Argument Summarization and its Evaluation in the Era of Large Language Models

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have revolutionized various Natural Language Generation (NLG) tasks, including Argument Summarization (ArgSum), a key subfield of Argument Mining (AM). This paper investigates the integration of state-of-the-art LLMs into ArgSum, including for its evaluation. In particular, we propose a novel prompt-based evaluation scheme, and validate it through a novel human benchmark dataset. Our work makes three main contributions: (i) the integration of LLMs into existing ArgSum frameworks, (ii) the development of a new LLM-based ArgSum system, benchmarked against prior methods, and (iii) the introduction of an advanced LLM-based evaluation scheme. We demonstrate that the use of LLMs substantially improves both the generation and evaluation of argument summaries, achieving state-of-the-art results and advancing the field of ArgSum. We also show that among the four LLMs integrated in (i) and (ii), Qwen-3-32B, despite having the fewest parameters, performs best, even surpassing GPT-4o, while LLaMA-3.3-70B consistently underperforms.

1 Introduction

In recent years, Large Language Models (LLMs) have significantly transformed various Natural Language Processing (NLP) and Generation (NLG) tasks. Their remarkable capabilities in understanding and generating human-like text promise new avenues for challenging tasks such as *Argument Summarization (ArgSum)*, a subfield of Argument Mining (AM) that focuses on distilling the essence of multiple arguments into concise representations (Friedman et al., 2021).[1]

With only a few recent exceptions (Li et al., 2024; Ziegenbein et al., 2024), however, ArgSum has up-to-date been mostly tackled with pre-LLM solutions, such as clustering techniques and earlier-generation pre-trained language models (Misra et al., 2016; Reimers et al., 2019; Ajjour et al., 2019; Wang and Ling, 2016; Schiller et al., 2021; Bar-Haim et al., 2020a,b; Alshomary et al., 2021; Li et al., 2023).

Thus, there is an urgent need for systematic analysis to understand how LLMs can be effectively utilized for both the generation and evaluation of argument summaries. This includes integrating LLMs into ArgSum frameworks to comprehensively assess their performance and developing suitable prompt-based evaluation schemes.

In this work, we aim to fill this gap by extensively exploring how LLMs can be leveraged for the ArgSum process, both for generating argument summaries and for their evaluation. Our contributions are: (i) We integrate LLMs into existing ArgSum systems, showing substantial performance gains. (ii) We introduce a new LLM-based ArgSum system, showing its superiority over existing approaches. (iii) We show that among the four LLMs used in (i) and (ii), the smallest one, Qwen3-32b, performs best, even surpassing GPT-4o, while LLaMA-3.3-70B consistently underperforms. (iv) We provide a new ArgSum evaluation dataset with human evaluation scores. (v) We develop a prompt-based ArgSum evaluation scheme, showing stronger correlation with human judgments than existing automatic evaluation metrics.

2 Related Work

2.1 Argument Summarization

Automatic Text Summarization (ATS) aims to condense the key ideas from one or more documents into a concise summary (Radev et al., 2002), while minimizing redundancy (Moratanch and Chitrakala, 2017). While *abstractive summarization* generates a summary including text units that do not necessarily appear in the source text, *extractive summarization* identifies the most important parts of a document and assembles them into a summary (Giarelis

[1] While past work on summarizing argumentative texts conveys different understandings of the task at hand, our understanding aligns with Key Point Analysis, introduced by Bar-Haim et al. (2020a,b).

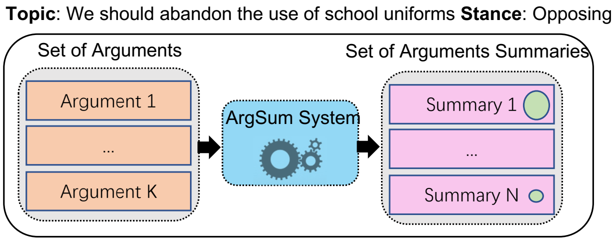

Figure 1: General procedure of ArgSum, where a set of K arguments on a certain debate topic and stance (example taken from Friedman et al. (2021)) is transformed to a set of N argument summaries along with their respective importance (indicated by the size of the green dots). It is expected that $K \gg N$ applies.

et al., 2023). ATS consists of several sub-areas like News Summarization (Sethi et al., 2017), Legal Document Summarization (Anand and Wagh, 2022), Scientific Paper Summarization (Zhang et al., 2018), and ArgSum (Bar-Haim et al., 2020a,b). Our focus is the latter.

Misra et al. (2016), Reimers et al. (2019) and Ajjour et al. (2019) treat the task of summarizing arguments as a clustering problem without providing easy-to-understand textual summaries. Wang and Ling (2016) frame ArgSum as claim generation, where a collection of argumentative sentences is summarized by generating a one-sentence abstractive summary that addresses the shared opinion of the inputs. Schiller et al. (2021) present an aspect-controlled argument generation model that enables an abstractive summatization of arguments.

Our understanding of ArgSum is in line with Key Point Analysis (KPA), introduced by Bar-Haim et al. (2020a,b), and is displayed in Figure 1. They aim to create an extractive summary consisting of the most prominent key points from a potentially large collection of arguments on a given debate topic and stance. Then, each source argument is classified according to the most suitable key point. Alshomary et al. (2021) perform the key point extraction by utilizing a variant of PageRank (Page et al., 1998). Li et al. (2023) extend KPA with a clustering-based and abstractive approach, using grouped arguments as input for a generation model to create key points. Khosravani et al. (2024) introduce a clustering-based and extractive approach, selecting the most representative argument within each cluster as a key point, determined by a supervised scoring model.

2.2 Evaluating NLG Systems

While automatic evaluation metrics such as BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004) correlate poorly with human judgments (Novikova et al., 2017), pre-trained transformer-based language models provide a more nuanced assessment of the performance of NLG systems (Celikyilmaz et al., 2021). BERTScore (Zhang et al., 2020) and MoverScore (Zhao et al., 2019) are reference-based metrics that leverage pre-trained embeddings obtained from BERT-based models. While BERTScore is based on the computation of cosine-similarities between the hypothesis and the reference, MoverScore determines an evaluation score by computing the Word Mover's Distance (Kusner et al., 2015) between both. BARTScore (Yuan et al., 2021) is based on the pre-trained sequence-to-sequence model BART and treats the evaluation task as a problem of text generation. BLEURT (Sellam et al., 2020) is reference-based and consists of a BERT-based model that is fine-tuned to predict human ratings in such a way that the metric is robust to quality shifts. MENLI (Chen and Eger, 2023) frames the evaluation task as a problem of Natural Language Inference (NLI), showing improved robustness. The most recent approaches to evaluation are LLM-based metrics, which can be leveraged in various ways: by comparing embeddings in terms of their cosine similarity (Es et al., 2024), by determining the sequence probability of the hypothesis given the respective source/reference (Fu et al., 2024), by utilizing suitable prompting strategies (Kocmi and Federmann, 2023; Liu et al., 2023; Fernandes et al., 2023; Leiter and Eger, 2024; Larionov and Eger, 2024), or by applying task-specific fine-tuning (Wang et al., 2024; Xu et al., 2023; Zhu et al., 2023). Some works show promising zero-shot results that are on-par with human-judgement (Leiter et al., 2023; Chang et al., 2024).

In this work, we leverage LLMs to evaluate ArgSum systems, which is different from evaluation of classical text generation systems, requiring different dimensions of evaluation (e.g., redundancy and coverage) and different mechanisms (e.g., ArgSum requires to compare m reference summaries to n generated summaries). To this end, we apply an LLM-based prompting approach and compare it with two existing ArgSum evaluation frameworks.

2.3 Argument Summarization and Evaluation with LLMs

Among the works that use LLMs for argument summarization or evaluation, Ziegenbein et al. (2024) use snippet generation and neutralization (a mix of extractive summarization and LLM prompting with reinforcement learning) for ArgSum in the context of argument search. They evaluate their approach automatically and manually, but do not apply LLMs for the evaluation. While their ArgSum task is different from ours, they also did not assess state-of-the-art LLMs like GPT-4o. Li et al. (2024) apply LLMs to argumentative essay summarization. They test a variety of state-of-the-art LLMs to generate reference summaries which are evaluated by humans. Their own summarization system, however, relies on smaller, instruction fine-tuned models rather than state-of-the-art LLMs. Different to our work, for the ArgSum evaluation, they only apply standard metrics like ROUGE.

Our work is the first *systematic* study on strategies for integrating LLMs with existing approaches for ArgSum and ArgSum evaluation.

3 Experimental Setup

3.1 Terminology

Most previous work on ArgSum can be categorized as either *classification-based* or *clustering-based* systems. Classification-based systems first generate a set of argument summaries based on all available source arguments. In a second step, they match each source argument to the most appropriate summary. Clustering-based systems first group all source arguments according to their similarity. Then, they generate a summary of the arguments for each cluster. In this work, we augment ArgSum systems of both types with LLMs. Details of those systems and how we integrate LLMs are specified in §3.2 and §3.3.

The systems we assess use two types of tools to perform ArgSum. While *Quality Scorers* assess the quality of an argument, *Match Scorers* determine how well an argument and a summary match. Both are realized by transformer-based language models that take task-specific textual inputs and output a respective score. The ArgSum systems considered in this work utilize Quality Scorers that are fine-tuned on the IBM-ArgQ-Rank-30kArgs (ArgQ) dataset by Gretz et al. (2020). The corresponding Match Scorers are fine-tuned on the ArgKP-2021 (ArgKP21) dataset by Friedman et al. (2021). Details on the required model fine-tuning for the ArgSum systems discussed in §3.2 and §3.3 are collected in Appendix A.

3.2 Classification-based Systems

We consider two classification-based ArgSum systems, which performed best in the Key Point Generation Track at the 2021 Key Point Analysis Shared Task (Friedman et al., 2021).

BarH To determine a set of potential argument summaries, referred to as candidates, BarH (Bar-Haim et al., 2020b) scores the source arguments with a Quality Scorer and selects those exceeding a threshold t_q, and also filters out arguments consisting of multiple sentences, arguments whose number of tokens exceeds a certain threshold n (=12), and arguments starting with pronouns. Subsequently, BarH applies a Match Scorer to match the remaining source arguments to the best fitting candidates. After ranking the candidates according to their number of matches, BarH minimizes redundancy by removing candidates whose match score with a higher-ranked candidate exceeds a threshold t_m. The remaining candidates are understood as the final argument summaries.

SMatchToPr To identify argument summary candidates, SMatchToPr (Alshomary et al., 2021) uses a variant of PageRank (Page et al., 1998). To this end, candidates are understood as nodes in an undirected graph, utilizing the match scores between each candidate pair as edge weights. Only nodes with edge weights above a threshold t_n are connected. Based on the resulting graph, an importance score is calculated for each candidate. Then, SMatchToPr minimizes redundancy by removing candidates whose match score with a higher-ranked candidate exceeds a threshold t_m. This results in the final set of argument summaries.

LLM Integration Given a set of arguments on a certain debate topic and stance, we apply a zero-shot prompting approach to instruct an LLM to generate either a set of candidates or argument summaries (see Appendix B.1). The resulting candidates or argument summaries are then further processed as usual in both BarH and SMatchToPr.

3.3 Clustering-based Systems

We consider an approach from Li et al. (2023) which demonstrated comparable performance to BarH and SMatchToPr. Further, we propose a new ArgSum approach that utilizes a Match Scorer for argument clustering.

USKPM For clustering arguments, USKPM (Li et al., 2023) utilizes the BERTopic framework (Grootendorst, 2022), which involves three steps. First, contextualized sentence embeddings of the arguments are created via SBERT (Reimers and

Gurevych, 2019). Second, UMAP (McInnes et al., 2018) is applied to reduce the embeddings' dimensionality. Third, the clustering of the reduced embeddings is performed by HDBSCAN (McInnes et al., 2017). Instances included in clusters with a size smaller than c are considered as unclustered. Since Li et al. (2023) state that it is reasonable to maximize the number of clustered arguments in order to increase the representativeness of the argument summaries to be generated, *Iterative Clustering* (*IC*) is proposed. IC is about incrementally assigning unclustered arguments to the most similar cluster in terms of cosine similarity.

Then, USKPM uses the instruction-tuned FLAN-T5 (Chung et al., 2022) to summarize the argument clusters, where the model input is formatted as follows: "summarize: {Stance} {Topic} {List of Arguments in Cluster}".

MCArgSum Our own approach, MCArgSum (Match Clustering based ArgSum), combines the use of a Match Scorer for argument clustering with an LLM-based cluster summarization. It is inspired by the redundancy reduction among candidates within BarH, where a Match Scorer is utilized to identify candidates addressing the same key point. We demonstrate that a Match Scorer can also be effectively used to group arguments addressing the same main statement. While the key idea of using a Match Scorer to group arguments is also proposed by Khosravani et al. (2024), our ArgSum system additional provides an abstractive summarization of argument clusters by incorporating an LLM.

Our ArgSum system utilizes Agglomerative Hierarchical Clustering (Day and Edelsbrunner, 1984) with the average linkage criterion in reference to Reimers et al. (2019) and a Match Scorer as pairwise similarity metric. To this end, we use the SBERT (Reimers and Gurevych, 2019) model all-mpnet-base-v2, fine-tuned on ArgKP21. While the threshold m determines the minimum match score required between two clusters to be merged, instances included in clusters with a size smaller than c are considered as unclustered.

To generate cluster summaries, our model uses LLM prompting in a zero-shot setting. We integrate a prompting strategy that summarizes all argument clusters simultaneously (global optimization). Details are given in Appendix B.2. After summarization, a post-processing step automatically extracts the argument summaries in the desired format.

3.4 Evaluation

Here, we describe the approaches used to evaluate ArgSum systems. These metrics are both set-based and reference-based, meaning a set of candidate summaries is compared to a set of reference summaries.

In accordance with previous work on generating argument summaries, we assess the two evaluation dimensions of *coverage* and *redundancy*. Coverage refers to the extent to which a set of argument summaries captures the central talking points of a debate. Redundancy is concerned with the extent of content overlap between the individual argument summaries (Bar-Haim et al., 2020b; Alshomary et al., 2021; Friedman et al., 2021; Li et al., 2023; Khosravani et al., 2024). Both criteria are closely related in order to assess the overall quality of a set of argument summaries, as high coverage can be achieved by generating many redundant argument summaries (Friedman et al., 2021).

Soft-Score Li et al. (2023) introduce three evaluation scores: *Soft-Precision* (*sP*), *Soft-Recall* (*sR*) and *Soft-F1* (*sF1*). While sP finds the most suitable reference summary for each candidate summary, sR finds the most suitable candidate summary for each reference summary. To compare references and candidates, Li et al. (2023) utilize a semantic similarity function. The final evaluation scores in terms of sP and sR are obtained by averaging the similarity scores of the respective best matches of references and candidates. Finally, the sF1 is the harmonic mean of sP and sR. Formally:

$$sP = \frac{1}{n} \cdot \sum_{a_i \in A} \max_{\beta_j \in B} f(a_i, \beta_j) \tag{1}$$

$$sR = \frac{1}{m} \cdot \sum_{\beta_j \in B} \max_{a_i \in A} f(a_i, \beta_j) \tag{2}$$

where f is a function evaluating the semantic similarity between two summaries; A and B are the sets of candidate and reference summaries, with n and m being their respective sizes. As similarity function, Li et al. (2023) suggest the use of BLEURT (Sellam et al., 2020) and BARTScore.

Coverage-Score (CS) The *Coverage-Score* (*CS*) (Khosravani et al., 2024) assesses the *coverage* of a set of candidate summaries, which is defined as the proportion of reference summaries covered by them. Each possible pair of candidates and references is scored by a Match Scorer and classified as matching or non-matching. The former corresponds to the case in which the respective match score is above a certain threshold t_M. Finally, the CS is derived as the proportion of references with at least one matching candidate. Formally:

$$CS = \frac{1}{m} \sum_{\beta_j \in B} \mathbb{1} \left[\sum_{a_i \in A} \mathbb{1} \left[match(a_i, \beta_j) > t_M \right] \geq 1 \right] \tag{3}$$

where *match* indicates the match score of two summaries; A and B are the sets of candidate and reference summaries, m is the size of B and t is the matching threshold. Khosravani et al. (2024) suggest the use of the Match Scorer inherent in BarH. A recommended threshold t_M is not provided.

LLM-based We introduce two one-shot prompting strategies for assessing ArgSum systems, focusing on the dimensions of coverage and redundancy. (i) We address coverage by instructing an LLM to count the number of reference summaries covered by a set of candidate summaries. Dividing this count of covered references by the total number of references results in an LLM-based coverage score. (ii) To assess redundancy, we instruct an LLM to count the number of unique main statements within a set of candidate summaries. The resulting uniqueness count is limited to the total number of candidates and a uniqueness score is derived by dividing the uniqueness count by the total number of candidates. Subsequently, we derive an LLM-based redundancy score as the complementary uniqueness score $(1 - \textit{uniqueness})$. The final LLM-based coverage and redunancy scores for a certain set of candidate summaries is obtained by averaging the results of 10 evaluation runs.

Human Evaluation To verify the reliability of the automatic evaluation metrics, we conduct a human evaluation of 126 generated argument summaries obtained from the ArgSum systems described in §3.2 and §3.3. We characterize a suitable set of argument summaries as consisting of succinct, non-redundant summaries that cover the main statements shared across the source arguments with adequate granularity. Thus, we assess the dimensions of coverage and redundancy, as introduced above.

The judgments are carried out by four experienced annotators with excellent knowledge within the field of NLP, especially argumentation. Initially, the four annotators are introduced to the task of ArgSum and provided with a description of the evaluation task. Guidelines can be found in Appendix C. The annotators are presented with a set of argument summaries (generated by the ArgSum systems discussed in Sections §3.2 and §3.3 and based on the arguments contained in ArgKP21) and the corresponding set of reference summaries. To assess coverage, they are asked to count the number of references that are covered by the set of generated summaries. The respective coverage score is the proportion of covered references out of the total number of references. We then ask the annotators to count the number of unique main statements within the set of generated summaries (permitted count is limited to the total number of generated summaries). Based on this, we derive a uniqueness score (ranging from zero to one) as the number of unique main statements divided by the total number of generated summaries. The redundancy score is the complementary uniqueness score.

In order to determine the inter-rater reliability, we average the Pearson correlation coefficients between each pair of the four annotators' scores for both dimensions. We report an average correlation of 0.697 for coverage and 0.722 for redundancy, indicating that the annotations are reliable. Pairwise correlations between annotators are shown in Figure 2 in the appendix.[2]

4 Results

In this section, we present the correlation of automatic metrics with human judgments in §4.1 and the evaluation of ArgSum systems in §4.2. Details on the experimental conditions, including data preprocessing, modifications to the ArgSum systems, hyperparameter settings, and hardware, are provided in Appendix D.

Data While ArgKP21 is used to train the Match Scorers utilized by BarH, SMatchToPr and MCArgSum, we use its test set to generate argument summaries in §4.1 and §4.2. This dataset consists of 27,519 pairs of arguments and key points, each labeled with a binary value that indicates whether the corresponding argument and key point are matching (1) or non-matching (0). While the pairs of arguments and key points cover 28 topics, each with supporting (1) and opposing (-1) stance, the dateset includes a train set of 24 topics, a development set of 4 topics and a test set of 3 topics.

[2] For annotator 1 (A1), the judgments for both stances of the third topic are missing, whereas the others (A2-A4) evaluated all three topics.

Temperature	LLM-based Coverage Score			LLM-based Redundancy Score		
	Across	Within	Runtime (s)	Across	Within	Runtime (s)
0.20	0.736	0.756 ± 0.100	495.1	0.798	0.697 ± 0.226	1305.3
0.30	0.725	0.747 ± 0.133	467.7	0.789	0.752 ± 0.096	1651.1
0.40	0.746	0.771 ± 0.112	529.3	0.817	0.758 ± 0.122	1515.4
0.50	0.742	0.757 ± 0.127	512.0	0.812	0.724 ± 0.210	1446.3
0.60	0.741	0.762 ± 0.122	629.7	0.837	0.795 ± 0.088	1359.9
0.70	0.755	0.789 ± 0.103	644.3	0.830	0.782 ± 0.112	1425.6
0.80	0.729	0.755 ± 0.108	612.4	0.828	0.762 ± 0.111	1431.1
0.90	0.754	0.782 ± 0.131	676.4	0.843	0.784 ± 0.109	1651.0
1.00	**0.767**	**0.803** ± 0.115	845.5	**0.852**	**0.824** ± 0.055	1649.1

Table 1: Pearson correlation between the LLM-based coverage and redundancy scores and the respective averaged human scores for different temperatures, along with the evaluation runtime, on ArgKP21. For the scenario within topics and stances, standard deviations are indicated alongside the correlation values.

We also consider the Debate dataset (Hasan and Ng, 2014) as a second independent evaluation data set in §4.2. Debate includes 3,228 argumentative text sequences filtered from posts on four different topics in an online debate forum. The text sequences are labeled with their reason within the respective topic and whether they are supporting (1) or opposing (-1). We consider the argumentative text sequences as arguments and the reasons as argument summaries. In contrast to ArgKP21, the dataset exclusively contains matching pairs.

Exemplary data points for ArgKP21 and Debate are presented in Table 4 and Table 5 in the appendix, respectively.

LLMs We integrated four LLMs into the ArgSum systems as described in §3.2 and §3.3, including GPT-4o,[3] LLaMa3.3-70b (Grattafiori et al., 2024), Qwen2.5-72b (Qwen et al., 2025) and Qwen3-32B (non-thinking mode) (Yang et al., 2025). GPT-4o-mini[4] was integrated into the LLM-based evaluation metric as discussed in §3.4, since it offers fast response times and is a cost-effective model version for the more quantity-based evaluation approach. We accessed OpenAI models via the official API[5] and open-source models via the OpenRouter API.[6]

4.1 Reliability of automatic metrics

To measure the quality of diverse automatic metrics, we correlate them to our human assessment of 126 argument summaries, see §3.4, where we average the four human assessments per instance, focusing on coverage as annotated by humans.

We consider two ways of computing correlations. (i) We calculate correlations **across** all topics and stances simultaneously. (ii) We calculate correlations **within** topics and stances and average the results. For the latter scenario, we also report the standard deviations, indicating the variability of reliability.

Soft-Score We apply the Soft-Score, explained in §3.4, with the following automatic metrics as similarity function f: (1) ROUGE 1, (2) BERTScore F1 (Zhang et al., 2020), (3) MoverScore (Zhao et al., 2019), (4) BARTScore (Yuan et al., 2021), (5) BLEURT (Sellam et al., 2020), (6) MENLI (Chen and Eger, 2023). Table 6 in the appendix shows the results.

First, we note that sP does not intuitively correspond to the annotation dimensions of coverage or redundancy in our human annotation — sP could be interpreted as the fraction of candidate summaries covered by the reference summaries, but not vice versa. Thus, it comes as no surprise that the correlation between sP and coverage is close to zero across all settings. The sR, which better matches the definition of coverage, performs clearly better, even if no strong correlations are observed. Across topics and stances, MENLI performs best (0.265) followed by BERTScore-F1 (0.254). The scenario within topics and stances generally yields better correlation results for the sR. While BERTScore-F1 exhibits the highest correlation at 0.402, MENLI (0.372) also achieves a moderate positive correlation with the human coverage scores. It is notable that BLEURT and

[3] https://platform.openai.com/docs/models/gpt-4o
[4] https://platform.openai.com/docs/models/gpt-4o-mini
[5] https://openai.com/index/openai-api/
[6] https://openrouter.ai/

BARTScore, suggested by Li et al. (2023), achieve the poorest results among all considered similarity functions.[7]

Coverage Score To examine the correlation of the CS with the averaged human coverage scores, we consider the Match Scorers of BarH, as proposed by Khosravani et al. (2024), as well as those of SMatchToPr and MCArgSum. Furthermore, we apply various values for the threshold t_M, which determines the match score for which an individual reference summary is understood as covered or not.

As depicted in Table 7 in the appendix, the CS with BarH's Match Scorer reaches a maximum correlation of 0.489 across and 0.698 within topics and stances. For the scenario across topics and stances, SMatchToPr performs even better and achieves a maximum correlation of 0.541. Within topics and stances, SMatchToPr reaches a maximum correlation of 0.6. The Match Scorer included in MCArgSum yields comparatively worse results, achieving a maximum correlation of 0.449 within and 0.551 across topics and stances. Regarding the matching threshold t_M, BarH's Match Scorer performs very stably across the considered parameter range, whereas this is not the case for both other variants.

To summarize, the CS provides considerably stronger correlations for the dimension of coverage compared to the Soft-Score.

LLM-based metric The LLM-based metrics for coverage and redundancy, described in §3.4, are examined regarding their respective criterion. Here, we investigate different values for the temperature, a parameter controlling the creativity or randomness in LLM-based text generation (Peeperkorn et al., 2024). The results are collected in Table 1.

The LLM-based score for coverage achieves a maximum correlation of 0.767 across and 0.803 within topics and stances. Consequently, it performs better than the Soft-Score and CS in all scenarios. The LLM-based metric for redundancy reaches also high correlations with a maximum value of 0.852 across and 0.824 within topics and stances. Thus, we exclusively use the LLM-based evaluation metrics to assess the argument summarization capability of ArgSum systems in §4.2.

4.2 System evaluation

Having identified the LLM-based evaluation metrics as the most reliable among those considered for both dimensions of coverage and redundancy, this section addresses their application in order to evaluate the ArgSum systems. In our investigations, we make use of a weighted evaluation score assessing both coverage and redundancy simultaneously. The Weighted Score ws for a certain set of argument summaries is defined as follows:

$$ws = \alpha \cdot c + (1 - \alpha) \cdot (1 - r) \qquad (4)$$

where c indicates the LLM-based coverage score and r indicates the LLM-based redundancy score. The weighting factor α is defined to be in the range [0, 1] and can be used to bias the Weighted Score either towards the coverage score or the redundancy score. For our investigations, we set the weighting factor to $2/3$, as we consider coverage to be more important than redundancy. We generate several argument summaries using various hyperparameter settings (see

	ArgKP21	Debate
Classification-based		
BarH	0.848	0.770
BarH+cand(gpt-4o)	0.877 ↑	0.847 ↑
BarH+cand(llama-3.3-70b)	0.845	0.807 ↑
BarH+cand(qwen-2.5-72b)	0.829	0.807 ↑
BarH+cand(qwen3-32b)	0.900 ↑	0.880 ↑
BarH+summ(gpt-4o)	0.859 ↑	0.839 ↑
BarH+summ(llama-3.3-70b)	0.812	0.784 ↑
BarH+summ(qwen-2.5-72b)	0.908 ↑	0.876 ↑
BarH+summ(qwen3-32b)	**0.925** ↑	**0.900** ↑
SMtPR	0.856	0.805
SMtPR+cand(gpt-4o)	0.884 ↑	0.869 ↑
SMtPR+cand(llama-3.3-70b)	0.816	0.815 ↑
SMtPR+cand(qwen-2.5-72b)	0.843	0.860 ↑
SMtPR+cand(qwen3-32b)	0.896 ↑	0.890 ↑
SMtPR+summ(gpt-4o)	0.853	0.842 ↑
SMtPR+summ(llama-3.3-70b)	0.796	0.795
SMtPR+summ(qwen-2.5-72b)	0.899 ↑	0.840 ↑
SMtPR+summ(qwen3-32b)	0.920 ↑	0.898 ↑
Clustering-based		
USKPM	0.800	0.833
MCArgSum(gpt-4o)	0.844	0.886
MCArgSum(llama-3.3-70b)	0.765	0.729
MCArgSum(qwen-2.5-72b)	0.847	0.880
MCArgSum(qwen3-32b)	0.853	0.898

Table 2: **Weighted Scores** of ArgSum systems on ArgKP21 and Debate datasets. For BarH and SMatchToPr (abbreviated as SMtPR), the variants with LLM-based candidates and summaries are indicated by +cand and +summ, respectively. Models in brackets indicate the LLMs integrated. We bold the best results on each dataset. ↑ indicates that classification-based systems with LLM integration outperform the original systems.

[7]We rescaled BARTScore according to Li et al. (2023) in order to obtain positive scores in the range from zero to one.

Appendix D.3) and report the best setting in terms of the Weighted Score for each ArgSum system. Since ArgSum is performed per topic and stance, the final evaluation score for each ArgSum system results as the average of the highest Weighted Scores within topics and stances. For simplicity, we refer to the averaged highest Weighted Score as the Weighted Score and the averaged coverage and redundancy score as the coverage and redundancy score, respectively.

Results The weighted scores of ArgSum systems are depicted in Table 2; we refer to Table 8 in Appendix E for full evaluation results including coverage and redundancy scores. For both **classification-based systems**, integrating an LLM generally improves performance. On ArgKP21, 9 out of 16 configurations of BarH and SMatchToPr with LLMs outperform their original versions in weighted scores. On Debate, all LLM-enhanced systems — except SMatchToPr+summ(llama-3.3-70b) — achieve higher weighted scores than their non-LLM counterparts. Qwen3-32B is the most effective LLM: it consistently boosts all generation systems and achieves the highest weighted scores across variants; BarH+summ(qwen3-32b) also ranks first on both datasets. GPT-4o performs slightly below Qwen-3-32B, improving scores in 7 out of 8 cases (except for SMatchToPr+summ). LLaMA-3.3-70B performs the worst, yielding the lowest weighted scores on both datasets (0.796 for SMatchToPr+summ(llama-3.3-70b) on ArgKP21 and 0.784 for BarH+summ(llama-3.3-70b) on Debate), and frequently ranks last among system variants. This is because, although systems using it achieve moderate coverage scores overall, they often exhibit higher redundancy scores compared to other systems. As for **clustering-based systems**, all MCArgSum variants except those with LLaMA-3.3-70B outperform USKPM in weighted scores (0.844-0.853 vs. 0.8 on ArgKP21; 0.880-0.898 vs. 0.833 on Debate). Similar to the classification-based systems, Qwen-3-32B performs best, ranking first on both datasets, while LLaMA-3.3-70B ranks last on both.

Qualitative inspection of the different systems' outputs (generated summaries) shows that the low scores of LLaMA-3.3-70B are to be attributed to the model's tendency to create very short summaries (bullet point style), while Qwen-3-32B and GPT-4o mostly produce full sentences (consistently across datasets). E.g., on the topic of "The USA is a good country to live in", LLaMA-3.3-70B creates summaries like "Offers freedom" or "Has many freedoms", while Qwen-3-32B produces summaries on the same topic like "The USA offers unparalleled freedom and the American dream." or "High levels of freedom and democratic values." We did not notice any systematic differences in the systems' outputs across topics and/or stances.

Overall, the integration of LLMs results in considerable improvements for classification-based as well as clustering-based ArgSum systems. On ArgKP21, classification-based systems outperform clustering-based ones on average, particularly those using LLM-based argument summaries, whereas both system types perform comparably on Debate.

The final choice of an ArgSum system should also depend on the runtime requirements. Using GPT-4o as an example, clustering-based systems are generally faster, with MCArgSum showing the best performance among all LLM-based ArgSum systems for both datasets. It required on average 3.779 seconds per topic and stance for ArgKP21 and 7.375 seconds for Debate (cf. hardware specifications in Appendix D.4).

5 Conclusion

Our proposed LLM-based ArgSum systems and metrics achieve state-of-the-art performance across the two datasets considered. MCArgSum, our newly proposed LLM-based ArgSum system outperforms existing approaches and has a runtime advantage against all other systems considered. ArgSum systems integrating Qwen-3-32B models achieve the state-of-the-art results. The LLM-based ArgSum evaluation scores we propose show very high correlation with human judgements and thus set a very reliable evaluation framework where reference summaries are available.

A few open questions and tasks remain: While we applied uniform prompts and parameter settings across all LLMs for consistency, optimizing them for each model may unlock further performance gains. Furthermore, we leave the application of reference-free evaluation strategies to future work.

Limitations

All inspected LLMs were trained on data that postdates the publication of Hasan and Ng (2014) and Friedman et al. (2021). Therefore, the evaluation datasets used in this work may have been seen during their training. However, similar limitations of potential data contamination are faced in many other recent problem settings as well; due to a lack of suitable ArgSum datasets, this issue is hard to avoid. We also point out that this work introduces a new evaluation benchmark for ArgSum systems, which could not have been seen by our employed LLMs. Additionally, prompts were initially designed for GPT-4o and applied uniformly across all LLMs, which may have resulted in an overestimation of GPT-4o's performance. Nevertheless, some LLMs still outperform GPT-4o in our evaluation. The system outputs included in the human evaluation do not cover those from ArgSum systems using the open-source LLMs.

Ethical Considerations

ArgSum systems could yield unreliable, factually incorrect, biased or even maliciously misleading summaries of the underlying source arguments — particularly, if certain arguments are misrepresented or filtered. Thus, the usage of ArgSum systems must always be made transparent, and recipients of the summarized arguments must interpret these with care.

We used ChatGPT solely for text refinement during the writing of this paper.

References

Yamen Ajjour, Milad Alshomary, Henning Wachsmuth, and Benno Stein. 2019. 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)*, pages 2922–2932, Hong Kong, China. Association for Computational Linguistics.

Milad Alshomary, Timon Gurcke, Shahbaz Syed, Philipp Heinisch, Maximilian Spliethöver, Philipp Cimiano, Martin Potthast, and Henning Wachsmuth. 2021. Key point analysis via contrastive learning and extractive argument summarization. In *Proceedings of the 8th Workshop on Argument Mining*, pages 184–189, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Deepa Anand and Rupali Wagh. 2022. Effective deep learning approaches for summarization of legal texts. *Journal of King Saud University - Computer and Information Sciences*, 34(5):2141–2150.

Roy Bar-Haim, Lilach Eden, Roni Friedman, Yoav Kantor, Dan Lahav, and Noam Slonim. 2020a. From arguments to key points: Towards automatic argument summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4029–4039, Online. Association for Computational Linguistics.

Roy Bar-Haim, Yoav Kantor, Lilach Eden, Roni Friedman, Dan Lahav, and Noam Slonim. 2020b. Quantitative argument summarization and beyond: Cross-domain key point analysis. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 39–49, Online. Association for Computational Linguistics.

Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. 2021. Evaluation of text generation: A survey.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. 2024. A survey on evaluation of large language models. *ACM Trans. Intell. Syst. Technol.*, 15(3).

Yanran Chen and Steffen Eger. 2023. MENLI: Robust evaluation metrics from natural language inference. *Transactions of the Association for Computational Linguistics*, 11:804–825.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

William H. E. Day and Herbert Edelsbrunner. 1984. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification*, 1:7–24.

Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024. RAGAs: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, St. Julians, Malta. Association for Computational Linguistics.

Patrick Fernandes, Daniel Deutsch, Mara Finkelstein, Parker Riley, André Martins, Graham Neubig, Ankush Garg, Jonathan Clark, Markus Freitag, and Orhan Firat. 2023. The devil is in the errors: Leveraging large language models for fine-grained machine translation evaluation. In *Proceedings of the Eighth Conference on Machine Translation*, pages 1066–1083, Singapore. Association for Computational Linguistics.

Roni Friedman, Lena Dankin, Yufang Hou, Ranit Aharonov, Yoav Katz, and Noam Slonim. 2021. Overview of the 2021 key point analysis shared task. In *Proceedings of the 8th Workshop on Argument Mining*, pages 154–164, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2024. GPTScore: Evaluate as you desire. In *Proceedings of the 2024 Conference of the North American

Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 6556–6576, Mexico City, Mexico. Association for Computational Linguistics.

Nikolaos Giarelis, Charalampos Mastrokostas, and Nikos Karacapilidis. 2023. Abstractive vs. extractive summarization: An experimental review. *Applied Sciences*, 13(13).

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly

Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The llama 3 herd of models.

Shai Gretz, Roni Friedman, Edo Cohen-Karlik, Assaf Toledo, Dan Lahav, Ranit Aharonov, and Noam Slonim. 2020. A large-scale dataset for argument quality ranking: Construction and analysis. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7805–7813. AAAI Press.

Maarten Grootendorst. 2022. Bertopic: Neural topic modeling with a class-based tf-idf procedure.

Kazi Saidul Hasan and Vincent Ng. 2014. 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)*, pages 751–762, Doha, Qatar. Association for Computational Linguistics.

Mohammad Khosravani, Chenyang Huang, and Amine Trabelsi. 2024. Enhancing argument summarization: Prioritizing exhaustiveness in key point generation and introducing an automatic coverage evaluation metric. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8212–8224, Mexico City, Mexico. Association for Computational Linguistics.

Tom Kocmi and Christian Federmann. 2023. Large language models are state-of-the-art evaluators of translation quality. In *Proceedings of the 24th Annual Conference of the European Association for Machine Translation*, pages 193–203, Tampere, Finland. European Association for Machine Translation.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 957–966, Lille, France. PMLR.

Daniil Larionov and Steffen Eger. 2024. Promptoptme: Error-aware prompt compression for llm-based mt evaluation metrics.

Christoph Leiter and Steffen Eger. 2024. PrExMe! large scale prompt exploration of open source LLMs for machine translation and summarization evaluation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11481–11506, Miami, Florida, USA. Association for Computational Linguistics.

Christoph Leiter, Juri Opitz, Daniel Deutsch, Yang Gao, Rotem Dror, and Steffen Eger. 2023. The Eval4NLP 2023 shared task on prompting large language models as explainable metrics. In *Proceedings of the 4th Workshop on Evaluation and Comparison of NLP Systems*, pages 117–138, Bali, Indonesia. Association for Computational Linguistics.

Hao Li, Viktor Schlegel, Riza Batista-Navarro, and Goran Nenadic. 2023. 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)*, pages 14064–14080, Toronto, Canada. Association for Computational Linguistics.

Hao Li, Yuping Wu, Viktor Schlegel, Riza Batista-Navarro, Tharindu Madusanka, Iqra Zahid, Jiayan

Zeng, Xiaochi Wang, Xinran He, Yizhi Li, and Goran Nenadic. 2024. Which side are you on? a multi-task dataset for end-to-end argument summarisation and evaluation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 133–150, Bangkok, Thailand. Association for Computational Linguistics.

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

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: NLG evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore. Association for Computational Linguistics.

Leland McInnes, John Healy, and Steve Astels. 2017. hdbscan: Hierarchical density based clustering. *Journal of Open Source Software*, 2(11):205.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. 2018. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861.

Amita Misra, Brian Ecker, and Marilyn Walker. 2016. Measuring the similarity of sentential arguments in dialogue. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 276–287, Los Angeles. Association for Computational Linguistics.

N. Moratanch and S. Chitrakala. 2017. A survey on extractive text summarization. In *2017 International Conference on Computer, Communication and Signal Processing (ICCCSP)*, pages 1–6.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for NLG. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252, Copenhagen, Denmark. Association for Computational Linguistics.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1998. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA. Association for Computational Linguistics.

Max Peeperkorn, Tom Kouwenhoven, Dan Brown, and Anna Jordanous. 2024. Is temperature the creativity parameter of large language models?

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 technical report.

Dragomir R. Radev, Eduard Hovy, and Kathleen McKeown. 2002. Introduction to the special issue on summarization. *Computational Linguistics*, 28(4):399–408.

Nils Reimers and Iryna Gurevych. 2019. 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)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Nils Reimers, Benjamin Schiller, Tilman Beck, Johannes Daxenberger, Christian Stab, and Iryna Gurevych. 2019. Classification and clustering of arguments with contextualized word embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 567–578, Florence, Italy. Association for Computational Linguistics.

Benjamin Schiller, Johannes Daxenberger, and Iryna Gurevych. 2021. Aspect-controlled neural argument generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–396, Online. Association for Computational Linguistics.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Prakhar Sethi, Sameer Sonawane, Saumitra Khanwalker, and R. B. Keskar. 2017. Automatic text summarization of news articles. In *2017 International Conference on Big Data, IoT and Data Science (BID)*, pages 23–29.

Lu Wang and Wang Ling. 2016. Neural network-based abstract generation for opinions and arguments. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 47–57, San Diego, California. Association for Computational Linguistics.

Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, Wei Ye, Shikun Zhang, and Yue Zhang. 2024. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization.

Wenda Xu, Danqing Wang, Liangming Pan, Zhenqiao Song, Markus Freitag, William Wang, and Lei Li. 2023. INSTRUCTSCORE: Towards explainable text generation evaluation with automatic feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5967–5994, Singapore. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. 2025. Qwen3 technical report.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021. Bartscore: Evaluating generated text as text generation. In *Advances in Neural Information Processing Systems*, volume 34, pages 27263–27277. Curran Associates, Inc.

Junsheng Zhang, Kun Li, and Changqing Yao. 2018. Event-based summarization for scientific literature in chinese. *Procedia Computer Science*, 129:88–92. 2017 INTERNATIONAL CONFERENCE ON IDENTIFICATION,INFORMATION AND KNOWLEDGEIN THE INTERNET OF THINGS.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. 2019. MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 563–578, Hong Kong, China. Association for Computational Linguistics.

Lianghui Zhu, Xinggang Wang, and Xinlong Wang. 2023. Judgelm: Fine-tuned large language models are scalable judges.

Timon Ziegenbein, Shahbaz Syed, Martin Potthast, and Henning Wachsmuth. 2024. Objective argument summarization in search. In *Robust Argumentation Machines*, pages 335–351, Cham. Springer Nature Switzerland.

A Details on Model Fine-tuning

BarH and SMatchToPr We fine-tuned the Match Scorers and Quality Scorers in BarH and SMatchToPr according to Bar-Haim et al. (2020b) and Alshomary et al. (2021), respectively. It is important to note that Bar-Haim et al. (2020b) do not specify which of the two quality scores (MACE-P and WA) in ArgQ should be used for training the Quality Scorer. Additionally, it is unclear whether a model with or without a pooling layer was used. Since the model without pooling layer and fine-tuned on MACE-P performs best in preliminary investigations, we applied it in BarH.

USKPM The fine-tuning of FLAN-T5 in USKPM was conducted as proposed by Li et al. (2023), though no specific learning rate was provided. Based on our observations, a learning rate of 4e-4 worked well and was therefore used for fine-tuning the model.

MCArgSum As Match Scorer, MCArgSum uses the SBERT model "all-mpnet-base-v2" fine-tuned on ArgKP21. The fine-tuning is conducted over 10 epochs with a learning rate of 5e-6 and contrastive loss. The best performing model on the development set was selected as final model.

B LLM Prompting

LLM prompting can be divided into a system message and a user message. The system message guides the LLM on its general behavior, while the user message specifies the exact task. We also utilize the system message to introduce the task at hand and to describe the desired appearance of the argument summaries.

B.1 Classification-based Systems

The proposed prompting strategy instructs the LLM to generate either candidates or argument summaries. In both cases, the prompt is divided into a system message and a user message. The following prompt template is applied for both generating candidates and argument summaries, but used differently. For generating candidates, we instruct the LLM to produce a large number of key points (12 to 20). In contrast, for argument summaries, we request fewer key points (4 to 8) and apply the optional user message to minimize redundancy. A description of the parameters and placeholders contained in the prompt template is given below.

System Message You are a professional debater and you can express yourself succinctly. If you are given a corpus of arguments on a certain debate topic and stance, you find `{num_kps}` appropriate salient single sentences, called key points, summarizing most of the arguments and providing a textual and quantitative view of the data. A key point can be seen as a meta argument why one is for or against a certain topic. Make sure that the generated key points summarize the majority of the arguments contained in the corpus. A key point should not exceed a length of `{kp_token_length}` tokens. Here are two examples of good key points: “School uniform reduces bullying” is an opposing key point on the topic “We should abandon the use of school uniform" and "Guns lead to accidental deaths” is a supporting key point on the topic "We should abolish the right to keep and bear arms".

User Message Please generate `{num_kps}` short (maximal length of `{kp_token_length}` tokens), salient and high quality `{stance}` key points on the topic “`{topic}`” so that they capture the main statements that are shared between most of the arguments based on the following corpus of arguments: `{arguments}`.

Optional User Message for generating argument summaries You should only generate as many key points as necessary to summarize the arguments contained in the corpus. This means you should preferably generate fewer key points than the maximum permitted number of `{max_num_kps}` key points instead of generating overlapping key points in terms of content.

Parameters/Placeholders

- `num_kps`: Number of key points (can be a fixed value or a range of values)
- `kp_token_length`: Maximum permitted number of tokens for key points
- `stance`: Stance of arguments (supporting or opposing)
- `topic`: Topic of arguments
- `arguments`: List of arguments
- `max_num_kps`: Maximum permitted number of key points

B.2 Clustering-based Systems

The prompting for LLM-based Cluster Summarization is divided into a system message and a user message. A description of the parameters and placeholders contained in the prompt template is given below.

System Message You are a professional debater and you can express yourself succinctly. If you are given a cluster of similar arguments on a certain debate topic and stance, you find a single appropriate salient sentences, called key point, capturing the main statement that is shared between most of the clustered arguments and providing a textual and quantitative view of the data. A key point can be seen as a meta argument why one is for or against a certain topic. Since argument clusters are not perfect, they may contain arguments that do not actually belong together. Therefore, make sure that a generated key point summarizes the majority of the arguments contained in the cluster. A key point should not exceed a length of `{kp_token_length}` tokens. Here are two examples of good key points: “School uniform reduces bullying” is an opposing key point on the topic “We should abandon the use of school uniform" and “Guns lead to accidental deaths” is a supporting key point on the topic “We should abolish the right to keep and bear arms”.

User Message Please generate a single short (maximal length of `{kp_token_length}` tokens), salient and high quality `{stance}` key point on the topic “`{topic}`” so that it captures the main statement that is shared among most of the clustered arguments for each of the following `{num_clusters}` clusters of similar arguments: `{clusters}`. Since argument clusters are not perfect, they may contain arguments that do not actually belong together. Therefore, make sure that each generated key point summarizes the majority of the arguments contained in the respective cluster. In addition, ensure that the generated key points do not overlap in terms of content. Do not deliver an explanation why you generated the key points or any other information. Only return the cluster ids and corresponding individual key points.

Parameters/Placeholders

- `kp_token_length`: Maximum permitted number of tokens for key points
- `stance`: Stance of arguments (supporting or opposing)
- `topic`: Topic of arguments
- `arguments`: List of arguments
- `num_clusters`: Number of clusters
- `clusters`: List of argument clusters, where each cluster consists of a cluster id and a list of the corresponding arguments

B.3 LLM-based Evaluation

For the evaluation, we only worked with user messages.

User Message for Coverage Evaluation Your task is to evaluate a set of generated summaries obtained from a collection of arguments against a set of reference summaries. The evaluation is conducted according to the criteria of coverage, meaning that the set of generated summaries aims to cover the main statements contained in the set of reference summaries. Since each reference summary addresses a unique main statement, you are asked to count the number of reference summaries that are covered by the set of generated summaries. If a reference summary is only partially covered by the set of generated summaries, an increase of the count by 0.5 is allowed. Your counts aim to correlate well with human judgments.

Make sure to always print the final count in the format "Coverage count: x.y" in a new line with no additional text in that line.

Example:

Set of Reference Summaries:

1. Banning guns would save lives
2. Guns can fall into the wrong hands
3. Guns lead to accidental deaths
4. Gun ownership allows for mass-shootings/general gun violence

Set of Generated Summaries:

1. Banning guns would save thousands of lives
2. Some people do not know how to handle firearms. This is a danger to them and others.
3. Guns kill people, they should be banned
4. Firearms can fall into the hands of potential murderers
5. Firearms are a disgrace to humanity.
6. Without weapons, there would be no war.

Coverage count: 3.5

Evaluation Procedure:

1. Read the reference summaries. Do not print them again.
2. Read the generated summaries. Do not print them again.
3. Go through the set of reference summaries and determine whether the reference summary at hand is covered by at least one generated summary.
4. Once you have done this for each reference summary, count the number of covered reference summaries and return the resulting coverage count.

Evaluation Task:

Set of Reference Summaries:

`reference_summaries`

Set of Generated Summaries:

`candidate_summaries`

User Message for Redundancy Evaluation Your task is to evaluate a set of arguments on a certain debate topic and stance according to their uniqueness. Since arguments can be formulated differently, but address the same aspect of a debate, your task is to count the number of unique main statements addressed by the set of arguments. If a main statement addressed by an argument is only partially unique because it is also in parts covered by another argument, an increase of the count by 0.5 is allowed. Your counts aim to correlate well with human judgments.

In the following, you are provided with an example, instructions for the evaluation procedure, and finally with your evaluation task.

Example:

Set of Arguments:

1. Banning guns would save lives
2. Guns can fall into the wrong hands
3. Guns lead to accidental deaths
4. Guns kill people, they should be banned
5. Gun ownership allows for mass-shootings/general gun violence
6. Some people do not know how to handle firearms. This is a danger to them and others.
7. Banning guns would save thousands of lives
8. Firearms can fall into the hands of potential murderers

Number of Unique Main Statements: 4

Explanation:

- Argument 1, 4, and 7 address the same main statement (guns kill people so without guns lives could be saved)
- Argument 2, 6, and 8 address the same main statement (guns could fall into the wrong hands, such as murders or people not knowing how to handle guns)
- Argument 3 addresses a unique main statement, focusing on accidents with guns
- Argument 5 addresses a unique main statement, focusing on intentional killing like terrorism or running amok

Notes:

- Arguments 1, 4, and 7 are quite general, and therefore differ from the others
- E.g., argument 3 could also be assigned to 1, 4, and 7. Nevertheless, it focuses on accidents and is more specific

Evaluation Procedure:

1. Read the arguments. Do not print them.
2. Go through the list of arguments, starting with the first argument.
3. Determine whether the argument at hand addresses a main statement of the debate.
4. Move on to the next one and consider whether it addresses a main statement and whether it has already been covered by previous arguments in the list.
5. Once you have done this for each argument, count the total number of unique main statements.
6. Return your uniqueness count in the format "Number of Unique Main Statements: x.y" in a new line with no additional text in that line. Always make this line the last line of your response and always include it.

Evaluation Task:

Set of Arguments: `candidate_summaries`

Number of Unique Main Statements:

Generation of Candidate Summaries and Reference Summaries Candidate Summaries and Reference Summaries are constructed by iterating over lists of generated and reference summaries, respectively. Each element in the list is formatted as an enumerated string, where each entry is prefixed with its index and a period. This ensures a structured representation of arguments or summaries for evaluation. Below is an example of how a list of three reference summaries would be converted into a formatted string:

Set of Reference Summaries:

1. Renewable energy reduces carbon emissions.
2. Solar panels provide long-term cost savings.
3. Wind power is a reliable energy source.

Similarly, a set of generated summaries follows the same structure, ensuring consistency in comparison.

C Human Evaluation

C.1 Introduction to ArgSum

A debate on a certain topic can be conducted using a variety of arguments for each side of the debate. Although some of these arguments refer to the same main statement, they can be formulated very differently. While the number of possible arguments seems to be almost infinite due to the possibility of different formulations, the number of possible main statements within a debate is limited.

Argument summarization is about summarizing a relatively large set of arguments on a certain debate topic and stance by generating a small set of argument summaries, each expressing one distinct main statement contained in the set of arguments. In addition, each argument is matched to the generated summary that conveys its main statement the best. Following is a simple example:

Topic:

We should abandon the use of school uniform

Stance:

Opposing

Set of Arguments:

1. School uniforms keep everyone looking the same and prevent bullying
2. School uniforms can help parents save money on outfit
3. School uniforms help stop bullying because when people are similarly dressed, nobody is made to feel inferior
4. It is cheaper for parents to buy school uniforms, which is helpful to parents that are struggling financially
5. School uniforms are substantially more affordable

Set of Summaries:

1. School uniforms reduce bullying
2. School uniforms save costs

Argument Summary Matches:

The matches are highlighted by the colored markings:

- Arguments 1 and 3 are matched to summary 1
- Arguments 2, 4 and 5 are matched to summary 2

C.2 Description of the Evaluation Task

This task is about determining how well a set of generated argument summaries serves as a summary of possible arguments on a certain debate topic and stance.

For this purpose, you are given a set of generated summaries and a set of reference summaries as well as the corresponding debate topic and stance. You have to carry out the following two instructions regarding the dimensions of coverage and uniqueness:

1. **Coverage**: Count the number of reference summaries that are covered by the set of generated summaries.
2. **Uniqueness**: Count the number of distinct/unique main statements contained in the set of generated summaries.

For both dimensions increments of 0.5 are allowed. In the case of coverage, this applies if a reference summary is only partially covered by the set of generated summaries. For the dimension of redundancy, this applies if there is a distinct main statement in the set of generated summaries that partially overlaps with another. For the case you are not sure, you can answer with -1. Following is an example:

Topic:

Routine child vaccinations should be mandatory

Stance:

Opposing

Set of Reference Summaries:

1. Mandatory vaccination contradicts basic rights
2. Routine child vaccinations are not necessary to keep children healthy
3. Routine child vaccinations, or their side effects, are dangerous
4. The parents and not the state should decide

Set of Generated Summaries:

1. Vaccinations violate free will and personal choice
2. Mandatory vaccines conflict with religious beliefs
3. Parents should have the right to decide
4. Children may suffer harmful effects from vaccines
5. Concerns about vaccine safety and side effects

Coverage:

3 (The second reference summary is not covered.)

Uniqueness:

3.5 (The first and third generated summaries address two different distinct main statements. The fourth and fifth generated summaries refer to the same distinct main statement. The second generated summary partially overlaps with the first one.)

D Experimental conditions

D.1 Data Preprocessing

To conduct our investigations on the test split of ArgKP21 as well as Debate, we performed two preprocessing steps. First, we remove arguments that do not have exactly one matching argument summary. The reason for this is that we aim to process only those arguments that have a well-defined reference summary. This is because the considered automatic evaluation metrics are reference-based. Including arguments without any reference could result in candidate summaries that are not captured by the references and thus bias the evaluation of ArgSum systems.

Second, we exclude arguments consisting of more than one sentence, as we consider an adequate argument to consist of a single sentence. This is particularly crucial for the argumentative text sequences contained in Debate. For the test split of ArgKP21, the pre-processing reduces the number of arguments from 732 to 428, while for Debate it is reduced from 3180 to 2321. Finally, to decrease the computational effort, we select only 50% of the arguments for each unique argument summary in Debate as our final dataset. This pre-processing step results in 1165 remaining arguments for Debate, while retaining each unique argument summary.

D.2 Modifications to ArgSum Systems

We had to apply three modifications to the `ArgSum` systems as proposed in §3. The first concerns the candidate selection in BarH and SMatchToPr. In cases where the proportion of candidates out of all arguments is below a certain threshold p_C, we fill this gap with the highest quality arguments not yet considered as candidates. In this way, we avoid cases in which no candidates are identified at all, as the Quality Scorer provides low scores across all arguments. Second, when selecting candidates in SMatchToPr, we delete arguments consisting of several sentences instead of separating them. Finally, we use the Quality Scorer included in BarH instead of TextRank for determining the order of arguments in the corresponding input list of Flan-T5 in USKPM.

D.3 Details on Hyperparameters

When applying BarH and SMatchToPr, we used the recommended parameter values from Bar-Haim et al. (2020b) and Alshomary et al. (2021), respectively. In case of USKPM and MCArgSum, we set the minimum cluster size c to 3. The similarity threshold for IC in USKPM was set to zero, meaning that we forced each unclustered argument to be assigned to an existing cluster. In addition, Table 3 includes the varying hyperparameter settings for the argument clustering inherent in USKPM and MCArgSum. For USKPM, we performed the clustering for each possible combination of the depicted parameter values.

D.4 Hardware

We conducted our experiments on a personal computer with an Apple M1 Max chip, which is designed as a system-on-a-chip. It includes a 10-core CPU (8 performance cores and 2 efficiency cores), a 32-core GPU, and a 16-core Neural Engine. The GPU has direct access to the entire main memory of 64GB. The system runs on macOS Sonoma 14.1.2 (64-bit). With the introduction of Metal support for PyTorch on macOS, utilizing the GPU for machine learning tasks has become accessible.[8] This setup was used for both training and inference of PyTorch models.

E Tables and Figures

[8] https://pytorch.org/blog/introducing-accelerated-pytorch-training-on-mac

	Parameter	Value Range	Steps
USKPM	Reduced embedding dimensionality	$[2, 5]$	1
	Number of neighboring samples used for the manifold approximation of UMAP	$[2, 5]$	1
	Minimum permitted distance of points in the low dimensional representation of UMAP	$[0, 0.4]$	0.2
MCArgSum	Minimum match score required between two clusters to be merged (m)	$[0.05, 0.95]$	0.025

Table 3: Hyperparameter settings of clustering-based ArgSum systems considered in our investigations.

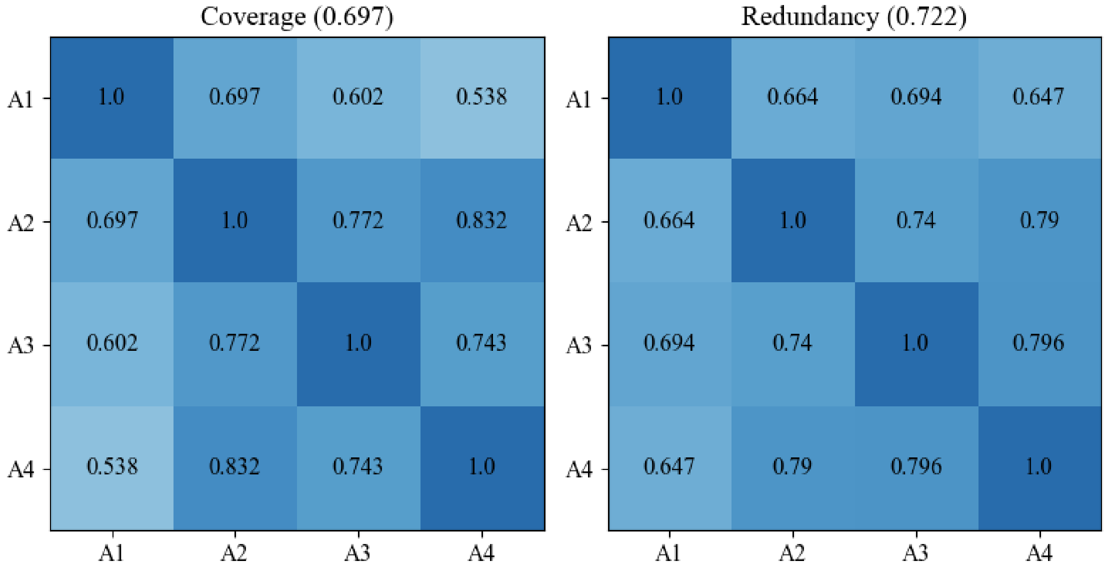

Figure 2: Pairwise Pearson correlation coefficient of the human judgments by the four annotators (A1-A4) for the criteria of coverage and redundancy. The averaged value across annotator pairs is indicated in the parentheses.

Topic	We should abandon the use of school uniform
Stance	-1
Argument	school uniforms cut down on bulling and keep everyone the same.
Key Point	School uniform reduces bullying
Label	1
Set	dev

Table 4: Exemplary data point of ArgKP21.

Topic	obama
Stance	-1
Argument	Where are those outspoken democrats who voted for him because they were told, no promised, that he would END THE WAR?
Argument Summary	Wars are still on

Table 5: Exemplary data point of Debate.

Similarity Function	Soft-Precision		Soft-Recall		Soft-F1		Run-time (s)
	Across	Within	Across	Within	Across	Within	
ROUGE 1	-0.118	-0.072 ±0.194	0.164	0.315 ±0.170	0.027	0.127 ±0.184	0.428
BERTSc. F1	-0.028	0.092 ±0.262	0.254	**0.402** ±0.175	0.121	0.240 ±0.242	354.2
MoverSc.	-0.046	0.044 ±0.227	0.156	0.310 ±0.204	0.069	0.191 ±0.207	55.93
BARTSc. CNN/DM	-0.146	-0.305 ±0.164	0.024	-0.011 ±0.283	-0.053	-0.132 ±0.264	84.33
BARTSc. Parabank	-0.271	-0.221 ±0.251	-0.012	0.112 ±0.339	-0.132	-0.022 ±0.320	41.09
BLEURT	-0.209	-0.218 ±0.289	0.033	0.138 ±0.247	-0.091	-0.055 ±0.294	487.3
MENLI	-0.154	-0.039 ±0.287	**0.265**	0.372 ±0.260	0.107	0.228 ±0.298	254.6

Table 6: Pearson correlation between the Soft-Score (incl. different similarity functions) and averaged human coverage scores, along with the evaluation runtime, on ArgKP21. For the scenario within topics and stances, standard deviations are indicated below the correlation values.

Threshold	**CS** (BarH)		**CS** (SMatchToPr)		**CS** (MCArgSum)	
	Across	Within	Across	Within	Across	Within
0.40	0.478	0.585 ±0.254	-0.092	-0.157 ±0.037	0.206	-0.057 ±0.223
0.45	0.475	0.605 ±0.248	0.163	-0.074 ±0.187	0.300	-0.019 ±0.307
0.50	0.465	0.627 ±0.251	0.174	-0.023 ±0.196	0.300	-0.019 ±0.307
0.55	0.462	0.657 ±0.273	0.378	0.249 ±0.307	0.281	-0.002 ±0.292
0.60	**0.489**	**0.698** ±0.222	0.469	0.338 ±0.371	0.415	0.137 ±0.390
0.65	0.464	0.676 ±0.218	0.465	0.411 ±0.256	**0.449**	0.256 ±0.357
0.70	0.458	0.657 ±0.233	0.457	0.404 ±0.212	0.369	0.297 ±0.295
0.75	0.466	0.658 ±0.197	**0.541**	0.550 ±0.182	0.379	0.347 ±0.319
0.80	0.429	0.591 ±0.154	0.511	**0.600** ±0.196	0.444	**0.551** ±0.193
0.85	0.414	0.556 ±0.201	0.468	0.558 ±0.085	0.364	0.421 ±0.270
0.90	0.295	0.504 ±0.249	0.238	0.261 ±0.070	0.316	0.401 ±0.129
Average Runtime (s)	70.302		20.961		14.689	

Table 7: Pearson correlation coefficient between the CS (incl. different Match Scorers) and averaged human coverage scores for different matching thresholds, along with the evaluation runtime, on ArgKP21. For the scenario within topics and stances, standard deviations are indicated alongside the correlation values.

	ArgKP21			Debate		
	Coverage	Redundancy	Weighted	Coverage	Redundancy	Weighted
Classification-based						
BarH	0.819	0.093	0.848	0.764	0.218	0.770
BarH+cand(gpt-4o)	0.904	0.177	0.877	0.843	0.146	0.847
BarH+cand(llama-3.3-70b)	0.829	0.125	0.845	0.806	0.192	0.807
BarH+cand(qwen-2.5-72b)	0.830	0.173	0.829	0.825	0.228	0.807
BarH+cand(qwen3-32b)	0.915	0.129	0.900	0.891	0.142	0.880
BarH+summ(gpt-4o)	0.813	0.048	0.859	0.797	0.075	0.839
BarH+summ(llama-3.3-70b)	0.843	0.250	0.812	0.824	0.295	0.784
BarH+summ(qwen-2.5-72b)	0.909	0.093	0.908	0.872	0.117	0.876
BarH+summ(qwen3-32b)	0.943	0.109	0.925	0.904	0.110	0.900
SMtPR	0.905	0.240	0.856	0.780	0.147	0.805
SMtPR+cand(gpt-4o)	0.912	0.172	0.884	0.862	0.116	0.869
SMtPR+cand(llama-3.3-70b)	0.898	0.348	0.816	0.893	0.343	0.815
SMtPR+cand(qwen-2.5-72b)	0.853	0.176	0.843	0.864	0.150	0.860
SMtPR+cand(qwen3-32b)	0.933	0.177	0.896	0.922	0.173	0.890
SMtPR+summ(gpt-4o)	0.803	0.048	0.853	0.785	0.044	0.842
SMtPR+summ(llama-3.3-70b)	0.837	0.287	0.796	0.846	0.306	0.795
SMtPR+summ(qwen-2.5-72b)	0.891	0.086	0.899	0.827	0.134	0.840
SMtPR+summ(qwen3-32b)	0.924	0.088	0.920	0.880	0.067	0.898
Clustering-based						
USKPM	0.824	0.249	0.800	0.806	0.112	0.833
MCArgSum(gpt-4o)	0.844	0.156	0.844	0.884	0.112	0.886
MCArgSum(llama-3.3-70b)	0.713	0.132	0.765	0.636	0.084	0.729
MCArgSum(qwen-2.5-72b)	0.809	0.079	0.847	0.887	0.134	0.880
MCArgSum(qwen3-32b)	0.839	0.119	0.853	0.896	0.099	0.898

Table 8: Coverage and redundancy scores as well as Weighted Scores for ArgKP21 (left) and Debate (right). For BarH and SMatchToPr (abbreviated as SMtPR), the variants with LLM-based candidates and summaries are indicated by +cand and +summ, respectively. Models in brackets indicate the LLMs integrated.