# 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, addressing the challenges of traditional evaluation metrics, which do not align well with human judgment. We propose a novel prompt-based evaluation scheme, and validate it through a novel human benchmark dataset. Our work makes three key contributions: the integration of LLMs into existing ArgSum frameworks, the development of a new ArgSum system benchmarked against prior methods, and 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.

## 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 (Petasis and Karkaletsis, 2016).

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 (Reimers et al., 2019; Ajjour et al., 2019; Misra et al., 2016; Wang and Ling, 2016; Schiller et al., 2021).

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 core contributions are: (i) We integrate LLMs into existing ArgSum systems and evaluation schemes, (ii) we introduce a new ArgSum system and benchmark existing work against it, (iii) we provide a new `ArgSum` evaluation dataset with human evaluation scores, and (iv) we develop a prompt-based ArgSum evaluation scheme and validate it in the context of existing work and our human evaluation.

We show that the use of LLMs substantially improves existing work in both the process of ArgSum as well as ArgSum evaluation and present several state-of-the-art results.

## 2 Related Work

### 2.1 Argument Summarization

Automatic Text Summarization (ATS) aims to condense 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 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 (Petasis and Karkaletsis, 2016). Our focus is the latter.

Wang and Ling (2016) treat 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. Friedman et al. (2021); Bar-Haim et al. (2020) introduce Key Point Analysis (KPA), which aims to create an extractive summary of the most prominent key points contained in a potentially large collection of arguments. Schiller et al. (2021) present an aspect-controlled argument generation model that enables an abstractive summatization of arguments. Li et al. (2023) extend KPA with a clustering-based, abstractive approach. While they use all arguments as input for a generation model to create key points, Khosravani et al. (2024) introduce an extractive approach by selecting the most representative argument within each cluster, which is determined by a supervised scoring model.

While there are some works on LLMs as summarization models for arguments (Li et al., 2024; Ziegenbein et al., 2024), this is the first *systematic* study on strategies for integrating LLMs with existing approaches for ArgSum. Furthermore, we also present a systematic overview on how to integrate LLMs into the evaluation of `ArgSum` systems.

### 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. 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 a gold summary of $m$ arguments to a generated summary of $n$ arguments). To this end, we apply an LLM-based prompting approach besides incorporating classical NLG metrics into `ArgSum` adjusted evaluation schemes.

## 3 Experimental Setup

### 3.1 Terminology

Our understanding of `ArgSum` is based on that of KPA, where `ArgSum` is performed per topic and stance. It is assumed that a set of arguments is given and that the `ArgSum` systems generate a set of argument summaries, as displayed in Figure 1.

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 group. 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 cor-

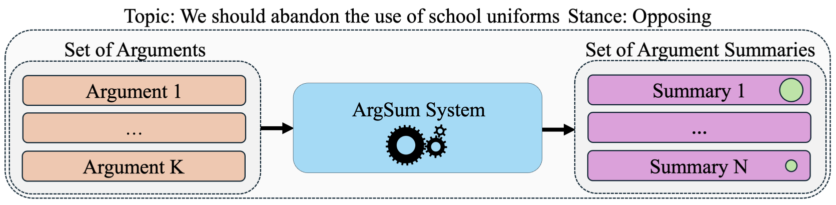
Topic: We should abandon the use of school uniforms Stance: Opposing
Set of Arguments
Argument 1
...
Argument K
ArgSum System
Set of Argument Summaries
Summary 1
...
Summary N

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)) as input 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 $K >> N$ applies.

responding Match Scorers are fine-tuned on the ArgKP-2021 (ArgKP21) dataset by Friedman et al. (2021).

### 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., 2020) scores the source arguments with a Quality Scorer and selects those exceeding a threshold $t_q$. 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. 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 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, inctances 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 the appendix. 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 criteria 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., 2020; Alshomary et al., 2021; Friedman et al., 2021; Li et al., 2023; Khosravani et al., 2024).

**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. Finally, the CS is derived as the proportion of references with at least one matching candidate. Formally:

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

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 $t$ is not provided.

**LLM-based** We introduce two LLM one-shot prompting strategies for assessing `ArgSum` systems, focusing on the criteria of coverage and redundancy. (1) 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. (2) 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 criteria 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 §C in the appendix. The annotators are presented with a set of generated argument summaries 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 criteria. 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 5 in the appendix.

## 4 Results

In this section, we present the correlation of automatic metrics with human judgements in §4.1 and the evaluation of `ArgSum` systems in §4.2.

**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.

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 7 and Table 8 in the appendix, respectively. The data preprocessing is described in Appendix §A.3.

**Representative LLM** As representative LLM, we utilize GPT-4o (`gpt-4o-2024-08-06`) for ArgSum, as well as GPT-4o-mini (`gpt-4o-mini-2024-07-18`) for evaluation, both provided by OpenAI's API service.[1] GPT-4o offers more qualitative responses for a suitable generation of argument summaries, while GPT-4o-mini was chosen for evaluation as it offers fast response times and is a cost-effective model version for the more quantity-based evaluation approach. We integrated GPT-4o into the `ArgSum` systems as described in §3.2 and §3.3. GPT-4o-mini was integrated into the automatic evaluation metrics as discussed in §3.4. Nevertheless, we continue to use the general term "LLM" to indicate that GPT-4o can be replaced with any other generative LLM.

[1] `https://platform.openai.com/docs/models/gpt-4o-mini`

### 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 the dimension of **coverage** as annotated by humans.

We consider two ways of computing correlations. 1) We calculate correlations **across** all topics and stances simultaneously. 2) 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.

**SoftScore** We then run **SoftScore**, explained in §3.4, with differently proposed automatic metrics as similarity functions $f$. In our case, we use: (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 3 in the appendix shows the results. First, we note that sP does not intuitively correspond to 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 BARTScore, suggested by Li et al. (2023), achieve the poorest results among all considered similarity functions.[2]

**Coverage Score** For **CS**, we also examine the correlation with the averaged human coverage scores. For the Match Scorers inside of CS, we consider those 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. 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).

As depicted in Table 1, 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.

Although the CS thus provides considerably stronger correlations for the criterion of coverage compared to the Soft-Score, it likely overfits the structure of ArgKP21 on which our conducted human evaluation is based.

**LLM-based metric** 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 a maximum correlation of 0.852 across and 0.824 within topics and stances, again performing considerably better than the Soft-Score and CS (see subsection B.1, Table 6). Thus, we exclusively use LLM-based scores in the remainder.

### 4.2 System evaluation

In this section, we evaluate the argument separation capabilities of clustering-based ArgSum systems in §4.2.1 and then evaluate the best ArgSum systems overall in §4.2.2.

#### 4.2.1 Argument Separaration Capability

We first evaluate the capability of **clustering-based** ArgSum systems (USKPM, MCArgSum and two MCArgSum versions with the Match Scorers of BarH and SMatchToPr) to separate arguments. To determine the argument separation capability of ArgSum systems, we use the ARI (Warrens and

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

| 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 1: Pearson correlation coefficient between the CS (incl. different Match Scorers) and averaged human coverage scores for different matching thresholds on ArgKP21.

van der Hoef, 2022), which measures the agreement between two data clusterings while accounting for the possibility of random clustering. The ARI ranges from -1 to 1, where lower values indicate poor separation, higher values indicate good separation, and values near zero suggest random clustering. Since both processed test datasets consist exclusively of arguments with exactly one matching summary, we treat the sets of arguments associated with each unique argument summary as reference clusters within topics and stances. The argument clusterings provided by the considered ArgSum systems substantially depend on the hyperparameters used. Therefore, we conduct several clustering runs with different parameter settings and select the best runs within topics and stances for each ArgSum system. The parameter settings taken into account are listed in Table 2 in the appendix.

**Results**: Overall, MCArgSum performs roughly 0.2-0.4 ARI points better than USKPM on ArgKP21 and 0.04-0.08 ARI points better on Debate (except with SmatchToPr), demonstrating that our LLM-based clustering approach can separate arguments substantially better than existing non-LLM-based argument clustering approaches. Detailed results are given in Table 4 and Table 5 as well as Figure 3 and Figure 4 in the appendix.

### 4.2.2 Argument Summarization Capability

Having identified the LLM-based evaluation metrics as the most reliable among those considered for both criteria 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. 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 $\alpha$ 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 and select 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. The results of evaluating the ArgSum systems with regard to their generated arguments summaries for ArgKP21 are depicted in Figure 2 (Top). For both classification-based systems, the integration of an LLM leads to improved performance. The variant of BarH without LLM integration has a slightly lower score compared to the LLM-based summary generation results. The variant of BarH with LLM-based argument candidate generation performs best

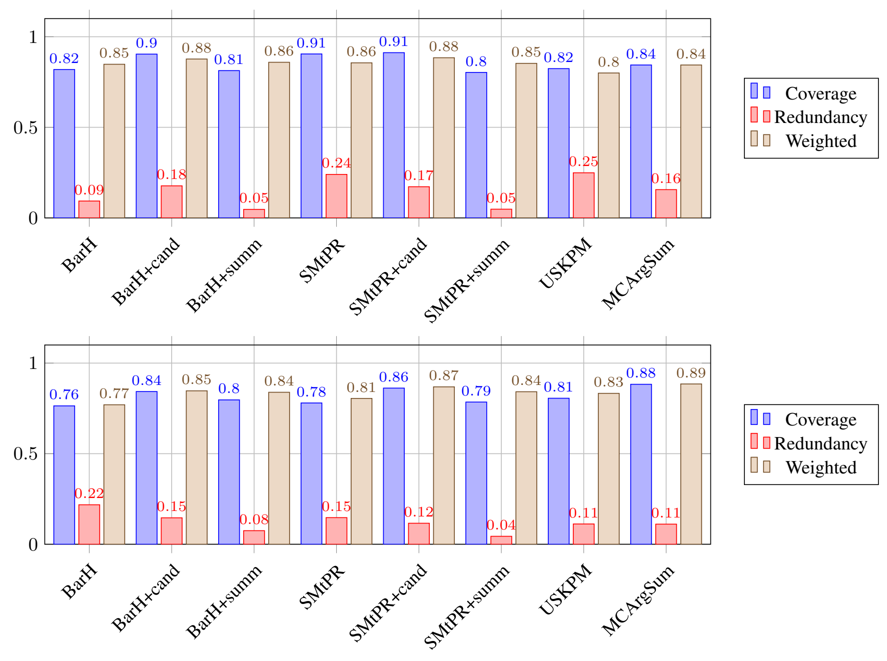

Figure 2: Top: Coverage, Redundancy, and Weighted metrics for the ArgKP21 dataset. Bottom: Coverage, Redundancy, and Weighted metrics for the Debate dataset. `SMatchToPr` is abbreviated as `SMtPR`, `+cand` and `+summ` are the variants with LLM candidates and summaries, respectively.

and achieves a weighted score of 0.88. Considering SMatchToPr, the variant with LLM-based argument candidate generation (0.88) also outperforms those with LLM-based summary generation (0.85) and without LLM integration (0.86). It is noteworthy that SMatchToPr without the integration of an LLM achieves the highest coverage score (0.91) among all `ArgSum` systems, but also comes with high redundancy (0.24). With regard to the clustering-based `ArgSum` systems, MCArgSum performs best and achieves the highest weighted score of 0.84, outperforming USKPM (0.80).

Considering the results for Debate, shown in Figure 2 (Bottom), the integration of LLM comes with advantages in all cases. As with `ArgKP21`, the variant integrating LLM for argument candidate generation generally performs best. However, the clustering-based `MCArgSum` provides even better results: It achieves a weighted score of 0.89, mostly as a result of its high coverage score (0.88).

The integration of LLMs results in considerable improvements for classifciation-based as well as clustering-based `ArgSum` systems. It is notable that `LLM`-based argument candiate generation in classification-based systems performs best for both datasets. The final choice of an ArgSum system should also depend on the runtime requirements. Here, 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 A.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 `ArgSum` system, achieves highest performance on the Debate dataset and has a runtime advantage against all other systems considered. 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: we did not consider open-source LLMs like Llama 3.X or Deepseek R1. While we do not expect substantial improvements to GPT-4o, it might be interesting to better understand the influence of the LLM on the prompting strategies relevant for the `ArgSum` systems and (even more) evaluation. Furthermore, we leave the application of reference-free evaluation strategies to future work.

## Limitations

Both model's (GPT-4o (`gpt-4o-2024-08-06`), GPT-4o-mini (`gpt-4o-mini-2024-07-18`)) training data includes information up to October 2023. ArgKP21, published in November 2021 (Friedman et al., 2021) and Debate, which dates back to 2014 (Hasan and Ng, 2014) could have been used in 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.

## Ethical Considerations

ArgSum systems could yield unreliable, factually incorrect or even maliciously misleading summaries of the underlying source arguments. Thus, recipients of the summarized arguments must interpret these with care.

## 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, Yoav Kantor, Lilach Eden, Roni Friedman, Dan Lahav, and Noam Slonim. 2020. 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).

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?

Georgios Petasis and Vangelis Karkaletsis. 2016. Identifying argument components through TextRank. In *Proceedings of the Third Workshop on Argument Mining (ArgMining2016)*, pages 94–102, Berlin, Germany. Association for Computational Linguistics.

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.

Matthijs Warrens and Hanneke van der Hoef. 2022. Understanding the adjusted rand index and other partition comparison indices based on counting object pairs. *Journal of Classification*, 39.

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.

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 Hyperparameters

### A.1 Fine-tuning

We fine-tuned the Match Scorers and Quality Scorers in BarH and SMatchToPr according to Bar-Haim et al. (2020) and Alshomary et al. (2021), respectively. It is important to note that Bar-Haim et al. (2020) 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. 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.

### A.2 Investigations

When applying BarH and SMatchToPr, we used the recommended parameter values from Bar-Haim et al. (2020) 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 2 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.

### A.3 Preprocessing

To conduct our investigations on the test split of ArgKP21 as well as Debate, we performed two pre-processing steps. First, we remov 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. Beyond that, including arguments with multiple references is not suitable for the evaluation of the argument separation capability addressed in §4.2.1. 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.

### A.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. [3] This setup was used for both training and inference of PyTorch models.

### A.5 Modifications to ArgSum Systems

We had to apply three modifications to the ArgSum systems as proposed in §4.2.2. 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.

[3] `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 2: Hyperparameter settings of clustering-based ArgSum systems considered in our investigations.

Table 2 includes the hyperparameter settings considered for the clustering runs in our investigations described in §4.2.1. For USKPM, we performed the clustering for each possible combination of the depicted parameter values. The minimum cluster size was constantly set to 3 for each system. In the case of USKPM, we set the similarity threshold for IC to zero, meaning that we forced each unclustered argument to be assigned to an existing cluster in order to avoid the presence of very small argument clusters.

## B Additional Results

### B.1 LLM-based Metric

We list the correlation between LLM-based coverage and redundancy scores and the respective averaged human scores for the used set of temperatures in Table Table 6.

## C Introduction to the Task of 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:

| 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 3: Pearson correlation coefficient between the Soft-Score (incl. different similarity functions) and averaged human coverage scores, along with the evaluation runtime. For the scenario within topics and stances, standard deviations are indicated below the correlation values. The three strongest positive correlations for both scenarios across sP, sR and sF1 are underlined in green, blue and orange.

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

| ArgSum System | **Scenario** | **ARI** | **Proportion Clustered** | **Runtime** |
|---|---|---|---|---|
| **USKPM** | Excl. Noise | 0.273 ±0.045 | 0.884 ±0.079 | 1.417 ±0.136 |
| | Incl. Noise | 0.253 ±0.057 | 1.000 ±0.000 | 1.478 ±0.113 |
| | IC | 0.268 ±0.052 | 1.000 ±0.000 | 4.377 ±4.378 |
| **MCArgSum** | Excl. Noise | 0.594 ±0.299 | 0.908 ±0.061 | 1.133 ±0.178 |
| | Incl. Noise | 0.547 ±0.255 | 1.000 ±0.000 | 1.133 ±0.178 |
| **MCArgSum** (BarH) | Excl. Noise | 0.725 ±0.182 | 0.858 ±0.064 | 53.86 ±25.19 |
| | Incl. Noise | 0.640 ±0.158 | 1.000 ±0.000 | 53.86 ±25.19 |
| **MCArgSum** (SMatchToPr) | Excl. Noise | 0.682 ±0.109 | 0.881 ±0.042 | 2.723 ±0.435 |
| | Incl. Noise | 0.639 ±0.113 | 1.000 ±0.000 | 2.723 ±0.435 |

Table 4: Argument separation capability of clustering-based ArgSum systems for ArgKP21. While ARI refers to the highest ARI from several examined parameter settings, the proportion of clustered arguments and the clustering runtime refer to this highest ARI. The scenarios are detailed in §4.2.1. All depicted values are averaged across topics and stances. Standard deviations are indicated behind each value.

| ArgSum System | **Scenario** | **ARI** | **Proportion Clustered** | **Runtime** |
|---|---|---|---|---|
| **USKPM** | Excl. Noise | 0.298 ±0.054 | 0.797±0.051 | 2.090 ±0.518 |
| | Incl. Noise | 0.232 ±0.049 | 1.000 ±0.000 | 2.044 ±0.507 |
| | IC | 0.262 ±0.054 | 1.000 ±0.000 | 15.38 ±9.633 |
| **MCArgSum** | Excl. Noise | 0.369 ±0.135 | 0.878 ±0.095 | 1.931 ±0.690 |
| | Incl. Noise | 0.315 ±0.123 | 1.000 ±0.000 | 1.931 ±0.690 |
| **MCArgSum** (BarH) | Excl. Noise | 0.411 ±0.085 | 0.789 ±0.079 | 201.9 ±166.7 |
| | Incl. Noise | 0.311 ±0.099 | 1.000 ±0.000 | 201.9 ±166.7 |
| **MCArgSum** (SMatchToPr) | Excl. Noise | 0.305 ±0.071 | 0.783 ±0.068 | 6.109 ±2.323 |
| | Incl. Noise | 0.245 ±0.045 | 1.000 ±0.000 | 6.109 ±2.323 |

Table 5: Argument separation capability of clustering-based ArgSum systems for Debate. While ARI refers to the highest ARI from several examined parameter settings, the proportion of clustered arguments and the clustering runtime refer to this highest ARI. All depicted values are averaged across topics and stances. Standard deviations are indicated behind each value.

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.)

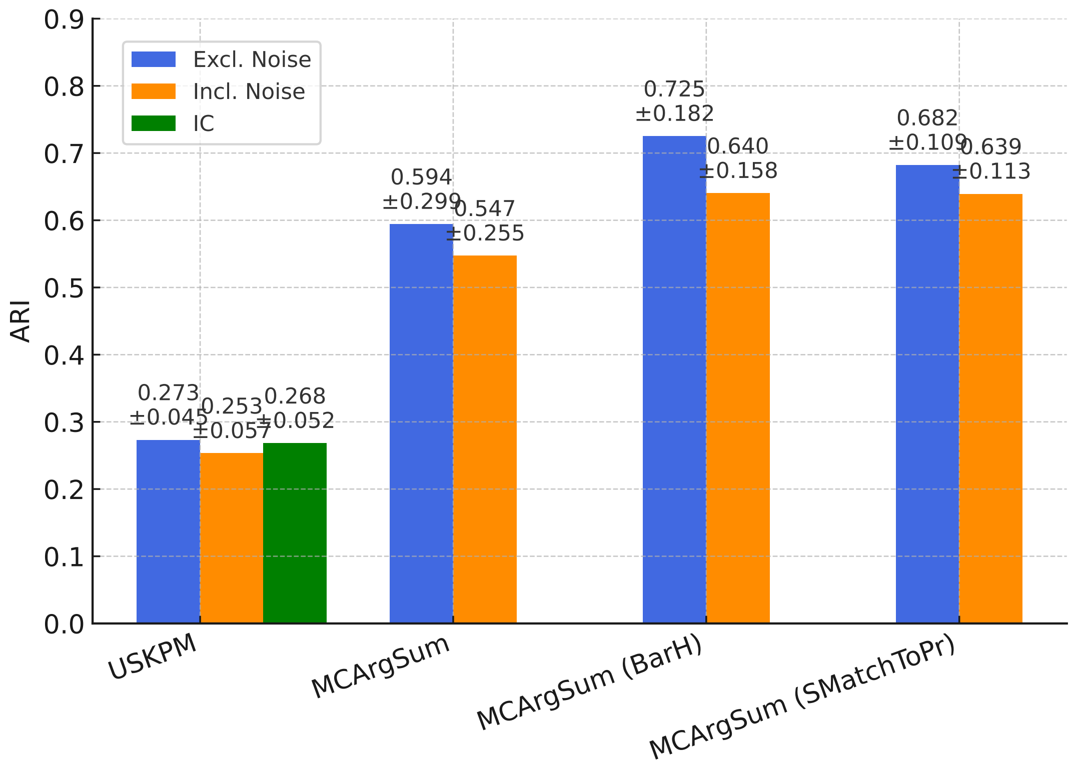

Figure 3: Highest ARI of clustering-based ArgSum systems from several examined parameter settings for ArgKP21. All depicted values are averaged across topics and stances. Standard deviations are indicated below the ARI values.

| Temper-ature | 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 6: Pearson correlation coefficient 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 below the correlation values. The strongest positive correlations for both scenarios within each LLM-based score are highlighted in bold.

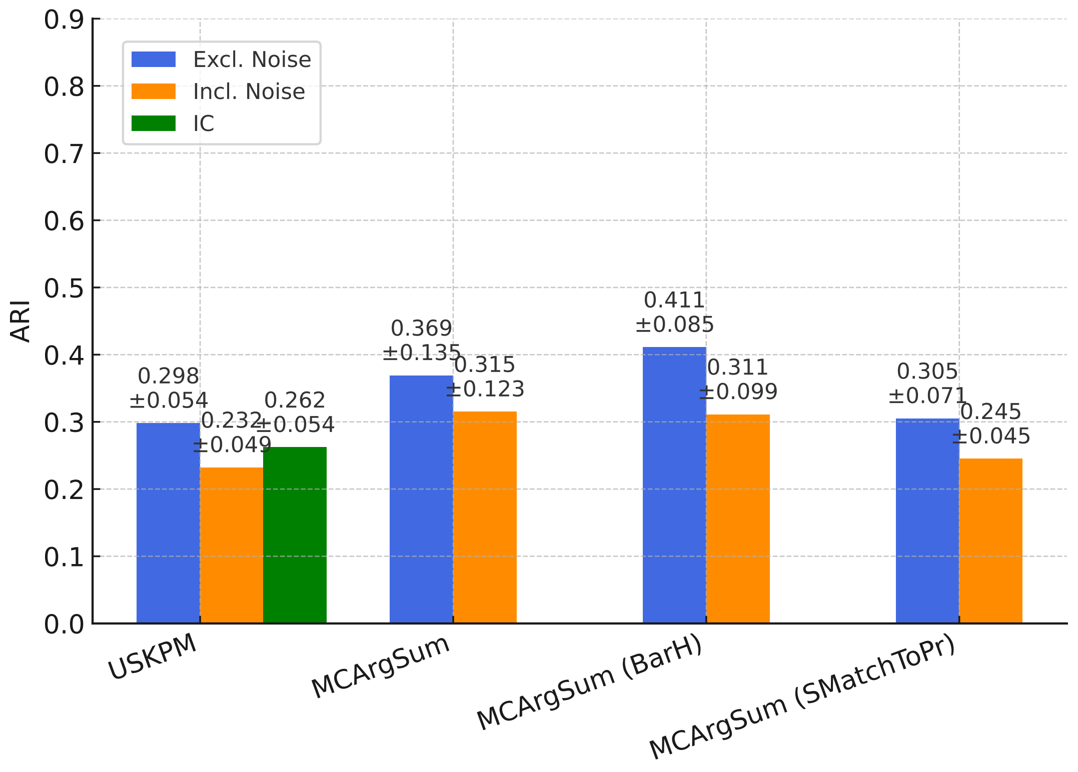

Figure 4: Highest ARI of clustering-based ArgSum systems from several examined parameter settings for Debate. All depicted values are averaged across topics and stances. Standard deviations are indicated below the ARI values.

| | |
|---|---|
| **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 7: Exemplary data point of ArgKP21.

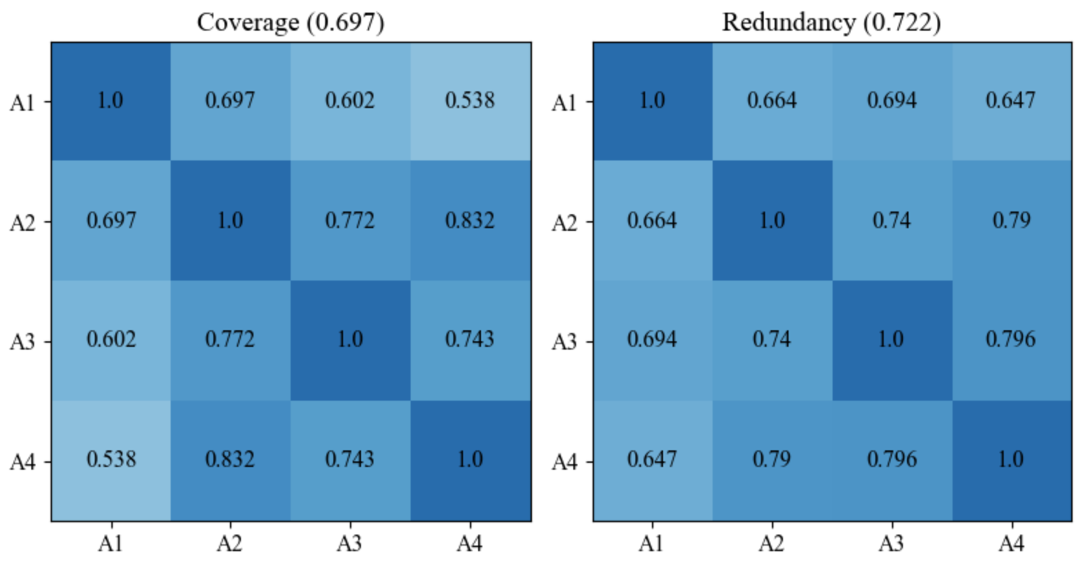

Figure 5: Pairwise Pearson correlation coefficient of the human judgments by the four annotators (A1-A4) for the criteria of coverage and redundancy. The averaged value across the unique annotator pairs is indicated in the parentheses.

| | |
|---|---|
| **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 8: Exemplary data point of Debate.