

LaMSUM: Creating Extractive Summaries of User Generated Content using LLMs

Garima Chhikara
Indian Institute of Technology Delhi
Delhi Technological University
New Delhi, India

Anurag Sharma
Indian Institute of Science Education
and Research Kolkata
Mohanpur, India

V. Gurucharan
Collaborative Dynamics
Texas, USA

Kripabandhu Ghosh
Indian Institute of Science Education
and Research Kolkata
Mohanpur, India

Abhijnan Chakraborty
Indian Institute of Technology
Kharagpur
Kharagpur, India

Abstract

Large Language Models (LLMs) have demonstrated impressive performance across a wide range of NLP tasks, including summarization. LLMs inherently produce abstractive summaries by paraphrasing the original text, while the generation of extractive summaries – selecting specific subsets from the original text – remains largely unexplored. LLMs have a limited context window size, restricting the amount of data that can be processed at once. We tackle this challenge by introducing LaMSUM, a novel multi-level framework designed to generate extractive summaries from *large* collections of user-generated text using LLMs. LaMSUM integrates summarization with different voting methods to achieve robust summaries. Extensive evaluation using four popular LLMs (Llama 3, Mixtral, Gemini, GPT-4o) demonstrates that LaMSUM outperforms state-of-the-art extractive summarization methods. Overall, this work represents one of the first attempts to achieve extractive summarization by leveraging the power of LLMs, and is likely to spark further interest within the research community.

1 Introduction

“Brevity is the soul of wit.”

– William Shakespeare, Hamlet, Act 2, Scene 2

Social media platforms like Facebook, X (formerly Twitter), and Reddit offer a medium for individuals to express their opinions and views on various subjects, leading to a diverse array of perspectives shared through social debates, critiques, and reviews [5, 11, 13, 75, 78]. With a vast amount of data being generated by users online, there is a growing need for summarization algorithms providing a precise and concise summary, eliminating the need for users to sift through numerous posts or reviews. Based on the resulting summary, summarization algorithms can be categorized as ‘extractive’ and ‘abstractive’. In extractive summarization, the aim is to select a subset representative of the original text [41, 66, 71, 72, 80]. In contrast, abstractive summarization aims to generate natural language summaries that capture the essence of the original text, often paraphrasing the content rather than directly extracting it [39, 76].

Extractive summarization is a crucial task with a rich body of literature, widely applied in summarizing legal cases, news articles, lectures, clinical notes, and social media content [3, 29, 35, 47]. For example, a Google search for a topic or hashtag displays a few tweets alongside the usual lists of websites and news articles.

The selection of tweets displayed in search results is similar to an extractive summarization task, where only a subset of tweets is selected [24]. Additionally, the Library of Congress only stores a selection of tweets as part of its archive to optimize storage space [12]. E-commerce platforms like Amazon display only a subset of reviews in the condensed view, such that the chosen reviews are representative of all the reviews submitted by the users. Such cases are instances of extractive summarization, where the target is to select the most relevant subset that can effectively summarize a topic or a product. When summarizing user generated data, it is crucial to quote the user’s exact words, making extractive summarization particularly valuable in such cases. In the process of summarization, each post or review is treated as a basic unit – often referred to as a *textual unit*.

In recent years, Large Language Models (LLMs) have exhibited high performance across various tasks, including summarization [7, 32, 51, 60]. Summaries generated by LLMs showcase high coherence and are overwhelmingly preferred by the human evaluators [42, 52]. However, using LLMs for extractive summarization has two significant limitations: (i) as generative models, LLMs naturally tend to perform abstractive summarization by paraphrasing rather than selecting the most relevant sentences; and (ii) due to the finite size of the context window, LLMs cannot handle long texts in a single input, underscoring the need for a method that allows for processing long text. Given the increasing reliance on LLMs for summarization, it is worth investigating the utility of LLMs for extractive summarization of large volumes of user-generated text.

To address these limitations, we present a novel framework LaMSUM (**L**arge Language **M**odel based **E**xtractive **S**UMmarization) that integrates LLM-generated summaries with voting algorithms borrowed from Social Choice Theory [4]. Voting algorithms are traditionally employed in decision making processes to aggregate preferences and opinions of a large population; the prime example being the electoral system, where voters’ preferences for candidates are gathered to elect a *winner* that best represents the voters’ preferences. Voting systems can be categorized into *single-winner* and *multi-winner* types, depending on the number of winners. In a single-winner system, only one candidate is elected, whereas a multi-winner system allows for the selection of multiple candidates.

Extractive summarization can be viewed as a multi-winner election, where the input units act as candidates, and the units chosen for the summary are considered as the winners. A multi-winner

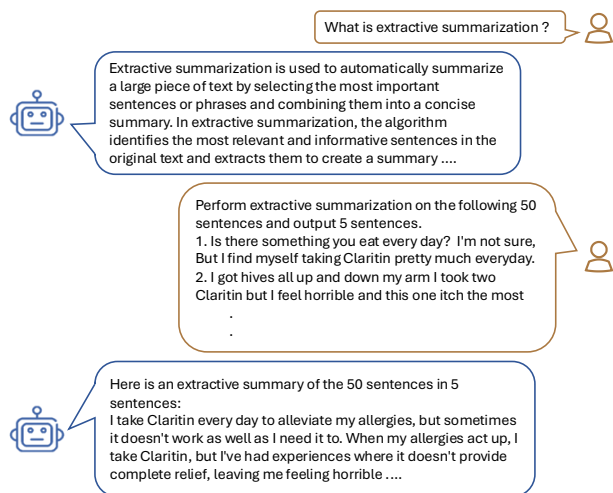


Figure 1: Current LLMs, by default, produce abstractive summaries. Llama3-70b-8192, despite specifically prompted for extractive summarization, generates abstractive summaries. This behavior underscores the need for a targeted approach to enable LLMs to effectively generate extractive summaries.

voting system includes approval-based committee (ABC) voting, where voters approve a subset of candidates without ranking, and ranked choice voting, where voters rank candidates by preference. In LaMSUM, we utilize two approval based voting algorithms – Plurality Voting [46] and Proportional Voting [37], and one ranked based voting algorithm – Borda Count [19]. Our judicial application of voting algorithms with a multi-level summarization framework ensures that LaMSUM outperforms the state-of-the-art fine-tuned summarization models. In summary, in this work, we make the following contributions:

- We propose a novel framework LaMSUM which can effectively summarize large (having >30K tokens) collection of user generated content.
- LaMSUM considers a multi-level summarization model that utilizes voting algorithms to combine outputs to generate robust summaries.
- Analyse whether an ensemble model with multiple LLMs in LaMSUM can lead to improved outcomes across different voting methods.

To our knowledge, this is the first work to implement extractive summarization of large user-generated texts using LLMs by combining summarization with voting algorithms. We believe this work can spawn further research in this direction.

2 Background and Related Work

In this section, we review the relevant prior works that provide the foundation for our current research.

Text Summarization Algorithms

With the growing volume of online data, the demand for algorithms that automatically shorten and summarize texts is increasing. Automatic summarization can be approached in two ways: extractive and abstractive. In extractive summarization, a subset of input collection is selected for the summary based on their perceived quality

and significance, aiming to represent a larger dataset with a concise sample. Abstractive summarization, on the other hand, generates natural language summaries that capture the most critical information from the original text. Both approaches seek to provide readers with a concise overview of the textual content. Over the years, many text summarization algorithms have been proposed in the literature; the reader can refer to [18, 25] for detailed surveys.

Extractive Summarization of User Generated Content

A large body of prior research has focused on summarizing long documents [1, 8, 16]. In fact, traditionally, summarization tasks have focused on summarizing a single document, such as a news article or a business report. In recent years, summarization has been increasingly applied on different types of *user generated text* (e.g., tweets, Facebook or Reddit posts) [28, 31, 44], where the task is to summarize short, independent posts written by many users. Several extractive summarization algorithms tailored specifically for user generated content have also been proposed [14, 33, 48, 55, 57, 67].

Large Language Models (LLMs) for Summarization

LLMs are now being extensively used for summarization [7, 32, 60]. Multiple works have proposed few-shot learning frameworks for the abstractive summarization of news, documents, webpages, and generic texts [5, 38, 61, 69, 73], but their primary focus remains on short documents that can fit in the LLM context window. Researchers have also observed that human evaluators are increasingly preferring LLM-generated summaries compared to other baselines [23, 42, 65, 74, 77]. Despite the advancements, recent studies have also uncovered factual inaccuracies and inconsistencies in LLM-generated summaries [36, 43, 58, 59].

Extractive Summarization through LLMs: The Current State

By default, LLMs produce abstractive summaries, meaning that the summary text is distinct from the input text, even when it is instructed to do otherwise. To illustrate this, we present a small example in Figure 1. An LLM, when prompted, could clearly explain extractive summarization, yet, when we instructed it to perform extractive summarization on a set of 100 sentences, it fails to do so and instead generated an abstractive summary. Prior to our current work, only two studies attempted to perform similar tasks. Zhang et al. [73] attempted summarization of short news articles using GPT 3.5, while Chang et al. [9] attempted abstractive summarization for book-length documents. However, both these approaches suffer from practical limitations such as lack of contextual dependencies in user generated text and the problem with positional bias in LLMs.

To the best of our knowledge, ours is the first attempt to perform extractive summarization on a large collection of user generated texts through LLMs, while tackling the challenge of positional bias. We describe our proposal in detail in the next section.

3 Generating Extractive Summaries through LLMs

In this section, we define the problem statement formally and introduce our novel summarization framework LaMSUM (**L**arge Language **M**odel based **E**xtractive **S**UMmarization) that leverages LLMs to summarize large user-generated text.

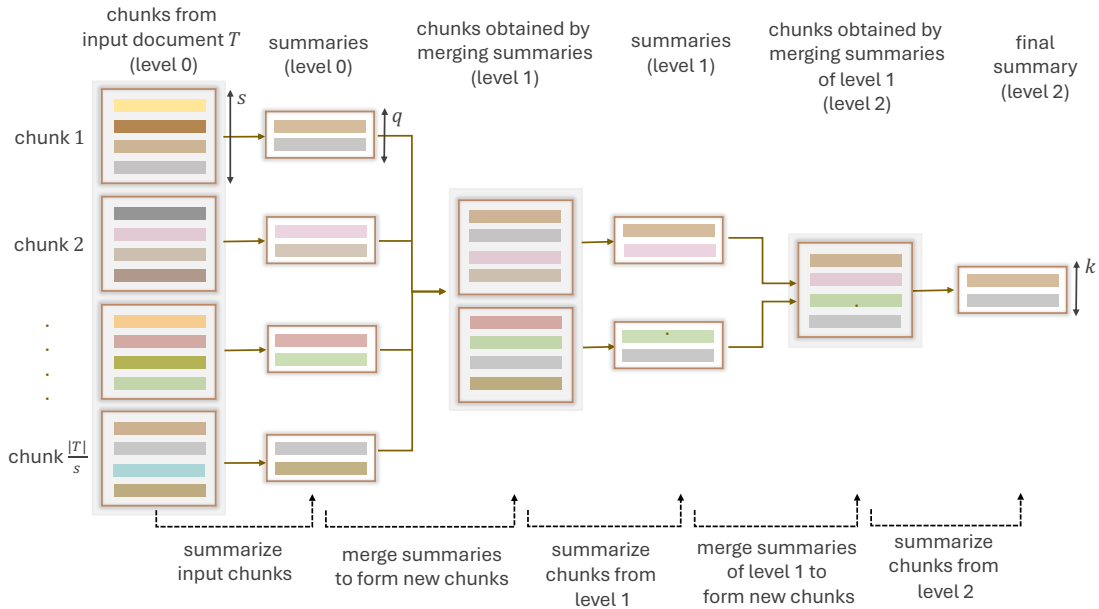


Figure 2: LaMSUM: Multi-level framework for extractive summarization of large user-generated text. Input set \mathcal{T} (level 0) is divided into $\lceil \frac{|\mathcal{T}|}{s} \rceil$ chunks each of size s . From each chunk a summary is produced of size q (refer Figure 3), q length summaries from $\lceil \frac{|\mathcal{T}|}{s} \rceil$ chunks are merged to form the input for the next level i.e., level 1. Iteratively the same procedure is repeated till we obtain a summary of size k . We set $q = k$ to ensure our algorithm can effectively handle the worst-case scenario where all the textual units in the final summary may come from the same input chunk (Section 3.2).

3.1 Task Formulation

Let $\mathcal{T} = \{t_1, t_2, \dots, t_N\}$ represent a collection of textual units, where each unit can be a tweet, a post, or a review. Our summarization algorithm takes \mathcal{T} and an integer k as input, where \mathcal{T} denotes the entire set of textual units and k specifies the desired number of units in the summary. Task is to output a summary $\mathcal{S} \subseteq \mathcal{T}$ such that $|\mathcal{S}| = k$. The summary \mathcal{S} would be evaluated based on its alignment with the preferences of gold standard summarizers. If the context window size of an LLM is W , we assume \mathcal{T} is too large to fit in a single context window.

3.2 Multi-Level Summarization

LLMs have a limited context window, making it impossible to input large text collections all at once. While recent models like GPT-4 support context windows of up to 128k tokens, they still cannot accommodate book-length inputs within a single window. Consequently, the input must be divided into smaller chunks to perform the desired task [9]. Thus, LaMSUM employs a multi-level framework for extractive summarization, enabling it to consider input data of any size (detailed in Figure 2)¹.

The set \mathcal{T} , which contains the original textual units, is provided as input at level 0 and is divided into $\lceil \frac{|\mathcal{T}|}{s} \rceil$ number of chunks of size s . From each chunk of size s , we generate a summary (discussed in Section 3.3) of size q (where $q < s$), and repeat this process for

all $\lceil \frac{|\mathcal{T}|}{s} \rceil$ chunks.² We then merge all these q length summaries obtained from level 0 to form an input for the next level i.e., level 1. We repeatedly perform this process until we obtain the final summary of length k . Note that the last chunk may be less than q in size, in such case we move all the textual units of the respective chunk to the next level (refer Algorithm 1).

An alternate strategy would be to divide the input \mathcal{T} into $\lceil \frac{|\mathcal{T}|}{s} \rceil$ chunks each of size s and from each chunk select $\frac{k \cdot s}{|\mathcal{T}|}$ sentences to be included in the summary. However, this approach assumes a uniform distribution of potential candidates across chunks that can be included in the final summary. In LaMSUM, we keep $q = k$ i.e., we extract k textual units from each chunk eliminating the chance of missing any potential candidate (discussed in Section 4.4). In the worst-case scenario, all k units in the final summary can come from a single chunk, and our algorithm can handle such cases effectively.

It is important to note that we are dealing with user-generated content, such as tweets, which lack contextual connections. Unlike book summarization, where chapters are interconnected and the context of previous chapters is crucial for summarizing the current one, tweets are generally standalone and contextually independent. Thus, our approach of independently deriving summaries from each chunk works well in our setup, as each textual unit operates independently of the others and there are no long-range dependencies.

²Note that a chunk of size s refers to a chunk containing s textual units. Likewise, a summary of size q indicates a summary of q textual units. $|\mathcal{T}|$ denotes the number of textual units present in \mathcal{T} .

¹Code is available at <https://anonymous.4open.science/r/LaMSUM/>

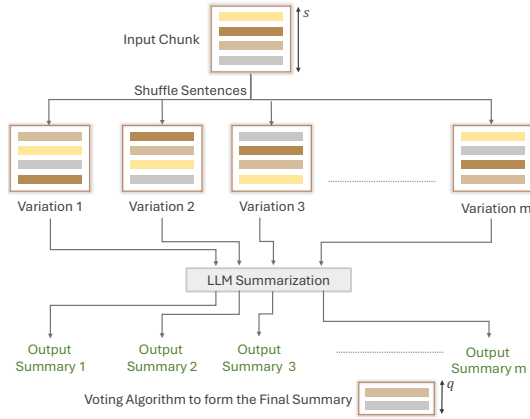


Figure 3: Textual units (e.g., posts) in the input chunk are shuffled to account for the positional bias. m different chunk variations are obtained through shuffling, which are subsequently summarized using LLMs. m summaries are then aggregated by voting algorithms to get the final summary.

3.3 Summarizing a Chunk

Next, we discuss how LaMSUM summarizes a chunk (Algorithm 2) by tackling the positional bias in LLMs and leveraging voting algorithms drawn from Social Choice Theory [4].

3.3.1 Tackling Positional Bias. Prior research [6, 34, 65, 73] has highlighted that summarization using LLM is prone to positional bias, i.e., the sentences located in certain positions, such as the beginning of articles, are more likely to be considered in the summary. To address this issue and generate a robust summary, we create m different variations by shuffling the textual units within the input chunk. This ensures that each unit has the opportunity to appear in different positions within the input text (refer Figure 3).

3.3.2 Zero-shot prompting. For each input chunk, we obtain m different summaries (one for each variation) by prompting the LLM. We employ the following two prompts to obtain the summaries –

(a) Select the most suitable units that summarize the input text.
Prompt: "Input consists of <chunk_size> sentences. Each sentence is present in a new line. Each sentence contains a sentence number followed by text. You are an assistant that selects best <summary_length> sentences (subset) which summarizes the input. Think step by step and follow the instructions. <sentences>"

(b) Generate a ranked list in descending order of preference.
Prompt: "Input consists of <chunk_size> sentences. Each sentence is present in a new line. Each sentence contains a sentence number followed by text. You are an assistant that outputs the sentences in the decreasing order of their relevance to be included in the summary. Think step by step and follow the instructions. <sentences> Remember that output should contain all the sentences in the decreasing order of their preference."

3.3.3 Output Calibration. LLMs may alter certain words from the input text while generating extractive summaries, as shown in Table 1. Thus, we perform additional checks to ensure that the

Algorithm 1 Algorithm for multi-level summarization

```

Input:  $\mathcal{T}, k, s, q, m$ 
 $S = \{\}$  ▷  $S$  stores the final summary
while  $|\mathcal{S}| < k$  do ▷ until  $k$  length summary is obtained
   $n_{chunks} = \lceil \frac{|\mathcal{T}|}{s} \rceil$  ▷ number of chunks in set  $\mathcal{T}$ 
   $L = \{\}$  ▷  $L$  stores the results of a given level
  for  $i \leftarrow 1$  to  $n_{chunks}$  do
     $si = (i - 1) * s$  ▷ starting index of chunk
     $ei = i * s$  ▷ ending index of the chunk
    if  $i = n_{chunks}$  then ▷ if last chunk
       $ei = |\mathcal{T}|$  ▷ ending index is equal to length of  $\mathcal{T}$ 
    end if
     $width = ei - si$  ▷ number of textual units in a chunk
    if  $width \leq q$  then ▷ if last chunk
       $L = L \cup t_{si} \cup t_{si+1} \cup \dots \cup t_{ei-1}$  ▷ add all textual units to the result
    else
       $L = L \cup \text{CHUNKRESULT}(\mathcal{T}, si, ei, q, m)$  ▷ add summary of each chunk to result  $L$ 
    end if
  end for
   $\mathcal{T} = L$  ▷ update the input  $\mathcal{T}$  for the next level
   $S = S \cup L$ 
end while
Output:  $S$ 

```

Algorithm 2 Algorithm for summarization of a chunk

```

function  $\text{CHUNKRESULT}(\mathcal{T}, si, ei, q, m)$ 
   $X = \{\}$ 
  for  $i \leftarrow 1$  to  $m$  do ▷ for each variation of a chunk
     $V = \text{SHUFFLE}(\mathcal{T}, si, ei, i)$  ▷ shuffle with random state  $i$ 
     $R = \text{LLM}(V, q)$  ▷ obtain summary from the LLM
     $C = \text{CHECK}(R, \mathcal{T}, si, ei)$  ▷ output calibration
     $X.add(C)$ 
  end for
  return  $\text{VOTING}(X, q)$  ▷ voting to obtain the final summary
end function

```

textual units selected in the summary are indeed a subset of \mathcal{T} . If the post selected by the LLM (say x) is not present in the original text \mathcal{T} , we identify the post with the closest resemblance to x by computing the edit distance [54]. LLMs may also hallucinate, generating new sentences rather than selecting units from the input. In such instances, the edit distance between the generated unit x and all the original textual units would be high. We discard these sentences from the output.

3.4 Reimagining Summarization as an Election

As mentioned earlier, for a given chunk, we obtain m summaries – one for each variation. We imagine the process of creating the final summary from these m summaries to be a multi-winner election, where the textual units in m summaries correspond to ballots (candidates) and the role of the voting algorithm is to pick q winners.

Original Post	LLM Modified Output
My allergies be acting up & Claritin be letting them ><	My allergies be acting up & Claritin be letting them.
I spent the extra money to have fast relief, but instead I am SUFFERING #gimmemy moneyback	I spent the extra money to have fast relief, but instead I am SUFFER

Table 1: Examples illustrating that LLMs when selecting textual units for summarization, often demonstrate a propensity to alter certain words or introduce new ones.

We employ three different voting methods, namely **Plurality Voting** [46], **Proportional Approval Voting (PAV)** [37] and **Ranked Choice Voting** [19] to determine the final summary. Due to the varying input requirements of different voting methods, changing both the prompting approach and the output generated by the LLM becomes imperative.

Plurality voting and proportional voting are approval-based voting methods where voters can select multiple candidates they approve of without indicating a specific preference order. In multi-winner plurality voting (also known as **block voting**), each voter casts multiple votes and the candidates are selected based on the number of votes polled. *In the context of summarization, a textual unit is treated as a candidate, and the LLM acts as the voter.* We select the textual units in the decreasing order of the votes polled, till we obtain a summary of size q . PAV evaluates the *satisfaction* of each voter in the election outcome. A voter’s satisfaction is measured based on – amongst the number of candidates they voted for, how many are selected in the election. In the realm of summarization, PAV selects the textual units based on the amount of support each unit receives in m summaries. Since both plurality and proportional are approval-based voting algorithms, the units are either approved or disapproved by the underlying LLM, with no explicit ranking or preference order. In this case, we prompt the LLM to *select the best $<q>$ sentences that summarize the input text* as shown in Section 3.3.2 (a).

On the other hand, ranked choice voting entails assigning a score to each textual unit and subsequently selecting the highest-scoring units for inclusion in the summary. For ranked voting, we use the Borda count, a positional voting algorithm [19]. In the Borda method, each candidate is assigned points corresponding to the number of candidates ranked below them: the lowest-ranked candidate receives 0 points, the next lowest gets 1 point, and so forth. The candidates with the highest aggregate points are declared as the winners. In ranked voting, we prompt the LLM to *output sentences in descending order of their suitability for the summary* as discussed in Section 3.3.2 (b).

It is important to note that the prompting technique and the output generated by LLM vary for different voting methods. In approval voting the output from LLM is a list of q textual units that LLM finds best suited to be included in the summary. Whereas in ranked choice voting, the output from LLM is a list of the same length as input i.e. s with all the units sorted in decreasing order of their preference towards the summary, and Borda Count [19] is used to identify the top q textual units. In the next section, we highlight how the voting-based summarization schemes outperform the *Vanilla* setup, which does not use voting.

Parameters	Claritin	US-Election	MeToo
#TextualUnits ($ \mathcal{T} $)	3998	2107	483
#InputWords	53609	35522	16737
AV #CTU (s)	200	150	75
#CSTU (q)	100	100	50
RV #CTU (s)	40	40	40
#CSTU (q)	20	20	20
#SummaryTU (k)	100	100	50

Table 2: Input parameters used for the proposed framework LaMSUM. #TextualUnits is the number of textual units in the input set i.e. $|\mathcal{T}|$. #InputWords represents the number of words present in the input set \mathcal{T} . #CTU is the number of textual units in a chunk i.e. chunk size represented as s . #CSTU is the number of textual units in the chunk summary represented as q . #SummaryTU is the number of textual units present in the final summary i.e. k . AV and RV represents Approval Voting and Ranked Voting respectively.

4 Experimental Setup

4.1 Dataset

Our experiments are conducted on *three* publicly available datasets, consisting of crowd-sourced data from X, listed in Table 2 [13]. The *Claritin* dataset contains 3,998 tweets about the benefits and the side-effects of the anti-allergic drug Claritin. *US-Election* dataset contains 2,107 tweets from 2016 US Presidential Election where people support and attack different political parties. *Me-Too* dataset includes 483 tweets from the October 2018 MeToo movement, where individuals recount the harassment cases they experienced.³ During pre-processing, we remove all web links and duplicate entries from the datasets. The datasets also include gold-standard summaries (reference summaries) which are human-generated i.e., the textual units that are strong candidates for inclusion in the summary are selected by the humans. *Claritin* and *US-Election* dataset, each have three gold standard summaries, comprising 100 textual units each. *Me-Too* dataset has two gold-standard summaries with 50 textual units each.

4.2 Large Language Models (LLMs)

LLMs are characterized by their extensive parameter sizes and remarkable learning abilities [10, 79]. In our work, we utilize three open-source LLMs and one proprietary LLM to conduct experiments: llama3-70b-8192 from Meta [62], mixtral-8x7b-32768 from Mistral AI [30], gemini-1.0-pro from Google [22] and gpt-4o-mini-2024-07-18 from OpenAI [50]. Across all experiments, we keep temperature, top probability and output tokens as 0, 0.8 and 8192 respectively.

³Dataset: <https://github.com/ad93/FairSumm/tree/master/Dataset>

4.3 Evaluation Metric

For evaluating the quality of summaries generated by LaMSUM, we report ROUGE-1, ROUGE-2, and ROUGE-Lsum scores [40]. ROUGE-1, ROUGE-2 and ROUGE-L respectively evaluate the overlap of unigrams, bigrams and longest common subsequence between the generated summary and the reference summary. ROUGE-Lsum is more suitable for extractive summarization, as it applies ROUGE-L at sentence level and then aggregates all the results to obtain the final score.

4.4 LaMSUM Input Parameters

Input parameters for LaMSUM (Algorithm 1), such as $|\mathcal{T}|$ (total number of textual units in the set), s (chunk size) and k (length of summary) are listed in Table 2 for different voting algorithms and datasets. The value of m (number of shuffling) for all the datasets was set to 5.

If $q \in [k, s)$, our proposed method can handle worst case scenario where all the textual units in the final summary may originate from a single chunk of level 0. As q approaches s , more levels are required to converge to the final summary. The optimal value of q that can handle worst case and also reduce the number of levels in multi-level summarization is k , thus we keep $q = k$ for experiments with approval voting. For instance, if s is 200 and q is 100, this indicates that only 50% of the units from each chunk advance to the next level.

In ranked voting algorithm, we maintain smaller value for chunk size (s) to ensure that the LLMs output, which is of size s , fits within the context window. Additionally, as chunk size increases, LLM often does not output all the sentences, instead produce generalized statements like “*similarly for other sentences we find the rank*”. Therefore, it is essential to keep the chunk size smaller. For ranked voting, we set s and q to 40 and 20 respectively, upholding the selection ratio of 50% at each level.

5 Experimental Evaluation

In this section, we present the empirical comparison of LaMSUM with competent baseline models and voting algorithms across datasets (refer Table 3).

5.1 Baseline Comparison

We compare LaMSUM with the pre-neural models (ClusterRank [21], DSDR [26], LexRank [20], SummBasic [49]), transformer based models (GPT2 [53], BERT [45], XLNET [70]) and with fine-tuned BERTSUM [41] model. As presented in Table 3 it is observed that LaMSUM outperform the state-of-the-art summarization models. Earlier work [73] reported that the ChatGPT model achieves lower ROUGE scores on CNN/DM and XSum dataset. But our results demonstrate that our proposed framework LaMSUM performs significantly better than other fine-tuned models for large user-generated text.

Results from the LLM exhibit variability when executed multiple times over the same input. We conducted experiments with LLMs for five iterations for the same input. Table 3 contains the maximum ROUGE score obtained from these five iterations and Table 6 (in supplementary material) displays the variance in the ROUGE scores when executed for five times.

Algorithm 3 Algorithm for summarization of a chunk in Vanilla LLM

```

function CHUNKRESULT( $\mathcal{T}, si, ei, q, m$ )
   $R = \text{LLM}(\mathcal{T}, si, ei, q)$        $\triangleright q$  textual units from  $\mathcal{T} \in [si, ei]$ 
   $C = \text{CHECK}(R, \mathcal{T}, si, ei)$      $\triangleright$  output calibration
  return  $C$ 
end function

```

5.2 Vanilla LLM vs. LaMSUM

Our proposed framework, LaMSUM, ensures robust summary generation by shuffling and employing a voting algorithm to select the best textual units for the summary. It is crucial to compare LaMSUM with a multi-level LLM that does not use shuffling and voting, which we call *Vanilla LLM*. Algorithm 3 outlines the steps used by *vanilla LLM* to find the chunk summary. Table 3 demonstrates that the *vanilla* multi-level LLM has lower ROUGE scores for each LLM compared to the proposed framework LaMSUM, indicating that shuffling and voting enhances the performance.

5.3 Which Voting Algorithm Holds the Lead?

We experimented with three voting algorithms, two approval-based and one ranked-based. Experimental results indicate that LLMs with approval voting perform the best compared to the ranked voting algorithm. We hypothesized that rank-based voting would yield better results, as it makes more informed decisions about the potential sentences to be included in the summary. Contrary to our expectations, rank-based algorithms performed even worse than neural and transformer-based models. This can be attributed to multiple factors: (i) LLMs hallucinate and output sentences in the same or in the reverse order as they were in the input. (ii) Occasionally, LLMs do not output all the sentences from the input, resulting in the padding of left-out sentences towards the end of the list, which disturbs the ranking and potentially affects the result. To overcome these problems, we kept the chunk size low as discussed in Section 4.4, but the results still did not surpass those of the approval-based voting algorithm.

Takeaway: LLMs, when prompted to *select sentences that can summarize the input*, perform better than when tasked to *rank the sentences in the order of their preference towards the summary*.

5.4 What Fails to Deliver Results?

To ensure extractive summarization, we tested an additional approach – each sentence is tagged with a sentence number, LLM is prompted to *select the best q sentences and output only the sentence numbers of the best q sentences*. Thereafter, the sentences corresponding to the sentence numbers can be retrieved. For instance, if s is 200 and q is 100, the task is to output the sentence numbers of the best 100 sentences from a pool of 200 sentences. In such cases, LLMs hallucinate and provide an output consisting of either all the odd number sentences or all the even number sentences.

Takeaway: For extractive summarization, relying solely on indexes may result in hallucination, underscoring the importance of emitting the input content and not the numbers.

Models		Claritin			US-Election			MeToo		
		R1	R2	RLSum	R1	R2	RLSum	R1	R2	RLSum
ClusterRank		50.08	14.17	48.64	55.18	13.09	53.51	55.81	55.81	52.48
DSDR		28.01	8.40	27.45	48.04	9.53	46.44	55.03	17.30	52.04
LexRank		45.04	19.71	44.74	42.63	10.78	41.64	42.70	11.32	40.91
SummBasic		58.25	19.29	56.76	55.36	12.43	53.94	58.23	21.53	56.07
GPT2		61.61	23.58	60.74	55.86	15.07	54.81	40.63	11.24	38.86
BERT		57.30	22.37	56.21	55.89	15.44	55.00	45.72	10.76	43.50
XLNET		55.52	21.37	54.75	56.48	15.72	55.41	36.58	08.50	34.48
BERTSUM [41]		57.87	22.75	55.96	59.00	17.51	57.41	57.11	23.08	54.84
Vanilla	Llama3	58.07	21.86	56.73	56.12	12.93	54.71	51.84	17.84	49.47
	Mixtral	57.80	20.14	55.81	57.11	14.20	55.79	49.63	11.29	46.16
	Gemini	49.70	18.33	48.78	51.00	12.36	49.19	45.51	8.95	42.51
	GPT-4o mini	62.93	24.82	61.08	55.06	15.01	53.88	36.30	7.05	34.54
Plurality Voting	Llama3	61.28	23.79	59.54	60.11	18.26	58.99	55.51	18.79	52.85
	Mixtral	59.13	22.90	57.40	59.55	16.01	58.12	55.41	17.62	52.63
	Gemini	55.43	19.32	53.35	58.83	16.25	57.54	57.95	21.47	54.87
	GPT-4o mini	64.20	26.71	62.66	58.15	15.91	56.78	54.55	19.66	51.9
Proportional Voting	Llama3	61.31	23.36	59.73	58.90	15.99	57.55	58.14	14.99	55.46
	Mixtral	60.30	24.03	58.58	58.46	15.74	57.28	57.36	22.97	54.82
	Gemini	61.77	19.83	59.81	57.92	15.85	56.49	54.02	13.95	50.90
	GPT-4o mini	64.13	26.75	62.30	58.68	19.08	57.62	48.55	13.23	45.35
Borda Count	Llama3	51.87	15.93	49.61	51.95	13.49	50.60	53.35	16.99	50.38
	Mixtral	56.07	22.28	54.65	48.71	13.37	47.48	53.24	18.80	50.96
	Gemini	51.45	16.92	49.39	54.01	13.38	52.56	50.13	21.45	47.96
	GPT-4o mini	58.01	18.08	56.05	53.10	15.94	51.63	50.10	17.82	47.75

Table 3: Table showing metric scores from different models for various datasets. Here, R1 = ROUGE-1 Score, R2 = ROUGE-2 Score, RLSum = ROUGE-LSum Score. The best value per dataset is shown in bold and clearly Approval voting outperforms all the other methods across all the evaluation measures.

Ensemble	Voting Method	R1	R2	RLSum
All 4 models	Plurality	54.16	20.79	53.00
	Proportional	54.70	19.95	53.14
	Ranked	51.78	16.97	50.24
Best 3 models	Plurality	58.88	21.74	57.60
	Proportional	58.04	22.78	56.71
	Ranked	57.22	19.51	55.33
Weak 3 models	Plurality	50.64	18.82	49.31
	Proportional	52.92	19.23	51.07
	Ranked	47.50	16.74	46.37

Table 4: Results obtained through ensembling of LLMs in LaMSUM for Claritin dataset. We compare 3 different cases: i) ensemble of all 4 models – Llama, Mixtral, Gemini and GPT-4o mini ii). ensemble of best 3 models – Llama, Mixtral and GPT-4o mini iii). ensemble of weak 3 models – Llama, Mixtral and Gemini.

6 Ensembling of LLMs in LaMSUM

Different LLMs can be considered as distinct experts, each offering their vote or opinion about the textual unit to be included in the summary. In LaMSUM, each expert (LLM) operated independently, determining which sentences were worthy of inclusion. We hypothesize that using an ensemble of LLMs in LaMSUM, where these experts collaborate and reach a consensus, has potential to improve performance of the overall system [56, 63].

We apply ensembling at each chunk as shown in Figure 4. For a given chunk i containing s textual units, the goal is to produce a final summary of size q by gathering opinions of all the experts. Ensembling takes place in two stages. Stage 1 - For each chunk we

obtain q length summary through different LLMs by using shuffling and voting method as demonstrated in Section 3.3. Stage 2 - To ensemble the q length summaries from different LLMs, we again leverage the voting methods. In stage 2 we make use of the same voting method that was used in stage 1 to maintain consistency. The summary generated from the ensembling process is treated as the final summary for chunk i and is carried forward to the next level.

As shown in Table 3, for Claritin dataset, GPT-4o mini outperforms the other models, while Gemini delivers weaker results. We evaluate three different scenarios based on the number of LLMs used in the ensembling process: i) all four models – Llama, Mixtral, Gemini and GPT-4o mini, ii) the top three models – Llama, Mixtral and GPT-4o mini, iii) the three weaker models – Llama, Mixtral and Gemini. Table 4 shows the results of ensembling in LaMSUM for different scenarios and voting methods.

In the approval-based voting algorithm, using all four models or just the top three models for ensembling doesn't significantly improve performance; the resulting performance after ensembling remains close to that of the weakest model in each case. However, ensembling with the ranked-based voting algorithm yielded interesting results. When ranked voting is applied to individual LLMs, it delivers the worst performance compared to other voting algorithms, as shown in Table 3. When using ranked voting with all four models in the ensembled LaMSUM, the performance is close to that of the weakest model – Gemini (49.39) < Llama (49.61) < Ensemble (50.24) < Mixtral (54.65) < GPT-4o mini (56.05). However,

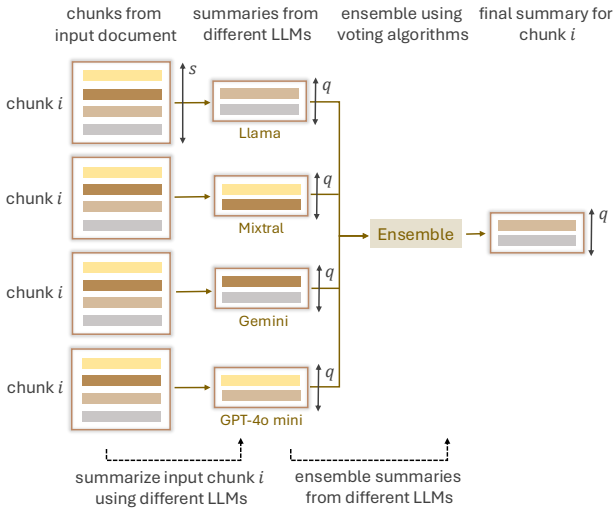


Figure 4: Ensembling applied at chunk using all the four models. q length summaries are obtained from each chunk through different LLMs using the shuffling method discussed in Section 3.3. All the summaries are ensembled through voting algorithms to achieve the final summary of length q . This final q length summary is thereafter forwarded to the next level.

when considering the ensembled LaMSUM with the top three models – Gemini (49.39) < Llama (49.61) < Mixtral (54.65) < Ensemble (55.33) < GPT-4o mini(56.05), ensemble outperforms Mixtral, and the ROUGE score increases from 50.24 to 55.33. While the ensembled LaMSUM with the top three models does not surpass GPT-4o mini, it does enhance the performance of the overall setup.

Takeaway: Ensemble of LLMs in LaMSUM yield good results with rank-based voting method, and can help improve the performance of the overall setup if we select models for ensembling that perform well individually in LaMSUM. Including all models in the ensemble could result in the weakest model becoming the bottleneck.

7 Does Fine-Tuning Help?

The performance of LLMs on specific downstream tasks, such as extractive summarization, can be suboptimal when not explicitly fine-tuned. LLMs tend to generate erroneous information while sounding persuasive and assured [2, 68]. In Section 2, we highlighted instances where LLMs exhibit nonplussed responses by generating abstractive summaries despite being prompted for extractive summarization. They also alter some words, so we perform output calibration in LaMSUM. To address these challenges, we explore the efficacy of instruction fine-tuning through explicit instructions [64] and compare it with LaMSUM.

We utilize the Claritin dataset for instruction fine-tuning as it has the highest number of tweets among all the three datasets. It is divided into training (60%), validation (10%), and test (30%) set. The US-Election and MeToo datasets are retained as additional test sets for evaluation. Instruction fine-tuning involves three key components: i) Instruction: The prompts used for zero-shot prompting,

Dataset	R1	R2	RLSum
Claritin	18.60	3.11	17.14
US-Election	26.55	14.81	25.05
MeToo	28.94	10.81	24.80

Table 5: Metric scores from fine-tuned llama3.1-8b for various datasets.

as detailed in Section 3.3.2. ii) Input: Set of input text \mathcal{T} to be summarized. iii) Output: The reference summary comprises sentences extracted from \mathcal{T} .

We fine-tune an open-source LLM, llama3.1-8b from Meta [17], using a 15 GB Tesla T4 GPU. During the fine-tuning phase, we set the learning rate, warmup ratio, and batch size to $2e-4$, 0.0, and 2, respectively. For this process, we employ PEFT (Parameter Efficient Fine Tuning), which involves freezing the layers of the pre-trained model and only fine-tuning the last few layers specific to the downstream task. We follow the QLoRA [15] method in PEFT training. QLoRA is an optimized variant of LoRA [27] that reduces the precision of weight parameters to 4-bit. This reduction in precision decreases the model size, which is advantageous in scenarios with limited memory available for fine-tuning.

Table 5 shows the results of the fine-tuned llama3.1-8b model. We observed that the outputs are robust and adhere closely to the instructions provided. The generated summaries contain the exact sentences extracted from the input text, but the performance of the fine-tuned model is inferior compared to LaMSUM.

Takeaway: Fine-tuning an LLM with the appropriate instructions, input, and output can facilitate extractive summarization. But fine-tuning requires a substantial amount of data with gold-standard summaries. With less data, the model may overfit, resulting in poor performance, as shown in the Table 5. Additionally, a model fine-tuned on one dataset may not perform well on another dataset. Our proposed LaMSUM addresses these issues by being effective when the input data is large enough to exceed a single context window but small enough to be used for fine-tuning.

8 Concluding Discussion

This work marks an early attempt to achieve extractive summarization of large user-generated text that exceeds a single context window using zero-shot learning. The proposed multi-level framework LaMSUM leverages approval based and ranked based voting algorithms to generate robust summaries. Experiments conducted on three distinct crowd-sourced datasets demonstrated the efficacy of LaMSUM, as it outperformed the results achieved by state-of-the-art fine-tuned models. Ensembling of LLMs in our proposed framework can enhance the performance of weaker models.

Note that there can be a concern regarding the potential data leakage, as the experiments involve newer LLMs that may have been exposed to the experimented datasets during their pre-training phase. In Section 5.2 we showcased that the *vanilla LLM*, which also includes LLM underperformed, whereas our proposed framework which generates robust summaries yielded good results. This highlights the efficacy of our model, even when it is exposed to data leakage.

Limitation

Our proposed framework, LaMSUM, very well handles text of any length, conditioned on the fact that the final summary fits within a single context window. Some modifications to LaMSUM may be necessary when the output summary exceeds the size of a single context window. We focused only on the user-generated text where each textual unit is independent of the others; future research could extend the framework to summarization tasks involving contextual dependency such as book summarization.

Ethical Considerations

Our research focuses on using LLMs to produce extractive summaries for user-generated text. Given the recent rise of LLMs and the growing interest in applying them across different research fields, we believe this research direction can help unveil the potential of LLMs. Below, we outline concerns that need to be considered and addressed in this research area:

- Bias and discrimination: LLMs are biased towards their training data, which can lead them to favor certain textual units when creating summaries.
- Transparency and accountability: LLMs are black boxes, with an opaque decision-making process, making it difficult to discern how or why specific textual units are chosen for summarization. This lack of transparency can create ethical issues, especially in scenarios where clear explanations for decisions are needed.
- Environmental impact: The training and operation of LLMs require substantial computational resources, which contribute to increased energy consumption and may have a negative environmental impact.

While LLMs can produce high-quality summaries, their use must be approached with careful consideration of potential ethical implications.

References

- [1] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, and Krysz Kochut. 2017. Text Summarization Techniques: A Brief Survey. *International Journal of Advanced Computer Science and Applications* (2017).
- [2] Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, and Pascale Fung. 2023. A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity. In *IJCNLP and AACL*.
- [3] Paheli Bhattacharya, Soham Poddar, Koustav Rudra, Kripabandhu Ghosh, and Saptarshi Ghosh. 2021. Incorporating domain knowledge for extractive summarization of legal case documents. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law*.
- [4] Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D Procaccia. 2016. *Handbook of computational social choice*. Cambridge University Press.
- [5] Arthur Brazinskas, Mirella Lapata, and Ivan Titov. 2020. Few-Shot Learning for Opinion Summarization. In *EMNLP*.
- [6] Hannah Brown and Reza Shokri. 2023. How (Un)Fair is Text Summarization? <https://openreview.net/forum?id=-UsbRlXzMG>
- [7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *NeurIPS*.
- [8] Daniel O. Cajueiro, Arthur G. Nery, Igor Tavares, Maisa K. De Melo, Silvia A. dos Reis, Li Weigang, and Victor R. R. Celestino. 2023. A comprehensive review of automatic text summarization techniques: method, data, evaluation and coding. arXiv:2301.03403 [cs.CL] <https://arxiv.org/abs/2301.03403>
- [9] Yapei Chang, Kyle Lo, Tanya Goyal, and Mohit Iyyer. 2024. BoookScore: A systematic exploration of book-length summarization in the era of LLMs. In *ICLR*.
- [10] 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. 2023. A Survey on Evaluation of Large Language Models.
- [11] Garima Chhikara, Kripabandhu Ghosh, Saptarshi Ghosh, and Abhijnan Chakraborty. 2023. Fairness for both Readers and Authors: Evaluating Summaries of User Generated Content. In *SIGIR*.
- [12] Library Congress. 2017. Update on the Twitter Archive at the Library of Congress. <https://blogs.loc.gov/loc/2017/12/update-on-the-twitter-archive-at-the-library-of-congress-2/>.
- [13] Abhisek Dash, Anurag Shandilya, Arindam Biswas, Kripabandhu Ghosh, Saptarshi Ghosh, and Abhijnan Chakraborty. 2019. Summarizing User-generated Textual Content: Motivation and Methods for Fairness in Algorithmic Summaries. *CSCW* (2019).
- [14] Abhisek Dash, Anurag Shandilya, Arindam Biswas, Kripabandhu Ghosh, Saptarshi Ghosh, and Abhijnan Chakraborty. 2019. Summarizing user-generated textual content: Motivation and methods for fairness in algorithmic summaries. *ACM CSCW* (2019).
- [15] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. arXiv:2305.14314 [cs.LG] <https://arxiv.org/abs/2305.14314>
- [16] Yue Dong. 2018. A Survey on Neural Network-Based Summarization Methods. *arXiv preprint arXiv:1804.04589* (2018).
- [17] Abhimanyu Dubey and et al. 2024. The Llama 3 Herd of Models. arXiv:2407.21783 [cs.AI] <https://arxiv.org/abs/2407.21783>
- [18] Wafaa S El-Kassas, Cherif R Salama, Ahmed A Rafea, and Hoda K Mohamed. 2021. Automatic text summarization: A comprehensive survey. *Expert Systems with Applications* (2021).
- [19] Peter Emerson. 2013. The original Borda count and partial voting. *Social Choice and Welfare* (2013).
- [20] Günes Erkan and Dragomir R. Radev. 2004. LexRank: Graph-based Lexical Centrality As Salience in Text Summarization. *Journal of Artificial Intelligence Research* (2004).
- [21] Nikhil Garg, Benoit Favre, Korbinian Reidhammer, and Dilek Hakkani-Tür. 2009. Clusterrank: a graph based method for meeting summarization. In *Proc. Interspeech*.
- [22] Google Gemini Team. 2023. Gemini: A Family of Highly Capable Multimodal Models. *arXiv preprint* (2023).
- [23] Tanya Goyal, Junyi Jessy Li, and Greg Durrett. 2023. News Summarization and Evaluation in the Era of GPT-3. arXiv:2209.12356 [cs.CL]
- [24] Andrew Griffin. 2015. Twitter and Google team up, so tweets now go straight into Google search results. <https://www.independent.co.uk/tech/twitter-and-google-team-up-so-tweets-now-go-straight-into-google-search-results-10262417.html>.
- [25] Vishal Gupta and Gurpreet Singh Lehal. 2010. A Survey of Text Summarization Extractive Techniques. *IEEE Journal of Emerging Tech. in Web Intelligence* (2010).
- [26] Zhanying He, Chun Chen, Jiajun Bu, Can Wang, Lijun Zhang, Deng Cai, and Xiaofei He. 2012. Document Summarization Based on Data Reconstruction. In *AAAI*.
- [27] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685 [cs.CL] <https://arxiv.org/abs/2106.09685>
- [28] David I. Inouye and Jugal K. Kalita. 2011. Comparing Twitter Summarization Algorithms for Multiple Post Summaries. In *IEEE SocialCom*.
- [29] Ruipeng Jia, Yanan Cao, Hengzhu Tang, Fang Fang, Cong Cao, and Shi Wang. 2020. Neural Extractive Summarization with Hierarchical Attentive Heterogeneous Graph Network. In *EMNLP*.
- [30] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Léo Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of Experts. arXiv:2401.04088
- [31] Wenjun Jiang, Jing Chen, Xiaofei Ding, Jie Wu, Jiawei He, and Guojun Wang. 2021. Review Summary Generation in Online Systems: Frameworks for Supervised and Unsupervised Scenarios. *ACM Trans. Web* (2021).
- [32] Hanlei Jin, Yang Zhang, Dan Meng, Jun Wang, and Jinghua Tan. 2024. A Comprehensive Survey on Process-Oriented Automatic Text Summarization with Exploration of LLM-Based Methods. arXiv:2403.02901 [cs.AI] <https://arxiv.org/abs/2403.02901>
- [33] Shawn M. Jones, Martin Klein, Michele C. Weigle, and Michael L. Nelson. 2023. Summarizing Web Archive Corpora via Social Media Storytelling by Automatically Selecting and Visualizing Exemplars. *ACM Trans. Web* (2023).

- [34] Taehee Jung, Dongyeop Kang, Lucas Mentch, and Eduard Hovy. 2019. Earlier Isn't Always Better: Sub-aspect Analysis on Corpus and System Biases in Summarization. In *EMNLP*.
- [35] Neel Kanwal and Giuseppe Rizzo. 2022. Attention-based clinical note summarization. In *Proceedings of the 37th ACM SIGAPP Symposium on Applied Computing*.
- [36] Philippe Laban, Wojciech Kryscinski, Divyansh Agarwal, Alexander Fabbri, Caiming Xiong, Shafiq Joty, and Chien-Sheng Wu. 2023. SummEdits: Measuring LLM Ability at Factual Reasoning Through The Lens of Summarization. In *EMNLP*.
- [37] Martin Lackner, Peter Regner, and Benjamin Krenn. 2023. abc voting: A Python package for approval-based multi-winner voting rules. *Journal of Open Source Software* (2023).
- [38] Md Tahmid Rahman Laskar, M Saiful Bari, Mizanur Rahman, Md Amran Hossen Bhuiyan, Shafiq Joty, and Jimmy Huang. 2023. A Systematic Study and Comprehensive Evaluation of ChatGPT on Benchmark Datasets. In *Findings of the Association for Computational Linguistics: ACL 2023*.
- [39] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *ACL*.
- [40] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*.
- [41] Yang Liu and Mirella Lapata. 2019. Text Summarization with Pretrained Encoders. In *EMNLP*.
- [42] Yixin Liu, Kejian Shi, Katherine He, Longtian Ye, Alexander Fabbri, Pengfei Liu, Dragomir Radev, and Arman Cohan. 2024. On Learning to Summarize with Large Language Models as References. In *NAACL*.
- [43] Zheheng Luo, Qianqian Xie, and Sophia Ananiadou. 2023. ChatGPT as a Factual Inconsistency Evaluator for Text Summarization. arXiv:2303.15621 [cs.CL]
- [44] S. Mackie, R. McCreadie, C. Macdonald, and I. Ounis. 2014. Comparing Algorithms for Microblog Summarisation. In *CLEF*.
- [45] Derek Miller. 2019. Leveraging BERT for Extractive Text Summarization on Lectures. arXiv:1906.04165 [cs.CL]
- [46] Ram Mudambi, Pietro Navarra, and Carmela Nicosia. 1996. Plurality versus Proportional Representation: An Analysis of Sicilian Elections. *Public Choice* (1996).
- [47] Rajdeep Mukherjee, Hari Chandana Peruri, Uppada Vishnu, Pawan Goyal, Sourangshu Bhattacharya, and Niloy Ganguly. 2020. Read what you need: Controllable Aspect-based Opinion Summarization of Tourist Reviews. In *SIGIR*.
- [48] Rajdeep Mukherjee, Uppada Vishnu, Hari Chandana Peruri, Sourangshu Bhattacharya, Koustav Rudra, Pawan Goyal, and Niloy Ganguly. 2022. MTLTS: A Multi-Task Framework To Obtain Trustworthy Summaries From Crisis-Related Microblogs. In *ACM WSDM*.
- [49] Ani Nenkova and Lucy Vanderwende. 2005. *The impact of frequency on summarization*. Technical Report. Microsoft Research.
- [50] OpenAI. 2024. GPT-4o mini: advancing cost-efficient intelligence. <https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/>.
- [51] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *NIPS*.
- [52] Xiao Pu, Mingqi Gao, and Xiaojun Wan. 2023. Summarization is (Almost) Dead. arXiv:2309.09558 [cs.CL]
- [53] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners.
- [54] E.S. Ristad and P.N. Yianilos. 1998. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1998).
- [55] Koustav Rudra, Niloy Ganguly, Pawan Goyal, and Saptarshi Ghosh. 2018. Extracting and Summarizing Situational Information from the Twitter Social Media during Disasters. *ACM Trans. Web* (2018).
- [56] Philipp Schoenegger, Indre Tuminauskaitė, Peter S. Park, and Philip E. Tetlock. 2024. Wisdom of the Silicon Crowd: LLM Ensemble Prediction Capabilities Rival Human Crowd Accuracy. arXiv:2402.19379 [cs.CY] <https://arxiv.org/abs/2402.19379>
- [57] B. Sharifi, M. Hutton, and J. K. Kalita. 2010. Experiments in Microblog Summarization. In *IEEE Conference on Social Computing*.
- [58] Derek Tam, Anisha Mascarenhas, Shiyue Zhang, Sarah Kwan, Mohit Bansal, and Colin Raffel. 2023. Evaluating the Factual Consistency of Large Language Models Through News Summarization. In *ACL*.
- [59] Liyan Tang, Igor Shalymov, Amy Wing mei Wong, Jon Burns, Jake W. Vincent, Yu'an Yang, Siffi Singh, Song Feng, Hwanjun Song, Hang Su, Lijia Sun, Yi Zhang, Saab Mansour, and Kathleen McKeown. 2024. TofuEval: Evaluating Hallucinations of LLMs on Topic-Focused Dialogue Summarization. arXiv:2402.13249 [cs.CL]
- [60] Liyan Tang, Zhaoyi Sun, Betina Idray, Jordan G Nestor, Ali Soroush, Pierre A Elias, Ziyang Xu, Ying Ding, Greg Durrett, Justin Rousseau, Chunhua Weng, and Yifan Peng. 2023. Evaluating large language models on medical evidence summarization. *medRxiv* (2023).
- [61] Yuting Tang, Ratish Puduppully, Zhengyuan Liu, and Nancy Chen. 2023. In-context Learning of Large Language Models for Controlled Dialogue Summarization: A Holistic Benchmark and Empirical Analysis. In *Proceedings of the 4th New Frontiers in Summarization Workshop*.
- [62] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models.
- [63] Pat Verga, Sebastian Hofstatter, Sophia Althammer, Yixuan Su, Aleksandra Piktus, Arkady Arkhangorodsky, Minjie Xu, Naomi White, and Patrick Lewis. 2024. Replacing Judges with Juries: Evaluating LLM Generations with a Panel of Diverse Models. arXiv:2404.18796 [cs.CL] <https://arxiv.org/abs/2404.18796>
- [64] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022. Finetuned Language Models are Zero-Shot Learners. In *International Conference on Learning Representations*.
- [65] Yunshu Wu, Hayate Iso, Pouya Pezeshkpour, Nikita Bhutani, and Estevam Hruschka. 2024. Less is More for Long Document Summary Evaluation by LLMs. In *EACL*.
- [66] Jiacheng Xu, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. Discourse-Aware Neural Extractive Text Summarization. In *ACL*.
- [67] Wei Xu, Ralph Grishman, Adam Meyers, and Alan Ritter. 2013. A Preliminary Study of Tweet Summarization using Information Extraction. In *Language in Social Media (LASM)*.
- [68] Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. 2024. Hallucination is Inevitable: An Innate Limitation of Large Language Models. arXiv:2401.11817 [cs.CL] <https://arxiv.org/abs/2401.11817>
- [69] Xianjun Yang, Yan Li, Xinlu Zhang, Haifeng Chen, and Wei Cheng. 2023. Exploring the Limits of ChatGPT for Query or Aspect-based Text Summarization. arXiv:2302.08081 [cs.CL]
- [70] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. *XLNet: generalized autoregressive pretraining for language understanding*. Curran Associates Inc.
- [71] Haopeng Zhang, Xiao Liu, and Jiawei Zhang. 2022. HEGEL: Hypergraph Transformer for Long Document Summarization. In *EMNLP*.
- [72] Haopeng Zhang, Xiao Liu, and Jiawei Zhang. 2023. DiffuSum: Generation Enhanced Extractive Summarization with Diffusion. In *ACL*.
- [73] Haopeng Zhang, Xiao Liu, and Jiawei Zhang. 2023. Extractive Summarization via ChatGPT for Faithful Summary Generation. In *EMNLP*.
- [74] Haopeng Zhang, Xiao Liu, and Jiawei Zhang. 2023. SummIt: Iterative Text Summarization via ChatGPT. In *EMNLP*.
- [75] Justine Zhang, Ravi Kumar, Sujith Ravi, and Cristian Danescu-Niculescu-Mizil. 2016. Conversational Flow in Oxford-style Debates. In *NAACL*.
- [76] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020. PEGASUS: pre-training with extracted gap-sentences for abstractive summarization. In *ICML*.
- [77] Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B. Hashimoto. 2024. Benchmarking Large Language Models for News Summarization. *Transactions of the Association for Computational Linguistics* (2024).
- [78] Yusen Zhang, Nan Zhang, Yixin Liu, Alexander Fabbri, Junru Liu, Ryo Kamoi, Xiaoxin Lu, Caiming Xiong, Jieyu Zhao, Dragomir Radev, Kathleen McKeown, and Rui Zhang. 2024. Fair Abstractive Summarization of Diverse Perspectives. In *NAACL*.
- [79] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A Survey of Large Language Models.
- [80] Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2020. Extractive Summarization as Text Matching. In *ACL*.