Automating Key-Point Analysis with LLMs: Argument-to-Key-Point Mapping

Jasper Eijkenboom[1]

Supervisor(s): Stephanie Tan[1]**, Edgar Salas Gironés**[1]

[1]EEMCS, Delft University of Technology, The Netherlands

Abstract

A well-functioning democracy depends on an informed population. To help informing citizens, summaries of arguments in political transcripts can be made. An approach to argument summarization is the creation of summaries through distillation of the arguments into higher-level key points. In this approach, mapping arguments to key points is an important subtask. This study examines how model selection, prompting strategy, choice of domain, and input batching influence the performance of large language models (LLMs) in matching arguments to key points. We introduce a self-annotated dataset from U.S. Congress committee transcripts and evaluate both generative and embedding-based models on this task. Generative LLMs (GPT-3.5-turbo, o4-mini) outperform both untuned and fine-tuned RoBERTa in zero-shot argument-to-keypoint mapping (up to 0.880 macro-F1), while sparse two-shot prompting yields no gains. Moderate batching (n=32) boosts throughput without losing accuracy. These results show that a fully automated KPA pipeline—argument extraction, keypoint generation, and mapping—is achievable with current LLMs.

1 Introduction

In a good democratic system, the population needs to access accurate, up-to-date information about politics. This is important to critically evaluate policies. Recently, with the rise of the internet, a lot of political data has become available. The noise created by the abundance of data makes it harder to stay informed as citizen, and may result in a lack of political engagement. Therefore, it might be useful to summarize political arguments such that the amount of information is condensed. This can be addressed by automatic summarization of arguments.

In general, the analysis of human texts is part of the field of Natural Language Processing (NLP). Argument summarization is a subfield of NLP focused on the automatic processing and analysis of argumentative discourse.

One approach in the argument summarization subfield is to distill arguments into a list of key points, in a task called Key Points Analysis (KPA) [1]. This approach allows readers to efficiently understand the core arguments without having to read full-length texts.

The Bar-Haim et al. paper described key points in the following way: "Key points may be viewed as high-level arguments. They should be general enough to match a significant portion of the arguments, yet informative enough to make a useful summary." One of the subtasks in KPA is argument-to-keypoint mapping (KPM). In this step, some number of arguments from the text are mapped to some number of keypoints. In most papers on this topic, the KPM subtask is executed using a BERT model, such as plain BERT or RoBERTa [1][9][10].

Due to recent advancements in generative Large Language Model (LLM) technology and size, and because of the strength of generative LLMs in text-based tasks, it is probable to suppose that the KPM task could be performed by a generative LLM. This has been done in a recent study[10], using GPT-3.5-turbo. In the paper, GPT-3.5-turbo performed poorly on the KPM subtask.

Since the published of that paper, multiple generative LLMs with significantly improved performance on general and complex benchmarks have been released [8]. Using newer generative LLMs may contribute to a better performance on the KPM task due to an increase in semantical understanding of the relation between argument and keypoint.

Using more accurate generative LLMs and using a prompting strategy that results in a higher accuracy makes research into the use of generative LLMs in the KPM subtask valuable.

Main Research Question

The aim of the study is as follows: How do model choice, prompting strategy, domain, and input batching influence LLM performance on argument-to-keypoint matching?

While the scope of this research question is broad, it can be reduced depending on time constraints, uninteresting results or other considerations. Therefore the following sub questions are:

Sub-Questions

1. **Model Comparison** How does accuracy differ between generative LLMs and (fine-tuned) embedding models?
2. **Prompting Strategy** Which prompting strategy (zero-shot or few-shot) performs best for each LLM?
3. **Domain Transfer** How well do models and prompts tuned on the Congress data set perform on the ArgKP data set?
4. **Input Batching Effects** Is it better, in terms of accuracy and cost, to send multiple single-pair prompts or one multi-pair prompt?

In this thesis, we will investigate LLM performance on summarization via KPA. In particular, we experimentally evaluate the performance on some generative LLMs on the KPM substep of the process proposed by Bar-Haim et al. (2020) [1].

We will start by giving a background of the problem and a summarization of the existing literature. Next to this, we will define important terms and dive into language models that can be used for this mapping task. After that, we will introduce a novel manually annotated data set from US House committee transcripts and describe the annotation process and outcomes. We will be performing the arguments-to-key-points mapping task using this data set. Thereafter, we will explain the experimental setup and its evaluation by F1 macro. Lastly, we will discuss the results.

2 Background

2.1 Definitions

We will start out by giving some definitions related to the context of KPM. Van der Meer et al. propose that "[a]n argument needs to at least (1) contain (informal) logical reasoning, (2) address a why question, and (3) have a non-neutral stance towards the issue being discussed[10]. A topic in the most general sense is the subject of what is being discussed. In the context of this research, we narrow the definition of a topic down to a policy proposal, e.g. "Abortion should be illegal". We will take both topic and policy proposal to mean the same thing from this point. Additionally, we define arguments to only be policy-related. Using these definitions we can summarize that, in the context of this research, the connection of a related policy proposal, argument and stance is that each argument provides the reasoning for the stance towards the policy proposal. Next, we define a key point to be a higher-level argument that has the ability to cover a multitude of arguments. An argument and a key point are matching if the argument is a more specific version of the key point.

2.2 Related Work

One approach to summarization of argumentative texts, such as parliamentary proceedings, is to present a list of argumentative Key Points.

Bar-Haim et al. [1] laid the groundwork for this approach, named Key Point Analysis (KPA). In order to execute the key points summarization, a three-step approach was proposed. In the first two steps, the arguments and key points are extracted from the text and in the third step, an arguments-to-key-points mapping (KPM) is made. This last step will be the focus of our experiment. The authors created the aptly named ArgKP dataset, containing 24 093 argument–key point pairs (6 515 arguments and 243 key points) across 28 controversial topics. Additionally, the authors showed that a domain expert can create set of key points per topic cover a significant portion of the arguments.In their experiments, unsupervised BERT-embedding similarity reached an F_1 of 0.403. A fine-tuned BERT-large on ArgKP attained an F_1 score of 0.684.

Van der Meer et al. [10] examine three dimensions Key Point Analysis across three datasets. They find that GPT-3.5-turbo, while strong at Key Point Generation, achieves only a mean Average Precision (mAP) of 0.17–0.46 in KPM, whereas a RoBERTa model reaches mAP up to 0.82. In the paper, GPT-3.5-turbo was queried for a valid JSON object. If the JSON syntax was invalid, the answer by the model was deemed incorrect. Having to create a JSON object using valid syntax next to performing the KPM subtask could explain the low performance of the model in the paper. Secondly, the model was given a batch prompt. In a batch prompt, multiple yes/no questions are asked simultaneously in the same prompt, instead of separately for each yes/no question. This approach has been shown to decrease accuracy when using large batch sizes [5].

2.3 Models and Prompting

We frame argument-to-keypoint mapping as a task for both (fine-tuned) embedding models and prompt-driven generative models. RoBERTa [6] is a large, pre-trained embedding model with a masked-language objective, meaning it seeks to find the best substitute for a masked placeholder in a text. It can be fine-tuned to certain tasks, causing better performance in that context. Generative Pre-trained Transformer (GPT) models (ChatGPT-3.5-turbo, o3, o4-mini) generate next tokens and do not rely on fine-tuning for performance.

Instead, performance can be tailored to a specific context based on the input structure. The model's input is called a prompt. Zero-shot prompts consist solely of a query without examples, whereas few-shot prompts include examples in the input. These added examples have been shown to improve performance [2]. Newer, larger models can perform Chain-of-Thought reasoning, in which models such as OpenAI's o3 and o4-mini publicly or internally note intermediate reasoning steps, increasing performance [11].

3 Methodology & Experimental Setup

3.1 Data

We created a data set from House Committee Hearing transcripts from the United States House of Representatives, hereafter referred to as the House Hearing Transcripts (HHT) data set. The representatives take part in committees or subcommittees that hold (generally public) hearings. In these hearings, witnesses are interviewed and discussions take place in relation to new or current operational legislation, or in relation to the functioning of the government[1]. The transcripts of the hearings are released to the public.

The following workflow was used for creating the dataset: The dataset was created in three phases; the argument-extraction phase, the key-point-generation phase and the key-point-mapping annotation phase.

In order to create the HHT data set, transcripts from 11 hearings out of a total of 1015 hearings from the 117th Congress (2021-2022) were selected. A list of selected hearings can be found in the Appendix (see Table 5).

In the first phase, each transcript was appended to a prompt (see Appendix A.1), and given to OpenAI o3 version xxxxx. Included in the prompt instructions were definitions of topic argument keypoint etc., and requested output of policy proposals, arguments, and stances. This resulted in 66 policy proposals and a total of 269 arguments.

An extracted argument is included if it satisfies the following criteria:

- The extracted argument exists verbatim in the transcript
- The argument follows our definition of an argument (See Section 2.2)
- The subject of argument is the related policy proposal
- The extracted stance is in accordance with the argument

Out of the 269 arguments, 143 arguments relating to 60 topics satisfied the criteria.

In the second phase, OpenAI o3. was prompted (See Appendix A.2) to give, for each of the 60 policy proposals, a response containing 4 key points with a Pro-stance and 4 key points with a Con-stance, resulting in 480 key points. Each key point was checked for following the definition for an argument and relating to the policy proposal. All 480 key points satisfied this criterion.

In the final phase, 3 students interested in politics were asked to write down per argument to which subset of the 4 topic/stance key point(s) it corresponds. This can either be 0,

[1]https://web.archive.org/web/20210602142207/https://www.senate.gov/reference/glossary_term/hearing.html

1 or multiple key points. A Venn diagram showing the overlap of annotator choices can be found in Figure 1.

Due to the use of more than 2 annotators, Fleiss' κ [3] was used to assess inter-annotator agreement, resulting in a κ of 0.609, which corresponds to a moderate-to-substantial agreement [4].

Arguments were assigned to key points if the majority, at least 2 out of 3 annotators, mapped it that way. The result in 119 matchings between an argument and a keypoint. To create the final data set, each of the 143 topic-argument-stance data point was appended once for each of the 4 stance-corresponding key points of the related topic, resulting in 572 data points. Each of the 119 data points with matching argument and key point was given a label of 1, and each non-matching row a label of 0. The resulting data set thus consists of two imbalanced classes; 119 matching (label 1) to 453 non-matching (label 0) data points.

3.2 Experiment Setup

We begin by partitioning the House Hearing Transcripts (HHT) dataset—572 argument–keypoint pairs—into an 80% test split (to evaluate in-domain performance), a 10% training split (for fine-tuning RoBERTa), and a 10% validation split (for early stopping during fine-tuning) and with stratified class labels.

Each argument–keypoint pair is presented to the model with a prompt in which it is asked to determine if the pair is a match (see Appendix Table 4); outputs "yes" and "no" are mapped to labels 1 and 0, respectively, and compared against the ground-truth label to compute accuracy and macro-F1 scores. We investigate the following variables:

Model Choice We compare RoBERTa-large in two configurations; out-of-the-box and fine-tuned. Additionally, we compare the performance of generative LLMs GPT-3.5-turbo and o4-mini. Generative models are queried with temperature set to 0.

Prompting Strategy Generative models were queried with zero-shot and few-shot prompts and asked to provide a yes/no decision on the matching of an argument-keypoint pair. Embedding models were queried with an analogous zero shot prompt, instead being asked to substitute a decision for a masked token. For the prompts, see Appendix Table 4

Domain Each model is assessed on two distinct domains: the HHT test set (80% of total, 95 positive, 362 negative examples) and a random balanced subset of 1 000 pairs drawn from the ArgKP dataset. The HHT test set has an average amount of 44.61 words per argument, the ArgKP subset has an average of 18.63 words per argument.

Batching To isolate the impact of grouping multiple argument–keypoint queries into a single prompt, we make the following distinction. *Single–pair:* (n=1) Each API call contains exactly one argument–keypoint pair. This condition establishes our baseline performance and is used in both zero-shot and few-shot prompts. *Multi–pair:* ($n > 1$) We concatenate n independent argument–keypoint pairs into a single prompt, numbered by index. Next, the model is to provide a "yes" or "no" answer for each index. We evaluate batch sizes $n \in \{32, 64, 128, 256\}$ for HHT, reusing the same prompt template described in Appendix Table 4 for each sub–query. After analysing the results, a selection of batching sizes will also be applied to the balanced ArgKP subset. The experiment will be performed by o4-mini, the only model with a large enough context window.

Fine-tuning We reformulate argument–keypoint pairs as a zero-shot prompt by injecting topic, argument text, key point text, and stance ("Pro"/"Con") into the template in Table 4. We load CSV splits into pandas, apply the template per row, and convert to a Hugging Face Dataset with a labels column. We tokenize all examples to a fixed length of 512 tokens (padding/truncation). The model is fine-tuned as a binary classifier. For fine-tuning, we use Hugging Face Trainer over 5 epochs with per-device batch size 8, learning rate 5e-05, weight decay 0.01, and 500-step linear warm up. We evaluate (accuracy + macro-F1), and apply early stopping after two non-improving validations.

Evaluation For every combination of model, prompt type, batching style, and domain, we report macro F1 and accuracy. As the HHT dataset has a large class imbalance, macro F1 score is the most important metric.

4 Results

4.1 Impact of Model Choice

Comparing model performance in Table 1, o4-mini decidedly outperforms GPT-3.5-turbo and the untuned RoBERTa-large model, achieving the highest macro F1 of 0.734 and accuracy of 0.803 under zero-shot prompting.

As noted earlier, given the pronounced class imbalance in the HHT dataset, the F1 score should be taken as the primary evaluation metric. Illustrating the need for this, the accuracy of the fine-tuned RoBERTa-large model rivals that of o4-mini. Inspecting the output data reveals that the fine-tuned model assigned only no (match) to every argument-keypoint pair, explaining the low macro F1 score of 0.442. In contrast, GPT-3.5-turbo attains a macro F1 of 0.531 (acc. 0.604), and vanilla RoBERTa-large performed worst with F1 0.387 (acc. 0.385).

4.2 Impact of Prompting Strategy

Investigating the same Table 1, neither generative model shows consistent performance improvement using the few-shot prompting strategy. GPT-3.5-turbo's macro F1 and accuracy both decrease slightly in the few-shot condition with F1 0.521 compared to the zero-shot F1 of 0.531 (acc. 0.595 vs. 0.604). Next, o4-mini also sees a small drop using the few-shot strategy (F1 0.707 vs. 0.734; Acc. 0.777 vs. 0.803), still showing the best performance out of all models.

4.3 Domain Transfer

Using the zero-shot prompting strategy on a balanced ArgKP sample, Table 1 again shows o4-mini's superior performance, scoring highest in both macro F1 and accuracy with a score of 0.880. GPT-3.5-turbo performs moderately well (F1 0.685, acc. 0.687).

Model →	GPT-3.5-turbo		o4-mini		RoBERTa-large		Fine-tuned RoBERTa-large	
Strategy ↓	Macro F1	Accuracy	Macro F1	Accuracy	Macro F1	Accuracy	Macro F1	Accuracy
Zero-shot prompt	0.531	0.604	0.734	0.803	0.387	0.385	0.442*	0.792*
Few-shot prompt	0.521	0.595	0.707	0.777				

Table 1: Performance of models by prompting strategy for both generative and discriminative models on the HHT test set (95 positive, 362 negative examples). *The fine-tuned model classified every pair as not matching.

Model	Macro F1	Accuracy
GPT-3.5-turbo	0.685	0.687
o4-mini	0.880	0.880
Base RoBERTa-large	0.506	0.516
Fine-Tuned RoBERTa-large	0.333*	0.500*

Table 2: Comparison of zero-shot KPM performance on balanced ArgKP sampled data set (n=1000; class balance is 50/50). *The fine-tuned model classified every pair as not matching.

In contrast, base RoBERTa-large attains only F1 0.506 (acc. 0.516), an increase compared to its performance on the HHT data set. The fine-tuned RoBERTa-large model once again classified every pair as not matching, explaining the low F1 of 0.333 despite a 0.500 accuracy. The difference in accuracy compared to its HHT performance is completely explained by the difference in class balance between datasets. These results show that generative LLMs, particularly o4-mini, deliver superior performance for argument-to-keypoint mapping compared to discriminative models.

4.4 Input Batching Effects

Batching	Macro F1	Accuracy
HHT (n=1)	0.707	0.777
HHT (n=32)	0.739	0.796
HHT (n=64)	0.712	0.777
HHT (n=128)	0.718	0.768
HHT (n=256)	0.602	0.630
ArgKP (n=1)	0.880	0.880
ArgKP (n=32)	0.885	0.885
ArgKP (n=256)	0.858	0.860

Table 3: Effect of single pair (n=1) prompting vs. batching (n >1) for o4-mini on both HHT and ArgKP datasets.

Finally, the influence of batching multiple argument–keypoint pairs in a single prompt was investigated using o4-mini and both datasets (see Table 3). On HHT, a small batch of 32 examples results in the highest macro F1 (0.739) and accuracy (0.796), improving over single-pair prompting (F1 0.707, Acc. 0.777). Larger batches ($n \leq 64$), see diminishing returns. For n = 128 the F1 drops to 0.718 (acc. 0.768), and for n = 256 performance decreases even further (F1 0.602, acc. 0.630).

A similar but less pronounced trend appears on ArgKP. A batch size of 32 slightly outperforms single-pair prompting (F1 0.885 vs. 0.880; acc. 0.885 vs. 0.880), while very large batches (n = 256) again see diminishing results (F1 0.858, acc. 0.860). These results indicate that modest batching can stay accurate while boosting throughput, but overly large batch sizes show lower performance.

5 Responsible Research

We reflect on the ethical consideration related to using language models for the KPM task.

Firstly, using KPM for summarizing political arguments must be done carefully to avoid misrepresenting politicians' statements, as inaccuracies could wrongly influence public opinion.

Secondly, generative LLMs, such as the models used in this study, have significant environmental impacts due to high energy consumption and substantial CO_2 emissions during training [7]. Running models locally for tasks like KPA and other NLP tasks would reduce environmental costs. However, suitable models are currently too large to run in typical local setups.

Regarding reproducibility, outputs from LLMs inherently include randomness, primarily introduced through parameters like temperature settings. Consequently, even though temperature was set to zero, randomness persists. Experimental results are thus not fully reproducible.

The data set creation process was thoroughly documented to facilitate transparency and reproducibility.

6 Discussion

6.1 Data and Annotation

Our self-annotated House Hearing Transcripts (HHT) dataset exhibits an inherently low argument density: only 143 of the 269 extracted spans met our formal criteria for arguments, yielding 572 argument–keypoint pairs after replication (119 positive, 453 negative). This sparsity reflects the nature of congressional hearings, where much of the discourse consists of procedural remarks, witness testimony, or claims without explicit informal logical reasoning. Consequently, the dataset contains a high proportion of non-matching pairs, introducing severe class imbalance.

Additionally, the lack of a debate format in subcommittee hearings causes many extractable policy proposals to only be related to a single argument. Topic–argument–keypoint groups with only one relevant argument cannot be used in this experiment, as there would be little use of summarization. The moderate-to-substantial inter-annotator agreement (Fleiss' $\kappa = 0.609$) points to some subjectivity in key-point assignment, suggesting that the used dataset is not of the highest quality. The employment of more annotators, the exclusion of annotators with low pair-wise agreement, and the inclusion of control questions with a pre-determined answer all could have contributed to a higher Fleiss' κ. An argument in support of the quality of the dataset could be made, as the possibility of consistently high performance by o4-mini points to a lack of noise and the existence of structure in the data. Finally, our reliance on ChatGPT (o3) for initial extraction and key-point generation may propagate model biases into the ground-truth labels, in turn biasing later evaluation by related (o4-mini) models.

6.2 Experiments and Results

Contrary to the expectation that few-shot demonstrations clarify mapping criteria, our two-shot setup (one positive, one negative exemplar) did not improve performance over zero-shot prompting. Both GPT-3.5-turbo and o4-mini showed slight performance drops in the few-shot condition (Table 1). The cause of this could be the low amount of examples (two; one positive and one negative). In the literature, few-shot prompts with tens of examples show a performance increase [2].

Generative LLMs outperformed discriminative RoBERTa variants across domains. In zero-shot on HHT, o4-mini achieved the highest macro-F1 (0.734) and accuracy (0.803), while fine-tuned RoBERTa collapsed to majority-class predictions, and untuned RoBERTa performed worse than its fine-tuned counterpart. On the balanced ArgKP sample, o4-mini further showed superior performance (F1 = 0.880), indicating its strength in zero-shot generalization. These results imply that generative LLMs have greater semantic understanding than task-specific fine-tuned models, and outperform them on small, imbalanced datasets.

We also observed modest benefits from batching multiple pairs in a single prompt, which aligns with the findings in [5]. For o4-mini on HHT, a batch size of 32 increased macro-F1 from 0.707 to 0.739 and accuracy from 0.777 to 0.796, though larger batches suffered diminishing returns. A similar effect appeared on ArgKP. This indicates that moderate batching can improve throughput without sacrificing quality, but overly large context windows lower matching-task performance.

Bar-Haim et al. [1] showed that fine-tuned BERT-large on ArgKP reached an F_1 of 0.684, which is comparable to our RoBERTa baseline on a subset of the same data set. The zero-shot performance of o4-mini (F_1 of 0.734) is an improvement.

Overall, our findings highlight both the promise and limitations of generative LLMs for argument-to-keypoint mapping. Both generative LLMs performed in zero-shot settings, yet their in-context learning requires sufficient examples to yield few-shot gains. The superior performance of o4-mini, and its tolerance to moderate batching shows its usefulness for NLP tasks such as argument-to-keypoint mapping.

7 Conclusion and Future Work

We investigated the feasibility of fully automated argument-to-keypoint mapping (KPM) using state-of-the-art large language models. Our zero-shot experiments showed that generative models (GPT-3.5-turbo and o4-mini) substantially outperform (fine-tuned) discriminative baselines (RoBERTa) on both our House Hearing Transcripts (HHT) and the balanced ArgKP sample, achieving up to 0.880 macro-F1. Moderate batching (up to 32 pairs) further improved performance without degrading accuracy. In contrast, our minimal two-shot few-shot setup failed to yield gains, indicating that reliable in-context learning demands substantially more examples.

These findings answer our core research questions: (1) generative LLMs can perform KPM effectively in zero-shot settings even on small, noisy, and imbalanced argument corpora; (2) sparse few-shot prompts may introduce noise rather than clarity; (3) generative LLMs use their deeper semantic understanding to generalize effectively across diverse datasets; and (4) modest batching increases throughput and maintains quality;

In sum, our work demonstrates that a fully automated KPA pipeline is within reach: argument extraction, key-point generation, and mapping can all be handled by competitive LLMs.

References

[1] Roy Bar-Haim, Lilach Eden, Roni Friedman, Yoav Kantor, Dan Lahav, and Noam Slonim. From arguments to key points: Towards automatic argument summarization. *arXiv preprint arXiv:2005.01619*, 2020.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[3] Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

[4] J Richard Landis and Gary G Koch. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174, 1977.

[5] Jianzhe Lin, Maurice Diesendruck, Liang Du, and Robin Abraham. Batchprompt: Accomplish more with less. *arXiv preprint arXiv:2309.00384*, 2023.

[6] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[7] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

[8] Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, et al. Humanity's last exam. *arXiv preprint arXiv:2501.14249*, 2025.

[9] Ahnaf Mozib Samin, Behrooz Nikandish, and Jingyan Chen. Arguments to key points mapping with prompt-based learning. *arXiv preprint arXiv:2211.14995*, 2022.

[10] Michiel Van Der Meer, Piek Vossen, Catholijn M Jonker, and Pradeep K Murukannaiah. An empirical analysis of diversity in argument summarization. *arXiv preprint arXiv:2402.01535*, 2024.

[11] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

A Prompts

A.1 Argument extraction prompt:

```
You now are ArgMineGPT. Your goal is to exhaustively find every arguable statement in a transcript, assign it a clear policy “topic,” and
    tag each argument’s stance toward that single proposal.

**Definition of an “arguable statement (argument):**
1. Contains informal logical reasoning (e.g., “because,” “therefore,” “so that”).
2. Addresses a why/should question (explicitly or implicitly).
3. Takes a non-neutral stance (Pro or Con) on exactly one policy proposal.

---

### Step 1: Verbatim Extraction & Stance Tagging
1. Scan the text in sequence and identify each contiguous span (up to 2 sentences) that meets criteria 1-3.
2. For each such span:
   - Output the **exact raw text** (letter-for-letter, in straight quotes).
   - Immediately append `(Speaker: <name/role>; Stance: Pro/Con)`-decide whether the speaker is arguing **for** or **against** the exact
       policy proposal.
   - If a speaker recommends adopting or strengthening that policy, tag “Pro.” If they argue against or reject that policy, tag “Con.”

---

### Step 2: Cluster by Single, Directional Policy Topics
1. After extracting all arguments (verbatim + stance), group them into at most eight (8) distinct topics-**each topic must name one
    single policy proposal**, e.g.:
   - “Mandatory COVID-19 Vaccination Policy”
   - “Prioritize Federal Funding for Rail Infrastructure”
   - “Maintain or Increase Federal Highway Spending”
   - “Expand Federal Grants for Rural Broadband”
   - “Strengthen Encryption Standards for Government Agencies”
2. **Ensure each topic has at least three (3) arguments total** (Pro + Con combined). If you find more than eight potential proposals,
    merge any that refer to the same underlying policy (e.g. “Rural broadband grants” + “Rural Internet funding” → “Expand Federal Grants
    for Rural Broadband”).
3. **Do not** use any “vs.” comparison or procedural label. Each topic must be a single actionable policy that one can be Pro or Con to.

---

### Step 3: Output Format
Produce your answer exactly in this format (no extra commentary):

Topic 1: <Exact Policy Proposal>
• Argument 1.1 (Speaker: <name/role>; Stance: Pro) - “<Full raw text of argument 1.1>”
• Argument 1.2 (Speaker: <name/role>; Stance: Con) - “<Full raw text of argument 1.2>”
• Argument 1.3 (Speaker: <name/role>; Stance: Pro) - “<Full raw text of argument 1.3>”
(You can have more than 3 arguments under a given topic.)

Topic 2: <Exact Policy Proposal>
• Argument 2.1 (Speaker: <name/role>; Stance: Con) - “<Full raw text of argument 2.1>”
• Argument 2.2 (Speaker: <name/role>; Stance: Pro) - “<Full raw text of argument 2.2>”
• Argument 2.3 (Speaker: <name/role>; Stance: Con) - “<Full raw text of argument 2.3>”
(You can have more than 3 arguments under a given topic.)

...
(up to Topic 8, each with  3 arguments) ...


- Number each argument as `Argument X.Y`, where X = topic number, Y = argument index within that topic (starting at 1).
- Always put the raw text in straight quotes (`“...”`) for exact copying.
- Ensure each topic label names one single policy proposal so “Pro” means “for that proposal” and “Con” means “against that proposal.”

---

### Step 4: Continuation & Exhaustiveness
- Process the transcript strictly in the order it appears to avoid skipping any arguable spans.
- Include **all** arguments, even minor or repeated ones-as long as they meet criteria 1-3.
- If your output exceeds one response, continue seamlessly in the next completion beginning at “Topic N+1:” where N is the last topic
    number you used.

---

**Now process the full transcript below:**
[PASTE FULL TRANSCRIPT HERE]
```

A.2 Keypoint generation prompt:

```
You will be given a policy proposal, and you need to think of Key Points in the sense of the Bar-Haim paper which is given together with
    this prompt. You need to create 4 Key Points that are for the proposal (Pro), and 4 Key Points that are against (Con)

**Do this for the policy proposal below:**

[Policy proposal]
```

A.3 KPM Prompts

Table 4: Overview of the language models and corresponding prompts for the KPM task

Model	base & fine-tuned RoBERTa-large	GPT-3.5-turbo & o4-mini
Zero-shot Prompt	`<s>Determine if the key point summarizes (part of) the argument: You’ll only respond with “yes” or “no”-no extra text.` `policy proposal: "{topic}" argument: "{argument}"` `stance: "{stance}" key point: "{key_point}"` `output: <mask> </s>`	`Determine if the key point summarizes (part of) the argument: You’ll only respond with “yes” or “no”-no extra text.` `policy proposal: "{topic}" argument: "{argument}"` `stance: "{stance}" key point: "{key_point}"` `output:`
Few-shot Prompt		`You are asked to determine if the key point summarizes (part of) the argument. You’ll only respond with “yes” or “no”-no extra text.` `Example 1:` `policy proposal: "Adopt Sen. Graham’s proposal for a nation-wide ban on abortions after 15 weeks of pregnancy"` `argument: "Their end game is a nationwide abortion ban that will rip away freedoms for millions of women and put our Nation’s healthcare providers at risk of imprisonment."` `stance: "Con"` `key point: "A federal mandate overrides states’ rights and conflicts with local self-governance."` `Output: no` `Example 2:` `policy proposal: "Make Paying Ransoms to Cyber-Criminals Illegal"` `argument: "Paying the hackers is sort of like aiding and abetting the next crime."` `stance: "Pro"` `key point: "Cutting off ransom funds prevents criminals from financing other illicit activities, including terrorism."` `output: yes` `Determine if the key point summarizes (part of) the argument. You’ll only respond with “yes” or “no”-no extra text:` `policy proposal: "{topic}" argument: "{argument}"` `stance: "{stance}" key point: "{key_point}"` `output:`

A.4 Table of transcripts

Table 5: United States House of representatives (117th Congress, 2021–2022) hearing transcripts (n=11) used in this study. Dates are presented in U.S. format (MM/DD/YYYY).

Serial No.	Committee	Date	Hearing Title
117-1	House Energy and Commerce	02/02/2021	NO TIME TO LOSE: SOLUTIONS TO INCREASE COVID-19 VACCINATIONS IN THE STATES
117-2	House Judiciary	02/11/2021	The U.S. Immigration System: The Need for Bold Reforms
117-12	House Judiciary	03/18/2021	Discrimination and Violence Against Asian Americans
117-35	House Energy and Commerce	05/26/2021	A Shot at Normalcy: Building COVID-19 Vaccine Confidence
117-42	House Energy and Commerce	06/13/2021	MEMBER DAY
117-58	House Education and Labor	09/21/2022	Examining the Administration of the Unemployment Insurance System
117-60	House Judiciary	03/29/2022	OVERSIGHT OF THE FEDERAL BUREAU OF INVESTIGATION, CYBER DIVISION
117-69	House Judiciary	05/19/2022	Oversight Hearing on Clemency and the Office of the Pardon Attorney
117-102	House Oversight and Reform	09/15/2022	Fueling the Climate Crisis: Examining Big Oil's Prices, Profits, and Pledges
117-103	House Financial Services	11/15/2022	Investing in our Rivals: Examining U.S. Capital Flows to Foreign Rivals and Adversaries Around the World
117-107	House Oversight and Reform	09/29/2022	Examining the Harm to Patients from Abortion Restrictions and the Threat of a National Abortion Ban

B Figures

B.1 Annotator Choice Venn Diagram

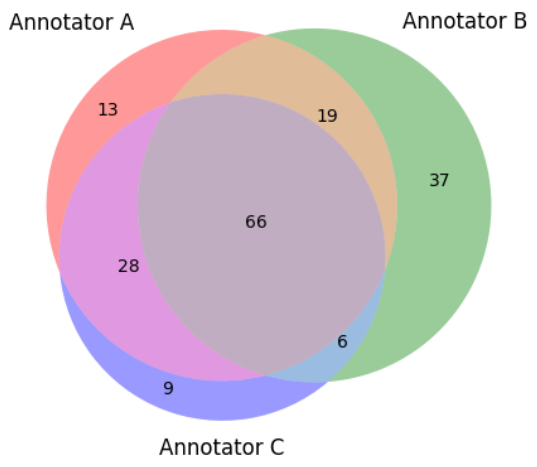

Figure 1: Venn diagram showing the overlap of argument-keypoint annotations by annotators A, B, and C.

C Data & Code Availability

All scripts and (raw) data for reproducing the experiments in this thesis are archived at github.