SpeechEE@XLLM25: Retrieval-Enhanced Few-Shot Prompting for Speech Event Extraction

Máté Gedeon
Budapest University of Technology and Economics
Budapest, Hungary
gedeonm01@gmail.com

Abstract

Speech Event Extraction (SpeechEE) is a challenging task that lies at the intersection of Automatic Speech Recognition (ASR) and Natural Language Processing (NLP), requiring the identification of structured event information from spoken language. In this work, we present a modular, pipeline-based SpeechEE framework that integrates high-performance ASR with semantic search-enhanced prompting of Large Language Models (LLMs). Our system first classifies speech segments likely to contain events using a hybrid filtering mechanism including rule-based, BERT-based, and LLM-based models. It then employs few-shot LLM prompting, dynamically enriched via semantic similarity retrieval, to identify event triggers and extract corresponding arguments. We evaluate the pipeline using multiple LLMs—Llama3-8B, GPT-4o-mini, and o1-mini—highlighting significant performance gains with o1-mini, which achieves 63.3% F1 on trigger classification and 27.8% F1 on argument classification, outperforming prior benchmarks. Our results demonstrate that pipeline approaches, when empowered by retrieval-augmented LLMs, can rival or exceed end-to-end systems while maintaining interpretability and modularity. This work provides practical insights into LLM-driven event extraction and opens pathways for future hybrid models combining textual and acoustic features.

1 Introduction

Information extraction aims at automatically identifying structured information, such as entities and their relations, from unstructured data (Bikel et al., 1997; Fei et al., 2023). A task in this domain is Event Extraction (EE) (Chen et al., 2015) searching for answers to questions like *what happened*, *who was involved*, and *where did it take place*. The source data can be text (Yang and Mitchell, 2016), but even images (Li et al., 2020) or videos (Chen et al., 2021). *Speech Event Extraction* (SpeechEE) (Kang et al., 2024) extends textual EE, with the purpose of identifying structured event information directly from the input of the spoken language. This task occupies a unique intersection between Automatic Speech Recognition (ASR) and Natural Language Processing (NLP), requiring not only accurate transcription but also the detection of event types, triggers, and arguments from possibly noisy spoken content.

Existing SpeechEE approaches can be broadly categorized into the methodologies *pipeline-based* and *end-to-end*. Pipeline-based architectures typically employ an ASR module to transcribe speech, followed by text-based event extraction using NLP techniques (Cao et al., 2022; Fei et al., 2024). These systems offer modularity and transparency, allowing separate optimization and analysis of ASR and extraction components. However, they are susceptible to cascading errors, where transcription inaccuracies can significantly impair downstream event extraction performance. Conversely, end-to-end approaches aim to bypass intermediate text by learning to map raw audio directly to structured outputs (Wang et al., 2024). While promising in reducing error propagation and potentially more efficient, these models often demand large-scale annotated audio-event datasets and are less interpretable, acting as opaque "black boxes" in many cases.

Recent progress in Large Language Models (LLMs) such as GPT-4 (OpenAI, 2024a) has opened new possibilities in pipeline-based architectures by enabling powerful few-shot and zero-shot learning capabilities. LLMs exhibit remarkable proficiency in extracting structured knowledge from unstructured text with minimal task-specific supervision (Zhang et al., 2022). When combined with state-of-the-art ASR systems, these models can form the backbone of robust SpeechEE systems that generalize well across domains and require minimal adaptation.

In this work, we present a pipeline-based SpeechEE framework that leverages semantic search-enhanced few-shot prompting with LLMs. Our system dynamically retrieves relevant examples from a support set and incorporates them into prompts to guide event extraction. Additionally, we introduce a classification mechanism to identify utterances likely to contain events, reducing false positives and improving extraction precision.

Our main contributions are as follows:

- We propose a multi-stage SpeechEE pipeline combining high-performance ASR with semantic search-enhanced prompting for event extraction using LLMs.
- We introduce a generalizable few-shot learning strategy based on semantic similarity, applicable to various text-related information extraction tasks.
- We develop a speech segment classification module that selectively filters utterances likely to contain events.
- We provide a detailed comparison of several configurations, offering practical insights for SpeechEE deployment.

The remainder of the paper is organized as follows. Section 2 reviews related work on SpeechEE and language models. Section 3 introduces the dataset. Section 4 describes the model architecture and pipeline components. Section 5 presents the experimental results and analysis. Section 6 outlines the results of the ablation study. Section 7 discusses key findings and limitations. Finally, Section 8 concludes the paper and suggests directions for future research.

2 Related Work

2.1 Event Extraction from Text

EE from textual data has been a long-standing task in information extraction, focusing on identifying event triggers and their semantic arguments. Early approaches were largely feature-based, relying on hand-engineered lexical, syntactic, and semantic features fed into statistical models such as maximum entropy classifiers, conditional random fields, or nearest-neighbor methods (Ahn, 2006) (Liao and Grishman, 2010).

With the advent of deep learning, neural-based models became dominant. Convolutional Neural Networks (CNNs) enabled automatic feature learning from word embeddings, replacing manual feature engineering (Nguyen and Grishman, 2015). With the ability to handle long-range dependencies, recurrent architectures achieved state-of-the-art results on the ACE2005 dataset (Nguyen et al., 2016).

Graph Convolutional Networks (GCNs) have also shown promise in the field by modeling syntactic structures directly from dependency trees. Unlike sequential models, GCNs exploit the syntactic proximity between triggers and arguments in a graph form, improving performance on long sentences with distant dependencies (Nguyen and Grishman, 2018; Liu et al., 2018).

More recently, Transformer-based solutions (Vaswani et al., 2017) also emerged, with (Paolini et al., 2021) introducing a framework, achieving new state-of-the-art results on joint entity and relation extraction using a generative transformer.

A recent survey categorizes modern approaches into sequence-based, graph neural, knowledge-enhanced, and prompt-based methods, highlighting the dominance of pretrained language models in capturing contextual event semantics (Xie et al., 2025).

2.2 Speech Processing and ASR

Automatic Speech Recognition (ASR) has witnessed remarkable progress in recent years, particularly with the introduction of transformer-based models. Whisper (Radford et al., 2022) represents a significant advancement in ASR, trained on a large and diverse dataset of 680,000 hours of multilingual and multitask supervised data. This approach has demonstrated robust performance across various domains and languages, making it suitable for real-world applications.

Similarly, Canary (Puvvada et al., 2024) offers an alternative approach to ASR that achieves competitive performance without relying on web-scale data. These models provide high-quality transcriptions that can serve as the foundation for downstream NLP tasks, including event extraction.

2.3 Speech Event Extraction

Speech Event Extraction (SpeechEE) is a relatively new research direction that aims to bridge ASR and event extraction. (Wang et al., 2024) introduced a novel benchmark for SpeechEE and proposed an end-to-end model for extracting events directly from audio. Their work highlights the challenges of extracting structured information from speech without relying on intermediate textual representations. Their end-to-end system demonstrated consistently superior performance compared to the employed pipeline-based baseline. They report that previous approaches to speech-based information extraction have largely relied on pipeline methods, where ASR is followed by text-based extraction. While these methods benefit from advances in both ASR and text-based event extraction, they often suffer from error propagation, where mistakes in transcription lead to extraction failures.

2.4 Large Language Models and Few-Shot Learning

Large Language Models (LLMs) have emerged as transformative tools in natural language processing (NLP), enabling significant progress across a wide range of tasks (Wu et al., 2024) including text generation, summarization, machine translation, and information extraction. Proprietary models like GPT-4o (OpenAI, 2024b) and o1-mini (OpenAI, 2024c) or open source models like Llama 3 (Meta, 2024) exemplify the scale and versatility of these models. Their capacity to perform tasks with minimal or no explicit training—known as few-shot or zero-shot learning—has shifted the paradigm from model-specific fine-tuning to prompt-based task generalization.

Few-shot learning in LLMs typically involves crafting prompts that include a handful of task-specific examples, guiding the model to infer patterns and generalize to unseen inputs. This method leverages the latent knowledge encoded in the pre-trained model, often yielding strong performance on tasks such as question answering, named entity recognition, and relation extraction (Gao et al., 2021; Min et al., 2022). Particularly for structured information extraction, few-shot prompting enables LLMs to identify and retrieve relevant spans of text even in the absence of large annotated datasets.

Our work builds upon these advances by integrating LLMs with semantic search techniques to dynamically select the most relevant examples for few-shot learning, thereby enhancing the models' ability to extract events from speech transcriptions.

3 Dataset

We use the SpeechEE shared task dataset[1], derived from ACE2005-EN+. The dataset is provided in a structured JSON format, where each entry consists of a unique `id`, an event `trigger` indicating the lexical anchor of the event, its corresponding `type`, and a list of associated `arguments`, each annotated with a semantic role. In total, the dataset includes 33 distinct event types and 22 argument roles, offering a diverse and challenging benchmark. It comprises 19,217 training instances, 901 development examples, and 676 test samples. An example of the data format is shown below:

```
{"id": "train-6",
 "event": [
   {"trigger": "election",
    "type": "Elect",
    "arguments": [
      {"name": "man", "role": "Person"}
    ]}
 ]}
```

4 Methodology

Our proposed pipeline for speech event extraction consists of several key components, as illustrated in Figure 1. The pipeline follows a modular approach, allowing for component-level evaluation and optimization.

4.1 ASR Transcription

The first step in our pipeline involves transcribing speech data into text. We experimented with two state-of-the-art ASR systems for this purpose, `Whisper large-v3` and `Canary 1b`. Both systems were used with their default configurations to transcribe the audio data. The resulting transcripts served as input for subsequent steps in the pipeline.

4.2 Event Presence Classification

A significant challenge in event extraction is distinguishing between segments that contain events and those that do not. Preliminary experiments revealed that applying LLMs directly to all transcripts resulted in numerous false positives, where models identified events, that were not annotated in the dataset.

To address this challenge, we implemented a classification step to determine whether a given transcript is likely to contain an event. We employed three different classification methods:

- Rule-based approach: This method flagged instances containing trigger words identified in the training set. We compiled a lexicon of trigger words based on the training data and used this to identify potential event-containing segments.

[1] https://xllms.github.io/SpeechEE/

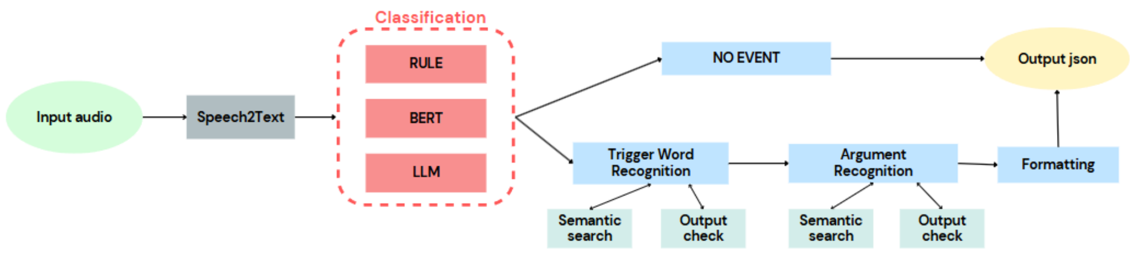

Figure 1: Overview of the SpeechEE Pipeline

- BERT-based classifier: We fine-tuned a BERT model on text embeddings to classify segments as either containing events or not. The model was trained on the transcribed training data with binary labels indicating event presence.

- LLM-based classification: We prompted OpenAI's `o1-mini` model to classify the presence or absence of events in transcripts. The model was given a short description of the task and asked to determine whether a given transcript contained an event.

For the BERT-based classification, we used the `all-MiniLM-L6-v2` sentence transformer model[2]. The training ran for 5 epochs, with a learning rate of $2e^{-5}$, batch size of 16, choosing the best model based on recall, as the goal was to have as few false negatives as possible.

To enhance classification reliability, we only considered transcripts in which all three models agreed on the presence of an event. This approach effectively reduced false positives during the classification stage. Table 1 presents the agreement matrix among the models, with bolded cells indicating cases where the filtering mechanism identified likely event presence.

Rule	BERT	LLM: NO	LLM: YES
NO	NO	258	27
	YES	61	39
YES	NO	17	27
	YES	19	**228**

Table 1: Agreement table between the three systems.

[2]`https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2`

4.3 Trigger Word Recognition

For segments classified as likely containing events, the next step involved identifying and classifying trigger words. Trigger words are specific words or phrases that signal the occurrence of an event. These events also have an event type, which needs to be recognized based on the context.

We evaluated three different LLMs for this task: `Llama3-8B`, `GPT-4o-mini`, and OpenAI's `o1-mini`. We deployed `Llama3-8B` locally on two NVIDIA GeForce RTX 2080 Ti GPUs, while the two OpenAI models were accessed via OpenAI's Batch API. Each model was prompted to extract trigger words from the transcript and classify them into predefined event categories based on the ACE2005 ontology. The prompt included a description of the task, examples of trigger words for different event types, and the transcript to be analyzed.

During our experiments, we observed that the Llama model occasionally produced outputs that did not comply with the expected format or failed to identify trigger words correctly. To address this issue, we implemented an automated verification step that checked the output format and re-executed queries when necessary to ensure consistency. This step was omissible for the OpenAI models.

4.4 Semantic Search-Enhanced Few-Shot Learning

A key component in our approach is the use of semantic search to dynamically select the most relevant examples for few-shot learning. As there are 33 classes, each with multiple argument types, even showing one example from each case would result in a very long prompt. Therefore, rather than using a fixed set of examples for all queries, we implemented a system that selected examples based on their semantic similarity to the current transcript. This retrieval-augmented few-shot approach increases contextual relevance and allows the model to better adapt to domain-specific nuances. By coupling LLMs with semantic retrieval, we aim to improve robustness and generalization in this complex setting.

The process looks like the following:

1. We created embeddings for all training examples using a the `all-MiniLM-L6-v2` sentence transformer model.

2. For each new transcript, we generated an embedding using the same model.

3. We retrieved the top ten most similar examples to the new transcript using the FAISS library (Douze et al., 2025).

4. We added these examples to the prompt (Appendix A.1).

This approach ensures that the LLM receives the most relevant and informative examples for each specific transcript, thereby improving its ability to identify and classify trigger words accurately.

4.5 Argument Extraction

Following trigger identification, the next step involved extracting event arguments and assigning roles. Event arguments are entities that participate in the event, and their roles define their relationship to the event (e.g., Agent, Entity, Place).

Similar to the trigger recognition phase, we used LLMs to extract arguments and assign roles (Appendix A.2). The prompt for this task included the transcript, the identified trigger word and event type, and examples of argument extraction for similar event types. We again employed semantic search to select the most relevant examples for few-shot learning, focusing on examples similar to the current transcript. Also we provided the dictionary of the possible argument types for each event type in the prompt.

4.6 Post-Processing

The final stage of our pipeline involved post-processing the extracted information to ensure uniform output formatting. We used an additional LLM call (Appendix A.3) to format the extracted events, triggers, and arguments into a structured JSON format consistent with the dataset specifications. This was necessary, because sometimes the models included the transcripts themselves in the output, or labeled the keys differently.

This step also included validation checks to ensure that the output met the expected schema and that all required fields were present. In cases where the output did not meet these requirements, additional LLM calls were made to correct and complete the information.

5 Results

We evaluated our event extraction pipeline using two primary metrics, following the evaluation protocol outlined by (Wang et al., 2024):

- **Trigger Classification (TC):** This metric assesses whether both the predicted event type and trigger span match the ground truth exactly. A prediction is considered correct only if both components align perfectly.

- **Argument Classification (AC):** This stricter metric evaluates the correctness of the predicted argument mention, its semantic role, and the associated event type, requiring full agreement across all elements.

For both tasks, we report precision (P), recall (R), and F1-score (F1), with the F1-score serving as the primary measure of overall performance. Table 2 summarizes the evaluation results across the three LLMs: `Llama3-8B`, `GPT-4o-mini`, and `o1-mini`. The ASR system used is indicated in parentheses—Whisper (W) or Canary (C). For `Llama3-8B` and `GPT-4o-mini`, all three stages of the pipeline utilized the respective models. In the case of `o1-mini`, however, the final formatting step was performed using `GPT-4o-mini`, as it achieved perfect results for this task, eliminating the need for the more expensive reasoning model.

Model	TC-R	TC-P	TC-F1	AC-R	AC-P	AC-F1
Llama3-8B (W)	24.1	57.6	33.9	11.2	18.3	13.9
GPT-4o-mini (W)	36.6	67.4	47.4	17.0	24.3	20.0
o1-mini (W)	59.2	65.9	62.4	26.9	27.1	27.1
Llama3-8B (C)	24.5	52.8	33.5	11.5	17.1	13.7
GPT-4o-mini (C)	36.8	65.5	47.1	16.8	23.2	19.5
o1-mini (C)	**60.8**	**66.0**	**63.3**	**28.0**	**27.6**	**27.8**

Table 2: Precision, recall, and F1-scores (%) for Trigger Classification (TC) and Argument Classification (AC).

The performance hierarchy among the models is consistent and clear: `o1-mini` substantially outperforms both `GPT-4o-mini` and `Llama3-8B` across all evaluation metrics. In TC, `o1-mini` achieved an F1-score of 63.3%, surpassing `GPT-4o-mini` by over 16 percentage points and `Llama3-8B` by nearly

30 points. Interestingly, while precision scores between o1-mini and GPT-4o-mini were relatively close, the major difference arose from recall, suggesting that GPT-4o-mini adopted a more conservative prediction strategy, prioritizing precision at the expense of coverage.

As anticipated, performance across all models declined on the more demanding AC task. Nevertheless, o1-mini again demonstrated a clear advantage, achieving an AC F1-score of 27.8%, more than double that of Llama3-8B and significantly ahead of GPT-4o-mini. Moreover, o1-mini maintained a balanced trade-off between precision and recall, whereas the other two models exhibited weaker recall, often failing to capture all valid argument mentions even when precision remained reasonable.

The consistent superiority of o1-mini across both tasks underscores the potential of specialized reasoning models in information extraction, particularly in achieving a more balanced and robust performance across precision and recall.

Regarding the choice of ASR system, switching between Whisper and Canary resulted in only minor differences (within 1% F1-score). Given the inherent nondeterminism of LLM outputs—even with identical inputs—the observed variations could stem even from stochastic model behavior rather than from the ASR systems themselves. Notably, while Llama3-8B and GPT-4o-mini performed slightly better with Whisper, o1-mini achieved marginally superior results with Canary. A more comprehensive study, involving multiple evaluation runs per prompt, would be required to draw stronger conclusions about ASR influence.

As a point of reference, (Wang et al., 2024) reported results on the ACE2005-EN+ dataset, achieving an F1-score of 61.1% for TC and 23.2% for AC. While our dataset is a modified version of ACE2005-EN+, it is not identical, and thus direct comparisons should be made cautiously. Nevertheless, our pipeline outperforms these baselines, particularly in AC, where we observe a notable improvement. This advancement likely stems from the incorporation of large language models after the transcription step, enabling more semantically coherent interpretations of spoken input instead of bypassing transcription entirely.

6 Ablation

To substantiate our initial observation that LLMs tend to produce a considerable number of false positives, we conducted an ablation study evaluating all three models under various classification configurations. The results, presented in Table 3 and Table 4, affirm the value of the classification component. We denote *one+* and *two+* to indicate that at least one or two of the three models classified the instance as containing an event, respectively.

Model	**without**	**Rule**	**BERT**	**o1-mini**	**one+**	**two+**	**three**
Llama3-8B	27.4	32.4	32.0	32.7	30.6	33.0	33.9
4o-mini	43.8	47.1	45.8	45.1	44.7	46.1	47.4
o1-mini	58.8	60.7	61.8	60.3	59.8	62.4	63.4

Table 3: F1-scores (%) on TC under different classification criteria.

When applying a single criterion, each model demonstrated optimal performance with a different method: o1-mini performed best with the *BERT-based* classifier, while 4o-mini and Llama3-8B yielded higher scores with the *Rule-based* and *LLM-based* classifiers, respectively. Although relying on the *one+* filtering resulted in marginally lower performance than the best individual setups, it still outperformed the baseline without classification. Notably, aggregating predictions with *two+* and *three* led to consistent F1-score improvements over the standalone classifiers.

Model	**without**	**Rule**	**BERT**	**o1-mini**	**one+**	**two+**	**three**
Llama3-8B	13.2	13.4	15.4	15.4	14.7	15.6	13.9
4o-mini	19.2	19.7	20.0	20.0	19.6	20.1	20.0
o1-mini	25.1	25.2	26.8	26.0	25.5	26.9	27.8

Table 4: F1-scores (%) on AC under different classification criteria.

In the AC task, the outcomes are more nuanced, indicating that strategies yielding gains in TC do not always translate to AC. For instance, while 4o-mini achieved the best TC performance with the *Rule-based* method (when considering single criterion), it was outperformed by other approaches in AC. Nonetheless, similarly to TC, both *two+* and *three* yielded improvements over the individual classifiers. However, unlike in TC, the *three* criterion did not universally result in the best performance, benefitting only the o1-mini model.

These consistent enhancements across both tasks confirm the effectiveness of the classification step in reducing false positives and improving overall model performance.

7 Discussion

Our experimental results demonstrate that a well-designed pipeline approach can achieve performance comparable or even superior to state-of-the-art end-to-end models for speech event extraction. This finding is significant as it challenges the assumption that end-to-end approaches necessarily outperform pipeline methods for complex tasks. Our results suggest that by leveraging advanced LLMs and intelligent example selection strategies, pipeline approaches can mitigate traditional weaknesses such as error propagation.

Moreover, this approach offer several advantages over end-to-end models, including modularity and interpretability, in the sense that the output of each stage can be examined and is humanly interpretable.

The minimal performance difference observed when comparing Whisper and Canary suggests that transcript quality variations within a certain threshold have limited impact on event extraction outcomes. Both Whisper and Canary have their own characteristics, when it comes to the transcription of names or cities, which are crucial in event extraction. The fact that this did not make too much of a difference, is encouraging for real-world applications, where perfect transcription cannot be guaranteed. However, it is important to note that the performance gap might be more pronounced when comparing with lower-quality ASR systems or when processing audio with significant noise, accents, or other challenging characteristics.

The substantial performance differences observed between the three LLMs highlight the critical role of LLM selection in event extraction efficacy. Model size seemed to be a factor, with the larger proprietary models clearly outperforming the locally runnable Llama model. The reasoning capability also largely seems to help the task, but to make these claims more confidently, a more thorough research would be needed across several models.

During development, we also observed that in several cases, the false positives produced by the LLMs were remarkably close to genuine events that should have been annotated. This suggests that, in some instances, the models may have correctly identified events that were inadvertently missed during the original annotation process.

8 Conclusion

This research presents an LLM-driven pipeline for speech event extraction that achieves performance comparable to state-of-the-art end-to-end models. Our approach combines ASR-generated transcripts with semantic search-enhanced few-shot learning to create a modular and interpretable framework for identifying events and their arguments from spoken language. By dynamically selecting examples based on semantic similarity to the current input, our approach ensures that the LLM receives the most relevant and informative context for each specific case. This is particularly important for event extraction, where different event types have distinct patterns, trigger words, and argument structures. The effectiveness of this approach suggests that similar techniques could be beneficial for other complex NLP tasks where pattern recognition plays a crucial role and where diverse examples exist in the training data.

The choice of ASR system showed limited impact on extraction performance, suggesting robustness to transcript quality variations within a reasonable range. However, LLM selection plays a critical role in event extraction efficacy, with larger, more capable models achieving significantly better results.

Several promising directions for future research emerge from this work, including hybrid approaches exploring methods that integrate transcript-based and direct audio-based features that could potentially combine the strengths of pipeline and end-to-end approaches. Our approach does not leverage speech-related cues present in the audio, which could potentially enhance performance if incorporated. With prompt engineering, the current prompting strategies could be further refined and may improve LLM performance without requiring additional computational resources. Another area could be exploring methods to reduce computational requirements, such as model distillation or selective component invocation.

This work advances spoken language understanding by demonstrating that modular pipelines can rival end-to-end models through strategic integration of LLMs and retrieval mechanisms. By decoupling transcription from extraction while maintaining cross-component optimization potential, our framework offers a practical pathway for deploying speech event extraction tools.

Acknowledgments

We thank the organizers of the SpeechEE shared task for providing the dataset and evaluation framework that enabled this study.

This paper benefited from the use of generative AI tools in accordance with the ACL Policy on AI Writing Assistance. Specifically, large language models were used for assistance in paraphrasing and polishing the original content. All such generated content was carefully reviewed and revised by the authors, and relevant citations were added where appropriate. The ideas and scientific contributions presented in this paper are entirely the authors own. The authors bear full responsibility for the accuracy and originality of the work.

Limitations

First, although the classification step successfully reduces false positives, it introduces an additional layer of complexity and latency into the pipeline. In real-time applications, this could pose practical constraints.

Second, although Whisper and Canary represent state-of-the-art multilingual ASR models, their performance varies considerably across languages, limiting generalizability in multilingual settings. This challenge is further compounded by the few-shot prompting technique, which is also expected to yield lower performance for low-resource languages due to limited linguistic and contextual coverage in training data.

Third, although semantic search-enhanced few-shot prompting improves LLM performance, it increases computational costs due to the need for embedding comparisons and dynamic prompt construction. This makes the system more resource-intensive, especially for large-scale or low-latency deployments.

Fourth, the argument classification task remains challenging, with overall performance still low despite LLM assistance. This is likely due to the difficulty of correctly identifying multiple, diverse argument roles within spoken inputs. Improving argument role assignment—especially for less frequent event types—requires further attention, potentially through the use of more structured reasoning or task-specific tuning.

Finally, while the `o1-mini` model consistently outperformed the other models in our experiments, it is a proprietary model, limiting reproducibility and potentially raising cost or accessibility concerns. Our pipeline's dependency on API-based models also poses challenges for deployment in privacy-sensitive or resource-constrained environments.

Future work should explore more efficient alternatives for few-shot prompting, more robust handling of ASR errors, and ways to make the pipeline more lightweight and adaptable to diverse real-world settings.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8, Sydney, Australia. Association for Computational Linguistics.

Daniel M. Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. 1997. Nymble: a high-performance learning name-finder. In *Fifth Conference on Applied Natural Language Processing*, pages 194–201, Washington, DC, USA. Association for Computational Linguistics.

Hu Cao, Jingye Li, Fangfang Su, Fei Li, Hao Fei, Shengqiong Wu, Bobo Li, Liang Zhao, and Donghong Ji. 2022. OneEE: A one-stage framework for fast overlapping and nested event extraction. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1953–1964, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Brian Chen, Xudong Lin, Christopher Thomas, Manling Li, Shoya Yoshida, Lovish Chum, Heng Ji, and Shih-Fu Chang. 2021. Joint multimedia event extraction from video and article. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 74–88, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 167–176, Beijing, China. Association for Computational Linguistics.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2025. The faiss library. *Preprint*, arXiv:2401.08281.

Hao Fei, Shengqiong Wu, Jingye Li, Bobo Li, Fei Li, Libo Qin, Meishan Zhang, Min Zhang, and Tat-Seng Chua. 2023. Lasuie: Unifying information extraction with latent adaptive structure-aware generative language model. *Preprint*, arXiv:2304.06248.

Hao Fei, Meishan Zhang, Min Zhang, and Tat-Seng Chua. 2024. Xnlp: An interactive demonstration system for universal structured nlp. *Preprint*, arXiv:2308.01846.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Jingqi Kang, Tongtong Wu, Jinming Zhao, Guitao Wang, Guilin Qi, Yuan-Fang Li, and Gholamreza Haffari. 2024. Towards event extraction from speech with contextual clues. *Preprint*, arXiv:2401.15385.

Manling Li, Alireza Zareian, Qi Zeng, Spencer Whitehead, Di Lu, Heng Ji, and Shih-Fu Chang. 2020. Cross-media structured common space for multimedia event extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2557–2568, Online. Association for Computational Linguistics.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797, Uppsala, Sweden. Association for Computational Linguistics.

Xiao Liu, Zhunchen Luo, and Heyan Huang. 2018. Jointly multiple events extraction via attention-based graph information aggregation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1247–1256, Brussels, Belgium. Association for Computational Linguistics.

Meta. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 300–309, San Diego, California. Association for Computational Linguistics.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 365–371, Beijing, China. Association for Computational Linguistics.

Thien Huu Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press.

OpenAI. 2024a. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

OpenAI. 2024b. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

OpenAI. 2024c. Openai o1 system card. *Preprint*, arXiv:2412.16720.

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, Rishita Anubhai, Cícero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. 2021. Structured prediction as translation between augmented natural languages. *ArXiv*, abs/2101.05779.

Krishna C. Puvvada, Piotr Żelasko, He Huang, Oleksii Hrinchuk, Nithin Rao Koluguri, Kunal Dhawan, Somshubra Majumdar, Elena Rastorgueva, Zhehuai Chen, Vitaly Lavrukhin, Jagadeesh Balam, and Boris Ginsburg. 2024. Less is more: Accurate speech recognition & translation without web-scale data. *Preprint*, arXiv:2406.19674.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust speech recognition via large-scale weak supervision. In *International Conference on Machine Learning*.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Neural Information Processing Systems*.

Bin Wang, Meishan Zhang, Hao Fei, Yu Zhao, Bobo Li, Shengqiong Wu, Wei Ji, and Min Zhang. 2024. Speechee: A novel benchmark for speech event extraction. In *ACM Multimedia*.

Shengqiong Wu, Hao Fei, Leigang Qu, Wei Ji, and Tat-Seng Chua. 2024. Next-gpt: Any-to-any multimodal llm. *Preprint*, arXiv:2309.05519.

Jianye Xie, Yulan Zhang, Huaizhen Kou, Xiaoran Zhao, Zhikang Feng, Lekang Song, and Weiyi Zhong. 2025. A survey of the application of neural networks to event extraction. *Tsinghua Science and Technology*, 30(2):748–768.

Bishan Yang and Tom M. Mitchell. 2016. Joint extraction of events and entities within a document context. In *Proceedings of the 2016 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 289–299, San Diego, California. Association for Computational Linguistics.

Ningyu Zhang, Luoqiu Li, Xiang Chen, Shumin Deng, Zhen Bi, Chuanqi Tan, Fei Huang, and Huajun Chen. 2022. Differentiable prompt makes pre-trained language models better few-shot learners. *Preprint*, arXiv:2108.13161.

A Appendix

A.1 Trigger Recognition Prompt

```
{"role": "system", "content":
 "Your job is to extract trigger words signaling events in a text, and classify its event type."},
{"role": "user", "content":
 "From the following TEXT, please extract the event type and its trigger word. It is a transcript of
 an audio, so there may be some mistakes.
 The possible event types are: [<event type list>].
 It is possible there are no events in the text.
 Below are examples demonstrating the required output format and some useful hints.
 Do not return the transcript, only the trigger word and event type."},
{"role": "user", "content": "TEXT: <text_input>"},
{"role": "user", "content": "EXAMPLES: <few-shot examples>"},
```

A.2 Argument Recognition Prompt

```
{"role": "system", "content":
 "Your job is to extract arguments for events in a text, and classify their role in that event."},
{"role": "user", "content":
 "From the following TEXT, please extract event arguments (usually one word or a name) and their role.
 It is a transcript of audio, so there may be mistakes.
 Use the provided event schema: <event schema>.
 An event may have no arguments.
 Examples are provided to guide selection and format."},
{"role": "user", "content": "TEXT: <text_input>, EVENT TYPE(s): <event types>"},
{"role": "user", "content": "EXAMPLES: <few-shot examples>"},
```

A.3 Post-processing Prompt

```
{"role": "system", "content":
 "Your job is to extract a JSON-like output from the end of a string. Only return the JSON."},
{"role": "user", "content":
 "From the following TEXT, extract data in the format of the example.
 If multiple triggers exist, return one entry per trigger.
 Transcriptions are unnecessary. Return JSON only."},
{"role": "user", "content": "TEXT: <text_input>"},
{"role": "user", "content":
 "EXAMPLE:
  [
    {
      "trigger": "deploy",
      "type": "Transport",
      "arguments": [
        {"name": "soldiers", "role": "Artifact"},
        {"name": "region", "role": "Destination"}
      ]
    }
  ]"},
```