

A Fully Automated Pipeline for Conversational Discourse Annotation: Tree Scheme Generation and Labeling with Large Language Models

Kseniia Petukhova, Ekaterina Kochmar

Mohamed bin Zayed University of Artificial Intelligence
{kseniia.petukhova, ekaterina.kochmar}@mbzuai.ac.ae

Abstract

Recent advances in Large Language Models (LLMs) have shown promise in automating discourse annotation for conversations. While manually designing tree annotation schemes significantly improves annotation quality for humans and models, their creation remains time-consuming and requires expert knowledge. We propose a fully automated pipeline that uses LLMs to construct such schemes and perform annotation. We evaluate our approach on speech functions (SFs) and the Switchboard-DAMSL (SWBD-DAMSL) taxonomies. Our experiments compare various design choices, and we show that frequency-guided decision trees, paired with an advanced LLM for annotation, can outperform previously manually designed trees and even match or surpass human annotators while significantly reducing the time required for annotation. We release all code and resultant schemes and annotations to facilitate future research on discourse annotation: <https://github.com/Kpetyxova/autoTree>.

1 Introduction

Discourse analysis is essential in NLP tasks like dialog management, generation, summarization, and emotion recognition (Liang et al., 2020; Chen et al., 2021; Shou et al., 2022). Traditionally, discourse annotation depends on manual expert labeling, which is costly and time-consuming. LLM-based annotation presents a promising alternative, enhancing speed, consistency, and cost-effectiveness (Gillardi et al., 2023; Hao et al., 2024). However, challenges such as biases and domain limitations necessitate careful prompt design and evaluation.

In Ostyakova et al. (2023), the authors explored using ChatGPT to automate discourse annotation for labeling chit-chat dialogs using the speech functions (SFs) taxonomy (Eggins and Slade, 2004). SFs categorize communicative acts in dialog, capturing speaker intentions and interactions in a hierarchical structure to analyze conversational flow



Figure 1: Example of speech function structure (Ostyakova et al., 2023).

(see Figure 1 for an example and Appendix A for the full label set). Ostyakova et al. (2023) conducted three sets of experiments: (1) Direct Annotation with an LLM assigning labels from a pre-defined list of SFs; (2) Step-by-Step Scheme with Intermediate Labels with an LLM selecting labels progressively from broad to specific categories; and (3) Complex Tree-Like Scheme with Yes/No Questions, using a complex tree-like annotation pipeline originally designed for crowdsourced annotation. Since *breaking a multi-label selection task into smaller sub-tasks using a tree structure* has improved human performance in complex discourse annotation (Scholman et al., 2016), the authors hypothesized that the same approach could enhance LLM-based annotation. Prior research also suggests that guiding models with tree-structured prompts significantly improves performance (Yao et al., 2024).

Ostyakova et al. (2023) found that the Tree-Like Scheme approach enhances LLM accuracy, achieving near-human performance, and suggested that LLMs could serve as a “silver standard” for annotation. However, newer and more powerful LLMs have been released since this work was published. These developments not only enhance the potential of LLMs to be used for annotation but also open the door to automating the creation of tree-like schemes,¹ enabling a fully automated pipeline. This is especially valuable for large taxonomies, such as Intelligent PAL (Morrison et al., 2014) or

¹We will refer to them as tree schemes.

TEACh-DA (Gella et al., 2022) for task-oriented systems, where manually creating tree schemes is complex and time-intensive.

This work automates tree schemes creation, making them usable for annotation by crowd-sourced workers and LLMs. A tree scheme is a decision tree that classifies dialog utterances through a series of questions, which can be binary or non-binary, using yes/no or open-ended formats. An example of a tree generated using the pipeline proposed in this work can be found in Appendix B.

2 Related Work

Discourse Analysis Researchers analyze discourse structures to improve dialog understanding and management, focusing on pragmatics and speaker intent. One of the key frameworks is the Dialog Act (DA) Theory (Jurafsky et al., 1998), which assigns pragmatic labels to utterances.

The SWBD-DAMSL scheme (Jurafsky et al., 1997a), initially created for casual conversations and widely applied to task-oriented systems, classifies dialog acts into 42 classes. Taxonomy of speech functions (Eggins and Slade, 2004) offers a hierarchical annotation approach, integrating DA principles and relational analysis.

Dialog acts are beneficial in task-oriented dialog agents. For example, Gella et al. (2022) introduce a scheme for embodied agents, improving natural language interactions and task success, and Leech and Weisser (2003) develop Speech Act Annotated Corpus (SPAAC) scheme, balancing specificity and generalizability in task-oriented dialogs.

LLMs for Discourse and Annotation

Ostykova et al. (2023) introduce a semi-automated approach for annotating open-domain dialogs using a taxonomy of speech functions and ChatGPT. Their study evaluates three methods: Direct Annotation (selecting from a complete label set), Step-by-Step (progressively narrowing choices), and Tree-Like Schemes (using hierarchical yes/no questions). The Tree-Like Scheme has performed best, particularly for rare classes, achieving high consistency when running the same annotation pipeline with ChatGPT three times (with Fleiss’ kappa of 0.83). However, expert input has remained essential for designing the annotation pipelines.

In addition, Yadav et al. (2024) and Tseng et al. (2024) explore GPT-4-based semantic annotation, emphasizing prompt design and model limitations.

Chen et al. (2024) investigate LLMs for event extraction, addressing data scarcity in fine-tuned models. Finally, Wu et al. (2024) introduce a rationale-driven collaborative framework that refines annotations through iterative reasoning, outperforming standard prompting.

3 Speech Functions Corpus

The experiments described in this work are based on the Speech Functions Corpus, a dataset of dialogs annotated with SFs. This corpus was developed by Ostykova et al. (2023), where three experts, each with at least a B.A. in Linguistics, annotated the DailyDialog dataset (Li et al., 2017) – a multi-turn casual dialog dataset – using the speech functions taxonomy (Eggins and Slade, 2004). The authors of the corpus reduced the original 45 classes proposed by Eggins and Slade (2004) to a more manageable set of 32 classes.

The tag set covers five functional dimensions: *turn management*, *topic organization*, *feedback*, *communicative acts*, and *pragmatic purposes*. While all dimensions are embedded within speech functions, they are distributed unevenly across tags, with individual speech functions incorporating between two and five dimensions. Figure 1 illustrates an example of a speech function that includes all dimensions: **React.Rejoinder.Support.Track.Clarify**. In this example: (1) **React** represents *turn management*, indicating a speaker change or a reaction to a previous utterance; (2) **Rejoinder** corresponds to *topic organization*, signifying active topic development that influences the dialog flow; (3) **Support** denotes *feedback*, showing that the speaker is supporting an interlocutor; (4) **Track** falls under *communicative acts*, identifying questions; (5) **Clarify** serves a *pragmatic purpose*, indicating a question aimed at obtaining additional information on the current conversation topic. Some SFs, however, cover fewer dimensions. For instance, **Open.Attend** represents only two: *turn management* (marking the conversation’s beginning) and *communicative acts* (a greeting).

The Speech Functions Corpus includes 64 dialogs, containing 1,030 utterances. Appendix C shows an example of an annotated dialog.

To evaluate the effect of the taxonomy size on the method proposed in this work, the existing taxonomy was converted into the following subsets:

- **Full Taxonomy:** This set includes all 32 speech functions labels from Ostykova et al.

(2023). A complete list of labels, along with their descriptions, examples, and frequency information, can be found in Appendix A.

- **The Top Level of the Taxonomy:** This subset consists of three classes corresponding to the turn management level of the speech functions taxonomy. Definitions for these labels were written manually and are detailed in Appendix D. This subset is particularly important because if the model fails to distinguish between these high-level categories, the entire hierarchical structure may be unreliable.
- **The Top Two Levels of the Taxonomy:** This subset includes six classes, representing a combination of the speech functions taxonomy’s turn management and topic organization dimensions. Manual definitions for these classes are provided in Appendix E. The motivation for analyzing this subset is similar to the previous one but with an additional level of complexity, making the classification task more challenging.
- **Top-20 Frequent Classes of the Taxonomy:** This subset comprises the 20 most frequently occurring classes in the Speech Functions Corpus. It is designed to evaluate how well the model handles frequent classes in the absence of rarer ones.

4 Framework for Tree Construction with LLMs

Traditionally, discourse annotation has been performed manually by experts or trained annotators, relying on predefined end-class descriptions. However, Scholman et al. (2016) investigates whether non-trained, non-expert annotators can reliably annotate coherence relations using a step-wise approach, which functions similarly to a decision tree. Their findings indicate that a structured step-wise method can indeed make discourse annotation more accessible to non-experts, facilitating large-scale annotation without requiring extensive training. This is achieved by using cognitively plausible primitives rather than relying on complex end labels. A similar observation is made in Ostyakova et al. (2023), further supporting the viability of this approach.

To construct an effective decision tree for annotation, it is essential to design questions that do not require expert-level knowledge of discourse but can instead be answered based on the utterance itself.

Ideally, related classes should be positioned closely within the tree, forming a hierarchical structure that reflects conceptual similarities between coherence relations. This hierarchical organization not only simplifies decision-making for annotators but also enhances consistency and reliability in annotation.

Tree Construction The pipeline for the tree construction process is illustrated in Figure 2. The core concept of the proposed algorithm is to use an advanced LLM to identify distinguishing features that allow it to divide a set of classes into two or more groups. Here, a group refers to a subset of the input classes that the model clusters together based on a shared property it identifies – framed through a classification question. For example, given the classes *Open.Demand.Fact* (requesting factual information at the beginning of a conversation), *Open.Demand.Opinion* (requesting an opinion at the beginning), *Open.Give.Fact* (providing factual information at the beginning), and *Open.Give.Opinion* (providing an opinion at the beginning), the model might divide them as *group 1*: *Open.Demand.Fact*, *Open.Demand.Opinion*, and *group 2*: *Open.Give.Fact*, *Open.Give.Opinion*. To do this, the LLM is provided with a table containing class names, definitions, and usage examples as input.

To enhance reasoning capabilities, we aim for the model to engage in an inner monologue (Zhou et al., 2024) before determining how to split the data. To achieve this, the LLM is prompted to first generate a set of questions about an utterance that can aid its understanding. Specifically, the model is instructed to formulate and answer three such reasoning question-answer pairs at the beginning of its response. Recent studies indicate that employing such self-questioning techniques helps models produce more flexible, meaningful outputs, indicating a deeper level of comprehension and reasoning (Sun et al., 2024). An example of the LLM’s reasoning output can be found in Appendix F.

After this reasoning step, the LLM generates a classification question to determine an utterance’s group, providing possible answers mapped to class groups. For example, the model might generate a classification question: *“Is this the beginning of a conversation?”* The possible answers (groups) could be: (1) *“Yes, this is the beginning of a conversation.”* and (2) *“No, the utterance continues the conversation.”* This question helps categorize utterances, splitting the taxonomy into conversa-

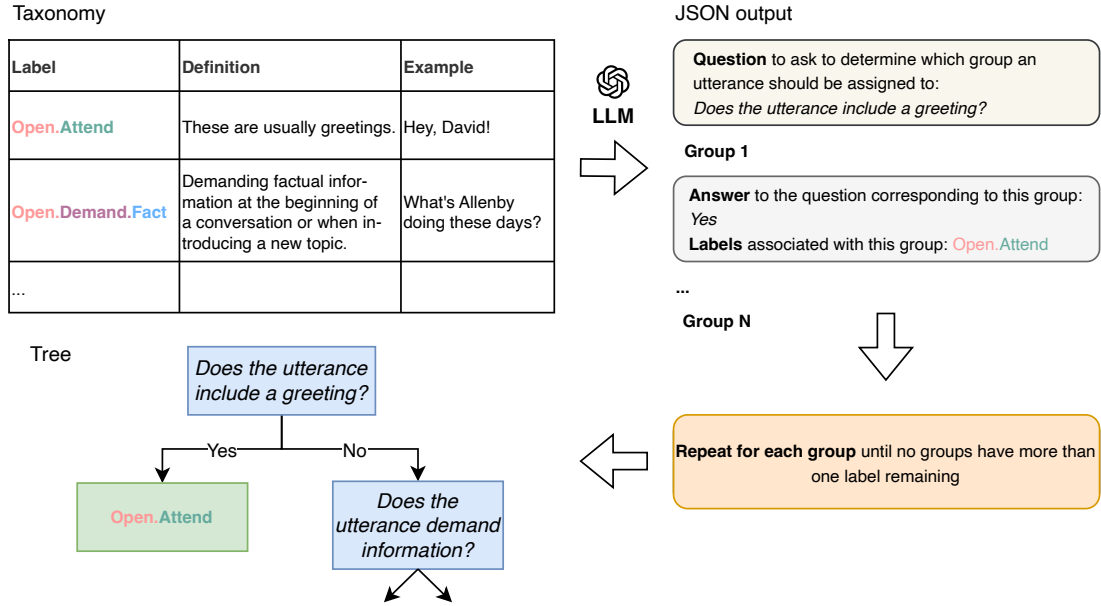


Figure 2: Pipeline for tree construction. An LLM formulates a classification question to split classes into groups, mapping possible answers to respective class groups. This process repeats recursively for created groups until all groups contain only one class. Finally, the grouped data is merged into a single tree structure in JSON format for annotation.

tion openings and other categories. The prompt template used for this process is detailed in Appendix G.

With taxonomy classes grouped, the process iterates through each group until no group contains more than one class. Once resolved, the resulting JSONs are merged into a single tree JSON for annotation. For this step, GPT-4 (gpt-4-0613) is used due to its ability to handle long contexts and generate valid JSON outputs (OpenAI, 2023). The temperature is set to 0.4, which is empirically chosen to balance creativity and reliability, allowing some variation while avoiding excessive randomness. In contrast, a temperature of 0 minimizes variability but may produce overly rigid outputs.

Annotation Using the constructed questions tree, dialogs are annotated by traversing the tree for each utterance until reaching a leaf node representing a class label. To ensure a fair comparison with the manually created tree from Ostyakova et al. (2023), the same annotation settings are applied. Specifically, the context length is set to 1, as this yielded the best results in ablation studies. The same model – ChatGPT (GPT-3.5-turbo) – is used, along with the optimal temperature of 0.9, as identified in the ablation studies. The prompt format remains con-

sistent with Ostyakova et al. (2023) and is shown in Appendix H. Additionally, open-source models’ performance is analyzed in Section 7.

Evaluation The framework’s performance is assessed by comparing the predicted labels from the annotation step to the gold references provided in Ostyakova et al. (2023), using the same 12 dialogs (189 utterances) as the authors, referred to here as the *development set*. To demonstrate generalizability across dialogs, we also evaluate it on another 12 dialogs (165 utterances) from the Speech Functions corpus, referred to as the *test set*. To further test generalizability across datasets, we evaluate it on a subset of the Switchboard Dialog Act Corpus — a dataset of telephone conversations annotated using the SWBD-DAMSL annotation scheme, as described in Section 8. The evaluation uses the same metrics as in Ostyakova et al. (2023): Weighted Precision (P_w), Weighted Recall (R_w), and Macro F1 ($F1$). Additionally, we report Weighted F1 ($F1_w$).

5 Tree Construction Approaches

Yes/No Questions vs. Open-Ended Questions The manually created tree in Ostyakova et al. (2023) is binary and composed exclusively of

yes/no questions. As the first approach explored in our work, we implement a binary yes/no question tree using the proposed framework. Additionally, we construct a binary tree with open-ended questions. The hypothesis behind this experiment is that allowing the model to generate any questions, rather than limiting it to yes/no questions, can result in a more flexible and nuanced tree for discourse annotation. The prompts used for constructing both the yes/no questions and the open-ended questions tree were developed through iterative refinement. This refinement process involved adjusting and testing prompts until they produced stable and consistent results, ensuring a valid JSON output with all necessary keys present. The final versions of these prompts are provided in Appendix I and Appendix J, respectively.

Binary vs. Non-Binary Trees Allowing the model to split nodes into multiple groups, rather than restricting it to two groups, has the potential to create a more granular and detailed annotation tree. However, this added granularity may introduce greater complexity and reduce consistency in the annotation process. This experiment is distinct from the previous one, where we only allowed binary open-ended questions and compared them to yes/no questions. As a result, both the questions and answers in that setting were more high-level, distinguishing data based on a single characteristic. In contrast, non-binary open-ended questions allow for greater granularity and specificity, enabling more nuanced differentiation within the data. Experiments described in this section examine the impact of restricting the model to binary splits compared to allowing multiple-group splits on the accuracy of the resulting annotation tree. Both the binary and non-binary trees discussed in this section use open-ended questions. A prompt that allows open-ended questions and splitting data into more than two groups is detailed in Appendix K.

Optimal Split Selection and Backtracking Inspired by Yao et al. (2024), we evaluate the impact of allowing the model to backtrack while constructing the annotation tree. In each iteration, the model generates three potential splits and assigns a score to each (the prompt used for scoring is provided in Appendix L). These splits are then evaluated using a pre-trained natural language inference (NLI)

model,² which classifies them as either contradictory, neutral, or non-contradictory. Among the non-contradictory options, the split with the highest score is selected. If the best-scoring split does not produce a viable partition, the model backtracks and evaluates the next-best option. The motivation behind this experiment is that multiple valid ways to create splits exist, and selecting the seemingly best option at each step may not always result in the most effective tree overall.

Frequency-Guided Optimal Split Selection and Backtracking Class frequency information can be used to guide the model in making splits and optimize the annotation process. In conversations, certain SFs occur more frequently than others. For instance, **Sustain.Continue.Prolong.Extend** appears 21.8% of the time, representing instances where a speaker adds information to their preceding statement. Similarly, **React.Rejoinder.Support.Track.Clarity** occurs 12% of the time, typically indicating a question aimed at obtaining additional information. Meanwhile, some SFs are relatively rare. The experiments described in this section aim to construct a decision tree that reflects the distribution of classes, making frequent classes easier and faster to reach compared to rare ones.

To achieve a frequency-guided tree, the prompt used to generate splits is modified as follows: at each step, the model is instructed to create a group containing only the most frequent class if one class is significantly more frequent than the others. The full prompt is provided in Appendix M.

6 Results & Analysis

Evaluation results for the Top Level, the Top Two Levels, and Top-20 class subsets on the development set are presented in Tables 1, 2 and 3 while the results for the complete SF taxonomy on the development and test sets are shown in Tables 4 and 5, respectively.³

Yes/No Questions vs. Open-Ended Questions

The findings indicate that *open-ended trees outperform yes/no trees across all metrics and data subsets*. However, for the complete SF taxonomy,

²<https://huggingface.co/cross-encoder/nli-deberta-v3-base>

³The best results for GPT-3.5 are highlighted in **bold**, while the second-best results are marked with an underline. The overall best results across all models are highlighted in **blue**.

Approach	P_w	R_w	$F1_w$	F1
Yes/no	0.55	0.34	0.37	0.33
Open-ended	0.68	0.63	0.61	0.63
Non-binary	0.74	0.72	0.70	0.73

Table 1: Evaluation of annotations on the **development** set using trees constructed through different methods for the **Top Level** of the SF taxonomy, with GPT-3.5 used for annotation.

Approach	P_w	R_w	$F1_w$	F1
Yes/no	0.49	0.22	0.26	0.20
Open-ended	0.70	0.65	0.65	0.43
Non-binary	0.60	0.48	0.45	0.48
W/ split selection	<u>0.67</u>	<u>0.62</u>	<u>0.62</u>	0.60
Freq.-guided split selection	0.63	0.45	0.41	0.55
W/ split selection	0.79	0.78	0.78	0.80
(GPT-4o for annotation)				
Freq.-guided split selection	0.80	0.78	0.78	0.80
(GPT-4o for annotation)				

Table 2: Evaluation of annotations on the **development** set using trees constructed through different methods for the **Top Two Levels** of the SF taxonomy (with GPT-3.5 used for annotation unless explicitly stated otherwise).

both yes/no and open-ended trees perform significantly worse than the manually created tree.

Binary vs. Non-Binary Trees The findings show that *allowing the model to split data into multiple groups generally outperforms restricting it to binary splits*. This performance difference is more pronounced when the number of classes is smaller. However, as the number of classes increases, the gap narrows, with weighted metrics occasionally favoring the binary approach, specifically showing (1) higher P_w and R_w for the Top Two Levels (6 classes); (2) higher R_w for the Top-20 classes; (3) higher P_w for the complete taxonomy (33 classes) on the test set. Nevertheless, higher macro metrics for non-binary trees suggest improved performance for smaller and less frequent classes. Based on these results, further experiments will allow the model to split data into more than two groups.

Optimal Split Selection and Backtracking For the Two-Level subset, performance improves compared to the approach that does not use split selection and backtracking. In the Top-20 subset, only the $F1$ metric shows an increase, indicating better performance for less frequent classes. For the complete SF taxonomy, performance on the development set remains comparable to the approach without split selection and backtracking, while on the test set, the $F1$ score is higher.⁴

⁴Results for the Top Level subset are unavailable, as these trees have only one level.

Approach	P_w	R_w	$F1_w$	F1
Yes/no	0.36	0.18	0.16	0.14
Open-ended	0.42	<u>0.40</u>	<u>0.37</u>	0.19
Non-binary	<u>0.51</u>	0.26	0.22	0.20
W/ split selection	0.39	0.36	0.34	0.27
Freq.-guided split selection	0.62	0.66	0.62	0.35
W/ split selection	0.56	0.55	0.52	0.37
(GPT-4o for annotation)				
Freq.-guided split selection	0.70	0.69	0.67	0.41
(GPT-4o for annotation)				

Table 3: Evaluation of annotations on the **development** set using trees constructed through different methods for the **Top-20** classes of the SF taxonomy (with GPT-3.5 used for annotation unless explicitly stated otherwise).

The lack of improvement in the complete SF taxonomy stems from the absence of an optimal split at the initial step, causing error propagation throughout the taxonomy. Specific issues include: (1) The model misassigned the **Open.Attend** category (which represents greetings at the beginning of a conversation) to the branch “The utterance involves a request for information,” and (2) It grouped all **React** classes under the branch “The dialog utterance involves a response to a request for information,” which does not accurately represent them. This misclassification often led the annotation model to misroute utterances to the **Sustain** branch instead, resulting in unreliable annotations. These issues are not due to the updated approach but rather to the fundamental challenge of generating meaningful splits when dealing with many classes. These errors propagate throughout the taxonomy if the model fails to establish a strong initial split.

Frequency-Guided Optimal Split Selection and Backtracking

The results indicate that the metrics for the Two-Level subset have decreased compared to the approach without frequency guidance (Table 2), while they have significantly increased for the Top-20 subset (Table 3). For the complete SF taxonomy, the metrics remained at the same level on the development set (Table 4) but improved on the test set (Table 5).

Manual analysis revealed that during the annotation step, the model frequently selected incorrect paths, often defaulting to upper-level classes. This behavior was especially prevalent in the most frequent class, **Sustain.Continue.Prolong.Extend**. Despite this, the tree structure appears logical, and the root question is straightforward: *Does the dialog utterance provide supplementary or contradictory information to the previous statement by the same speaker?* The response options for this question are:

Approach	P _w	R _w	F1 _w	F1
Manually created tree from Ostyakova et al. (2023) (crowdsourced annotation of the full dataset)	0.71	0.60	-	0.46
Manually created tree from Ostyakova et al. (2023) (ChatGPT for annotation)	0.67	0.62	-	0.43
Yes/no	0.37	0.25	0.24	0.13
Open-ended	0.38	0.23	0.21	0.23
Non-binary	0.39	0.34	0.31	0.16
W/ split selection	0.36	0.38	0.35	0.16
Freq.-guided split selection	0.31	0.43	0.34	0.19
W/ split selection (GPT-4o for annotation)	0.57	0.53	0.51	0.32
Freq.-guided split selection (GPT-4o for annotation)	0.83	0.75	0.74	0.60
W/ split selection (Llama-3.1-8B-Instruct for annotation)	0.56	0.40	0.41	0.24
Freq.-guided split selection (Llama-3.1-8B-Instruct for annotation)	0.45	0.48	0.41	0.30
W/ split selection (Mistral-7B-Instruct-v0.3 for annotation)	0.48	0.50	0.54	0.31
Freq.-guided split selection (Mistral-7B-Instruct-v0.3 for annotation)	0.33	0.44	0.32	0.18

Table 4: Evaluation of annotations on the **development** set (except for the first line) using trees constructed through different methods (with GPT-3.5 used for annotation unless explicitly stated otherwise).

Approach	P _w	R _w	F1 _w	F1
Yes/no	0.20	0.21	0.17	0.12
Open-ended	0.35	0.22	0.21	0.16
Non-binary	0.31	0.31	0.27	0.16
W/ split selection	0.48	0.18	0.16	0.17
Freq.-guided split selection	0.43	0.42	0.37	0.23
Freq.-guided split selection (GPT-4o for annotation)	0.77	0.68	0.67	0.46

Table 5: Evaluation of annotations on the **test** set using trees constructed through different methods (with GPT-3.5 used for annotation unless explicitly stated otherwise).

(1) “Dialog utterances that provide supplementary or contradictory information to the previous statement by the same speaker”; (2) “Dialog utterances that do not provide supplementary or contradictory information to the previous statement by the same speaker”. While the question is specific, emphasizing conditions about the same speaker and the addition of information, the model often ignored these requirements. In numerous cases, the first response option was incorrectly assigned, even at the start of a conversation.

The distinguishing characteristic of this tree is the heightened granularity and specificity of the questions and labels, with each step designed to determine whether the utterance fits a particular class using a single, targeted question. To assess whether the frequency-guided tree presents too significant a challenge for the GPT-3.5 model and whether it might perform better with a more advanced model, GPT-4o ([Hurst et al., 2024](#)) was used during the annotation step (see Section 6.1). This approach allowed for a direct comparison between GPT-4o and GPT-3.5. Section 6.2 also examines whether the observed differences in metrics are the same for trees created without frequency guidance.

6.1 GPT-4o for Annotation

Evaluation results comparing annotations by GPT-4o and GPT-3.5 on trees constructed using frequency-guided optimal split selection and backtracking for the Two-Level and Top-20 class subsets are presented in Tables 2 and 3, while Tables 4 and 5 show the results for the complete SF taxonomy on the development and test sets, respectively. We note that not only are the differences in metrics highly pronounced, but also **the P_w, R_w, and F1 scores for annotations on both the development and test sets for the complete SF taxonomy surpass those obtained for the entire dataset annotated by crowdsourcers in [Ostyakova et al. \(2023\)](#)**. This finding underscores that the proposed Frequency-Guided Optimal Split Selection approach, combined with an advanced LLM for annotation, may both outperform manually constructed trees and improve traditional human-driven annotation processes.

6.2 Do Annotation Gaps Persist in Non-Frequency-Guided Approaches?

This section examines whether the substantial differences observed between annotations generated by GPT-4o and GPT-3.5 also occur in the non-frequency-guided optimal split selection and backtracking approach.

Tables 2, 3, 4 and 5 presents performance metrics for annotations produced using GPT-3.5 and GPT-4o with the optimal split selection algorithm, both with and without frequency guidance. For the frequency-guided approach, the performance gap between GPT-3.5 and GPT-4o becomes more pronounced as the number of classes increases. In contrast, there is no substantial difference in performance for the non-frequency-guided approach

Level	Metric	Split-Sel.		Freq-Guided	
		GPT-3.5	GPT-4o	GPT-3.5	GPT-4o
1	Acc.	94.71	74.07	85.71	98.41
	Err%	21.28	55.06	25.00	6.38
2	Acc.	93.85	82.14	67.74	95.12
	Err%	23.40	28.09	46.30	17.02
3	Acc.	87.35	86.73	78.10	80.13
	Err%	44.68	16.85	21.30	65.96
4	Acc.	97.18	100.00	72.00	98.48
	Err%	4.26	0.00	6.48	2.13
5	Acc.	86.36	100.00	88.89	88.24
	Err%	6.38	0.00	0.93	8.51
6	Acc.	-	-	100.00	100.00
	Err%	-	-	0.00	0.00

Table 6: Accuracy and error percentages (relative to the total number of errors) at each tree depth level produced by GPT-3.5 and GPT-4o, using trees constructed with frequency-guided and non-frequency-guided split selection algorithms.

when switching from GPT-3.5 to GPT-4o. This underscores the clear advantage of the frequency-guided approach when paired with a more advanced model.

6.3 Error Distribution across Depth Levels

Table 6 decomposes accuracy by the depth at which an annotator model first diverges from the gold path. Several clear trends emerge. First, the use of frequency-guided optimal split selection and backtracking in combination with GPT-4o for annotation nearly eliminates root-level mistakes: the annotator model correctly selects the branch in 98% of the cases. A similar trend is evident at the second level of the tree, where using GPT-4o with a frequency-guided tree yields 95% accuracy, significantly outperforming the 68% accuracy observed with GPT-3.5 in the same context.

For GPT-4o on frequency-guided trees, errors primarily propagate downward, concentrating at the third level of the tree, where 66% of all errors occur. Most third-level errors arise from misclassifying *React.Rejoinder.Support.Response.Resolve* (response that provides the information requested in the question) as *React.Respond.Confront.Reply.Disagree* (negative response). These misclassifications occur because the annotator model confuses the group label “Utterances that involve a positive or negative response to a previous statement” with “Utterances that provide the information requested in the question.” The semantic overlap between these groups indicates that the issue lies more in the class definitions rather than a serious annotation error. Another frequent source of error involves

the vague *Other utterances* group. Utterances of class *React.Respond.Support.Register* (e.g., “Yeah,” “Hmm...,” “Right”) were grouped under *Other utterances* during tree construction but later misannotated as *React.Respond.Support.Reply.Affirm* (positive answers or confirmations, e.g., “Yes”). Despite explicit instructions to avoid creating *Other* groups during the tree-creation stage, the model occasionally disregarded these instructions and included them, contributing to misclassification. Notably, fewer than 10% of errors occur beyond the third level, suggesting most challenges arise earlier in the tree.

6.4 Consistency of Tree Generation

An important aspect to evaluate is the *consistency* of the proposed framework: if the decision tree is generated multiple times, how similar or different will the resulting trees be? Table 7 shows annotation performance on the test set using three trees built with the frequency-guided split selection algorithm. The first and third runs yield nearly identical results, while the second performs slightly worse. Manual inspection confirms that all three trees are broadly similar, with the drop in the second run due to its poor handling of the frequent class – *React.Respond.Support.Develop.Extend*. The primary difference between this second tree and the others lies in its structure: it immediately separates the labels *Extend*, *Enhance*, and *Elaborate* into three distinct terminal nodes at a single step. In contrast, the other two trees first group *Enhance* and *Elaborate* together, distinguishing them from *Extend*, and only in the subsequent step split the remaining group into separate nodes. Overall, the framework demonstrates strong consistency across runs.

Run	P _w	R _w	F1 _w	F1
1	0.77	0.68	0.67	0.46
2	0.66	0.64	0.62	0.41
3	0.76	0.66	0.68	0.43

Table 7: Evaluation of annotations on the test set using trees generated across three runs with the frequency-guided optimal split selection approach. Annotations were produced by GPT-4o.

6.5 Cost Analysis

This section provides a cost analysis for creating trees for SFs and annotating data using these trees.

Generating a non-binary tree for the full SF taxonomy with GPT-4 costs approximately \$0.40 and

takes 2 minutes (max depth: 3). Annotation with GPT-4o costs \$0.24 per dialog, taking 50 seconds.

Using frequency-guided optimal split selection and backtracking, tree creation costs \$5.48 (with \$4.05 for split-candidates and \$1.43 for scoring) and takes 32 minutes. Without optimal split selection and backtracking, a frequency-guided tree costs \$1.83. In this case, the maximum tree depth is around 7. Annotation costs approximately \$0.36 per dialog and takes about 35 seconds.

For comparison, GPT-3.5 annotation with a manually created tree costs \$0.03–\$0.07 per dialog (Ostykova et al., 2023). Crowdsourced annotation costs \$0.12–\$0.22 per dialog, averaging 29 minutes per annotation. While the authors do not specify the time required for tree creation, assuming it exceeds half an hour is reasonable.

Based on these estimates, annotating the entire dataset (64 dialogs) using only human resources would cost approximately \$10.88 and take 31 hours plus additional time for tree creation. In contrast, our best approach, frequency-guided optimal split selection with GPT-4o, would cost around \$20.84 but reduce the total time to approximately 1 hour and 25 minutes, offering significant efficiency and quality benefits despite the higher cost.

7 Open-Source Models for Annotation

Table 4 also presents the results of using two open-source models, Mistral-7B-Instruct-v0.3⁵ and Llama-3.1-8B-Instruct,⁶ compared to the closed-source models GPT-3.5 and GPT-4o for the annotation steps with two approaches, frequency-guided and non-frequency-guided optimal split selection with backtracking, which were selected based on their strong performance with other models.

These results indicate that despite having fewer parameters than the closed-source models, both open-source models notably outperform GPT-3.5. Specifically, when using the non-frequency-guided approach, both open-source models achieve performance close to GPT-4o while markedly surpassing GPT-3.5. However, with the frequency-guided approach, Llama continues to outperform GPT-3.5 noticeably, but both models’ performance metrics fall short of GPT-4o. This trend underscores the

⁵<https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3>

⁶<https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct>

substantial performance gains achieved by combining the frequency-guided approach with GPT-4o, as discussed in Section 6.2.

8 Evaluation on Switchboard Dialog Act Corpus

To assess the generalizability of our approach, we evaluate two configurations on the SWBD-DAMSL annotation scheme (42 classes). The configurations are: (1) open-ended questions with a non-binary tree and (2) frequency-guided optimal split selection with backtracking. These configurations were selected as they represent the best non-frequency-guided and frequency-guided approaches. We use the resulting trees to annotate a randomly sampled subset of the Switchboard Dialogue Act Corpus (Jurafsky et al., 1997b; Shriberg et al., 1998; Stolcke et al., 2000),⁷ consisting of 260 utterances.⁸

Table 8 presents evaluation metrics using GPT-4o. The results confirm that the proposed framework is effective and generalizable across different taxonomies, including large-scale ones like SWBD-DAMSL, with the Frequency-Guided Optimal Split Selection achieving a Weighted F1 score of 0.61.

Approach	P _w	R _w	F1 _w	F1
Non-binary	0.63	0.47	0.48	0.18
Freq.-guided split selection	0.65	0.63	0.61	0.23

Table 8: Evaluation of dialog act annotations from the SWBD-DAMSL annotation scheme, generated by GPT-4o, on a randomly selected set of 260 utterances from the Switchboard Dialogue Act Corpus using non-binary open-ended questions tree and a frequency-guided optimal split selection tree.

9 Conclusions

We conducted experiments on generating tree schemes for discourse taxonomies using LLMs. This paper proposes a framework that supports the entire pipeline, from tree construction to dialog annotation. Our configuration with frequency-guided tree creation demonstrates that using LLMs for both tree scheme generation and annotation can yield results that surpass manual tree construction and crowdsourced annotation while significantly reducing the time required for the entire process.

⁷GPL-2.0 license

⁸We initially selected 300 utterances, but 40 were annotated as “+”, which, in this taxonomy, indicates that the utterance continues the label of the preceding one. Since these cases do not require actual annotation but rather a repetition of the previous label, we excluded them from the analysis.

Limitations

A key limitation of the proposed method is the restricted set of models that can be used. The tree creation process requires an advanced model, and the annotation step also benefits from using a more sophisticated model. Another limitation is that non-frequency-guided configurations still underperform compared to manually created trees. This highlights the importance of class frequency information in achieving optimal performance.

Potential directions for future research, motivated by the current limitations, include: (1) exploring larger open-source models for improved taxonomy generation and annotation; (2) conducting experiments on other domains, such as classroom discourse and task-oriented dialog systems; (3) incorporating human feedback to allow the model to self-correct and improve annotation accuracy; (4) enabling self-refinement of the taxonomy by adapting to new, previously unseen dialog examples; and (5) allowing the annotation step to select multiple candidate branches/labels, followed by a final evaluation step that explicitly compares the utterance against the chosen candidates' class definitions to determine the most suitable class.

Ethical Considerations

Dialog data often contains personal or sensitive information, making it essential to anonymize and handle data securely when applying the proposed approach to individual datasets. This is crucial for protecting privacy rights. Beyond this consideration, we do not anticipate any significant risks associated with this work or the use of the proposed framework.

References

- Ruirui Chen, Chengwei Qin, Weifeng Jiang, and Dongkyu Choi. 2024. Is a Large Language Model a Good Annotator for Event Extraction? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17772–17780.
- Yulong Chen, Yang Liu, Liang Chen, and Yue Zhang. 2021. DialogSum: A real-life scenario dialogue summarization dataset. *arXiv preprint arXiv:2105.06762*.
- Suzanne Eggins and Diana Slade. 2004. *Analysing casual conversation*. Equinox Publishing Ltd.
- Spandana Gella, Aishwarya Padmakumar, Patrick Lange, and Dilek Hakkani-Tur. 2022. Dialog Acts for Task-Driven Embodied Agents. *arXiv preprint arXiv:2209.12953*.
- Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. 2023. ChatGPT outperforms crowd workers for text-annotation tasks. *Proceedings of the National Academy of Sciences*, 120(30):e2305016120.
- Jing Hao, Yuxiang Zhao, Song Chen, Yanpeng Sun, Qiang Chen, Gang Zhang, Kun Yao, Errui Ding, and Jingdong Wang. 2024. Fullanno: A data engine for enhancing image comprehension of mllms. *arXiv preprint arXiv:2409.13540*.
- Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. GPT-4o system card. *arXiv preprint arXiv:2410.21276*.
- Dan Jurafsky, Elizabeth Shriberg, Barbara Fox, and Traci Curl. 1998. Lexical, prosodic, and syntactic cues for dialog acts. In *Discourse relations and discourse markers*.
- Daniel Jurafsky, Rebecca Bates, Noah Coccaro, Rachel Martin, Marie Meteer, Klaus Ries, Elizabeth Shriberg, Andreas Stolcke, Paul Taylor, and Carol Van Ess-Dykema. 1997a. Automatic detection of discourse structure for speech recognition and understanding. In *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*, pages 88–95. IEEE.
- Daniel Jurafsky, Elizabeth Shriberg, and Debra Bisasca. 1997b. Switchboard SWBD-DAMSL shallow-discourse-function annotation coders manual, draft 13. Technical Report 97-02, University of Colorado, Boulder Institute of Cognitive Science, Boulder, CO.
- Geoffrey Leech and Martin Weisser. 2003. Generic speech act annotation for task-oriented dialogues. In *Proceedings of the corpus linguistics 2003 conference*, volume 16, pages 441–446. Lancaster: Lancaster University.
- Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. Dailydialog: A manually labelled multi-turn dialogue dataset. *arXiv preprint arXiv:1710.03957*.
- Kaihui Liang, Austin Chau, Yu Li, Xueyuan Lu, Dian Yu, Mingyang Zhou, Ishan Jain, Sam Davidson, Josh Arnold, Minh Nguyen, et al. 2020. Gunrock 2.0: A user adaptive social conversational system. *arXiv preprint arXiv:2011.08906*.
- Donald Morrison, Benjamin Nye, Borhan Samei, Vivek Varma Datla, Craig Kelly, and Vasile Rus. 2014. Building an intelligent pal from the tutor. com session database phase 1: Data mining. In *Educational Data Mining 2014*.
- OpenAI. 2023. *GPT-4 Technical Report*.

- Lidiia Ostyakova, Veronika Smilga, Kseniia Petukhova, Maria Molchanova, and Daniel Kornev. 2023. Chatgpt vs. crowdsourcing vs. experts: Annotating open-domain conversations with speech functions. In *Proceedings of the 24th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 242–254.
- Merel Scholman, Jacqueline Evers-Vermeul, Ted JM Sanders, et al. 2016. A step-wise approach to discourse annotation: Towards a reliable categorization of coherence relations. *Dialogue & Discourse*, 7(2):1–28.
- Yuntao Shou, Tao Meng, Wei Ai, Sihan Yang, and Keqin Li. 2022. Conversational emotion recognition studies based on graph convolutional neural networks and a dependent syntactic analysis. *Neurocomputing*, 501:629–639.
- Elizabeth Shriberg, Rebecca Bates, Paul Taylor, Andreas Stolcke, Daniel Jurafsky, Klaus Ries, Noah Coccaro, Rachel Martin, Marie Meteer, and Carol Van Ess-Dykema. 1998. Can prosody aid the automatic classification of dialog acts in conversational speech? *Language and Speech*, 41(3–4):439–487.
- Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Marie Meteer, and Carol Van Ess-Dykema. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3):339–371.
- Guohao Sun, Can Qin, Jiamian Wang, Zeyuan Chen, Ran Xu, and Zhiqiang Tao. 2024. Sq-llava: Self-questioning for large vision-language assistant. In *European Conference on Computer Vision*, pages 156–172. Springer.
- Yu-Min Tseng, Wei-Lin Chen, Chung-Chi Chen, and Hsin-Hsi Chen. 2024. Are Expert-Level Language Models Expert-Level Annotators? *arXiv preprint arXiv:2410.03254*.
- Jianfei Wu, Xubin Wang, and Weijia Jia. 2024. Enhancing text annotation through rationale-driven collaborative few-shot prompting. *arXiv preprint arXiv:2409.09615*.
- Sachin Yadav, Tejaswi Choppa, and Dominik Schlechtweg. 2024. Towards Automating Text Annotation: A Case Study on Semantic Proximity Annotation using GPT-4. *arXiv preprint arXiv:2407.04130*.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.
- Junkai Zhou, Liang Pang, Huawei Shen, and Xueqi Cheng. 2024. [Think Before You Speak: Cultivating Communication Skills of Large Language Models via Inner Monologue.](#)

A Taxonomy of Speech Functions

Label	Definition	Example	Freq. (%)
Open.Attend	These are usually greetings.	<i>Hey, David!</i>	1.6
Open.Demand.Fact	Demanding factual information at the beginning of a conversation or when introducing a new topic.	<i>What's Allenby doing these days?</i>	2.7
Open.Demand.Opinion	Demanding judgment or evaluative information from the interlocutor at the beginning of a conversation or when introducing a new topic.	<i>Do we need Allenby in this conversation?</i>	1.1
Open.Give.Fact	Providing factual information at the beginning of a conversation or when introducing a new topic.	<i>You met his sister.</i>	1.8
Open.Give.Opinion	Providing judgment or evaluative information at the beginning of a conversation or when introducing a new topic.	<i>This conversation needs Allenby.</i>	0.9
Open.Command	Making a request, an invitation or command to start a dialog or discussion of a new topic.	<i>Could you tell me about your wedding?</i>	1.1
React.Rejoinder.Support.Track.Probe	Requesting a confirmation of the information necessary to make clear the previous speaker's statement.	<i>Because Roman lives in Denning Road also?</i>	1.9
React.Rejoinder.Support.Track.Check	Getting the previous speaker to repeat an element or the entire statement that the speaker has not heard or understood.	<i>Straight into the what?</i>	0.9
React.Rejoinder.Support.Track.Clarity	Asking a question to get additional information on the current topic of the conversation. Requesting to clarify the information already mentioned in the dialog.	<i>What, before bridge?</i>	12.0
React.Rejoinder.Support.Track.Confirm	Asking for a confirmation of the information received.	[David: <i>Well, he rang Roman, he rang Roman a week ago.</i> Nick: <i>Did he?</i>]	1.6
React.Rejoinder.Support.Response.Resolve	The response provides the information requested in the question.	[Fran: <i>Oh what is it called?</i> Brad: <i>PhD in Science.</i>]	8.7
React.Rejoinder.Confront.Response.Re-challenge	Offering an alternative position, often an interrogative sentence.	[David: <i>No, Messi is the best</i> Nick: <i>PAUSE</i> David: <i>The best is Pele</i>]	0.2
React.Rejoinder.Confront.Challenge.Rebound	Questioning the relevance or reliability of the previous statement, often an interrogative sentence.	[David: <i>This conversation needs Allenby.</i> Fay: <i>Oh he's in London. So what can we do?</i>]	0.5
React.Rejoinder.Confront.Challenge.Detach	Terminating the dialogue.	<i>So stick that!</i>	0.5
React.Rejoinder.Confront.Challenge.Counter	Dismissing the addressee's right to his/her position.	<i>You don't understand, Nick.</i>	1.2
React.Rejoinder.Confront.Response.Refute	Rejecting a transition to a new topic.	[David: <i>I'm out.</i> Fay: <i>You can't do that, it's my birthday.</i>]	0.1
React.Respond.Support.Register	A manifestation of emotions or a display of attention to the interlocutor.	<i>Yeah. Right. Hmm...</i>	6.0
React.Respond.Support.Engage	Drawing attention or a response to a greeting.	<i>Hey! Hi-hi.</i>	0.6
React.Respond.Support.Reply.Accept	Expressing gratitude.	<i>Thank you!</i>	1.2

React.Respond.Support.Reply.Affirm	A positive answer to a question or confirmation of the information provided. Yes/its synonyms or affirmation.	[Nick: <i>He went to London.</i> Fay: <i>He did.</i>	3.7
React.Respond.Support.Reply.Acknowledge	Indicating knowledge or understanding of the information provided.	<i>I know. I see. Oh yea.</i>	1.1
React.Respond.Support.Reply.Agree	Agreement with the information provided. In most cases, the information that the speaker agrees with is new to him.	<i>Yes. Right.</i>	3.8
React.Respond.Support.Develop.Extend	Adding supplementary or contradictory information to the previous statement.	David: [<i>That's what the cleaner—your cleaner lady cleaned my place though.</i>] Nick: <i>She won't come back to our place.</i>	8.6
React.Respond.Support.Develop.Enhance	Adding details to the previous statement, adding information about time, place, reason, etc.	[Fay: <i>He kept telling me I've got a big operation on with.</i>] Nick: <i>The trouble with Roman though is that—you know he does still like cleaning up.</i>	0.4
React.Respond.Support.Develop.Elaborate	Clarifying/rephrasing the previous statement or giving examples to it. A declarative sentence or phrase (may include <i>for example, I mean, like</i>).	[Nick: <i>Cause all you'd get is him bloody raving on.</i>] Fay: <i>He's a bridge player, a naughty bridge player.</i>	0.2
React.Respond.Confront.Reply.Disavow	Denial of knowledge or understanding of information.	<i>I don't know. No idea.</i>	0.4
React.Respond.Confront.Reply.Disagree	Negative answer to a question or denial of a statement. No, negative sentence.	[David: <i>Is he in London?</i>] Nick: <i>No.</i>	2.0
React.Respond.Confront.Reply.Contradict	Refuting previous information. Sentence with opposite polarity. If the previous sentence is negative, then this sentence is positive, and vice versa.	[Fay: <i>Suppose he gives you a hard time, Nick?</i>] Nick: <i>Oh I like David a lot.</i>	0.4
React.Respond.Command	Making a request, an invitation, or command in response to previous information.	<i>Could you tell me about your wedding?</i>	-
Sustain.Continue.Monitor	Checking the involvement of the listener or trying to pass on the role of speaker to them.	<i>You know? Right?</i>	0.2
Sustain.Continue.Command	Making a request, an invitation, or command to continue the dialog or discussion without changing the speaker.	<i>Could you tell me about your wedding?</i>	-
Sustain.Continue.Prolong.Extend	Adding supplementary or contradictory information to the previous statement. Used only when the speaker remains the same as in the previous utterance.	<i>Just making sure you don't miss the boat. I put it out on Monday mornings. I hear them. I hate trucks.</i>	21.8
Sustain.Continue.Prolong.Enhance	Adding details to the previous statement, adding information about time, place, reason, etc. Used only when the speaker remains the same as in the previous utterance.	<i>Nor for much longer. We're too messy for him.</i>	5.1
Sustain.Continue.Prolong.Elaborate	Clarifying/rephrasing the previous statement or giving examples to it. Used only when the speaker remains the same as in the previous utterance.	<i>Yeah but I don't like people... um... I don't want to be INVOLVED with people.</i>	7.9

Table 9: Taxonomy of speech functions (“-” indicates that these labels are counted together with **Open.Command**.)

B Example of Tree Scheme

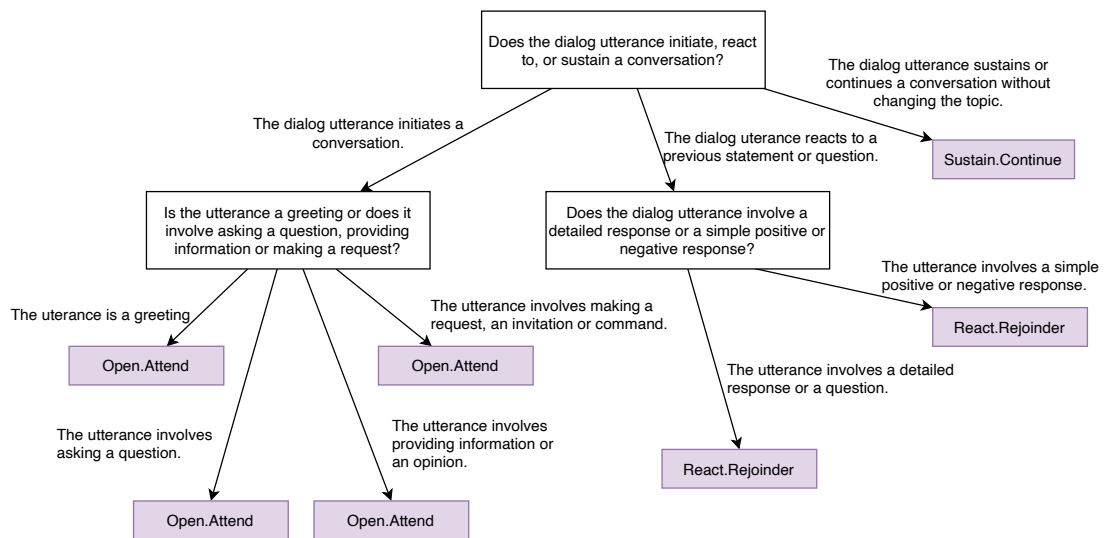


Figure 3: An example of a tree scheme generated by GPT-4 using our proposed approach, where nodes represent questions about an utterance, arrows indicate possible answer choices and purple leaves correspond to taxonomy labels.

C Example of dialog annotation with Speech Functions

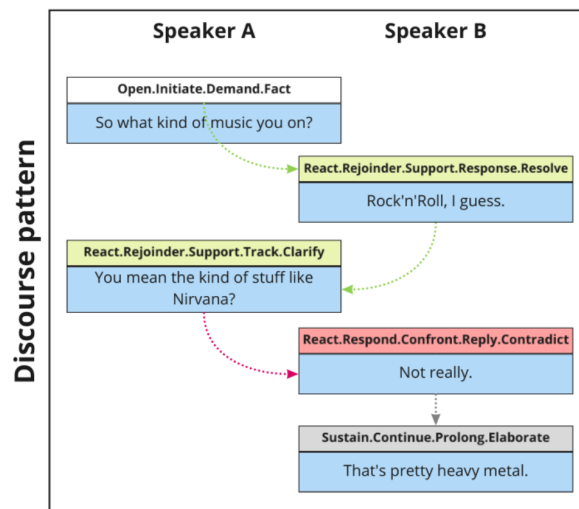


Figure 4: Example of dialog annotation with speech functions (Ostyakova et al., 2023).

D The Top Level Labels of SF taxonomy

Label	Definition
Open	Open utterances are statements or actions that initiate a conversation or introduce a new topic within an ongoing discussion. These may include greetings, questions, requests, invitations, or the sharing of information.
React	React utterances are responses to the interlocutor's statements. These may include answers to questions, follow-up questions, emotional reactions, sharing information, expressions of agreement or disagreement, and more.
Sustain	Sustain utterances are those that extend the speaker's own preceding statements by adding information, providing new details, or rephrasing. The "Sustain" label is applied only when the current and preceding utterances are made by the same speaker. These utterances cannot take the form of questions, except when the question serves to confirm that the listener is paying attention.

Table 10: Definitions of Open, React and Sustain labels written manually.

E The Top Two Level Classes of SF taxonomy

Label	Definition
Open.Demand	Questions at the beginning of a conversation or when introducing a new topic.
Open.Give	Providing information or opinion at the beginning of a conversation or when introducing a new topic.
Open.Command	Making a request, an invitation or command to start a dialog or discussion of a new topic.
Open.Attend	These are usually greetings.
Sustain.Continue	These are used only when there is no change in the speaker from the previous utterance, except for cases when the utterance is a reply to a greeting. The Sustain.Continue class involves adding additional information or details to the speaker's previous statement. It can also include questions intended to check if the interlocutor is listening, as well as requests or invitations.
React.Rejoinder	These include any type of question or detailed response to the interlocutor's questions, expressions of emotion, and grounding utterances such as "hmm," "aha," and similar reactions.
React.Respond	These include positive or negative responses to questions and expressions of understanding or misunderstanding. They also include the provision of new information or details, similar to Sustain.Continue. However, in this case, the speaker of the current utterance differs from the speaker of the previous one (this rule regarding speakers applies only to cases similar to Sustain.Continue).

Table 11: Definitions of the two highest-level labels.

F Example of the Model's Reasoning Output

Input Data

Label	Definition	Example	Frequency (%)
Open.Attend	These are usually greetings.	Hey, David!	1.6
Open.Demand.Fact	Demanding factual information at the beginning of a conversation or when introducing a new topic.	What's Allenby doing these days?	2.7
Open.Demand.Opinion	Demanding judgment or evaluative information from the interlocutor at the beginning of a conversation or when introducing a new topic.	Do we need Allenby in this conversation?	1.1
Open.Give.Fact	Providing factual information at the beginning of a conversation or when introducing a new topic.	You met his sister.	1.8
Open.Give.Opinion	Providing judgment or evaluative information at the beginning of a conversation or when introducing a new topic.	This conversation needs Allenby.	0.9
Open.Command	Making a request, an invitation or command to start a dialog or discussion of a new topic.	Could you tell me about your wedding?	1.1



LLM Output

"question_1": "Does the dialog utterance demand information or provide information?",
 "answer_1": "If the dialog utterance is asking for information, it is a demand. If it is providing information, it is a give.",
 "question_2": "Does the dialog utterance demand or provide factual information or opinion-based information?",
 "answer_2": "If the dialog utterance is asking for or providing factual information, it is a fact. If it is asking for or providing opinion-based information, it is an opinion.",
 "question_3": "Is the dialog utterance a command or a greeting?",
 "answer_3": "If the dialog utterance is a request, invitation, or command, it is a command. If it is a greeting, it is an attend.",
 "question_to_define_groups": "Is the dialog utterance demanding information, providing information, a command, or a greeting?",

Figure 5: An example of question-answer pairs generated by GPT-4 to reason about identifying utterances of different classes.

G Prompt template for Tree Construction

You are a linguist tasked with constructing a decision tree to distinguish between different classes within a provided taxonomy. Follow these instructions step-by-step:

1. Analyze the provided taxonomy and its descriptions.
2. Generate three questions based on the descriptions of the labels (NOT the labels themselves) that will help you determine how to split the data into groups. The questions must focus solely on the content or characteristics of the dialog utterances.
3. Answer these questions based on the provided data and descriptions.
4. Propose a final, clear question that can effectively classify dialog utterances into two or more groups.
 - Prohibited: Directly asking about the labels, categories, or grouping all taxonomy labels into a single group.
5. Form the groups based on the answers to your final question. The following rules are critical:
 - Each label can belong to only one group (no duplicates).
 - Avoid ambiguity in group descriptions—do not refer to the labels explicitly.

Output Format:
Provide your response as a valid JSON with the following structure:

```
{
  "question_1": "Your first generated question.",
  "answer_1": "Your answer to question_1.",
  "question_2": "Your second generated question.",
  "answer_2": "Your answer to question_2.",
  "question_3": "Your third generated question.",
  "answer_3": "Your answer to question_3.",
  "question_to_define_groups": "The final question to split the data into groups.",
  "groups": [
    {
      "label": "The answer/description that defines the utterances of the first group.",
      "data": ["List of taxonomy labels belonging to the first group."]
    },
    {
      "label": "The answer/description that defines the utterances of the second group.",
      "data": ["List of taxonomy labels belonging to the second group."]
    },
    ...
  ]
}
```

Description:
{description}

Taxonomy:
{taxonomy}

Figure 6: The prompt template used for all experiments with tree construction.

H Prompt template for the annotation step

TASK: You will see the part of the dialog between 2 speakers. Answer Question about Current Utterance. You must analyze the relations between the Current Utterance and the Previous Context.

Previous Context:
{previous_context}

Current Utterance:
{current_utterance}

Question: {question}
Possible Answers: {possible_answers}

Remember that the Question is about the Current Utterance.
You must select one answer from Possible Answers and reply only with "Answer 1", "Answer 2", etc., without any additional explanation.

Figure 7: The prompt template used for the annotation step.

I Prompt for Constructing a Yes/No Questions Tree

```
SPLIT_INTO_GROUPS_SYSTEM = """"\nYou are a linguist building a decision tree to distinguish between different classes. Follow these steps:\n\n1. Analyze the provided taxonomy.\n2. Generate three questions that will help you understand how to split the data into two groups. These questions should focus on the descriptions of the labels, NOT on the labels themselves.\n3. Answer these questions based on the data.\n4. Propose a clear yes/no question to classify the dialog utterance into one of the two groups.\n5. **Do not** ask questions about labels or categories directly; focus solely on the content of the dialog utterances.\n6. **Absolutely prohibited:** Grouping all taxonomy labels into one and naming the other group as "none," "other," "miscellaneous," "unknown," or similar.\n7. The split must always involve two meaningful groups based on content.\n\nFormat your response as a valid JSON:\n{{\n  "question_1": "Your first generated question.",\n  "answer_1": "Your answer to question_1.",\n  "question_2": "Your second generated question.",\n  "answer_2": "Your answer to question_2.",\n  "question_3": "Your third generated question.",\n  "answer_3": "Your answer to question_3.",\n  "question_to_define_groups": "The proposed Yes/No question to split the data into groups.",\n  "group_1_label": "Yes/No",\n  "group_2_label": "Yes/No",\n  "group_1_data": ["List of taxonomy labels for the first group."],\n  "group_2_data": ["List of taxonomy labels for the second group."]\n}}\n\nYou must always split the taxonomy into two **meaningful** groups. Creating vague or catch-all groups, such as "none," "other," "miscellaneous," or "unknown," is **strictly prohibited** and will result in an incorrect response. The split must be content-based and meaningful.""\"\n\nSPLIT_INTO_GROUPS_HUMAN = """"\nDescription:\n{description}\n\nTaxonomy:\n{taxonomy}\n"""
```

Figure 8: The prompt used for constructing trees with yes/no questions.

J Prompt for Constructing an Open-Ended Questions Tree

```
SPLIT_INTO_GROUPS_SYSTEM = """"\nYou are a linguist building a decision tree to distinguish between different classes. Follow these steps:\n\n1. Analyze the provided taxonomy.\n2. Generate three questions that will help you understand how to split the data into two groups. These\nquestions should focus on the descriptions of labels, NOT on the labels themselves.\n3. Answer these questions based on the data.\n4. Propose a clear question to classify the dialog utterance into two groups.\n5. **Do not** ask questions about labels or categories directly; focus solely on the content of the dialog\nutterances.\n6. **Absolutely prohibited:** Grouping all taxonomy labels into one and naming the other group as "none,"\n"other," "miscellaneous," "unknown," or similar.\n7. The split must always involve two meaningful groups based on content.\n\nFormat your response as a valid JSON:\n{{\n  "question_1": "Your first generated question.",\n  "answer_1": "Your answer to question_1.",\n  "question_2": "Your second generated question.",\n  "answer_2": "Your answer to question_2.",\n  "question_3": "Your third generated question.",\n  "answer_3": "Your answer to question_3.",\n  "question_to_define_groups": "The proposed question to split the data into groups.",\n  "group_1_label": "Answer to the question that defines the first group.",\n  "group_2_label": "Answer to the question that defines the second group.",\n  "group_1_data": ["List of taxonomy labels for the first group."],\n  "group_2_data": ["List of taxonomy labels for the second group."]\n}}\n\nYou must always split the taxonomy into two **meaningful** groups. Creating vague or catch-all groups, such\nas "none," "other," "miscellaneous," or "unknown," is **strictly prohibited** and will result in an incorrect\nresponse. The split must be content-based and meaningful.""\"\n\nSPLIT_INTO_GROUPS_HUMAN = """"\nDescription:\n{description}\n\nTaxonomy:\n{taxonomy}\n"""
```

Figure 9: The prompt used for constructing trees with open-ended questions.

K Prompt for Constructing a Non-Binary Open-Ended Questions Tree

```
SPLIT_INTO_GROUPS_SYSTEM = ""\nYou are a linguist building a decision tree to distinguish between different classes. Follow these steps:\n\n1. Analyze the provided taxonomy.\n2. Generate three questions that will help you understand how to split the data into groups. These questions should focus on the descriptions of labels, NOT on the labels themselves.\n3. Answer these questions based on the data.\n4. Propose a clear question to classify the dialog utterance into groups.\n5. **Do not** ask questions about labels or categories directly; focus solely on the content of the dialog utterances.\n6. **Absolutely prohibited:** Grouping all taxonomy labels into one and naming the other group as "none," "other," "miscellaneous," "unknown," or similar.\n7. The split must always involve meaningful groups based on content.\n\nFormat your response as a valid JSON:\n{\n  \"question_1\": \"Your first generated question.\",\n  \"answer_1\": \"Your answer to question_1.\",\n  \"question_2\": \"Your second generated question.\",\n  \"answer_2\": \"Your answer to question_2.\",\n  \"question_3\": \"Your third generated question.\",\n  \"answer_3\": \"Your answer to question_3.\",\n  \"question_to_define_groups\": \"The proposed question to split the data into groups.\",\n  \"groups\": [\n    {\n      \"label\": \"Answer to the question that defines the first group.\", # The answer must not contain labels!\n      \"data\": [\"List of taxonomy labels for the first group.\"]\n    },\n    {\n      \"label\": \"Answer to the question that defines the second group.\", # The answer must not contain labels!\n      \"data\": [\"List of taxonomy labels for the second group.\"]\n    },\n    ...\n  ]\n}\n\nYou must always split the taxonomy into **meaningful** groups. Creating vague or catch-all groups, such as \"none,\" \"other,\" \"miscellaneous,\" or \"unknown,\" is **strictly prohibited** and will result in an incorrect response. The split must be content-based and meaningful.\"`\n\nSPLIT_INTO_GROUPS_HUMAN = ""\nDescription:\n{description}\n\nTaxonomy:\n{taxonomy}\n""
```

Figure 10: The prompt used for constructing non-binary trees with open-ended questions.

L Prompt For Scoring Splits

```
SCORE_SPLITS_SYSTEM = """"\nYou are a linguist tasked with evaluating a decision tree designed to distinguish between different classes. Follow these steps:\n\n1. **Analyze the provided taxonomy** to understand the categories and their relationships.\n2. **Evaluate the proposed splits** to assess how well they divide the data into meaningful and distinct groups.\n3. **Score each split** as "bad," "good," or "great," based on its effectiveness in separating the data:\n   - **Bad:** Creates vague or ambiguous groups such as "none," "other," "miscellaneous," or "unknown."\n   - **Good:** Provides reasonable separation, though improvements may be possible.\n   - **Great:** Clearly and effectively separates the data into meaningful, well-defined groups.\n\nFormat your response as valid JSON in the following structure:\n{\n  "split_1": {\n    "thought": "Your analysis and reasoning for the first split.",\n    "score": "bad/good/great"\n  },\n  "split_2": {\n    "thought": "Your analysis and reasoning for the first split.",\n    "score": "bad/good/great"\n  },\n  ...\n}\n"""\n\nSCORE_SPLITS_HUMAN = """"\nTaxonomy:\n{taxonomy}\n\nProposed splits:\n{splits}\n"""
```

Figure 11: The prompt used to score split-candidates.

M Prompt For Frequency-Guided Tree Creation

```
SPLIT INTO GROUPS SYSTEM = ""
You are a linguist tasked with constructing a decision tree to distinguish between different classes within a provided taxonomy. Follow these instructions step-by-step:

1. Analyze the provided taxonomy and its descriptions.
2. Generate three questions based on the descriptions of the labels (NOT the labels themselves) that will help you determine how to split the data into groups. The questions must focus solely on the content or characteristics of the dialog utterances.
   - At this step, ensure you understand that one group must contain the single, most frequent label.
3. Answer these questions based on the provided data and descriptions.
4. Propose a final, clear question that can effectively classify dialog utterances into two or more groups.
   - Mandatory: One group must contain only the single, most frequent label.
   - Prohibited: Directly asking about the labels, categories, or grouping all taxonomy labels into a single group.
5. Form the groups based on the answers to your final question. The following rules are critical:
   - One group must exclusively contain the single, most frequent label.
   - Each label can belong to only one group (no duplicates).
   - You must split the taxonomy into at least two groups.
   - Avoid ambiguity in group descriptions—do not refer to the labels explicitly.

### Output Format:
Provide your response as a valid JSON with the following structure:

{
  "question_1": "Your first generated question.",
  "answer_1": "Your answer to question_1.",
  "question_2": "Your second generated question.",
  "answer_2": "Your answer to question_2.",
  "question_3": "Your third generated question.",
  "answer_3": "Your answer to question_3.",
  "question_to_define_groups": "The final question to split the data into groups.",
  "groups": [
    {
      "label": "The answer/description that defines the utterances of the first group (single most frequent label).",
      "data": ["The single most frequent taxonomy label."]
    },
    {
      "label": "The answer/description that defines the utterances of the second group.",
      "data": ["List of taxonomy labels belonging to the second group."]
    },
    ...
  ]
}

SPLIT INTO GROUPS HUMAN = ""
Description:
{description}

Taxonomy:
{taxonomy}
""
```

Figure 12: The prompt is designed to create a decision tree that enables splitting data into more than two groups and that specifically instructs the LLM to form one group that exclusively contains the single most frequent class.