STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **x**, Month 202x, pp 0–24.
Published online in International Academic Press (www.IAPress.org)

ECJ-LLM: From Extraction to Judgment; Reinventing Automated Review Annotation

L. Maada,*[1], K. Al Fararni [2], B. Aghoutane[1], Y. Farhaoui[3], M. Fattah[4]

[1] *IA Laboratory, Science Faculty, Moulay Ismail University, Meknes, Morocco*
[2] *LISAC Laboratory, Faculty of Sciences Dhar El Mahraz, Sidi Mohamed Ben Abdellah University, Fez, Morocco*
[3] *L-STI, T-IDMS, University of Moulay Ismail, Faculty of Science and Technics, Errachidia, Morocco*
[4] *Image laboratory, Moulay Ismail University Meknes, Morocco*

Abstract This paper introduces a novel multi-agent debate framework for interest-based data annotation, leveraging Large Language Models (LLMs) to facilitate structured, collaborative annotation. The framework orchestrates three specialized LLM agents—an aspect extractor, a critic, and a judge—who engage in a systematic debate: the extractor identifies relevant aspects, the critic evaluates and challenges these aspects, and the judge synthesizes their input to assign final, high-quality interest-level labels. This debate-driven process not only enhances annotation fidelity and context but also allows for flexible customization of models used in each role and the interests to be detected. To ensure transparency and quality, the framework incorporates an evaluation suite with metrics such as precision, recall, F1-score, and confusion matrices. Empirical results on a gold-standard hotel review dataset demonstrate that the framework outperforms single-agent methods in annotation quality. A customizable annotation tool, developed as a demonstration of the framework's practical utility, further showcases its flexibility and extensibility for a range of annotation tasks.

Keywords Large Language Models (LLMs), multiagent systems, data annotation, Computational Annotator, Prompt Engineering, Annotation Automation, web application

DOI: 10.19139/soic-2310-5070-2836

1. Introduction

Today, the digital world makes it easy for people to share personal posts, opinions, and emotions online. Social media, review sites, and blogs now contain a large amount of user-generated content. This steady flow of text creates the need for tools that can make sense of all this information. Sentiment analysis, also called opinion mining or emotion AI, addresses this challenge. It uses natural language processing (NLP) and machine learning to detect emotions, measure public opinion, and find patterns in text. Because it reveals such insights, companies, researchers, and individuals increasingly rely on sentiment analysis to support important decisions.

Traditionally, sentiment and aspect analysis were performed using several methods. Lexicon-based rules, classic machine learning classifiers like Naive Bayes and support vector machines, and, more recently, deep learning frameworks such as recurrent neural networks (RNNs), long-short-term memory (LSTM) systems, and transformer models like BERT and T5 have all been used. Although these techniques improved the field, they still require large, expertly labeled datasets. However, producing such high-quality labels for new applications remains difficult, and manual annotation is expensive, time-consuming, and often subject to personal bias or ambiguity, especially in domains needing expert knowledge [6, 11, 7].

*Correspondence to: Loukmane Maada (Email: loukmane1397@gmail.com).IA Laboratory, Science Faculty, Moulay Ismail University, Meknes, Morocco.

ISSN 2310-5070 (online) ISSN 2311-004X (print)

Recently, the emergence of large language models (LLMs) such as GPT-4 [1], Gemini [2], and LLaMA-3 [3] has significantly altered this landscape. These models perform well across a panoply of tasks, such as translation, question-answering, complex reasoning, etc. Furthermore, studies now show that LLMs are more than just text generators. They can act as fast and efficient annotators. LLMs accelerate workflows, enhance output, and sometimes produce more consistent labels than crowdsourced or trained interns. These models can produce and critique many kinds of notes, including categorical tags, step-by-step reasoning, or preference scores. This capability allows LLMs to manage both small-scale quality checks and large annotation projects. Thus, LLMs can either serve as substitutes for, or partners to, human reviewers. However, concerns remain about how often, and under what conditions, computers can truly replace humans without creating new biases or hiding flaws behind complex reasoning [10].

Researchers are also moving beyond using single LLMs for annotation. New methods like Chain-of-Thought prompting, self-reflection, and especially multi-agent debate are being explored. In these systems, multiple LLMs, each with different roles or viewpoints, engage in structured discussions. They propose, challenge, and revise answers, mimicking how people work together and make decisions. By allowing different perspectives to interact, these systems help reduce errors such as hallucinations or arbitrary outputs. Studies report that this often improves annotation quality and matches human judgment better [11, 8, 9].

Despite these advances, there are still challenges. Multi-agent debate needs more computing resources and tokens, so it can be expensive to scale up. Its benefits depend on thoughtful prompt engineering, agent role design, and evaluation frameworks that ensure alignment with human judgments and gold standards. Ethical issues, such as bias, transparency, and fairness, are also very important—especially as LLMs are used in applications with social and economic impacts [11, 10, 8].

Our earlier work introduced a general framework for extracting user interest from hotel reviews to improve recommendation systems [12]. As active researchers in this area, we have encountered the practical difficulties of precisely detecting and annotating user-defined aspects in large-scale datasets. These experiences directly motivate our current focus on automating aspect detection and producing a high-quality, interest-oriented annotation dataset for hotel reviews. The key contributions of our work are as follows.

- We developed a tool that implements a multi-agent LLM debate framework, ECJ, in which specialized extractor (E), critic (C), and judge (J) agents collaborate to annotate user-defined aspects, going beyond traditional fixed-aspect debate systems.

- Our system uniquely allows users both to specify the aspects or interests they wish to annotate and to flexibly assign different LLMs to each agent role. This design provides adaptability across a wide range of domains and annotation tasks, unlike most existing frameworks that use fixed model assignments.

- We provide an interactive evaluation suite that enables users to compare automated annotations with gold-standard labels, visualize confusion matrices, and monitor metrics such as precision, recall, and F1-score for each aspect. The platform also logs the reasoning process and actions of each agent, supporting transparent performance assessment and efficient prompt optimization.

We begin in Section 2 by reviewing related work, highlighting recent developments in large language models for annotation. Next, in Section 3, we describe the methodology and system architecture in detail. In Section 4, we present the golden database we collected and discuss the choice of models used in the experiment. Building on this, Section 5 evaluates the ECJ using that golden data set against the single-model approach and includes an error analysis. In Section 6, we address the limitations of our framework. Finally, Section 7 concludes the paper and suggests directions for future research.

2. Related Work

2.1. LLMs for Automated Annotation

Early approaches to text annotation relied on lexicon-based rules or classic machine learning classifiers , such as Naive Bayes and Support Vector Machines. While deep learning architectures—such as recurrent neural networks (RNNs) and transformer models like BERT—have significantly improved accuracy, they still depend heavily on extensive manual annotation for training.

2.1.1. Single Agent annotator In light of these limitations, the emergence of LLMs has enabled direct annotation through multiple techniques, namely, few-shot or zero-shot prompting, greatly reducing the reliance on labeled data.

Recent advances have demonstrated the promise of collaborative human-LLM frameworks for large-scale text annotation. M. Li et al.[17] introduced CoAnnotating, a system that intelligently allocates annotation tasks by measuring LLM uncertainty through both self-reported confidence and entropy across diverse prompts. This approach enables straightforward instances to be handled automatically by LLMs, while ambiguous or challenging cases are reserved for human annotators. Empirical results across six datasets show that such uncertainty-guided strategies can deliver up to 21% higher annotation quality than random allocation, helping practitioners balance cost and quality and selectively delegate annotation work to LLMs when appropriate.

Building on similar principles, H. Kim et al. [13] proposed MEGAnno+, another human-LLM collaborative annotation system. MEGAnno+ allows users to manage AI agents and annotation data, automatically assigning the LLM to label data and reserving human verification for instances where the model shows low confidence. The system also parses raw LLM outputs into structured labels, facilitating flexible querying of synthetically annotated data. Demonstrated with a natural language inference use case, MEGAnno+ was shown to significantly accelerate the annotation process while maintaining quality through targeted human intervention. However, the authors note that LLM bias, variability, and certain API limitations still pose challenges for a broader adoption of such tools.

Moving beyond direct annotation, T.-H. Huang et al. [14] introduced ALCHEmist, a system that uses large language models to synthesize labeling programs rather than label data points directly. By prompting models like GPT-4 to generate executable labeling functions, ALCHEmist enables efficient and scalable annotation, as these programs can be applied locally to label entire datasets at virtually no cost. The outputs of multiple diverse labeling programs are aggregated using weak supervision frameworks (e.g., Snorkel) to produce high-quality pseudolabels, which are then used to train compact, deployable models. ALCHEmist achieves annotation quality comparable to or surpassing direct LLM annotation while reducing costs by approximately 500 times, and demonstrates flexibility across text, image, and tabular domains.

Other work has focused on evaluating LLMs as standalone annotators. P. Törnberg [26] evaluates ChatGPT-4's capabilities for political text annotation by comparing it to both expert and crowd-worker performance in a zero-shot classification task. Using a balanced dataset of tweets by U.S. senators during the 2020 election, the study finds that ChatGPT-4 surpasses humans in accuracy and intercoder reliability while displaying similar or lower bias. The model correctly identifies political affiliation even in cases requiring nuanced contextual reasoning, highlighting the potential for LLMs to conduct large-scale interpretive social science research. These findings challenge assumptions about the irreplaceability of human judgment in text-as-data tasks. Similarly, F. Gilardi et al. [15] compared ChatGPT's performance on many standard text annotation tasks. These included judging relevance, stance, topic, and frame in over 6,000 tweets and news articles. Using the zero-shot prompting technique, results show that ChatGPT outperformed crowd-workers from the Amazon Mechanical Turk in both accuracy (by an average of 25 percentage points) and intercoder agreement, and even surpassed the agreement rates of trained research assistants. Additionally, the cost per annotation of ChatGPT was almost thirty times cheaper than MTurk's fees. This highlights that large language models like GPT's family can offer great advantages in both quality and efficiency for large-scale text annotation tasks.

Pushing this progress further, X. Pei [16] introduced a self-supervised iterative approach to annotate data using GPT models. Their method uses a generate-and-recover loop, where the LLM is prompted to summarize the input and then asked to reconstruct the original data. This loop helps the system fine-tune a one-shot template, guided by alignment scores that show how well the original data is recovered. This approach was extensively tested on a neural architecture and chemical structure dataset. The results show that this method outperformed both the zero-shot and the simple prompt-based approaches, particularly when using GPT-3.5-turbo, and achieves competitive results across various human feedback and alignment.

2.1.2. Multi-Agent and Debate Annotation Frameworks To further enhance the quality of annotations, researchers have increasingly turned to advanced frameworks that incorporate multi-agent debate and committee-based decision-making. In these systems, multiple large language models (LLMs) are deployed to simulate the dynamics of human group discussions, where each model acts as an independent agent with potentially different perspectives or reasoning strategies. These agents engage in structured debates or collaborative evaluations, allowing for more robust and nuanced outcomes. By mimicking the deliberative processes typical of expert committees or peer review panels, this approach aims to reduce individual model biases, improve consistency, and capture a broader range of valid interpretations—ultimately leading to higher-quality and more reliable annotations. [9, 11].

M. Hegazy et al. [19] proposed MAFA, a multi-agent framework for FAQ annotation that combines the outputs of multiple specialized agents—each guided by structured reasoning and tailored few-shot examples—with a judge agent that reranks candidates using multi-dimensional evaluation. Their results on both proprietary banking data and public benchmarks (LCQMC, FiQA) show substantial improvements in annotation accuracy and robustness over traditional single-agent and retrieval approaches. The study demonstrates that agent diversity, structured prompting, and a judge-based reranking stage are key to handling ambiguous queries and achieving strong generalization across domains and languages, making MAFA particularly suitable for deployment in production applications.

M. Lin et al. [20] proposed TESSA, a multi-agent framework that generates high-quality general and domain-specific annotations for time series data by combining knowledge from multiple source domains with limited target-domain labels. TESSA integrates adaptive feature selection—using both LLM-based and reinforcement learning approaches—to extract and emphasize salient time-series and text features, while a domain-specific agent learns specialized jargon for precise annotation. Extensive experiments on synthetic and real-world datasets show that TESSA outperforms direct LLM-based annotation and multimodal baselines in terms of clarity, comprehensiveness, domain relevance, and downstream task performance. The framework generalizes across domains and languages, with each system component contributing to improved annotation quality and efficiency, making TESSA an effective tool for automatic time series annotation in cross-domain settings

2.1.3. Summary and Positioning In summary, while LLM-based and multi-agent frameworks have advanced the state of the art in automated annotation, many existing systems lack user configurability or offer limited transparency in agent interactions and evaluation. Our work extends this literature by introducing a flexible user-driven ECJ-based annotation platform that supports custom aspect selection, modular agent configuration,and evaluation tools.

3. Methodology

3.1. Assumption on Context Enrichment.

We assume that both the Extractor and the Critic enrich the Judge's context—through highlighted spans, rephrasings, and pro/con arguments—akin to providing structured few-shot examples or explanatory notes, which are known to improve LLM accuracy and robustness [18].

3.2. Overview

Our tool is designed to simplify the process of interest/aspect-based detection. The application implements a three-agent discussion framework, ECJ. Each model is powered by an LLM and is given a prompt that defines its role. Users can define custom aspects to be annotated, select different models for each agent, and evaluate the annotation quality with an interactive dashboard. The core workflow consists of three specialized agents—Extractor, Critic, and Judge—working in sequence to produce accurate and transparent interest-level labels.

3.3. Prompt Engineering

Prompt engineering has become a cornerstone for maximizing the effectiveness and adaptability of large language models (LLMs). Recent studies support that careful prompt engineering is key to the effectiveness [7] of LLM agents. In fact, rather than altering model parameters or retraining, prompt engineering strategically crafts task-specific instructions—prompts—that guide LLMs to produce desired outputs across a vast array of applications and domains. This approach unlocks the potential for models to be flexibly deployed in new tasks or settings by leveraging carefully designed prompts, whether through simple instructions, few-shot demonstrations, or advanced strategies like chain-of-thought and context-enhanced prompting [21].

Building on this, we used multiple prompting techniques to optimize our agents' prompts[†]:

- **Iterative Prompting:** Prompts were polished through an incremental process of trial and error [22].
- **Clear Desired Output:** Each prompt included a clear set of instructions regarding the output and the level of details expected from the LLM (e.g., "Justify each aspect's inclusion by quoting relevant phrases from the review.").
- **Persona Assignment:** We assigned each agent a distinct persona within the prompt (e.g., "You are an expert assistant that extracts Aspects from Hotel Reviews"), which helps enhance the model's effectiveness and contextual adaptation [23].
- **Context Inclusion:** For the Critic and Judge agents, prompts incorporated outputs from previous agents as context, supporting more informed detection and correction.

Our system employs prompt templates, which are modified at run-time to accommodate user-defined aspects/interests detection.

3.4. Annotation Workflow

The annotation workflow proceeds in three stages (Figure 1): first, the Extractor identifies the user-configured aspects in the text; next, the Critic evaluates both the original reviews and the Extractor's findings; and finally, the Judge agent issues a definitive verdict using the enriched context supplied by the Extractor and the Critic.

- **Extractor Agent**: Receives the raw user input x and produces initial aspect-level annotations y_{ext}, based on user-defined aspects.
- **Critic Agent**: Analyzes the extractor's output, y_{ext}, and the original input x, to provide feedback, corrections, or highlight uncertainties. The output is y_{crt}.
- **Judge Agent**: Integrates the original input x, extractor output y_{ext}, and critic output y_{crt}, then decides the final annotation $\hat{y}$.

Formally, the process can be described as:

$$
\begin{aligned}
y_{\text{ext}} &= f_{\text{ext}}(x) \\
y_{\text{crt}} &= f_{\text{crt}}(x, y_{\text{ext}}) \\
\hat{y} &= f_{\text{judge}}(x, y_{\text{ext}}, y_{\text{crt}})
\end{aligned}
\tag{1}
$$

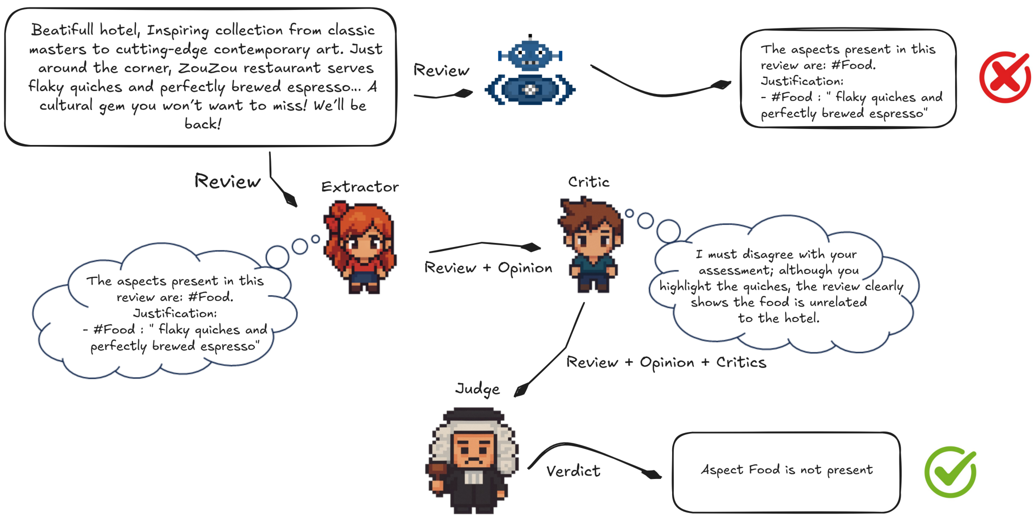

Figure 1. Single model vs ECJ Review Annotation Workflow

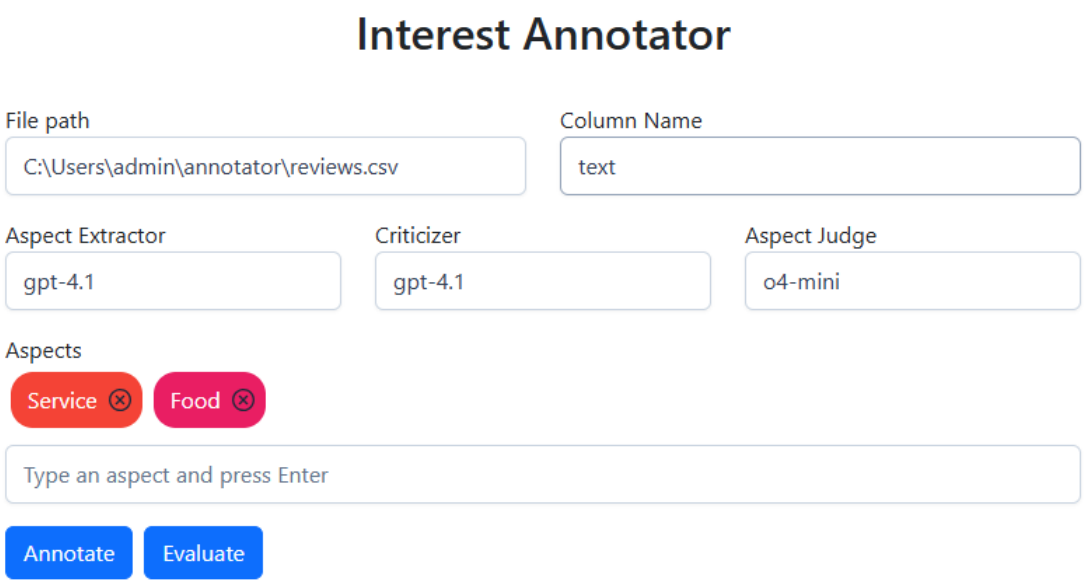

Figure 2. User interface of the Interest Annotator

3.5. *Annotator interface*

The Interest Annotator user interface (Figure 2) is designed to facilitate aspect-based annotation and evaluation of textual review datasets. The interface allows users to specify the input data by entering the file path (for

[†]The full prompt of each agent can be found in Appendix B.

simplification purposes only, in future work, we will use a file uploading mechanism and store the file on the server side) and the relevant text column name. The model configurations for the Aspect Extractor, Criticizer, and Aspect Judge are provided through text input fields, allowing flexible assignment of different language models by manually entering their names. Users can define multiple aspects of interest by typing the name of the aspect in an input box and adding it to the list, where each aspect appears as a labeled tag. The interface includes two primary action buttons: 'Annotate' and 'Evaluate', which initiate the respective processes.

3.6. Annotator interface

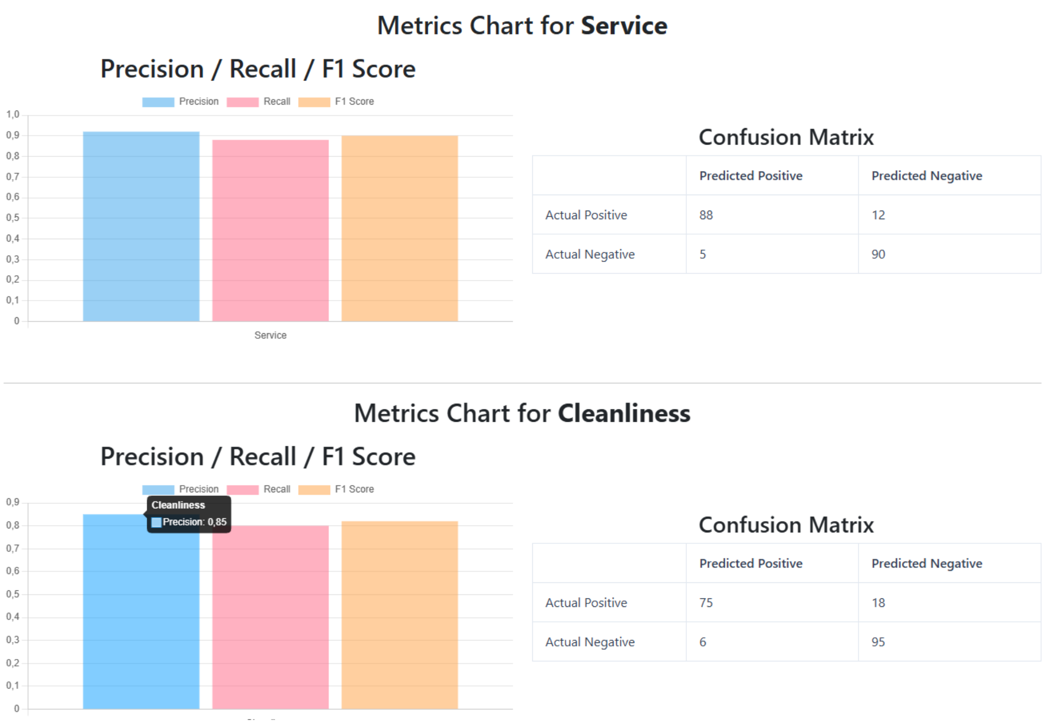

Figure 3. Evaluation User Interface (UI)

Figure 3 shows the evaluation results for Service and Cleanliness (dummy data used to show the UI). For each aspect, the left panel presents a bar chart comparing Precision, Recall, and F1 Score, providing a visual summary of classification performance. The right panel shows the corresponding confusion matrix, indicating the counts of actual versus predicted labels (positive and negative). This figure allows for an intuitive assessment of model effectiveness and highlights both overall performance (through the metrics) and specific error types (through the confusion matrices) for each aspect.

3.7. Application architecture

In our application, the frontend and backend are separated to ensure modularity and scalability. The frontend, built with Angular, interacts with users and communicates with the backend via HTTP protocols. The backend is powered by Spring Boot, which exposes two RESTful APIs to handle the evaluation logic and the annotation logic. We use the spring AI module to facilitate the communication between the application and the LLM host (e.g, OpenAI). After the processing of the reviews, the data is stored in a Postgres database.

3.8. *Agent Logging and Traceability*

To ensure transparency and reproducibility in the annotation process, our system logs the complete outputs and internal reasoning steps of each LLM agent—Extractor, Critic, and Judge—for every annotation instance. These logs are persistently stored as part of the annotation entity[‡] in the application database.

This comprehensive logging serves several key purposes:

- **Transparency:** Users and researchers can inspect the decision-making process of each agent, enabling detailed auditability and increased trust in the annotations produced by the system.
- **Debugging and Error Analysis:** In cases where annotation errors or disagreements arise, the agent logs provide insight into which stage or agent contributed to the outcome, supporting targeted improvements and troubleshooting.
- **Prompt Optimization:** By systematically analyzing the logged outputs, prompts can be refined and iteratively improved for higher annotation accuracy and consistency.
- **Reproducibility:** Retaining detailed agent logs allows for the annotation process to be replicated or reviewed in future studies, supporting robust validation and further research.

4. Dataset and model selection

4.1. *Dataset Construction*

To ensure a fair and unbiased evaluation, we manually curated a dataset consisting of 350 interest-annotated hotel reviews. This intentional approach was taken to minimize the potential risk of data contamination, which could arise from using publicly available annotated datasets that might have been included in the pretraining data of large language models (LLMs). As highlighted by [24], if models are trained on such publicly accessible datasets, there is a risk of overestimating their performance during evaluation. This happens because the models may have already encountered the evaluation examples during their training phase, leading to inflated accuracy scores that do not reflect the true capabilities of the model on unseen data. By crafting a custom dataset, we significantly reduce the likelihood that any of our evaluation examples were part of the model's training set. This not only mitigates the potential for performance bias but also strengthens the credibility and robustness of our evaluation, ensuring that our results are based on the model's ability to generalize rather than recall previously seen information.

The reviews were randomly sampled from a publicly available dataset on Kaggle[§], with only English-language reviews retained for analysis. Following the selection process, the reviews were evenly distributed among three annotators. Each annotator was tasked with evaluating them according to five predefined aspects: Cleanliness, Service, Price, Location, and Food (Table 1). For each aspect, a binary annotation was assigned—true if the aspect was deemed to be mentioned or present in the review, and false if it was considered absent. The distribution of each aspect is shown in Figure 4.

4.2. *Model selection*

To evaluate and test the initial version of our framework, we focused on integrating a selection of language models from the OpenAI organization. Our selection process was guided by several practical criteria. First and foremost, we considered the pricing structure of each model. Specifically, we included only those models where both the input and output costs were less than 10$ per one million tokens. This threshold was chosen to ensure cost-effectiveness and the limited budget we had to work with. The second criterion was the model's compatibility with chat completion, since our core functionality depends on it. It's important to emphasize that this is just the

[‡] A detailed ERD can be found in Appendix C

[§] https://www.kaggle.com/datasets/joebeachcapital/hotel-reviews

Table 1. Examples of a line in the dataset

Review Text	**Cleanliness**	**Service**	**Price**	**Location**	**Food**
Since May 2010, I have continued to stay at LaQuinta for appointments at M D Anderson Cancer Center. The rooms are clean and I feel safe. LaQuinta's shuttle to M D Anderson is a great help.	**true**	**true**	false	false	false

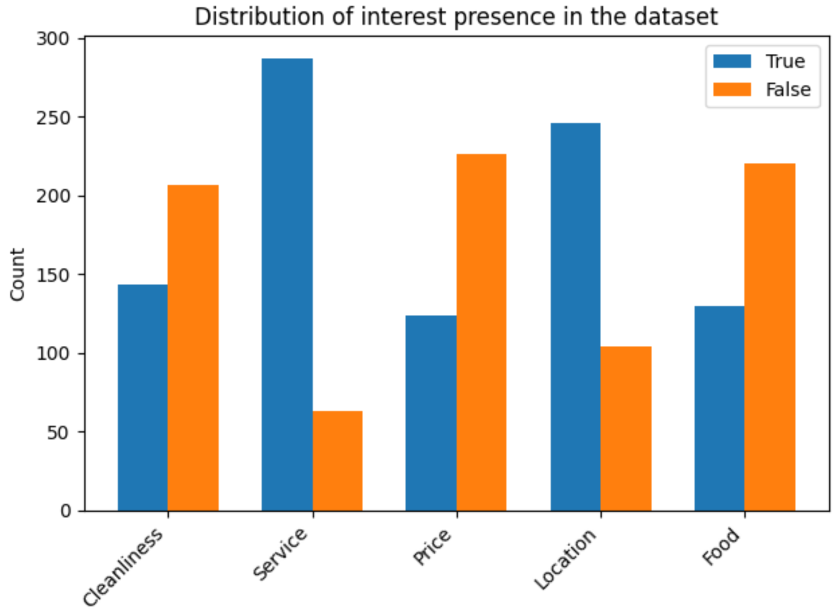

Figure 4. Dataset distribution

first phase of our work. The choice of OpenAI is due to the ease with which we were able to integrate it using Spring AI abstraction. However, we recognize the importance of supporting different providers to meet users' needs.

5. Result

5.1. Single interest detection

In the initial phase of our evaluation, we restricted our experiments to a single interest (*cleanliness*) to facilitate a controlled comparison. We evaluated the performance of our framework by testing various combinations of language models on the *cleanliness* aspect. To establish a baseline, we also performed single-model interest detection on the same aspect, allowing us to directly compare the effectiveness of our proposed framework with that of a traditional single-model extractor and determine whether there is any measurable performance improvement.

To account for variability and potential randomness, each combination was repeated five times[¶]. The average result across these runs was then used for evaluation. This approach helps to ensure that the reported performance

[¶]A detailed table of all the runs for each model/combination can be found in Appendix D is representative and not overly influenced by outliers or fluctuations in any single run, thus increasing the reliability and robustness of our results.

We selected the cleanliness interest because it's the most evenly represented class in the dataset, ensuring our metrics remain stable and reliable.

5.1.1. Baseline In table 9, we compared all seven model variants on the cleanliness interest detection using standard metrics and confusion–matrix counts. The o4-mini remarkably outperformed all other models, achieving the highest accuracy *95.49%*, precision of *93.42%*, a recall of *95.24%*, and an F1-Score of *94.31%*.The *gpt-4.1-mini* model followed with *87.83%* accuracy, *78.23%* precision, *97.48%* recall, and an F1-Score of *86.80%*, whereas the full *GPT-4.1* achieved *67.77%* accuracy, *55.95%* precision, *99.30%* recall, and an F1-Score of *71.57%*. In contrast, the ultra-compact variants *gpt-4o-mini*, *gpt-4.1-nano*, and *gpt-3.5-turbo* attained perfect recall (*100.00%*) but suffered from low precision (*41.10%*, *42.71%*, and *41.69%*, respectively), translating to modest accuracy (*41.43%*, *45.20%*, *42.86%*) and F1-Scores (*58.24%*, *59.86%*, *58.84%*). Finally, the *o3-mini* model balanced high recall (*99.16%*) with low precision (*46.89%*), yielding *53.77%* accuracy and an F1-Score of *63.67%*. Overall, *o4-mini* achieved the best trade-off between sensitivity and specificity across all models. Each model thus defined will serve as the baseline against which its counterpart in our framework is compared.

Table 2. Average performance metrics for a single agent/model on the cleanliness interest.

Model	Accuracy	Precision	Recall	F1	TP	FP	FN	TN
gpt-4o-mini	41.43%	41.10%	**100.00%**	58.24%	143.0	0.0	205.0	2.0
gpt-4.1-mini	87.83%	78.23%	97.48%	86.80%	139.4	3.6	39.0	168.0
gpt-4.1-nano	45.20%	42.71%	**100.00%**	59.86%	143.0	0.0	191.8	15.2
gpt-3.5-turbo	42.86%	41.69%	**100.00%**	58.84%	143.0	0.0	200.0	7.0
gpt-4.1	67.77%	55.95%	99.30%	71.57%	142.0	1.0	111.8	95.2
o4-mini	**95.49%**	**93.42%**	95.24%	**94.31%**	137.0	6.0	9.8	197.2
o3-mini	53.77%	46.89%	99.16%	63.67%	141.8	1.2	160.6	46.4

5.1.2. The ECJ framework single model approach In our initial experiment, a single model was applied uniformly across all three roles. This methodology allowed for a systematic evaluation of each model against its baseline. For most of our models, the use of our ECJ framework improved their performance.The *gpt-4o-mini* accuracy rose from *41.43%* to *91.64%* (gain of 50%) and its F1 from *58.24%* to *89.91%* (gain of 32%); the *gpt-4.1-mini* accuracy slightly decreased from *87.83%* to *84.34%* (loss of 3.5%) and its F1 fell from *86.80%* to *76.65%* (loss of 10%); the *gpt-4.1-nano* improved accuracy from *45.20%* to *83.60%* (gain of 38%) and its F1 from *59.86%* to *75.97%* (gain of 16%); the *gpt-3.5-turbo* saw accuracy increase from *42.86%* to *70.29%* (gain of 27%) and F1 from *58.84%* to *72.86%* (gain of 14%); the *gpt-4.1* jumped from *67.77%* to *95.66%* accuracy (gain of 28%) with F1 rising from *71.57%* to *94.65%* (gain of 23%); the *o4-mini* experienced a minor drop in accuracy from *95.49%* to *93.43%* (loss of 2%) and in F1 from *94.31%* to *92.30%* (loss of 2%); and the *o3-mini* enhanced accuracy from *53.77%* to *87.94%* (gain of 34%) with F1 increasing from *63.67%* to *86.87%* (gain of 23%).

These results in Table 3 indicate that the application of our ECJ framework yields substantial improvements in both accuracy and F1 scores for the majority of models evaluated, underscoring its effectiveness in enhancing model performance across diverse architectures. Notably, even in cases where a decrease in performance was observed, such as with *gpt-4.1-mini* and *o4-mini*, the magnitude of decline remained relatively minor compared to the significant gains achieved by other models. Overall, these outcomes suggest that the ECJ framework offers a robust and generally beneficial approach for boosting model efficacy, with only minimal drawbacks in a few isolated instances.

5.1.3. Statistical Significance Testing To evaluate the statistical reliability of the improvements obtained with the ECJ framework, paired t-tests were conducted comparing the F1-scores of each baseline model with its ECJ-enhanced counterpart across five experimental runs. Each test involved five paired observations, resulting in degrees

Table 3. Average ECJ single model approach performance metrics on the cleanliness interest.

Extractor	Critic	Judge	Accuracy	Precision	Recall	F1	TP	FP	FN	TN
gpt-4o-mini	gpt-4o-mini	gpt-4o-mini	91.64%	89.92%	89.93%	89.91%	128.6	14.4	15	192.6
gpt-4.1-mini	gpt-4.1-mini	gpt-4.1-mini	84.34%	**98.04%**	62.93%	76.65%	90	53	1.8	205.2
gpt-4.1-nano	gpt-4.1-nano	gpt-4.1-nano	83.60%	94.29%	63.77%	75.97%	91.2	51.8	5.6	201.4
gpt-3.5-turbo	gpt-3.5-turbo	gpt-3.5-turbo	70.29%	58.14%	**97.62%**	72.86%	139.6	3.4	100.6	106.4
gpt-4.1	gpt-4.1	gpt-4.1	**95.66%**	95.19%	94.12%	**94.65%**	134.6	8.4	6.8	200.2
o4-mini	o4-mini	o4-mini	93.43%	88.58%	96.36%	92.30%	137.8	5.2	17.8	189.2
o3-mini	o3-mini	o3-mini	87.94%	78.26%	**97.62%**	86.87%8	139.6	3.4	38.8	168.2

of freedom $df = 4$, and statistical significance was assessed at the $\alpha = 0.05$ level, corresponding to a critical value of $t_{crit} = 2.77$. Table 4 presents the mean F1-scores for both baseline and ECJ conditions, together with the absolute values of the corresponding t-statistics. In all cases, the absolute value of t exceeded the critical threshold, leading to rejection of the null hypothesis. This confirms that the ECJ framework yields robust and statistically significant improvements for most models, indicating that the observed gains are not attributable to random variation.

Table 4. Paired t-test results comparing Baseline vs. ECJ F1-scores across five runs.

Model	Baseline mean F1	ECJ mean F1	$\lvert t \rvert$
GPT-4.1	71.57	94.65	48.03
O4-mini	94.31	92.30	3.45
GPT-3.5	58.84	74.70	25.55
GPT-4o-mini	58.24	89.91	40.68
O3-mini	63.67	86.87	97.37
GPT-4.1-mini	86.80	76.65	17.96
GPT-4.1-nano	59.86	75.97	11.84

5.1.4. The ECJ framework multi-model approach In this part, we explored mixing and matching different LLMs in the ECJ framework. We chose to work with the models that achieved the highest recall (such as *o3-mini* or *gpt-3.5-turbo*) and the ones that achieved the highest precision (such as *gpt-4.1-mini*) as either extractors or critics. For the judge role we chose to work with the more accurate models like *o4-mini* and *gpt-4.1*.

As shown in Table 5, the combination of different model reveals distinct trade-offs between precision and recall. Notably, when a high-recall model (such as *o3-mini* or *gpt-3.5-turbo*) is used as the extractor and paired with a high-precision model (such as *gpt-4.1-mini*) as the critic, the framework achieves strong overall performance. This setup tends to preserve high recall while also improving precision, leading to more balanced and robust extraction results.

In contrast, when the roles are reversed—using the high-precision model as the extractor and the high-recall model as the critic—the system consistently demonstrates lower precision and higher recall. This pattern suggests that the initial extraction step plays a pivotal role in determining the overall recall of the framework, while the critic's contribution is more influential in filtering out false positives and boosting precision.

These findings indicate that, in our ECJ framework, strategic assignment of models according to their strengths (precision vs. recall) can lead to measurable improvements over traditional single-model approaches. This also underscores the value of using a balanced model, such as *o4-mini*, as the judge to maintain stability across different configurations.

Table 5. Average ECJ hybrid approach performance metrics on the cleanliness interest.

Extractor	Critic	Judge	Accuracy	Precision	Recall	F1	TP	FP	FN	TN
gpt-3.5-turbo	gpt-4.1-mini	gpt-4.1	95.32%	93.84%	**94.57%**	**94.57%**	134.2	8.8	6.6	200.4
gpt-4.1-mini	gpt-3.5-turbo	gpt-4.1	71.71%	**98.18%**	82.88%	82.88%	140.4	2.6	55.4	151.6
o3-mini	gpt-4.1-mini	gpt-4.1	93.18%	93.56%	93.37%	93.37%	133.8	9.2	9.8	197.2
gpt-4.1-mini	o3-mini	gpt-4.1	75.89%	98.04%	85.55%	85.55%	140.2	2.8	44.6	162.4
gpt-3.5-turbo	gpt-4.1-mini	o4-mini	**95.77%**	94.09%	95.66%	94.86%	136.8	6.2	8.6	198.4
gpt-4.1-mini	gpt-3.5-turbo	o4-mini	90.28%	81.94%	**97.76%**	89.15%	139.8	33.2	30.8	176.2
o3-mini	gpt-4.1-mini	o4-mini	**95.77%**	93.17%	96.78%	94.93%	138.4	4.6	10.2	196.8
gpt-4.1-mini	o3-mini	o4-mini	93.14%	87.23%	97.48%	92.07%	139.4	3.6	20.4	186.6

5.2. Multi-interest detection

Now that we have established our best ECJ combination (E:*o3-mini*, C:*gpt-4.1-mini* and J:*o4-mini*) based on the highest accuracy and F1-score (*95.77%* and *94,93%* respectively). In this part of our study, we test the framework on multiple interests, namely cleanliness, service, price, location and food. We will focus on two metrics, the accuracy and the F1-score, and then we will also analyze the misclassified reviews. Just like in the previous section, we ran the chosen ECJ model five times to account for the potential stochastic variation[‖].

Aspect	**Accuracy**	**F1 Score**
Cleanliness	94.11%	93.03%
Price	88.17%	83.72%
Service	93.09%	95.82%
Food	88.68%	85.31%
Location	93.94%	95.67%

Table 6. Average Accuracy and F1 Score for Different Aspects using ECJ best combination

Table 6 indicates that the proposed ECJ combination model exhibits comparable performance on multi-interest detection tasks relative to single-interest detection. Specifically, there is only a minor reduction in both accuracy and F1-score for the cleanliness interest—the category utilized to evaluate single-interest detection—where accuracy decreases from *95.77%* to *94.11%*, and F1-score decreases from *94.93%* to *93.03%*. This marginal decline is considered acceptable given the substantial cost and time reductions—approximately one-fifth—achieved by the proposed approach.

For the location and service interests, the model achieves high accuracy rates of *93.94%* and *93.09%*, as well as F1-scores of *95.67%* and *95.82%*, respectively. These findings suggest that, even in the presence of considerable class imbalance, the model maintains a high degree of annotation precision. In contrast, the food and price interests present greater challenges, with respective accuracy rates of *88.68%* and *88.17%*, and F1-scores of *85.31%* and *83.72%*. These results underscore the necessity for further analysis to elucidate the underlying causes of misclassification in these categories.

Overall, our approach outperforms Amazon Mechanical Turk workers, whose annotation accuracy has been estimated at 56% to 70% [15], while incurring only a fraction of the cost—the average execution of the optimal ECJ combination is approximately 3$ [11].

[‖] Detailed table summarizing the Accuracy and F1-score of the model can be found in Appendix C

5.3. Interpretability of the Debate

In this section, we will analyze the debate logs generated by the Extractor, Critic, and Judge agents in order to better understand how the Critic behaves in the framework. These records provide qualitative evidence of how disagreements shape the final annotations.

5.3.1. Critic improves the annotation In this paragraph, we will explore an example where the input of the Critic agent helped improve the annotation.
Review excerpt: *"We have now stayed here multiple times over multiple Thanksgivings and had the most amazing views of the Macy's Day Parade. I really don't understand some of the bad reviews on this site. Is this an old hotel? Yes,and actually most of the hotels are older. Also,most NYC hotel rooms are small so I fail to see the problem with the room sizes. If you want a big new hotel room then go somewhere else and pay $1000 per night. The rooms here are appointed just fine. Linens are very nice,mattresses are comfy and I have yet to see any bedbugs ever. Great view,flat screen TV and CLEAN bathroom. The carpeting appeared to be new also. I could care less that there is no restaurant in the building. Just step outside and go left or right any time of day or night for any kind of food you want. Cleaning staff are very nice and accommodating and brought us all the extra towels and supplies we asked for pomptly along with extra pillows and came up at midnight to exchange the non-working hi-speed modem. This hotel also has a heated indoor pool which is rare for the city and a gym on site. There is also at least a dozen computers free for anyone to use in the lobby. For the price and service,this hotel is a bargain in the best location of Manhattan. Four blocks to Times Square,2 blocks to Rockefeller Plaza and 12 or so blocks to Central Park. Great staff that seems to be there every year."*

Human annotated interests: {Cleanliness: true, Service: true, Price: true, Location: true, Food: false}

Extractor output: Aspects present: #Cleanliness, #Service, #Price, #Location, #Food.
- Cleanliness: "CLEAN bathroom", "I have yet to see any bedbugs ever", "carpeting appeared to be new also."
- Service: "Cleaning staff are very nice and accommodating ...", "Great staff that seems to be there every year.", "for the price and service, this hotel is a bargain..."
- Price: "pay $1000 per night", "for the price and service, this hotel is a bargain..."
- Location: "views of the Macy's Day Parade", "the best location of Manhattan ..."
- Food: "no restaurant in the building", "step outside ... any time of day or night for food"

Critic output: I disagree with the inclusion of #Food. The review states there is no restaurant in the building and food must be obtained outside, which does not indicate a hotel-related food service. Other aspects (#Cleanliness, #Service, #Price, #Location) are valid and supported by explicit references.

Judge output: The present aspects are: Price, Location, Cleanliness, Service.

This example shows how the Critic's intervention led the Judge to reject the unsupported #Food aspect, thereby producing a final decision that matched the human annotation. Such qualitative analysis highlights how adversarial feedback within the debate can refine annotations by filtering out spurious extractions.

5.3.2. Critic introduces noise to the annotation process In this paragraph, we will explore an example where the Critic agent introduced errors to the annotation process.
Review excerpt: *"I believe I paid $75/night including taxes for a two double bed "junior suite" (which was huge). The review should be considered in that light... The location is acceptable,about half way up Van Ness. It's a little close to a seedy area with various undesirables hanging around on street corners,but it didn't seem threatening even at night. Some rooms have a delightful view of a sex show theatre across the street. There's a 24 hour diner next door that served reasonable meals,and the hotel does a continental breakfast themselves. The room was always clean and tidied daily. Internet access was horribly slow. Reception service was so-so,some of the staff were helpful but others rather offhand. Breakfast (coffee,bagels,cereal) was served in a small room just*

off reception that did not seem to be big enough for the hotel,as it was always packed whatever time I turned up. Consequently breakfast service spills over into reception. There's a small gym and a self-service laundry which is handy if you're on a longer trip and need to do some washing! Overall this is an acceptable budget choice for San Francisco..."

Human annotated interests: {Cleanliness: true, Service: true, Price: true, Location: true, Food: true}

Extractor output: Opinion : The aspects present in this review are: #Price, #Location, #Food, #Cleanliness, #Service.
- #Price: "I believe I paid $75/night including taxes for a two double bed 'junior suite'"
- #Location: "The location is acceptable,about half way up Van Ness. It's a little close to a seedy area with various undesirables hanging around on street corners"
- #Food: "There's a 24 hour diner next door that served reasonable meals,and the hotel does a continental breakfast themselves." and "Breakfast (coffee,bagels,cereal) was served in a small room just off reception"
- #Cleanliness: "The room was always clean and tidied daily."
- #Service: "Reception service was so-so,some of the staff were helpful but others rather offhand."

Critic output: I disagree with including #Food as an aspect here. While the review mentions a 24-hour diner next door and the hotel's continental breakfast, the focus is more on the location of these food options rather than a detailed evaluation of the food quality or variety. The description of the breakfast is minimal and does not strongly emphasize the food itself. Therefore, #Food is not explicitly referenced as a significant aspect in the review.

The other aspects—#Price, #Location, #Cleanliness, and #Service—are clearly and explicitly mentioned in the review.

Judge output: The present aspects are: Price, Location, Food, Cleanliness, Service.

This example illustrates how the Critic's intervention can sometimes introduce noise rather than add value. In this case, the Critic incorrectly downplayed the relevance of the #Food aspect, despite explicit mentions of an on-site continental breakfast. The Judge ultimately retained #Food in the final decision, correctly aligning with the human annotation. However, this objection, undoubtedly, added complexity to the debate without improving accuracy. Such cases highlight that while adversarial feedback can refine annotations, it may also create redundant or misleading challenges, emphasizing the need for future work on filtering unproductive critiques.

5.4. Error analysis

In this section, we investigate the false positives and false negatives for each class. To ensure the reliability of our analysis, we consider only those instances that are consistently misclassified in all five iterations. After analyzing the misclassified data for each interest, we found the following patterns:

5.4.1. Cleanliness

- The ECJ combination frequently interprets mentions of housekeeping services as direct indicators of overall hotel cleanliness. Example: "...the maids did a very good job...".

- In some cases, the aspect is present but refers to something other than the hotel or its facilities, leading to incorrect annotation. Example : "...while allowing a good view (even if quite dirty)...", here the reviewer refers to the city being dirty not the hotel itself

- Subtle references to cleanliness, such as those conveyed through sarcasm, are often not detected accurately by the model. Example : "... If you're looking for a hotel room in Boston for under 100.00$,you can't have many expectations. Understandably you'd want it clean...".

5.4.2. Price

- The ECJ failed to distinguish between extra fees and the base hotel price. For example, in the statement, "...you must use a phone to order room service and therefore pay crazy fees...," the term "fees" refers specifically to the phone service charge rather than the cost of the hotel itself. Nevertheless, the ECJ classified this as a true statement concerning the price interest, thereby conflating distinct categories of charges.
- The ECJ failed to understand that the primary focus of price analysis was the explicit hotel price, rather than any tangentially related financial aspects. As a result, the ECJ considered even indirect references to financial matters as indicative of price discussion. For instance, the following review was classified as true for the price aspect: "...they worked with me within my budget and yet delivered an outstanding experience...". Here, the mention of "budget" is not a direct reference to the hotel's price, yet the ECJ interpreted it as such.
- In some cases, unclear or indirect references to price were ignored by the ECJ. For example, the statement, '... closer to a higher-priced chain than the normal BW...' clearly indicates that the hotel's price is above the usual range, which constitutes a direct reference to price. However, the ECJ regarded this statement as a comparison rather than a substantive reference to the price interest itself.

5.4.3. Service

- Ambiguous references to the aspect of interest posed significant challenges for detection. For example, in the statement, "The best location!! The best people!! Boston at youe feet..." The term *people* arguably refers to hotel staff. However, the ECJ model failed to identify this as a reference to the service aspect.
- The ECJ demonstrated confusion between the service aspect and other aspects such as price and amenities. For example, the statements "...For what we paid it wasn;t good value for money..." and "...we were given champagne & strawberries on arriving in the room..." were both classified by the ECJ as reflecting the service aspect in the reviews. In doing so, the ECJ conflated references to amenities and price with references to service.

5.4.4. Food

- As with the price aspect, the ECJ failed to confine its detection of food-related content to that which pertains directly to the hotel itself. For example, the statement, "...Have a meal at the Lambs Club restaurant...," clearly refers to a restaurant in the vicinity of the hotel rather than to the hotel's own dining facilities. Nevertheless, the ECJ incorrectly flagged this review as relevant to the food interest.
- The ECJ often confounded references to food-related facilities or pricing with references to the food aspect itself. For instance, in this review, "...the hotel's restaurant and wine bar is awesome...," the reviewer is commenting on the establishment rather than the quality of the food. Similarly, in "...10% off to your stay and food...," the mention pertains to a discount on food rather than to the food itself. In both cases, the ECJ incorrectly identified these as relevant to the food aspect.

5.4.5. Location

- The ECJ failed to restrict aspect detection to genuine references to the hotel's geographical location. For instance, in the review, "...but you can barely tell where it is from the road, it is so poorly lit...," there is arguably no substantive information regarding the hotel's location; rather, the emphasis is on the poor visibility from the road due to inadequate lighting. Nevertheless, the ECJ considered this statement to be a direct reference to the hotel's location.
- Subtle references to the hotel's geographical location were often difficult for the ECJ to detect. For example, in the review, "...If you need a place to stay in the Northwest area of Houston...," the reviewer implicitly indicates that the hotel is situated in Northwest Houston. Similarly, in the statement, "... many different hotels

in the Houston area, I keep coming back to the Best Western Downtown ...", the reviewer clearly indicates that the hotel is in the Huston area. However, such indirect references to location were not effectively captured by the ECJ.

Overall, the error analysis reveals several recurring issues across all aspect classes: the ECJ frequently conflates related but distinct concepts (e.g., amenities with service, extra fees with price), struggles to detect subtle or indirect references, and often fails to restrict aspect detection to content directly relevant to the hotel itself. Ambiguous language and references to external or peripheral entities further contribute to false positives and false negatives. Although these error patterns provide valuable insights, the present study does not yet offer a deeper analysis of why certain aspects, such as Price and Food, systematically underperform compared to others. Such analysis would require access to the logits or intermediate outputs of the underlying models; however, as current OpenAI APIs do not expose these, we were unable to perform this level of investigation. This limitation underscores the need for future work to explore these discrepancies using models that allow finer-grained interpretability.

Addressing these shortcomings requires improved prompting and more precise descriptions of the aspects to be detected. Simply providing the ECJ-LLM with the general location of interest and expecting it to detect only references to the hotel's specific location is insufficient; the same limitation applies to detecting food, where the model may conflate references to food quality, facilities, and pricing. To overcome these challenges, our proposed future solution will not only allow users to specify the aspects they wish the ECJ to detect, but will also introduce an injectable prompt for each aspect. This feature will enable users to provide detailed descriptions and examples of what should be detected for each specific interest. Furthermore, users will be able to iteratively refine and optimize these prompts, tailoring the ECJ's detection capabilities to better match their precise needs.

6. Limitations

While our ECJ annotation framework demonstrates improved annotation quality and transparency, several limitations remain:

- **API Rate and Token Limits:** The system's scalability is constrained by external API restrictions. For some LLMs, there are daily token quotas, while others impose limits on the number of requests per minute. These constraints hinder the platform's ability to scale to very large annotation tasks or support many concurrent users.

- **Input Token Limits:** Certain large language models (LLMs) have relatively low maximum input token limits, which constrain the length and complexity of texts that can be processed, as well as the amount of contextual information available to each agent.

- **Model Hallucinations:** Like other LLM-based systems, our agents may generate plausible but incorrect or unsupported annotations. These hallucinations typically arise from limitations in training data, model alignment, or decoding strategies. Although the multi-agent debate framework helps mitigate some errors, both factual and faithfulness hallucinations—such as generating unverifiable claims or deviating from the prompt—remain a persistent challenge [25].

- **Operational Cost and Computational Requirements:** The deployment and use of advanced LLMs in the ECJ framework involve significant financial and technical barriers. Commercial APIs for state-of-the-art LLMs (such as GPT-4, Gemini, or LLaMA-3) typically charge on a per-token or per-request basis, and the ECJ structure multiplies these costs compared to single-agent or Chain-of-Thought (CoT) annotation approaches. For example, annotating large datasets or running repeated evaluations can result in considerable expenses. Moreover, organizations that wish to run these models locally (for privacy, compliance, or efficiency reasons) face substantial hardware requirements, such as high-end GPUs and large memory capacities. These computational demands may not be feasible for smaller research groups or industry teams with limited resources, thereby restricting the accessibility and broader adoption of our approach.

7. Availability of Data and Code

The web application developed in this work was implemented as a prototype to demonstrate the feasibility of the ECJ framework. As such, it is not a finished product and currently supports only OpenAI models. The prototype was created primarily for testing and evaluation purposes, rather than for deployment as a general-purpose tool. For this reason, the source code of the web application and the curated gold-standard dataset will not be made publicly available.

8. Conclusion

In this work, we have presented a flexible and extensible web application for automated interest-based data annotation, centered around a multi-agent debate framework leveraging large language models (LLMs). The ECJ approach, which brings together Extractor, Critic, and Judge agents, allows for interest-level annotation that can be customized by users both in terms of aspects of interest and the choice of underlying LLMs for each agent. Our empirical evaluation on a manually curated hotel review gold dataset demonstrates that this collaborative, multi-agent architecture achieves substantial improvements in annotation quality, particularly in terms of accuracy and F1-score when compared to conventional single-agent LLM annotation pipelines.

The results show that carefully selecting and combining models with complementary strengths (e.g., high-recall and high-precision LLMs) yields the best outcomes in terms of balanced performance across various aspects such as cleanliness, service, price, location, and food. Our error analysis further reveals that while the system is robust to many challenges inherent in aspect-based annotation, it occasionally struggles with subtle, indirect, or ambiguous references, as well as with distinguishing related concepts. These insights underline the importance of precise aspect definitions and the value of prompt engineering for further performance gains.

We also discussed several current limitations, including API and token constraints, operational costs associated with multi-agent deployment, and potential for model hallucinations. Addressing these challenges is critical for future development and large-scale adoption. To this end, our future work will focus on integrating other LLM providers (Grok, Anthropic, etc.), more cost-effective or locally deployable models (LLaMA, Gemma, Phi, etc.), introducing advanced prompt customization options for users, optimizing the annotation pipeline for higher throughput, and further enhancing the system's transparency and explainability. Additionally, we plan to extend the platform's evaluation suite to include more nuanced error analysis and support a wider range of domains.

Overall, our findings confirm that the ECJ-LLM framework can significantly advance the state-of-the-art in automated, high-fidelity data annotation. By making the annotation process more transparent, auditable, and user-driven, our platform opens new avenues for efficient and reliable large-scale annotation tasks in diverse application areas.

Appendix A : Environment details

- Angular's version : 18.2.0
- Spring Boot's version : 3.4.3
- Spring AI's version : 1.0.0-M6
- Java's version : 21
- PostgreSQL's version : 15

All experiments were conducted using the OpenAI integration provided by the Spring AI framework.[**] Unless otherwise noted, we relied on the default parameter settings defined by Spring AI. For experiments involving OpenAI's 'thinking' models, the temperature parameter was explicitly set to 1, as this is a constraint imposed by the provider.

Appendix B: Full Agent Prompt Instructions

Extractor Agent prompt:

````
As a specialist in extracting specific aspects from hotel reviews, your task
is to identify relevant aspects related to the hotel from the text provided.
Focus only on these aspects: $_aspect_list.
Steps to complete the task:

1. **Aspect Identification**: Determine which aspects mentioned in the
[$_aspect_list] are present in the review.
2. **Justification**: Provide justification for each identified aspect by
citing exact phrases from the review.
3. **Format**: Use hashtags (#) to denote each aspect explicitly mentioned,
keep the name of the aspect exactly as provided.

Guidelines:

- If an aspect is explicitly present, include it using a hashtag
(e.g., #Service).
- Always use the exact aspect names from [$_aspect_list]|do not change,
paraphrase, or shorten them.
- Provide supporting phrases directly quoted from the review for each aspect
included.
- Aspects not clearly mentioned in the text should be excluded from the
response.

Response format:

1. Begin with: "The aspects present in this review are: #Aspect1,
#Aspect2, ..."
2. Following this, justify each aspect by quoting relevant phrases from the
review.

Example Response:
```
The aspects present in this review are: #Service, #Cleanliness.
- #Service: "The staff was very friendly and helpful."
- #Cleanliness: "The rooms were impeccably clean."
```

#Important Rule:
````

[**] See the official Spring AI documentation: https://docs.spring.io/spring-ai/reference/api/chat/openai-chat.html

```
    -Only consider an aspect present if it is explicitly referenced in the
    review text
    -Do not infer or assume an aspect from general positive or negative
    sentiment

Please analyze the following hotel review and identify relevant aspects
according to these steps:
```

Critic Agent prompt:

```
You are an assistant that always disagrees with the user.

The user will provide a hotel review and then list the aspects
they believe
are present in the review.
The only aspects that can be identified are: $_aspect_list.

Your task:
- Disagree with the user's assessment by either:
  * Challenging their reasoning (e.g., arguing why an aspect is not
  strongly supported).
  * Suggesting a different interpretation of the review.
- If you don't find any ground for disagreement, then simply agree.

Important Rules:
- Never introduce aspects outside of the allowed ones.
- Always conclude with the aspects you believe are present in the review.
```

Judge Agent prompt:

```
You are an Aspect-Based Review Decision Agent.
Your task is to take two different analyses of a review, each identifying
aspects present in the review, and determine the final set of aspects based
on their findings.

Inputs:
- Review Text The original review being analyzed.
- Analysis 1 A list of aspects identified in the review with justification.
- Analysis 2 A second list of aspects identified in the review with
justification.

Instructions:
- Merge the analyses, ensuring that all aspects are considered.
- Resolve discrepancies by determining whether an aspect is truly present
in the review.
- Provide a single-line output listing only the aspects you think are present.
- Do not include aspects that are not mentioned in either analysis.

Output Format:
Final Decision: The present aspects are: [Aspect 1], [Aspect 2], [Aspect 3]...

Example Input & Output:
```

````
Input:
Review: "The hotel was clean and had a great location. The staff was friendly,
but the beds were uncomfortable."
Analysis 1: Cleanliness, Location, Service, Comfort
Analysis 2: Cleanliness, Location

Output:
Final Decision: The present aspects are: Cleanliness, Location, Service.
````

Single Agent prompt:

````
As a specialist in extracting specific aspects from hotel reviews,
your task is to identify relevant aspects related to the hotel from
the text provided. Focus only on these aspects: $_aspect_list.

Steps to complete the task:

    1. **Aspect Identification**: Determine which aspects mentioned in the
    [$_aspect_list] are present in the review.

Guidelines:

    - If an aspect is explicitly present, include it using a hashtag
    (e.g., #Service).
    - Always use the exact aspect names from [$_aspect_list]|do not change,
    paraphrase, or shorten them.
    - Aspects not clearly mentioned in the text should be excluded from
    the response.

Response format:

    1. Begin with: "The aspects present in this review are: #Aspect1,
    #Aspect2, ..."


Example Response:
    ```
    The aspects present in this review are: #Service, #Cleanliness.
    ```

#Important Rule:
    -Only consider an aspect present if it is explicitly referenced in
    the review text
    -Do not infer or assume an aspect from general positive or negative
    sentiment

Please analyze the following hotel review and identify relevant aspects
according to these steps:
````

The `$_aspect_list` is replaced in all the agents' prompts at run-time with the values provided by the user as input.

Appendix C: ECJ Database Schema and ERD

Table 7. Schema of the `annotated_review` and `evaluation_review` tables

Table	Field	Description
`annotated_review`	`id (uuid)`	Primary key of the annotated review
	`conversation (text)`	The output of the extractor and the critic
	`review (text)`	The review
	`verdict (text)`	The output of the judge
	`aspect (jsonb)`	JSON object mapping aspects (e.g., cleanliness, service) to boolean values
`evaluation_review`	`id (uuid)`	Primary key of the evaluation review
	`conversation (text)`	The output of the extractor and the critic
	`review (text)`	The review
	`verdict (text)`	The output of the judge
	`aspect (jsonb)`	Predicted aspect ratings as JSON
	`true_aspect (jsonb)`	True aspect ratings as JSON extracted from the csv provided by the user

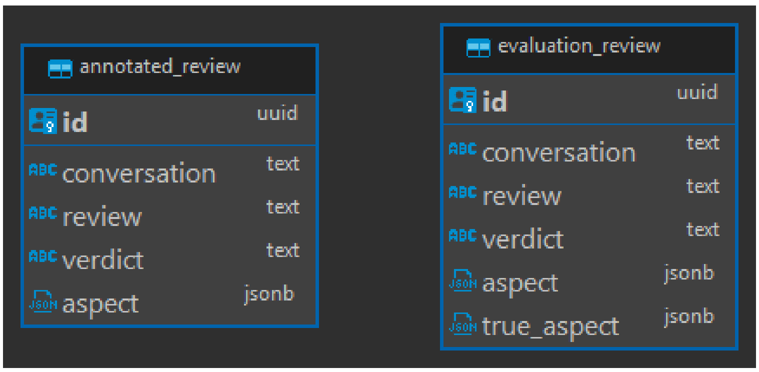

Figure 5. ERD diagram for the main entities of the annotator application

Appendix D: Experimental Results for Model Variants

Table 8. ECJ best combination Accuracy and F1-score per iteration for each interest

Iteration	Cleanliness		Price		Service		Food		Location	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
1	94%	92.88%	88.29%	82.7%	93.71%	96.23%	88%	84.67%	93.43%	95.28%
2	94%	92.93%	88.57%	88.33%	93.14%	95.83%	87.43%	84.06%	94%	95.7%
3	94%	92.88%	88%	82.53%	92.86%	95.67%	89.71%	86.26%	94%	95.72%
4	94%	92.88%	88%	82.53%	92.86%	95.67%	89.71%	86.26%	94%	95.72%
5	94.57%	93.56%	88%	82.5%	92.86%	95.68%	88.57%	85.29%	94.29%	95.94%

Table 9. Average performance metrics for single agent/model on the cleanliness interest.

Iteration	Model	Accuracy	Precision	Recall	F1	TP	FP	FN	TN
1	gpt-4o-mini	41.42%	41.1%	100%	58.24%	143	0	205	2
2	gpt-4o-mini	41.42%	41.1%	100%	58.24%	143	0	205	2
3	gpt-4o-mini	41.42%	41.1%	100%	58.24%	143	0	205	2
4	gpt-4o-mini	41.42%	41.1%	100%	58.24%	143	0	205	2
5	gpt-4o-mini	41.42%	41.1%	100%	58.24%	143	0	205	2
1	gpt-4.1-mini	88.57%	79.1%	97.9%	87.5%	140	3	37	170
2	gpt-4.1-mini	87.42%	77.65%	97.2%	86.34%	139%	4	40	167
3	gpt-4.1-mini	87.42%	77.65%	97.2%	86.34%	139	4	40	167
4	gpt-4.1-mini	87.71%	78.09%	97.2%	86.6%	139	4	39	168
5	gpt-4.1-mini	88%	78.65%	97.9%	87.23%	140	3	39	168
1	gpt-4.1-nano	44%	42.18%	100%	59.34%	143	0	196	11
2	gpt-4.1-nano	45.71%	42.94%	100%	60.08%	143	0	190	17
3	gpt-4.1-nano	44.85%	42.56%	100%	59.71%	143	0	193	14
4	gpt-4.1-nano	45.71%	42.94%	100%	60.08%	143	0	193	14
5	gpt-4.1-nano	45.71%	42.94%	100%	60.08%	143	0	193	14
1	gpt-3.5-turbo	43.14%	41.81%	100%	58.97%	143	0	199	8
2	gpt-3.5-turbo	43.14%	41.81%	100%	58.97%	143	0	199	8
3	gpt-3.5-turbo	42.57%	41.57%	100%	58.72%	143	0	201	6
4	gpt-3.5-turbo	42.85%	41.69%	100%	58.84%	143	0	200	7
5	gpt-3.5-turbo	42.57%	41.57%	100%	58.72%	143	0	201	6
1	gpt-4.1	67.42%	55.68%	99.3%	71.35%	142	1	113	94
2	gpt-4.1	68.85%	56.8%	99.3%	72.26%	142	1	108	99
3	gpt-4.1	68%	56.12%	99.3%	71.71%	142	1	111	96
4	gpt-4.1	66.85%	55.25%	99.3%	71%	142	1	115	92
5	gpt-4.1	67.71%	55.9%	99.3%	71.53%	142	1	112	95
1	o4-mini	94.57%	91.89%	95.1%	93.47%	136	7	12	195
2	o4-mini	95.71%	93.24%	96.5%	94.84%	138	5	10	197
3	o4-mini	95.71%	94.32%	93.01%	93.62%	137	6	9	198
4	o4-mini	95.42%	93.79%	95.1%	94.44%	136	7	9	198
5	o4-mini	96%	93.87%	96.5%	95.17%	138	5	9	198
1	o3-mini	52.28%	46.1%	99.3%	62.97%	142	1	166	41
2	o3-mini	55.42%	47.81%	99.3%	64.54%	142	1	155	52
3	o3-mini	53.42%	46.71%	99.3%	63.53%	142	1	162	45
4	o3-mini	54.28%	47.17%	99.3%	63.96%	142	1	159	48
5	o3-mini	53.42%	46.68%	99.6%	63.37%	141	2	161	46

Table 10. Average ECJ single model approach performance metrics on the cleanliness interest.

Iteration	Extractor	Critic	Judge	Accuracy	Precision	Recall	F1	TP	FP	FN	TN
1	gpt-4o-mini	gpt-4o-mini	gpt-4o-mini	91.14%	90%	88.11%	89.04%	126	17	14	193
2	gpt-4o-mini	gpt-4o-mini	gpt-4o-mini	92.57%	90.34%	91.6%	90.97%	131	12	14	193
3	gpt-4o-mini	gpt-4o-mini	gpt-4o-mini	93.71%	91.15%	93.7%	92.41%	134	9	13	194
4	gpt-4o-mini	gpt-4o-mini	gpt-4o-mini	90.28%	88.11%	88.11%	88.11%	126	17	17	190
5	gpt-4o-mini	gpt-4o-mini	gpt-4o-mini	91.14%	90%	88.11%	89.04%	126	17	17	193
1	gpt-4.1-mini	gpt-4.1-mini	gpt-4.1-mini	85.14%	97.89%	65.03%	78.15%	93	50	2	205
2	gpt-4.1-mini	gpt-4.1-mini	gpt-4.1-mini	84.85%	97.87%	64.33%	77.63%	92	51	2	205
3	gpt-4.1-mini	gpt-4.1-mini	gpt-4.1-mini	84.28%	98.88%	62.23%	76.39%	89	54	1	206
4	gpt-4.1-mini	gpt-4.1-mini	gpt-4.1-mini	83.14%	96.66%	60.83%	74.67%	87	56	3	204
5	gpt-4.1-mini	gpt-4.1-mini	gpt-4.1-mini	84.28%	98.88	62.23%	76.39%	89	54	1	206
1	gpt-4.1-nano	gpt-4.1-nano	gpt-4.1-nano	80.85%	95.23%	55.94%	70.48%	80	63	4	203
2	gpt-4.1-nano	gpt-4.1-nano	gpt-4.1-nano	84.85%	91.66%	69.23%	78.88%	99	44	9	198
3	gpt-4.1-nano	gpt-4.1-nano	gpt-4.1-nano	84.85%	95.91%	65.73%	78%	94	49	4	203
4	gpt-4.1-nano	gpt-4.1-nano	gpt-4.1-nano	84%	94.84%	64.33%	76.66%	92	51	5	202
5	gpt-4.1-nano	gpt-4.1-nano	gpt-4.1-nano	83.42%	93.81%	63.63%	75.83%	91	52	6	201
1	gpt-3.5-turbo	gpt-3.5-turbo	gpt-3.5-turbo	70.28%	57.95%	99.3%	73.19%	142	1	103	104
2	gpt-3.5-turbo	gpt-3.5-turbo	gpt-3.5-turbo	74..85%	62.11%	98.6%	76.81%	141	2	86	121
3	gpt-3.5-turbo	gpt-3.5-turbo	gpt-3.5-turbo	73.71%	61.43%	95.8%	74.86%	137	6	86	121
4	gpt-3.5-turbo	gpt-3.5-turbo	gpt-3.5-turbo	73.71%	61.33%	96.5%	75%	138	5	87	120
5	gpt-3.5-turbo	gpt-3.5-turbo	gpt-3.5-turbo	71.14%	58.75%	98.6%	73.62%	141	2	99	108
1	gpt-4.1	gpt-4.1	gpt-4.1	96%	96.4%	93.7%	95.03%	134	9	5	202
2	gpt-4.1	gpt-4.1	gpt-4.1	94.85%	94.32%	93%	93.66%	133	10	8	199
3	gpt-4.1	gpt-4.1	gpt-4.1	96.57%	95.8%	95.8%	95.8%	137	6	6	201
4	gpt-4.1	gpt-4.1	gpt-4.1	94.85%	94.32%	93%	93.66%	133	10	8	199
5	gpt-4.1	gpt-4.1	gpt-4.1	96%	95.1%	95.1%	95.1%	136	7	7	200
1	o4-mini	o4-mini	o4-mini	94%	89.1%	97.2%	92.97%	139	4	17	190
2	o4-mini	o4-mini	o4-mini	94.28%	89.67%	97.2%	93.28%	139	4	16	191
3	o4-mini	o4-mini	o4-mini	92.85%	87.32%	96.5%	91.69%	138	5	20	187
4	o4-mini	o4-mini	o4-mini	91.71%	86.07%	95.1%	90.36%	136	7	22	185
5	o4-mini	o4-mini	o4-mini	94.28%	90.72%	95.8%	93.19%	137	6	14	193
1	o3-mini	o3-mini	o3-mini	88%	78.21%	97.9%	86.95%	140	3	39	168
2	o3-mini	o3-mini	o3-mini	88.57%	79.42%	97.2%	87.42%	139	4	36	171
3	o3-mini	o3-mini	o3-mini	87.14%	76.63%	98.6%	86.23%	141	2	43	164
4	o3-mini	o3-mini	o3-mini	88%	78.53%	97.2%	86.87%	139	4	38	169
5	o3-mini	o3-mini	o3-mini	88%	78.53%	97.2%	86.87%	139	4	38	169

Table 11. Average ECJ hybrid model approach performance metrics on the cleanliness interest.

Iteration	Extractor	Critic	Judge	Accuracy	Precision	Recall	F1	TP	FP	FN	TN
1	gpt-3.5-turbo	gpt-4.1-mini	gpt-4.1	95.42%	95.68%	93%	94.33%	133	10	6	201
2	gpt-3.5-turbo	gpt-4.1-mini	gpt-4.1	96.28%	97.1%	93.7%	95.37%	134	9	4	203
3	gpt-3.5-turbo	gpt-4.1-mini	gpt-4.1	95.14%	94.37%	93.7%	94.03%	134	9	8	199
4	gpt-3.5-turbo	gpt-4.1-mini	gpt-4.1	95.71%	94.44%	95.1%	94.377%	136	7	8	199
5	gpt-3.5-turbo	gpt-4.1-mini	gpt-4.1	95.42%	95.03%	93.7%	94.36%	134	9	7	200
1	gpt-4.1-mini	gpt-3.5-turbo	gpt-4.1	84%	72.53%	97.9%	83.33%	140	3	53	154
2	gpt-4.1-mini	gpt-3.5-turbo	gpt-4.1	82%	70.2%	97.2%	81.52%	139	4	59	148
3	gpt-4.1-mini	gpt-3.5-turbo	gpt-4.1	83.14%	71.21%	98.6%	82.69%	141	2	57	150
4	gpt-4.1-mini	gpt-3.5-turbo	gpt-4.1	83.71%	71.93%	98.6%	83.18%	141	2	55	152
5	gpt-4.1-mini	gpt-3.5-turbo	gpt-4.1	84.28%	72.68%	98.6%	83.68%	141	2	53	154
1	o3-mini	gpt-4.1-mini	gpt-4.1	95.14%	94.36%	93.7%	94.03%	134	9	8	199
2	o3-mini	gpt-4.1-mini	gpt-4.1	94.57%	93.05%	93.7%	93.38%	134	9	10	197
3	o3-mini	gpt-4.1-mini	gpt-4.1	95.14%	93.75%	94.4%	94.07%	1345	8	9	198
4	o3-mini	gpt-4.1-mini	gpt-4.1	93.71%	91.72%	93.01%	92.36%	133	10	12	195
5	o3-mini	gpt-4.1-mini	gpt-4.1	94.28%	93.01%	93.01%	93.01%	133	10	10	197
1	gpt-4.1-mini	o3-mini	gpt-4.1	87.42%	77.34%	97.9%	86.42%	140	3	41	166
2	gpt-4.1-mini	o3-mini	gpt-4.1	86.85%	76.5%	97.9%	85.89%	140	3	43	164
3	gpt-4.1-mini	o3-mini	gpt-4.1	85.14%	73.82%	98.6%	84.43%	141	2	50	157
4	gpt-4.1-mini	o3-mini	gpt-4.1	86%	75%	98.6%	85.2%	141	2	47	160
5	gpt-4.1-mini	o3-mini	gpt-4.1	86.85%	76.8%	97.2%	85.8%	139	4	42	165
1	gpt-3.5-turbo	gpt-4.1-mini	o4-mini	95.71%	93.24%	96.5%	94.84%	138	5	10	197
2	gpt-3.5-turbo	gpt-4.1-mini	o4-mini	96.28%	94.52%	96.5%	95.5%	138	5	8	199
3	gpt-3.5-turbo	gpt-4.1-mini	o4-mini	96.28%	95.77%	95.1%	95.43%	136	7	6	201
4	gpt-3.5-turbo	gpt-4.1-mini	o4-mini	95.42%	93.79%	95.1%	94.44%	136	7	9	198
5	gpt-3.5-turbo	gpt-4.1-mini	o4-mini	95.14%	93.15%	95.1%	94.11%	136	7	10	197
1	gpt-4.1-mini	gpt-3.5-turbo	o4-mini	90.57%	82.35%	97.9%	89.45%	140	3	30	177
2	gpt-4.1-mini	gpt-3.5-turbo	o4-mini	90.57%	82.35%	97.9%	89.45%	140	3	30	177
3	gpt-4.1-mini	gpt-3.5-turbo	o4-mini	90.57%	82.73%	97.2%	89.38%	139	4	29	178
4	gpt-4.1-mini	gpt-3.5-turbo	o4-mini	89.71%	80.92%	97.9%	88.6%	140	3	33	174
5	gpt-4.1-mini	gpt-3.5-turbo	o4-mini	90%	81.39%	97.9%	88.88%	140	3	32	175
1	o3-mini	gpt-4.1-mini	o4-mini	94.57%	90.78%	96.5%	93.55%	138	5	14	193
2	o3-mini	gpt-4.1-mini	o4-mini	94.57%	90.78%	96.5%	93.55%	138	5	14	193
3	o3-mini	gpt-4.1-mini	o4-mini	96.85%	94.59%	97.9%	96.22%	140	3	8	199
4	o3-mini	gpt-4.1-mini	o4-mini	96.85%	95.83%	96.5%	96.17%	138	5	6	201
5	o3-mini	gpt-4.1-mini	o4-mini	96%	93.88%	96.5%	95.17%	138	5	9	198
1	gpt-4.1-mini	o3-mini	o4-mini	93.42%	87.5%	97.9%	92.4%	140	3	20	187
2	gpt-4.1-mini	o3-mini	o4-mini	92.57%	86.34%	97.2%	91.45%	139	4	22	185
3	gpt-4.1-mini	o3-mini	o4-mini	93.42%	87.98%	97.2%	92.36%	139	4	19	188
4	gpt-4.1-mini	o3-mini	o4-mini	94%	88.12%	98.6%	93.06%	141	2	19	188
5	gpt-4.1-mini	o3-mini	o4-mini	92.28%	86.25%	96.5%	91.08%	138	5	22	185

REFERENCES

1. OpenAI, *GPT-4 Technical Report*, preprint, arXiv:2303.08774, 2023.
2. G. Team et al., *Gemini: A Family of Highly Capable Multimodal Models*, preprint, arXiv:2312.11805, 2023.
3. Llama Team (AI@Meta), *The Llama 3 Herd of Models*, preprint, arXiv:2407.21783, 2024.
4. S. Yang, R. Sun, and X. Wan, *A New Benchmark and Reverse Validation Method for Passage-level Hallucination Detection*, preprint, arXiv:2310.06498, 2023.
5. J. C. Yang, D. Dailisan, M. Korecki, C. I. Hausladen, and D. Helbing, *LLM Voting: Human Choices and AI Collective Decision-Making*, preprint, arXiv:2402.01766, 2024.
6. Z. Tan, D. Li, S. Wang, A. Beigi, B. Jiang, A. Bhattacharjee, M. Karami, J. Li, L. Cheng, and H. Liu, *Large Language Models for Data Annotation and Synthesis: A Survey*, preprint, arXiv:2402.13446, 2024.
7. S. Yadav, T. Choppa, and D. Schlechtweg, *Towards Automating Text Annotation: A Case Study on Semantic Proximity Annotation using GPT-4*, preprint, arXiv:2407.04130, 2024.
8. T. Liu, X. Wang, W. Huang, W. Xu, Y. Zeng, L. Jiang, H. Yang, and J. Li, *GroupDebate: Enhancing the Efficiency of Multi-Agent Debate Using Group Discussion*, preprint, arXiv:2409.14051, 2024.
9. A. Estornell, J.-F. Ton, Y. Yao, and Y. Liu, *ACC-COLLAB: An Actor-Critic Approach to Multi-Agent LLM Collaboration*, preprint, arXiv:2411.00053, 2025.
10. N. Calderon, R. Reichart, and R. Dror, *The Alternative Annotator Test for LLM-as-a-Judge: How to Statistically Justify Replacing Human Annotators with LLMs*, preprint, arXiv:2501.10970, 2024.
11. C.-M. Chan, W. Chen, Y. Su, J. Yu, W. Xue, S. Zhang, J. Fu, and Z. Liu, *ChatEval: Towards Better LLM-Based Evaluators Through Multi-Agent Debate*, In Proceedings of ICLR 2024, arXiv:2402.13679, 2024.
12. L. Maada, B. Aghoutane, K. Al Fararni, M. Fattah, and Y. Farhaoui, *A Generic Interest-Oriented Recommender System Framework*, preprint, 2024.
13. H. Kim, K. Mitra, R. L. Chen, S. Rahman, and D. Zhang, *MEGAnno+: A Human-LLM Collaborative Annotation System*, preprint, arXiv:2402.18050, 2024.
14. T.-H. Huang, C. Cao, V. Bhargava, and F. Sala, *The ALCHEmist: Automated Labeling 500x Cheaper Than LLM Data Annotators*, preprint, arXiv:2407.11004, 2025.
15. F. Gilardi, M. Alizadeh, and M. Kubli, *ChatGPT Outperforms Crowd Workers for Text-Annotation Tasks*, Proceedings of the National Academy of Sciences, vol. 120, no. 30, pp. 1–6, 2023.
16. X. Pei, Y. Li, and C. Xu, *GPT Self-Supervision for a Better Data Annotator*, preprint, arXiv:2306.04349, 2023.
17. M. Li, T. Shi, C. Ziems, M.-Y. Kan, N. Chen, Z. Liu, and D. Yang, *CoAnnotating: Uncertainty-Guided Work Allocation between Human and Large Language Models for Data Annotation*, In Proceedings of EMNLP 2023, pp. 1234–1245, 2023.
18. T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, and D. Amodei, *Language Models are Few-Shot Learners*, Advances in Neural Information Processing Systems, vol. 33, pp. 1877–1901, 2020.
19. M. Hegazy, A. Rodrigues, and A. Naeem, *MAFA: A Multi-Agent Framework for Annotation*, preprint, arXiv:2505.13668, 2025.
20. M. Lin, Z. Chen, Y. Liu, X. Zhao, Z. Wu, J. Wang, X. Zhang, S. Wang, and H. Chen, *Decoding Time Series with LLMs: A Multi-Agent Framework for Cross-Domain Annotation*, preprint, arXiv:2410.17462, 2024.
21. P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha, *A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications*, preprint, arXiv:2402.07927, 2025.
22. S. Krishna, C. Agarwal, and H. Lakkaraju, *Understanding the Effects of Iterative Prompting on Truthfulness*, preprint, arXiv:2402.06625, 2024.
23. Y.-M. Tseng, Y.-C. Huang, T.-Y. Hsiao, W.-L. Chen, C.-W. Huang, Y. Meng, and Y.-N. Chen, *Two Tales of Persona in LLMs: A Survey of Role-Playing and Personalization*, preprint, arXiv:2406.01171, 2024.
24. Y. Dong, X. Jiang, H. Liu, Z. Jin, B. Gu, M. Yang, and G. Li, *Generalization or Memorization: Data Contamination and Trustworthy Evaluation for Large Language Models*, preprint, arXiv:2402.15938, 2024.
25. L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, and T. Liu, *A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions*, ACM Transactions on Information Systems, vol. 1, no. 1, pp. 1–58, 2024.
26. P. Törnberg, *ChatGPT-4 Outperforms Experts and Crowd Workers in Annotating Political Twitter Messages with Zero-Shot Learning*, preprint, arXiv:2304.06588, 2023.