Auto Review: Second Stage Error Detection for Highly Accurate Information Extraction from Phone Conversations

Ayesha Qamar, Arushi Raghuvanshi, Conal Sathi, and **Youngseo Son**
Infinitus Systems, Inc.
{ayesha, arushi, conal, youngseo.son}@infinitus.ai

Abstract

Automating benefit verification phone calls saves time in healthcare and helps patients receive treatment faster. It is critical to obtain highly accurate information in these phone calls, as it can affect a patient's healthcare journey. Given the noise in phone call transcripts, we have a two-stage system that involves a post-call review phase for potentially noisy fields, where human reviewers manually verify the extracted data—a labor-intensive task. To automate this stage, we introduce ***Auto Review***, which significantly reduces manual effort while maintaining a high bar for accuracy. This system, being highly reliant on call transcripts, suffers a performance bottleneck due to automatic speech recognition (ASR) issues. This problem is further exacerbated by the use of domain-specific jargon in the calls. In this work, we propose a second-stage postprocessing pipeline for accurate information extraction. We improve accuracy by using multiple ASR alternatives and a pseudo-labeling approach that does not require manually corrected transcripts. Experiments with general-purpose large language models and feature-based model pipelines demonstrate substantial improvements in the quality of corrected call transcripts, thereby enhancing the efficiency of ***Auto Review***.

1 Introduction

A key use case for Conversational AI systems in industry is collecting information (Gnewuch et al., 2017). One critical application is healthcare benefit verification, where information about a patient's insurance coverage is gathered from an insurance company over the phone. These extracted values, such as patient group numbers and drug coverage details, are essential for treatment approval and directly impact a patient's healthcare journey (Buker, 2023). Given the high-stakes nature of this task, ensuring the accuracy of extracted data is crucial.

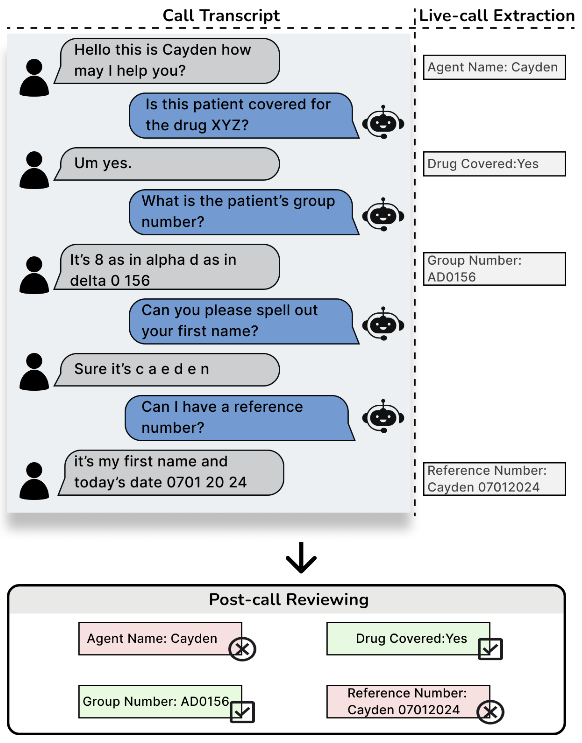

Call Transcript
Live-call Extraction
Hello this is Cayden how may I help you?
Is this patient covered for the drug XYZ?
Um yes.
What is the patient's group number?
It's 8 as in alpha d as in delta 0 156
Can you please spell out your first name?
Sure it's c a e d e n
Can I have a reference number?
it's my first name and today's date 0701 20 24
Agent Name: Cayden
Drug Covered:Yes
Group Number: AD0156
Reference Number: Cayden 07012024
Post-call Reviewing
Agent Name: Cayden
Drug Covered:Yes
Group Number: AD0156
Reference Number: Cayden 07012024

Figure 1: An excerpt from a dummy chat, along with the field values extracted during the call, is passed to the post-call reviewing module for verification. The noisy ASR transcripts can contribute to errors in the extracted data; this is exacerbated for domain-specific jargon such as group number and rare agent names.

While extensive research has focused on conversation navigation techniques—such as intent prediction, slot filling, and dialogue state tracking (McTear, 2022)—there has been comparatively less emphasis on ensuring the accuracy of extracted information in AI-driven conversations with task-specific context. In real-world applications, automated phone call outputs often contain errors due to ASR challenges, including background noise, domain-specific jargon, and complex alphanumeric sequences. To maintain data reliability, it is crucial to incorporate automated error correction methods or human-in-the-loop verification where necessary. Unlike prior work that focuses on ASR error correction for grammatical mistakes, our goal is to improve the accuracy of extracted informational fields. Since creating datasets for ASR error correction is time-consuming and labor-intensive, we propose using a pseudo-labeling technique with Large Language Models (LLMs).

Given the real-time constraints of compute and latency during live calls, we introduce **Auto Review**, a two-stage pipeline that enhances post-call information extraction. The first stage involves a conversational AI system that navigates live calls and extracts key field values. However, it does not guarantee that the extracted values are highly accurate. The second stage performs an automated review, flagging potential errors for human review or approving the accurate values. This second stage significantly reduces manual human review time while maintaining high accuracy.

We evaluate LLMs as a reviewing agent in two distinct settings: direct verification, where a model determines whether an extracted field value in the first stage is correct, and direct extraction, where a model identifies the correct value directly from the transcript. We compare multiple LLMs and feature-based models, analyzing their trade-offs in precision, recall, and computational efficiency.

The main contributions of this paper can be summarized as:

- We introduce a two-stage pipeline for accurate and efficient information extraction in the healthcare benefit verification domain. This approach saves human review time while ensuring high accuracy in the final outputs delivered to clients.
- To address domain-specific errors in ASR transcripts, we propose a pseudo-label generation technique leveraging LLMs.
- We conduct a comprehensive evaluation of LLMs for information verification in both generative and discriminative settings, analyzing the trade-offs between the two approaches.

2 Related Work

ASR Error Correction Most research on ASR error detection and correction focuses on grammatical mistakes (Li and Wang, 2024; Ma et al., 2023). Loem et al. (2023) demonstrated that GPT-3, in zero-shot and few-shot settings, can perform grammatical error correction. Davis et al. (2024) used LLM prompting techniques to address grammatical issues, while Wang et al. (2024) combined rule-based methods with generative models to introduce artificial errors that mimic real-world patterns. Shen et al. (2022) highlighted how the scarcity of errors in training data limits a model's ability to correct them effectively. Unlike these approaches, our focus is on correcting informational fields rather than grammatical issues. We leverage domain-specific context and frequent ASR error patterns to improve accuracy in benefit verification.

Previous work has focused on correcting named entity errors in ASR text. For instance, Pusateri et al. (2024) use a retrieval-augmented approach, while Saebi et al. (2021) leverage external knowledge sources like knowledge graphs. However, in our healthcare phone conversations, sensitive and context-dependent information (e.g., personal health data) is often not available in public knowledge bases and can only be captured live during the call.

Many studies use supervised fine-tuning as a post-processing step to reduce ASR errors (Errattahi et al., 2016; Radhakrishnan et al., 2023). Some approaches (Ebadi et al., 2024) avoid relying on manually corrected transcripts by using the inherent knowledge of LLMs to correct errors. In contrast, we don't have manually corrected transcripts, and few-shot LLMs were ineffective, as they haven't been exposed to our domain-specific data during pre-training.

Output Extraction Dialogue state tracking (DST) in task-oriented dialogues involves intent recognition, which can be viewed as output extraction based on the user turns (Li et al., 2024). This process fills predefined slot-value pairs according to the domain and task requirements. In healthcare benefit verification, this translates to extracting specific fields necessary to confirm patient benefits (Feng et al., 2023). Retrieval-augmented strategies have been explored for DST (King and Flanigan, 2023), and LLMs have been applied to intent and entity extraction for live conversations (Luo et al., 2024). While our first-stage live call system incorporates elements of these approaches, it does not achieve the required accuracy given our healthcare-specific constraints on latency and compute resources. To address this, we introduce a second-stage system that refines outputs in a post-processing step, improving overall accuracy.

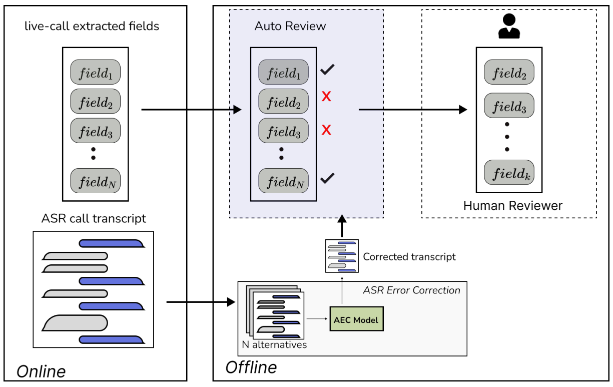


Figure 2: The *auto review* pipeline consists of an online and an offline component. The fields that do not get auto-approved are passed to a human reviewer for correction.

Field	Error Rates	Mean Edit	STDV
Agent Name	10.80%	3.23	2.89
Reference Number	12.90%	7.05	6.43
Group Number	9.80%	3.76	7.76

Table 1: Error rates denote the ratio of incorrectly extracted live-call values for each field. Mean edit and STD denote mean and standard deviations of edit distances of live-call extracted values that contain errors.

Dataset Type	Calls	AVG	STDV
Train	6,652	907	316.09
Validation	383	926	329.26
Test	2,260	939	356.79

Table 2: Patients benefit verification phone calls. AVG: average number of words, STDV: standard deviation.

3 Two Stage Pipeline for Highly Accurate Information Extraction

Our automation pipeline for verifying patient insurance benefits involves two stages. First, a live-call conversational AI model engages with an insurance representative to collect the necessary benefit information. Second, an auto-review AI model validates the collected data based on the full call context, patient details, and domain knowledge.

The goal is to ensure the accuracy of the information and automate healthcare processes. In cases where the data may be uncertain, a human is brought in for review. For high-confidence fields, we can automatically approve the data, significantly reducing human involvement and improving operational efficiency without compromising quality.

In the second stage, the auto-review AI models verify the accuracy of the information collected. We define auto-reviewing as the process of assessing whether each extracted value from a call transcript is correct. As shown in the conversation snapshot in Figure 1, some information may be updated or corrected during the call.

To support large-scale industrial deployment, we prioritized cost-effective model design, considering trade-offs between model complexity and performance. Our objective is to deploy efficient and scalable models that maintain comparable performance to larger alternatives, as long as differences are not statistically significant. The models evaluated in this paper represent a simplified component of a broader production pipeline used in our industrial setting.

4 Data Description

We collected 9,456 benefit verification calls between February and July 2024 for our experiments. Calls from February 1st to July 3rd were used for training, calls from July 5th for validation, and calls from July 10th to 12th for evaluation[1]. The dataset details are given in Table 2. The dataset includes call audio, ASR transcripts, extracted field values, and human-verified gold field values.

The field values in our healthcare domain include alphanumeric strings (e.g., insurance agent name, patient group number), booleans (e.g., medication coverage), and dates (e.g., effective dates of insurance plans). Alphanumeric fields typically exhibit the highest error rates due to ASR mistranscriptions caused by homophones, background noise, and similar-sounding names. We focus on alphanumeric fields for three reasons: 1) they have the highest correction rates, 2) they vary greatly in value, and 3) they are most prone to ASR errors. Therefore, we discuss three key alphanumeric fields with the highest correction rates: Agent Name, Reference Number, and Group Number[2]. The first-stage conversational AI models were generally accurate, with target output fields having an error correction rate of 10-13%, and their mean edit distances ranging from 3.23 to 7.05 (see Table 1).

5 Auto-Review Model

We developed two primary approaches for automatically reviewing benefit information, both of which take the call transcript as input. The first,

[1]No calls were collected over the weekend.

[2]Multimodal LLMs performed poorly when directly extracting from call audio recordings (see C.1).

Direct Extraction, extracts the field values, while the second, ***Direct Verification***, uses the live-call values and determines, in a discriminative setting, whether they are correct.

5.1 Direct Verification

In this approach, both the transcript and the live-call field value are provided as input. The *live-call* value is defined as the field value extracted by our real-time system, which may also involve human in the loop. This setting is akin to binary classification.

Input: [Transcript][Live-call Extracted Field Value] Is the field value correct? Output: Yes/No

5.2 Direct Extraction

Here, the model receives the call transcript along with the field name and is tasked with extracting the relevant value from the transcript. The value extracted in this setting is referred to as the *post-call* value.

Input: [Transcript] What is the field value? Output: Post-call Extracted Field Value

After the extraction, we convert the task back to a review process by comparing the extracted field value with the live-call field value. If the live-call field value matches the post-call extracted value, we consider it to be correct.

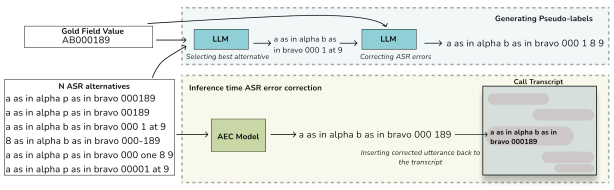
5.3 Error Patterns

A major source of incorrect predictions at this stage stems from errors in the call transcripts, which can result in either incorrect field values being approved or correct ones being missed.

Our task faces two main challenges: 1) detecting errors in call-level field extraction, which is a highly imbalanced classification problem, and 2) auto-correcting detected errors, which requires understanding ASR error patterns. One common error pattern involves similar pronunciations, such as a mistranscribed reference number (Rina A 01012024 instead of Sabrina A 01012024). Another common issue arises from inaccurate long sequence transcripts, such as missing or redundant digits (e.g., ‘10001234’ missing a 0, or ‘1234560’ with an extra 0). These ASR errors present a bottleneck for the auto-review process.

6 Error Handling

Traditional ASR error correction models aim to detect and correct all errors in a transcript (Lu et al., 2019). In contrast, our focus is not on correcting



Gold Field Value
AB000189
N ASR alternatives
a as in alpha p as in bravo 000189
a as in alpha p as in bravo 00189
a as in alpha b as in bravo 000 1 at 9
8 as in alpha b as in bravo 000-189
a as in alpha p as in bravo 000 one 8 9
a as in alpha p as in bravo 00001 at 9
Generating Pseudo-labels
LLM
Selecting best alternative
a as in alpha b as in bravo 000 1 at 9
LLM
Correcting ASR errors
a as in alpha b as in bravo 000 1 8 9
Inference time ASR error correction
AEC Model
a as in alpha b as in bravo 000 189
Inserting corrected utterance back to the transcript
Call Transcript
a as in alpha b as in bravo 000189

Figure 3: An overview of the ASR error handling component. n ASR alternatives are used to generate the pseudo-labels that are then used for training the AEC model. During inference, the corrected utterances are inserted back into the transcript.

Algorithm 1 Correcting ASR transcript using gold field value

```
1: Input: $ASR_N$ (list of ASR alternatives), $field_{gold}$ (corrected field value)
2: $ASR_{best} \leftarrow f^{LLM}_{best\_alternative}(ASR_N, field_{gold})$
3: $ASR_{corr} \leftarrow f^{LLM}_{correct\_transcript}(ASR_{best}, field_{gold})$
4: return $ASR_{corr}$
```

grammatical errors, but on ensuring the accuracy of the information relevant to benefit verification. As noted in recent studies (Zhu et al., 2021), using n-best alternatives significantly improves error correction. In our experiments, providing multiple transcript alternatives improves data extraction performance. Therefore, we use n-alternatives at both the pseudo-label generation and error correction stages[3]

6.1 Generating Pseudo-Labels

Manually curating an error correction dataset from a large number of calls is expensive and time-consuming. Instead, we leverage existing ASR transcripts and human-reviewed field values from past calls to create a specialized dataset for error correction.

To generate pseudo-labels, we prompt an LLM to correct noisy transcripts so that the information aligns with the gold field value. In initial experiments, we found that when multiple errors were present in a transcript[4], the LLM struggled to correct all of them. To address this, we use n-alternatives and break pseudo-label generation into two steps. First, we provide the LLM[5] with all n-alternatives and the gold field value, asking it to choose the best alternative, we formalize this as $f^{LLM}_{best\_alternative}(ASR_N, field_{gold})$. Then, using the selected alternative and the gold value, we prompt the LLM again to correct the transcript, we call this function $f^{LLM}_{correct\_transcript}(ASR_{best},$

[3]Please refer to C.1 and C.3 for more details about our main model architecture decision.

[4]The best transcript returned by the ASR model may not be the most accurate for benefit verification.

[5]We use the Gemini model for generating pseudo-labels.

Model	Agent Name			Reference Number			Group Number		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
XGBoost	0.9570	0.6617	0.7824	0.9636	0.8598	0.9088	0.9749	0.8969	0.9343
XGBoost + AED	0.9494	0.7634	0.8463	0.9732	0.8637	0.9152	0.9532	0.7523	0.8409
XGBoost + AEC	0.9567	0.6682	0.7868	0.9739	0.8506	0.9081	0.9562	0.6605	0.7813
XGBoost + AED + AEC	0.9508	0.7569	0.8429	0.9689	0.8773	0.9208	0.9531	0.7405	0.8335
Gemini 1.5	0.9563	0.8472	0.8985	0.9499	0.7541	0.8408	0.9796	0.6656	0.7927
Gemini 1.5 + AEC	0.9602	0.8011	0.8734	0.9569	0.7221	0.8231	0.9815	0.5979	0.7431
GPT 3.5	0.9373	0.8829	0.9093	0.9355	0.7953	0.8598	0.9493	0.9508	0.9500
GPT 3.5 + AEC	0.9415	0.8626	0.9003	0.9432	0.8138	0.8737	0.9506	0.9574	0.9540
Fine-tuned GPT 3.5 + AEC	0.9192	0.9985	**0.9572***	0.9386	0.9942	**0.9656***	0.9556	0.9933	**0.9741***

Table 3: Model performance for the *Direct Verification* setting in correctly reviewing Agent Name, Reference Number, and Group Number. Fine-tuned GPT 3.5 + AEC refers to the model fine-tuned for auto-reviewing using corrected transcripts. The results highlighted in gray are from the fine-tuned model, all other models have not been fine-tuned. (AED: ASR Error Detection, AEC: ASR Error Correction, GPT 3.5: GPT 3.5 Turbo). McNemar's tests were conducted on the best-performing model for each field against its baseline (XGBoost), and all comparisons showed statistically significant improvements ($* : p < 0.001$)

$field_{gold}$). Figure 3 gives the workflow on pseudo-label generation. In all experiments, we set $n = 10$ [6]. Detailed prompts are described in Appendix D, and the algorithm for locating utterances is presented in Appendix B.

6.2 Automatic Error Correction Model

For the ASR Error Correction (AEC) model, we use Mistral (Jiang et al., 2023) as the base model for error handling tasks[7]. The AEC model focuses exclusively on correcting utterances containing key field values. We first isolate those utterances for each field type. The corresponding pseudo-labels are generated only during the training phase. We provide n alternatives as input to the model and train using the pseudo-labels. Given the n alternatives, the AEC model is trained to output a single correct transcript. After the correction, the corrected utterances are inserted back to their original place in the full call transcript.

6.3 Automatic Error Detection Model

Error detection can be considered a component of the full auto-correction pipeline (Fang et al., 2022; Leng et al., 2023) and can be easily integrated into various ML models as an additional feature. To assess its impact, we examine the effect of incorporating a simple error detection signal into our production-level model.

The ASR Error Detection (AED) model is trained similarly to the AEC model but differs in its output. Instead of generating a corrected transcript, the AED model produces a binary classification: *True* if the first of the n alternatives is noisy and *False* otherwise. To adapt the AEC training data for this task, we label an instance as *True* if the best alternative differs from the pseudo-corrected transcript and *False* otherwise.

7 Results

7.1 Evaluation Setting

The goal of both *Direct Extraction* and *Direct Verification* is to determine whether a given live-call field value is correct. If the gold field value is the same as the live-call value and the model predicts it as correct, we consider that a correct prediction. Since our primary focus is on 'auto-approval', we evaluate results specifically for that class.

Given the dataset's high imbalance, we report precision, recall, and F1 scores. For *Direct Extraction*, we also measure exact match and normalized edit distance. The baseline in both evaluation settings is the model that is just provided the best ASR transcript, without any error correction [8].

[6]Additional details on the choice of n are given in appendix C.3

[7]We chose Mistral due to its open-source availability and, in our preliminary experiments with random subset samples, performed better than LLaMA-8B-instruct.

[8]We measure the efficacy of the error correction model by evaluating directly on the downstream task of benefit verification as opposed to intrinsic evaluation metrics such as ROUGE, since we do not have gold corrected transcripts.

Field Value	Precision↑	Recall↑	F1↑	Accuracy↑	NED↓
			Gemini		
Agent Name	0.9756	0.4568	0.6223	0.4403	0.2263
Reference Number	0.9791	0.2958	0.4544	0.2785	0.4083
Group Number	0.9942	0.3508	0.5186	0.3475	0.2673
Average	0.9830	0.3746	0.5318	0.3554	0.3006
			Gemini + AEC		
Agent Name	0.9776	0.4772	0.6413	0.4594	0.2187
Reference Number	0.9787	0.3574	0.5236	0.3383	0.3822
Group Number	0.9916	0.4262	0.5961	0.4248	0.2292
Average	**0.9823**	**0.4203**	**0.5870**	**0.4075**	**0.2767**

Table 4: Performance metrics in the *Direct Extraction* setting. 'Gemini' is the baseline that only gets the best ASR transcript while 'Gemini+AEC' gets the corrected transcript as input. *NED: Normalized Edit Distance*

7.2 Base Models

For off-the-shelf LLMs, we report results on GPT (Brown et al., 2020; Achiam et al., 2023) and Gemini (Team et al., 2023) models with noisy ASR transcripts as baseline and after performing error correction. The detailed prompts can be found in Appendix D. We also integrate the AEC model into the auto-review model used in a feature-based model architecture. We use XGBoost model architecture so we can leverage all of the statistical and historical features[9] and LLM models (e.g., field value extractions using LLMs) as features for making final auto-approval decisions. We do not compare against other specialized error correction models, as they either focus on grammatical error correction (Li and Wang, 2024; Ma et al., 2023) or rely on specialized knowledge graphs (Saebi et al., 2021) or manual annotations.

7.3 Analysis

Our goal is to assess the impact of ASR error correction on the overall performance of the **Auto Review** pipeline. Ultimately, the choice of model depends on the specific use case and the acceptable trade-off between precision and recall.

Direct Verification Table 3 presents the results for direct verification. We first examine the XGBoost model within the feature-based pipeline. Adding a simple binary feature for AED (indicating whether the transcript is noisy) improves performance for two out of three fields. Further incorporating corrected transcripts, the 'XGBoost + AED + AEC' model significantly enhances the F1 score for 'Agent Name' (0.7824→0.8428) and achieves the best performance on 'Reference Number' (0.9088→0.9208). The 'Gemini 1.5 + AEC' model improves precision across all fields but at the cost of reduced recall. In contrast, 'GPT 3.5 + AEC' enhances overall performance across all fields, except for a slight recall drop in 'Agent Name'. Notably, it achieves the highest accuracy for 'Group Number'. Fine-tuned GPT model with AEC obtained the highest F1 score on all fields by improving the recall substantially but resulted in a lower precision. Compared to LLMs, XGBoost models achieve higher precision but lower recall. This is due to their reliance on specialized regular expressions for field formats [10] as well as historical and statistical features. However, these constraints limit their generalization to diverse cases.

Direct Extraction Unlike the direct verification approach, the AEC model does not receive the live-call extracted field value as input. Instead, it extracts the field value directly from the ASR transcript. This extracted value is then compared to the live-call field values as an additional validation step. If both values match, the system auto-approves the result; otherwise, it requests a second human review. As shown in Table 5, this method results in lower recall, as the model often fails to approve correct values due to variations in ASR outputs. For instance, as illustrated in Figure 1, the *direct verification* model may approve the live-call group

[9]Features include textual features extracted from live-call field values (e.g., regular expression patterns for expected formats for each field), call STT transcripts and statistical and historical features extracted from benefit verification client and call recipient insurance company.

[10]e.g., predefined patterns for group numbers, reference numbers, and agent name capitalization

number despite minor errors in the transcript (e.g., ignoring an incorrect ‘8’). In contrast, the *direct extraction* model may output alternative values such as ‘8D0156’ or ‘AD0156’, increasing susceptibility to ASR errors. However, this approach achieves significantly higher precision. After applying ASR error correction, precision remains stable across all fields, while recall improves substantially, yielding an average F1 score improvement of **5.5%**. While failing to auto-approve correct values is undesirable, it is preferable to approving incorrect extractions and passing them to customers.

A hybrid model combining both settings could be implemented in production. *Direct verification* would be applied to less critical fields [11], leading to a higher overall F1 score and saving time on review. *Direct extraction* would be reserved for critical fields, approving them under a more stringent setting.

8 Conclusion

We introduced **Auto Review**, a two-stage pipeline that enhances information extraction from healthcare phone calls. Our approach reduces human verification while maintaining high accuracy. The second stage involves an ASR error correction framework, leveraging n-best ASR alternatives to generate pseudo-labels for training an error correction model. This framework is adaptable across domains, provided some past manually reviewed data is available. Results show that ASR error correction improves precision and recall across key fields, with *Direct Verification* offering higher recall and *Direct Extraction* achieving higher precision.

The results reported in this paper reflect the isolated performance of a model component within a larger production system. In real-world deployment, additional pipeline components—including human-in-the-loop mechanisms and cross-field verification models—contribute to significantly higher precision. This underscores the complementary role of system-level engineering in achieving production-grade performance alongside core model development.

9 Ethical Statement

All experiments described in this paper were conducted in compliance with applicable privacy and data protection regulations. Specifically, interactions with third-party models, including OpenAI’s GPT-3.5 Turbo and Google’s Gemini, were governed by appropriate Business Associate Agreements (BAAs) if required under the Health Insurance Portability and Accountability Act (HIPAA). These controls were designed to ensure that no Protected Health Information (PHI) was exposed to external service providers for training or other purposes beyond our immediate use case, and that at no point was PHI stored in third-party companies or used to improve or fine-tune the third-party models themselves.

For model inferences in our main experiments with GPT-3.5 Turbo and Gemini 1.5 Pro APIs, the total estimated cost was $303, based on publicly available pricing at the time of experimentation. This included approximately $260 for Gemini 1.5 Pro with audio input, $20 for Gemini 1.5 Pro with text input, and $23 for GPT-3.5 Turbo (16k context) with text input.

[11]Critical fields are those where incorrect values can have a significant negative impact on customers.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

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

Kirk Lorne Buker. 2023. *Financial Impact When a Health System Automates Manual Insurance Verification Processes*. Northcentral University.

Christopher Davis, Andrew Caines, Øistein E. Andersen, Shiva Taslimipoor, Helen Yannakoudakis, Zheng Yuan, Christopher Bryant, Marek Rei, and Paula Buttery. 2024. Prompting open-source and commercial language models for grammatical error correction of English learner text. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11952–11967, Bangkok, Thailand. Association for Computational Linguistics.

Nima Ebadi, Kellen Morgan, Adrian Tan, Billy Linares, Sheri Osborn, Emma Majors, Jeremy Davis, and Anthony Rios. 2024. Extracting biomedical entities from noisy audio transcripts. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 7023–7034.

Rahhal Errattahi, Asmaa El Hannani, Hassan Ouahmane, and Thomas Hain. 2016. Automatic speech recognition errors detection using supervised learning techniques. In *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, pages 1–6. IEEE.

Zheng Fang, Ruiqing Zhang, Zhongjun He, Hua Wu, and Yanan Cao. 2022. Non-autoregressive Chinese ASR error correction with phonological training. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5907–5917, Seattle, United States. Association for Computational Linguistics.

Yujie Feng, Zexin Lu, Bo Liu, Liming Zhan, and Xiao-Ming Wu. 2023. Towards llm-driven dialogue state tracking. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 739–755.

Ulrich Gnewuch, Stefan Morana, and Alexander Maedche. 2017. Towards designing cooperative and social conversational agents for customer service. In *ICIS*, pages 1–13.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Brendan King and Jeffrey Flanigan. 2023. Diverse retrieval-augmented in-context learning for dialogue state tracking. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5570–5585, Toronto, Canada. Association for Computational Linguistics.

Yichong Leng, Xu Tan, Wenjie Liu, Kaitao Song, Rui Wang, Xiang-Yang Li, Tao Qin, Ed Lin, and Tie-Yan Liu. 2023. Softcorrect: Error correction with soft detection for automatic speech recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13034–13042.

Wei Li and Houfeng Wang. 2024. Detection-correction structure via general language model for grammatical error correction. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1748–1763, Bangkok, Thailand. Association for Computational Linguistics.

Zekun Li, Zhiyu Chen, Mike Ross, Patrick Huber, Seungwhan Moon, Zhaojiang Lin, Xin Dong, Adithya Sagar, Xifeng Yan, and Paul Crook. 2024. Large language models as zero-shot dialogue state tracker through function calling. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8688–8704, Bangkok, Thailand. Association for Computational Linguistics.

Mengsay Loem, Masahiro Kaneko, Sho Takase, and Naoaki Okazaki. 2023. Exploring effectiveness of GPT-3 in grammatical error correction: A study on performance and controllability in prompt-based methods. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 205–219, Toronto, Canada. Association for Computational Linguistics.

Yiting Lu, Mark JF Gales, Kate M Knill, P Manakul, Linlin Wang, and Yu Wang. 2019. Impact of asr performance on spoken grammatical error detection. ISCA.

Xiang Luo, Zhiwen Tang, Jin Wang, and Xuejie Zhang. 2024. Zero-shot cross-domain dialogue state tracking via dual low-rank adaptation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5746–5765, Bangkok, Thailand. Association for Computational Linguistics.

Rao Ma, Mark J. F. Gales, Kate M. Knill, and Mengjie Qian. 2023. N-best t5: Robust asr error correction using multiple input hypotheses and constrained decoding space. In *INTERSPEECH 2023*, pages 3267–3271.

Michael McTear. 2022. *Conversational ai: Dialogue systems, conversational agents, and chatbots*. Springer Nature.

Ernest Pusateri, Anmol Walia, Anirudh Kashi, Bortik Bandyopadhyay, Nadia Hyder, Sayantan Mahinder, Raviteja Anantha, Daben Liu, and Sashank Gondala. 2024. Retrieval augmented correction of named entity speech recognition errors. *arXiv preprint arXiv:2409.06062*.

Srijith Radhakrishnan, Chao-Han Yang, Sumeer Khan, Rohit Kumar, Narsis Kiani, David Gomez-Cabrero, and Jesper Tegnér. 2023. Whispering LLaMA: A cross-modal generative error correction framework for speech recognition. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10007–10016, Singapore. Association for Computational Linguistics.

Mandana Saebi, Ernie Pusateri, Aaksha Meghawat, and Christophe Van Gysel. 2021. A discriminative entity aware language model for virtual assistants. In *Interspeech*.

Kai Shen, Yichong Leng, Xu Tan, Siliang Tang, Yuan Zhang, Wenjie Liu, and Edward Lin. 2022. Mask the correct tokens: An embarrassingly simple approach for error correction. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10367–10380.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan

Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Yixuan Wang, Baoxin Wang, Yijun Liu, Qingfu Zhu, Dayong Wu, and Wanxiang Che. 2024. Improving grammatical error correction via contextual data augmentation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10898–10910, Bangkok, Thailand. Association for Computational Linguistics.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. 2024. LlamaFactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 400–410, Bangkok, Thailand. Association for Computational Linguistics.

Linchen Zhu, Wenjie Liu, Linquan Liu, and Edward Lin. 2021. Improving asr error correction using n-best hypotheses. In *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 83–89. IEEE.

A Training Description

For the AEC model, we use Mistral-7B-Instruct-v0.3, which was trained with a batch size of 16, gradient accumulation step set to 2 using 1 A100 GPU. Training took 9 hours. In all our AEC experiments, the number of alternatives, *n*, is fixed to 10. LoRA (Hu et al., 2022) is used for parameter-efficient training using the LLaMA-Factory library (Zheng et al., 2024). We use the Gemini 1.5 model to generate the pseudo-labels. Google STT model is used as the base STT model for all ASR transcripts[12].

B Relevant Utterance Isolation

Algorithm 2 presents the algorithm to isolate only those utterances from the call transcripts that are highly likely to contain the field value information we want to extract. It starts collecting agent utterances after the conversational AI model asks for information regarding that field, those trigger questions are pre-defined and passed to the algorithm in field_triggers.

Algorithm 2 Extract Utterances for Fields of Interest

```
Require: call_transcript (list of tuples with speaker and utterance), field_triggers (list of trigger utterances)
1: Initialize an empty list $agent\_responses$
2: Set $collect\_responses \leftarrow$ false
3: for each $(speaker, utterance)$ in $call\_transcript$ do
4:     if not $collect\_responses$ and $utterance$ contains any phrase in $field\_triggers$ then
5:         $collect\_responses \leftarrow$ true
6:     else if $collect\_responses$ then
7:         if $speaker$ = Agent then
8:             Append $utterance$ to $agent\_responses$
9:         else if $speaker$ = AI Model then
10:            $collect\_responses \leftarrow$ false
11:        end if
12:    end if
13: end for
14: return $agent\_responses$
```

Field Value	Precision	Recall	F1
Gemini with Audio			
Agent Name	0.9838	0.1205	0.2148
Reference Number	0.9816	0.3875	0.5556
Group Number	0.9965	0.4323	0.6030
XGBoost Model			
Agent Name	0.9570	0.6617	0.7824
Reference Number	0.9636	0.8598	0.9088
Group Number	0.9749	0.8969	0.9343

Table 5: Performance metrics for Agent Name, Reference Number, and Group Number in the *Direct Extraction* setting using Gemini with audio input. The audio-based model suffers from very low recall.

C Preliminary Experiments

C.1 Experiments with Gemini using Audio Input

For our preliminary analysis, we experimented with off-the-shelf multimodal LLM (Gemini 1.5) with the same prompt we used for ASR text transcript direct extraction (Table 11, Table 12) except for

[12]In preliminary experiments, we found fine-tuning ASR helped improving the general performance metric such as word error rate (WER) but observed the similar issues especially from unseen field values. See more details in C.2

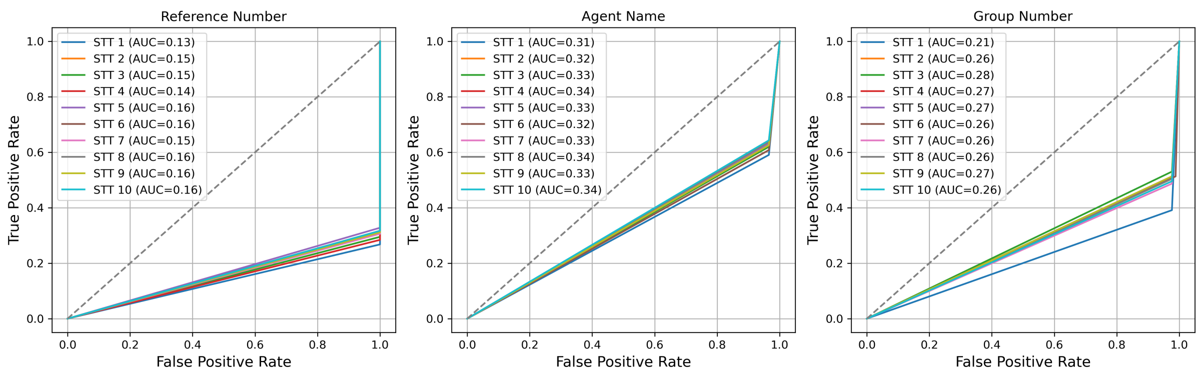


Figure 4: The ROC curves for the three field types when using different numbers of transcript alternatives as input. The Gemini model is provided the transcript to extract the field value, which is then compared with the gold field value. Providing multiple alternatives improves performance.

the instruction which tells to use the attached audio instead of the text providing the call audio recording. Gemini obtained high precision overall, but its recall is too low to effectively reduce human review time in industry settings with a large number of concurrent phone calls. When we analyzed false positive auto-approved samples, it made similar mistakes with ASR models incorrectly adding 0 or missing a few digits for long alphanumeric field values or misspelling rare agent names with more common names. Thus, we designed ASR error detection and correction models focusing on the field values of the data types that are highly vulnerable to such errors and cannot be resolved by off-the-shelf LLMs or other feature-based models.

C.2 Experiments with ASR systems

We conducted preliminary experiments using Google STT and Whisper (Whisper Large V3[13]) to choose the most suitable ASR system for our field value output extraction tasks. Although Google STT obtained a higher performance than Whisper, it was not available for fine-tuning so we fine-tuned Whisper model using the subset of our full data to explore the best ASR system options (785 outputs for training set, 390 outputs for validation set and 510 outputs for test set). We found that our fine-tuned Whisper model improved the general evaluation metrics but we still observed similar issues with mistranscripts with digits or letters missing for long sequence outputs; especially with the patterns which did not exist in training set (see more details in Table 6). Thus, collecting ground truth labels for all such cases required a large human labeling effort and it was not scalable for our task with real world data so we chose off-the-shelf Google STT for our main experiments.

ASR System	Word Error Rates	Norm. Edit
Google STT	0.602	0.430
Whisper	0.757	0.485
FT Whisper	0.349	0.216

Table 6: Performance metrics for ASR systems on task output transcription. FT Whisper: fine-tuned Whisper, Norm. Edit: normalized edit distance (edit distance between the transcript and the ground truth divided by the maximum value among the lengths of the two).

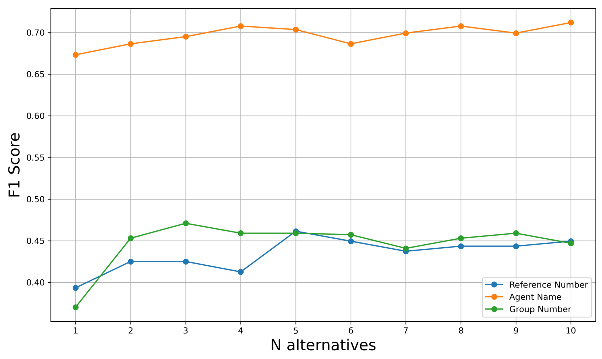


Figure 5: F1 scores for LLM performance across the three field types, based on correctly extracting field values from transcripts. The input transcript to the LLM includes multiple ASR alternatives. A significant performance improvement is observed when incorporating multiple alternatives instead of relying solely on the best one.

C.3 Number of ASR Alternatives

We conducted experiments to assess the impact of using multiple ASR alternatives on field value extraction. Using a subset of 200 calls, we measured

[13]https://huggingface.co/openai/whisper-large-v3

LLM performance in extracting three key fields. Specifically, we prompted Gemini to extract field values from transcripts while varying the number of ASR alternatives, with $n = 1$ corresponding to using only the best transcript. The prompts for these experiments follow the *Direct Extraction* approach and are detailed in Table 12 and Table 11. As shown in Figure 5 and Figure 4, incorporating multiple ASR alternatives significantly improves performance across all field values. Since the optimal value of n varies by field type and we want to train a single cohesive AEC model, we chose $n = 10$ in all our experiments.

D Model Prompts

The prompts used for all the experiments are given below. The in-context examples used in the experiments have been removed because they contain sensitive patient information.

<INSTRUCTIONS> You are “{our conversational AI model name}”, a digital assistant calling a healthcare insurance company to get benefits information for a member. Given the STT transcript of phone conversations between you and the health insurance company agent, check if all of your answers to the given questions are correct. Please respond using “correct” or “incorrect”, checking whether all the answers to the questions in the call are correct or not, and provide your reasoning in JSON format. Here are example cases for each answer:
1. “correct”: select this option only if all the answers are correct based on the call transcript.
2. “incorrect”: select this option if you see any of the answers to the questions is incorrect.
Below are sample responses and reasons:
Reason: Among 4 questions asked, the answer to the second question should have been “True”. // Your response: {“response”: “incorrect”}
Reason: All of the answers to the given 5 questions are correct. // Your response: {“response”: “correct”}
Reason: There was one question and the agent could not provide the answer and the answer was “agent did not provide this information”. // Your response: {“response”: “correct”} **</INSTRUCTIONS>**

<TARGET_QUESTION_GUIDELINES> Some additional guidelines for specific questions with examples for the questions of “agentName”, “referenceNumber”, and “groupNumber”:
1. Note if the agent spells it out or uses nato alphabet. For example, if the agent says “c as in Charlie 2 n as in Nancy 3 c as in Tango G is in gold”, you should collect “C2N3TG”. With STT mistranscriptions, you should follow the nato alphabet over the spelling.
2. Unless there is a word or name used, capitalize all letters and remove any spaces. For example, if the agent says “group number is 123 456 789”, you should collect “1234567890”.
3. There might be speech to text transcription errors (e.g. “8” instead of "H" or “for” instead of “4”) For example, they might say “C like Tango” and in this case you should get the spelling to include T, not C.
</TARGET_QUESTION_GUIDELINES>

<TARGET_QUESTION_EXAMPLES> [reason // questions // your response]
- Reason: “the agent spelled out their name as Jane and said C like Tango” Question: “Question 1: agentName? Answer: ’Jane T”’ // Your response: {{“response”: “correct”}} Reason: “the agent gave their name as Jane and said his last name initial is O as in Oscar and said there were no reference numbers” // Question: "Question 1: agentName? Answer: ’Jane O’. Question 2: referenceNumber? Answer: ’Jane O 06242024’" // Your response: {{“response”: “correct”}}
- Reason: “the agent said t i a b for boy so likely the last name initial is B so the first name is Tia” // Question “agentName”: “Tia B”, “referenceNumber”: “12345”}}
- Reason: “the agent said d a r a for alpha my initial so likely A is their last name initial so the first name is Dar” // Question: "Question 1: agentName? ’Dar A”’ // Your response: {{“response”: “correct”}}
- Reason: "the agent said their name was j a qu a i d i a last initial K so their name is Jaquaidia K and they said the reference number was their name and the date" // Question: "Question 1: agentName? ’Jaquaidia K’. Question 2: referenceNumber? ’Jaquaidia K 06012024’// Your response: "response": "correct"
- Reason: "the agent said their name was Jasmine but spelled it out as J A S M I N so with that spelling their name must be Jasmin" // Question: "Question 1: agentName? ’Jasmine’" // Your response: {{“response”: “incorrect”}} - Reason: “the agent said their name was Sam but spelled it out as s a m y r so with that spelling their name must be Samyr” // Question: “Question 1: agentName? ’Samyr”’ // Your response: {{“response”: “correct"}}
- Reason: “the agent spelled their name as ’p as in paul n as in nancy o t t r i c last initial is d’ so their name is Pnottric D and gave no reference number" // Question: “Question 1: agentName? ’Pnottric D’. Question 2: referenceNumber? ’Pnottric D 06012024”’ // Your response: {{“response”: “correct”}}
</TARGET_QUESTION_EXAMPLES>
Below is the STT transcript of the call.
[transcript]

Answer if all of the following questions and answer pairs are correct in the JSON format as in the example in the instruction
[question_answer_pairs]

Table 7: *Direct Verification* prompt used for all fields.

</INSTRUCTIONS> You are a capable annotator who can identify and correct issues in STT transcript. You will be given alternative STT transcripts and corresponding extracted name. Pick the best alternative that most correctly corresponds to the given extracted name. The best alternative is defined as: The alternative transcript from which we should be able to extract the name that matches the given extracted name. If there are multiple names present, usually we only care about the last name. Ignore the name "{our conversational AI model name}" if it is present in the transcript. The alternative transcripts are separated by "#". Give the output in json format of {{"Output": best_transcript}}
</INSTRUCTION>
<EXAMPLES>
Here are some examples of the STT transcripts along with the extracted value and the outputs separated by "//" (i.e., STT transcripts, extracted name // your output):
[Examples]
</EXAMPLES>
Now provide your answer from the following STT transcripts and extracted value:
[Input]

Table 8: Pseudo-label generation prompt for selecting the best alternative.

<INSTRUCTIONS> You are a capable annotator who can identify and correct issues in STT transcript. You will be given STT transcript and corresponding extracted value. If the transcript is correct, you will simply return the transcript and if the transcript is wrong compared to the correctly extracted value, you need to correct the transcript appropriately. Pay special attention to the number of zeros in the extracted value and compare with the noisy transcript. Do not capitalize letters in the transcript if they are not originally capitalized, even if the extracted value has capitalized letters. Give the output in json format of {{"Output": corrected_transcript}}
</INSTRUCTIONS>
<EXAMPLES> Here are some examples of the STT transcript along with the extracted value and the outputs separated by "//" (i.e., STT transcript, extracted value // your output):
[Examples]
</EXAMPLES>
Now provide your answer from the following STT transcript and extracted value: [Input]

Table 9: Pseudo-label generation prompt for error correction.

<PROMPT>You are a capable annotator who can identify and correct issues in ASR transcript. You will be given a list of noisy ASR outputs, separated by "#". Output the best possible ASR alternative. In some cases, the correct output will be one of the provided alternatives, in other cases you will have to identify patterns across the alternatives and output a cohesive correct transcript.
</PROMPT>
[Input]

Table 10: Automatic error correction model prompt.

<INSTRUCTIONS>Given a transcript, extract the underlying group number value. Give the output in json format of {{"Output": extracted value}}
</INSTRUCTIONS>
<EXAMPLES> Here are some examples of the transcript along with the extracted output separated by "//" (i.e., text // your output):
[Examples]
</EXAMPLES>
Now provide your answer from the following text:
[Input]

Table 11: Direct extraction prompt for Group Number.

<INSTRUCTIONS> Given a transcript, extract the underlying name. Ignore "{our conversational AI model name}" if it appears in the transcript. If there are multiple names, extract the last one. Capitalize the first name initial and last name initial. Give the output in json format of {{"Output": extracted value}}
</INSTRUCTIONS>
<EXAMPLES> Here are some examples of the transcript along with the extracted output separated by "//" (i.e., text // your output):
[Examples]
</EXAMPLES>
Now provide your answer from the following text:
[Input]

Table 12: *Direct Extraction* prompt for Agent Name and Reference Number.

<INSTRUCTIONS> You are "{our conversational AI model name}", a digital assistant calling a healthcare insurance company to get benefits information for a member. Given the STT transcript of phone conversations between you and the health insurance company agent, check if all of your answers to the given questions are correct. Please respond using "correct" or "incorrect", checking whether all the answers to the questions in the call are correct or not, and provide your reasoning in JSON format. Here are example cases for each answer:
1. "correct": select this option only if all the answers are correct based on the call transcript.
2. "incorrect": select this option if you see any of the answers to the questions is incorrect.
Below are sample responses and reasons:
Reason: Among 4 questions asked, the answer to the second question should have been "True". // Your response: {"response": "incorrect"}
Reason: All of the answers to the given 5 questions are correct. // Your response: {"response": "correct"}
Reason: There was one question and the agent could not provide the answer and the answer was "agent did not provide this information". // Your response: {"response": "correct"} **</INSTRUCTIONS>**

<TARGET_QUESTION_GUIDELINES> Some additional guidelines for specific questions with examples for the questions of "agentName", "referenceNumber", and "groupNumber":
1. Note if the agent spells it out or uses nato alphabet. For example, if the agent says "c as in Charlie 2 n as in Nancy 3 c as in Tango G is in gold", you should collect "C2N3TG". With STT mistranscriptions, you should follow the nato alphabet over the spelling.
2. Unless there is a word or name used, capitalize all letters and remove any spaces. For example, if the agent says "group number is 123 456 789", you should collect "1234567890".
3. There might be speech to text transcription errors (e.g. "8" instead of "H" or "for" instead of "4") For example, they might say "C like Tango" and in this case you should get the spelling to include T, not C.
</TARGET_QUESTION_GUIDELINES>

<TARGET_QUESTION_EXAMPLES> [reason // questions // your response]
- Reason: "the agent spelled out their name as Jane and said C like Tango" Question: "Question 1: agentName? Answer: 'Jane T"' // Your response: {{"response": "correct"}} Reason: "the agent gave their name as Jane and said his last name initial is O as in Oscar and said there were no reference numbers" // Question: "Question 1: agentName? Answer: 'Jane O'. Question 2: referenceNumber? Answer: 'Jane O 05012024'" // Your response: {{"response": "correct"}}
- Reason: "the agent said t i a b for boy so likely the last name initial is B so the first name is Tia" // Question "agentName": "Tia B", "referenceNumber": "12345"}}
- Reason: "the agent said d a r a for alpha my initial so likely A is their last name initial so the first name is Dar" // Question: "Question 1: agentName? 'Dar A"' // Your response: {{"response": "correct"}}
- Reason: "the agent said their name was j a qu a i d i a last initial J so their name is Jaquaidia K and they said the reference number was their name and the date" // Question: "Question 1: agentName? 'Jaquaidia K'. Question 2: referenceNumber? 'Jaquaidia K 06012024'// Your response: "response": "correct"
- Reason: "the agent said their name was Jasmine but spelled it out as J A S M I N so with that spelling their name must be Jasmin" // Question: "Question 1: agentName? 'Jasmine'" // Your response: {{"response": "incorrect"}} - Reason: "the agent said their name was Sam but spelled it out as s a m y r so with that spelling their name must be Samyr" // Question: "Question 1: agentName? 'Samyr"' // Your response: {{"response": "correct"}}
- Reason: "the agent spelled their name as 'p as in paul n as in nancy o t t r i c last initial is g' so their name is Pnottric G and gave no reference number" // Question: "Question 1: agentName? 'Pnottric G'. Question 2: referenceNumber? 'Pnottric G 06012024"' // Your response: {{"response": "correct"}}
</TARGET_QUESTION_EXAMPLES>
Below is the STT transcript of the call.
[transcript]

Answer if all of the following questions and answer pairs are correct in the JSON format as in the example in the instruction
[question_answer_pairs]

Table 13: *Direct Verification* prompt used for all fields.