PharmaSUG China 2024 - Paper AI-10070

Medical Coding with Large Language Models in the AI+HI Paradigm

Hao Chen, Xuejie Zhou, and Leo Li, Beigene, Inc.

ABSTRACT

Traditional medical coding processes are labor-intensive and heavily reliant on the experience of coding and medical knowledge. To identify a proper code for a verbatim term, it often requires multiple searches within the conventional dictionary browser or other sources to confirm the medical concept. This paper introduces an innovative AI-native medical coding application that leverages Large Language Models (LLMs) within the AI+HI (Artificial Intelligence + Human Intelligence) paradigm to enhance coding efficiency and accuracy. By utilizing advanced LLM techniques such as Prompting, Retrieval-Augmented Generation (RAG), and Agents, our application can effectively interpret the diverse medical terminology and match it with the MedDRA code.

The application integrates a ChatBot into the platform, supporting users both Chat mode and Batch processing mode. It also provides the coding logic in the results, enabling users to engage with the LLM model for informed decision-making. Comprehensive testing on historical medical coding records demonstrates the system's impressive performance and reveals how AI technology, domain-specific knowledge, and human guidance can be integrated to continuously improve performance.

Our approach highlights the potential of integrating AI and HI in medical coding, with GPT-4 showing excellent performance after comparing various open-source and closed-source models. The application offers customizable prompts, robust data protection, and the potential for further enhancements using proprietary algorithms. This innovative solution and framework promise to revolutionize the medical coding process and extend to other tasks such as programming, translation, and writing in the clinical development process, leading to improved efficiency, accuracy, and cost-effectiveness in the pharmaceutical industry.

INTRODUCTION

The conventional process of medical coding is labor-intensive and relies heavily on the expertise of clinical professionals. This traditional method faces challenges due to the diverse expressions of medical terminology, which frequently differ from the standardized terms found in the MedDRA dictionary. Leveraging the latest advancements in AI, particularly in Large Language Models (LLMs), our platform significantly enhances text comprehension capabilities. We utilize these sophisticated LLMs to increase both the efficiency and precision of medical coding. Our system is adept at interpreting the nuances of general medical terminology and mapping them accurately to standardized MedDRA codes.

SYSTEM ARCHITECTURE

USER INTERFACE (UI):

Streamlit App: A web application built with Streamlit, an open-source framework for rapidly developing data applications with intuitive UI, ideal for machine learning and data science projects. Support multi-round chat functionality to allow users to utilize the GPT model more effectively. This feature will support detailed discussions and thought processes, enabling users to make informed final decisions.

ALGORITHM:

Batch Embedding: Processes large batches of text data to generate embeddings, which are vector representations of text used to understand semantic similarities.

RAG (Retrieval-Augmented Generation): Utilizes a retrieval-based approach to enhance the generation of responses by fetching relevant context from a database or document collection.

Fast_Top_match: A method used to quickly identify the top matching terms from the LLT dictionary based on the input data.

GPT_rerank: Employs GPT models to re-rank the matched results to ensure the best matching terms based on context and relevance are presented first.

Result Format Control and Validation: Ensures that the output is formatted correctly and validates it against expected standards.

Accuracy Evaluation: Assesses the algorithm's performance by comparing the model's predictions with ground truth data.

DATA:

LLT Dictionary: Contains the MedDRA Lowest Level Terms used in medical coding, serving as the foundational dataset for the application.

Historical Relation Data: Past records of AETERM to LLT mappings, which provide a reference for the algorithm to learn from and improve its predictions.

MODEL:

LLM Model (text-embedding-ada-002): A specific model for generating text embeddings, which is used to understand the meaning and context of medical terms.

Azure OpenAI GPT3.5, GPT4: The application uses both GPT-3.5 and the more advanced GPT-4 for generating and re-ranking predicted medical codes, leveraging these models via the Azure OpenAI service.

SERVER:

Rstudio Connect Server: A publishing platform for Shiny applications, R Markdown reports, and more, used to host and manage the Streamlit application.

Databricks (next stage): A platform for big data analytics and artificial intelligence, future move towards a more scalable and comprehensive data processing environment.

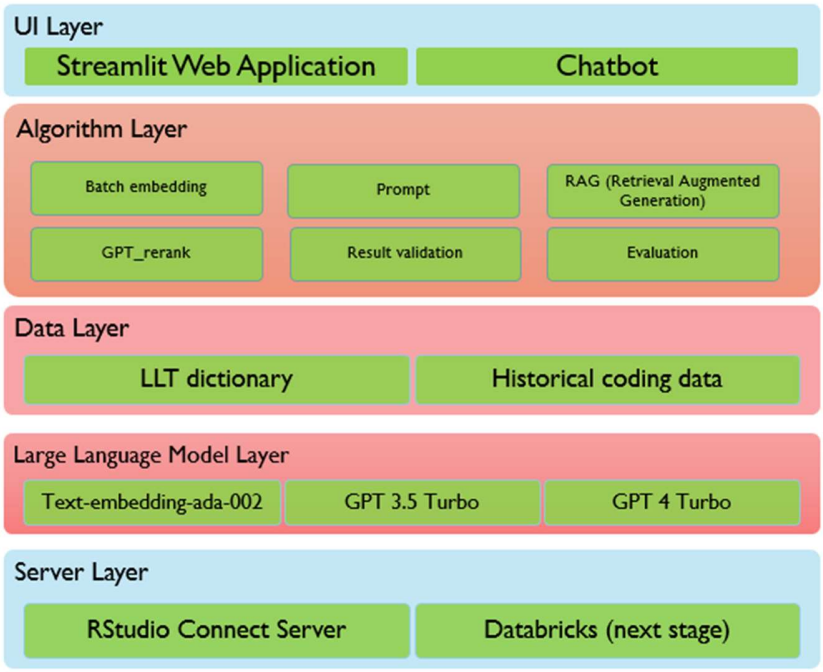


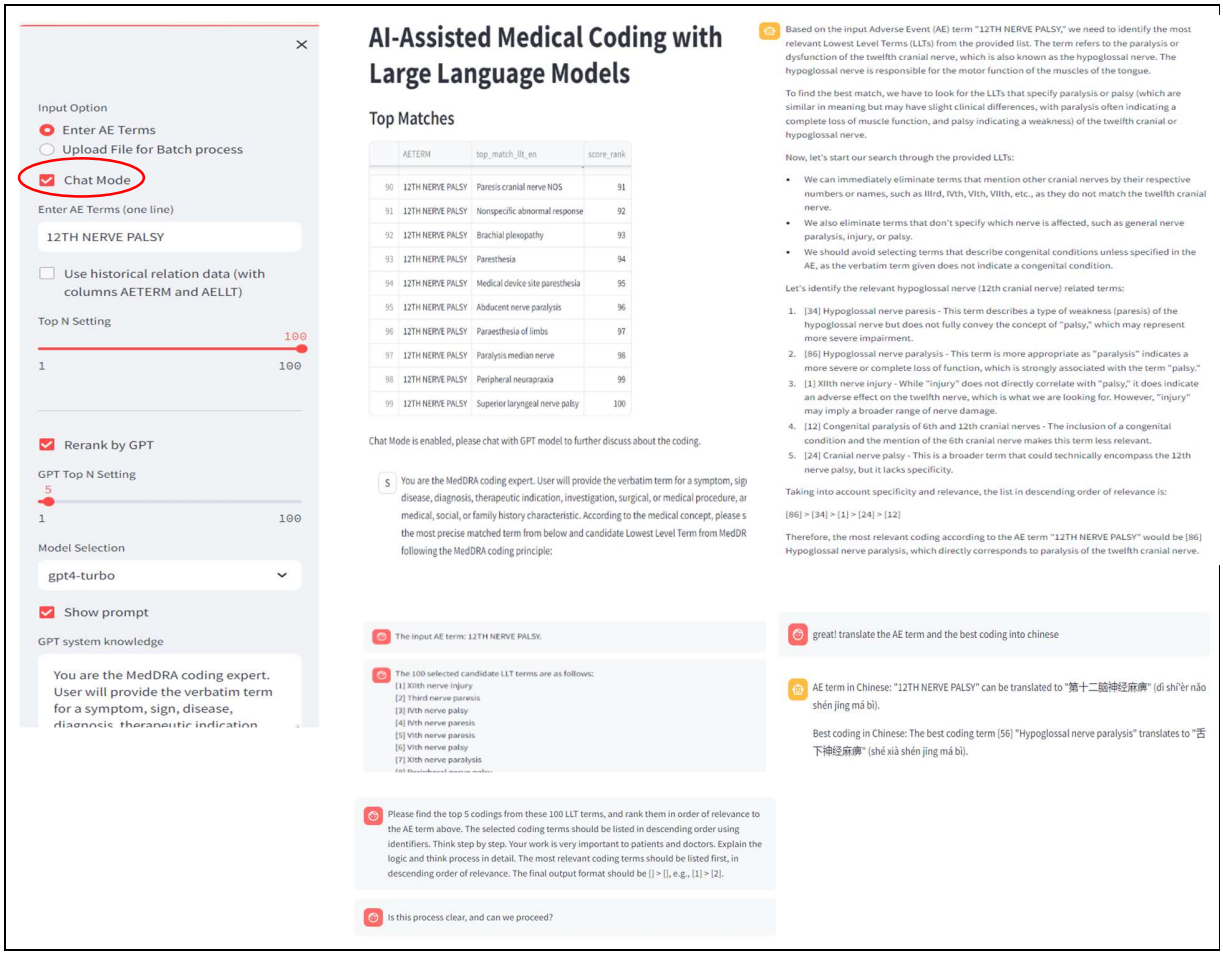
Figure 1. System Architecture

UI DESIGN

The UI design is user-friendly. Users can either type in several AE terms or upload an Excel file containing a batch of AE terms. The model will then calculate the similarity based on semantic meaning and select the top N candidates as specified by the user. Subsequently, users can choose either the GPT-3.5 or GPT-4 model to further refine the candidate pool. Additionally, users have the option to input medical coding logic through the UI, which will guide GPT during the selection process. If the user provides ground truth data, the system will generate a success rate and a summary plot.

All matches will display a similarity score and ranking from GPT, and users can further filter the results by score and download the data in an Excel file.

Meanwhile, the system provides the chatMode option, with multi-round chat functionality to allow users to utilize the GPT model more effectively. This feature will support detailed discussions and thought processes, enabling users to make informed final decisions.



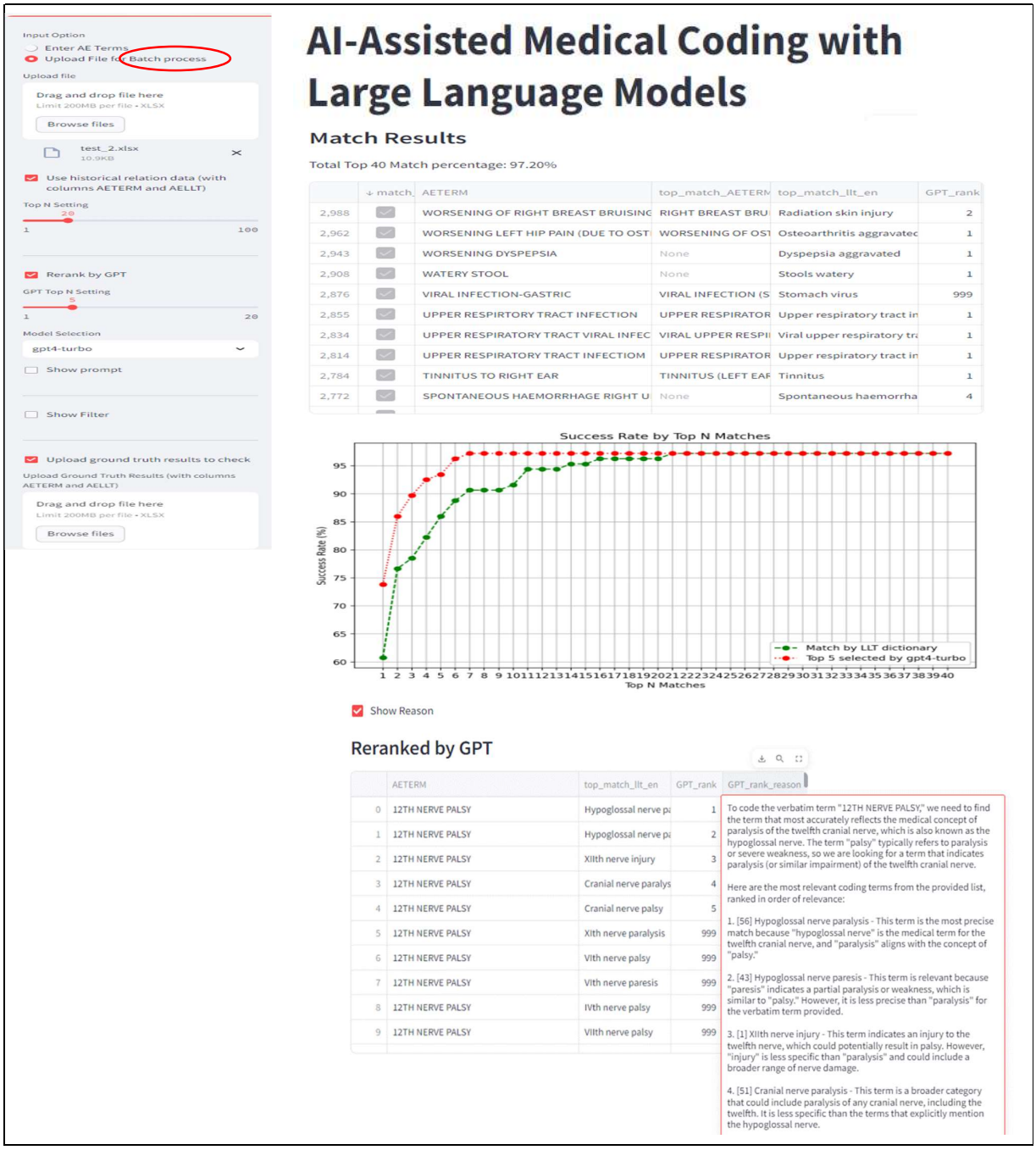


Table 1. UI Design

WORKFLOW AND EXPERIMENTAL DESIGN

For each verbatim Adverse Event (AE) term in a case study, we first apply a traditional rule-based method that uses exact text matching to associate the terms with the Lowest Level Terms (LLTs) from a dictionary. We then search for any matching AE terms in the historical AE term-LLT relationship data.

For those AE terms that remain unmatched, we employ a Large Language Model to interpret the text meaning and select the top N candidates from each group, assigning scores to indicate their level of similarity.

Finally, we use a prompt to direct GPT to narrow down the candidates to the top 5 LLTs. This step ensures that we identify the most relevant LLTs with a high degree of probability and rationale.

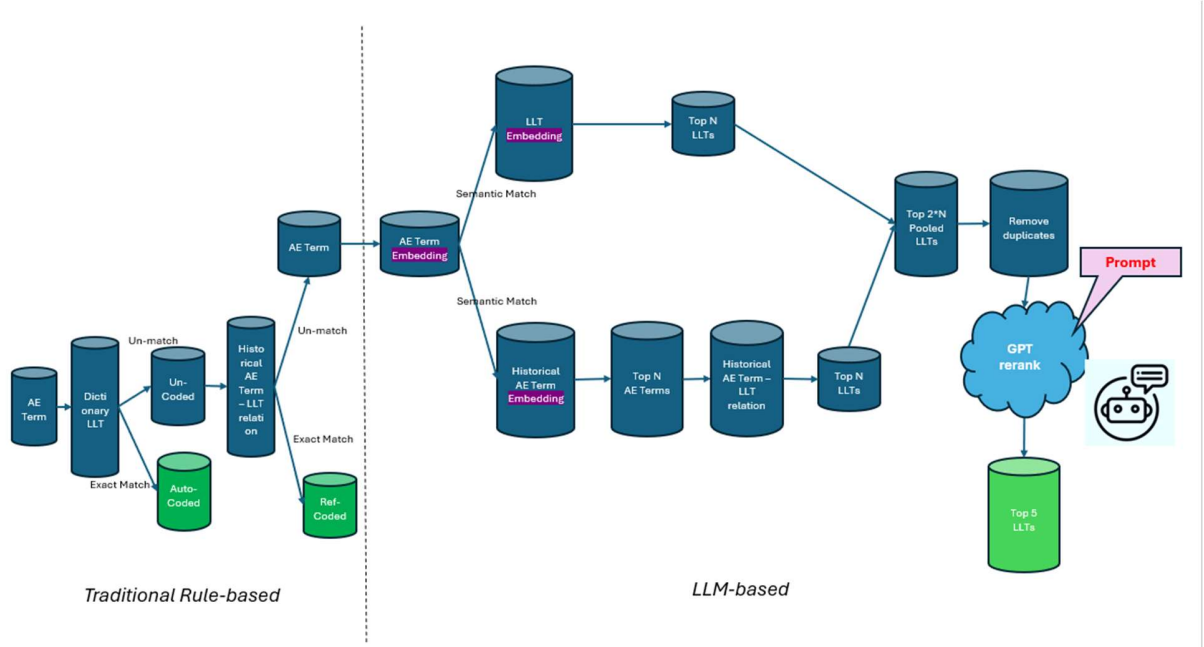


Figure 2. Workflow

TESTING RESULTS

We conducted thorough testing on a dataset of 8,350 unique historical medical coding records, after excluding records with exact matches between Adverse Event Terms (AETERM) and Low-Level Terms (LLT). Our goal was to identify the most accurate LLT from our comprehensive dictionary, which contains approximately 78,000 entries, for each AETERM in our test data. The test set included 835 records, constituting 10% of the total dataset, while the remaining 7,515 records, representing 90% of the dataset, served as a reference for historical coding patterns.

We utilized the Success Rate (also known as Hit Rate) to measure the system's performance. This rate is defined as the proportion of queries for which the correct answer appears within the top 'N' retrieved results. In simple terms, it represents the frequency at which our system accurately identifies the correct answer within its preliminary attempts.

PERFORMANCE BREAKDOWN BY METHOD:

The success rate for an exact match using either the LLT dictionary or historical coding is 0%, since we have excluded such cases from the testing data. This approach allows us to specifically evaluate scenarios where traditional rule-based matching would fail.

Direct Semantic Match Using LLT Dictionary: Success rate begins at 46% for the top 1 match and incrementally increases to 84% for the top 10 matches, reaching 90% for the top 30 matches.

Historical Coding Semantic Match: Starts identically to the dictionary match at 46%, but plateaus at a lower rate, peaking at 69% for the top 10 matches and 73% for the top 30 matches.

Combination of LLT Dictionary and Historical Coding: This method shows a significant improvement, starting at 51% and rising to 88% success rate for the top 10 matches, 95% for the top 30 matches, with the highest success rate of 96% from all 60 candidates.

Top 5 Predictions by GPT-3.5: Demonstrates a strong start at 59%, reaching a high of 88% by the top 5 matches, and 92% for the top 10 matches.

Top 5 Predictions by GPT-4: Exhibits the best performance with an impressive 68% success rate for the top 1 match, 91% for the top 5 matches, and 95% for the top 10 matches.

The comparison underscores the advanced capabilities of GPT models, particularly GPT-4, which significantly improves success rates. For instance, GPT-4's success rate at the top 6 matches (94%) surpasses the 93% success rate at top 20 matches achieved without the use of a GPT model. Our approach also benefits from learning from historical coding data, which can provide more appropriate candidates for AI models to refine further.

Method/Top N	1	2	3	4	5	6	7	8	9	10	20	30	60
Exact Match by LLT dictionary	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	-
Exact Match by historical coding	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	-
Semantic Match by LLT dictionary	46%	61%	69%	74%	77%	80%	81%	82%	83%	84%	89%	90%	-
Semantic Match by historical coding	46%	57%	60%	63%	64%	66%	68%	69%	69%	69%	72%	73%	-
Semantic Match by LLT dictionary and historical coding	51%	68%	74%	78%	82%	84%	86%	87%	87%	88%	93%	95%	96%
Top 5 selected by GPT-3.5	59%	76%	84%	87%	88%	91%	91%	91%	92%	92%	94%	96%	-
Top 5 selected by GPT-4	68%	81%	88%	89%	91%	94%	94%	95%	95%	95%	96%	96%	-

Table 2. Success Rate for Different Methods within Multiple Models

KEY FEATURES

HIGH ACCURACY AND EFFICIENCY

- ✓ In instances where the case extract match would fail, the system generally achieves a success rate of approximately 80% within the top 2 codings, 90% within the top 4-5 codings, and 95% within the top 10-20 codings.
- ✓ The overall success rate is expected to increase when extract match is also applied in actual studies.
- ✓ For 100 AETERMs, AI can complete the analysis in 2-10 minutes(GPT3.5/GPT4), allowing users to proceed with further analysis promptly.

AI NATIVE - FULLY LEVERAGING THE CAPABILITIES OF LLM

- ✓ Semantic matching through the strong embedding capability of LLM: Converts medical and coding terms into vectors, matching them by semantic similarity.
- ✓ Selection of top coding terms through Q&A with the LLM, informed by professional knowledge.

EFFECTIVE UTILIZATION OF HISTORICAL CODING KNOWLEDGE

- ✓ Corporate data and knowledge continuously enhance the model's performance, even when an exact match cannot be found in historical data.

CO-PILOT STYLE - INTEGRATION OF AI AND HUMAN INTELLIGENCE

- ✓ AI searches a comprehensive dictionary and selects the best candidates. Users can use system prompts and update them based on their expertise.
- ✓ Human review and decision-making:
    - Offers multiple parameters for flexible configuration, such as TopN, Threshold, GPT rank model, GPT TopN, etc.
    - Supports both batch coding and individual coding processes.
    - Incorporate multi-round chat functionality to allow users to utilize the GPT model more effectively. This feature will support detailed discussions and thought processes, enabling users to make informed final decisions.

INDEPENDENTLY DEVELOPED ALGORITHMS

- ✓ No dependencies on other software or open-source packages, except for the Azure OpenAI API.
- ✓ Features embedding, RAG, ReRank, and End-to-End solutions.
- ✓ High potential for future enhancement with more detailed in-house algorithms to meet customized business needs.
- ✓ Can be leveraged for other AI projects.

DATA PROTECTION - ALL WORK CONDUCTED IN A CORPORATE ENVIRONMENT

- ✓ Data is uploaded and analyzed on the Corporate RStudio Connect Server.
- ✓ Prompts and candidate LLT terms are analyzed on the Corporate Azure OpenAI and Databricks platforms.

CONCLUSION

The application offers flexible prompts, robust data protection, and excellent performance for coding. Our further plan as followings:

- Extensive testing of additional prompts will be conducted, employing detailed coding logic that considers the various levels of the MedDRA hierarchy, ranging from System Organ Class (SOC) to High-Level Group Term (HLGT), High-Level Term (HLT), Preferred Term (PT), and Lowest Level Term (LLT).
- Conduct tests with medical-specific embedding models to determine the most effective at interpreting and generating medical content, thereby enhancing the GPT model's accuracy for medical applications.
- To further narrow the scope before deploying the GPT model, an extra layer of re-ranking will be implemented, utilizing a range of algorithms. This strategy is designed to improve both the success rate and the efficiency of the application.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Hao Chen
BeiGene, Inc.
hao1.chen@beigene.com

Xuejie Zhou
BeiGene, Inc.

xuejie.zhou@beigene.com

Leo Li
BeiGene, Inc.
you.li@beigene.com