PharmaSUG 2024 - Paper PO-451

Simplifying Edit Check Configuration

Vandita Tripathi and Manas Saha, Tata Consultancy Services

ABSTRACT

Clinical Data Management is a pivotal process in clinical trial, and during clinical data management, majority of the data is directly captured into electronic case report forms (eCRFs). Without good quality data, there is no accurate statistical analysis. Hence maintaining Data Integrity and data quality in the eCRF Forms has a critical influence on data management and study success.

Edit checks are the first line of defense against manual data entry errors. They are small programs or algorithms that are used to validate the data to make sure it meets standards/requirements, these programs alert the user that data entry fields have (or may have) errors. Hence setting up and configure the edit checks correctly is a crucial step in designing eCRF forms.

In most eCRF Form designing solutions, configuring the edit checks is a cumbersome process. The form designer or edit check programmer needs to either use programing languages or use a low-code-no-code based UI tool to manually add the validation conditions and alert messages against each field that requires validation. Often this requires a long time to upskill the edit check programmer. Also manually setting up the edit checks requires a lot of time and effort.

An AI-based solution that can understand Natural language input and translate the same into edit check validation algorithm can solve this problem. The solution will accept a natural language plain English prompt from the form designer specifying the validation requirements, then the AI will translate the validation requirements into software programs/algorithms and will attach the algorithms in the eCRF fields. When a data entry operator fills the form, these algorithms will trigger to detect any discrepancy, and will warn the user accordingly.

This solution has the potential to greatly reduce the complexity of designing eCRF forms and edit checks. This solution does not need the operator to have programing skills, and it can drastically cut down the time and manual effort needed to design eCRF forms.

INTRODUCTION

Edit checks are crucial for ensuring data quality in a clinical trial. thus, configuring the edit checks correctly is a very important step for the study success. However, in most eCRF Form designing solutions, configuring the edit checks is not a very easy process. Often this requires custom software development by experienced developers. Some tools provide low-code no-code approach to setup and configure edit checks, but this approach also requires having experienced edit check programmers who are well familiar with using that tool.

The purpose of this paper is to present a new AI-Driven approach to overcome the challenges faced by the clinical trial industry for edit check setup. The solution implements cutting edge technologies like machine learning, contextual understanding, and Natural language processing to understand the custom edit check requirements from plain English text. Once understood, the system can use generative AI to automatically build the edit check software algorithm or code required to validate the forms.

In the following sections, this paper highlights the current challenges the industry is facing, as well as how the new solution can help to overcome the current challenges. This paper will also highlight the approach, architecture, and technology behind the new solution, and how the emerging technologies like artificial intelligence and generative AI has the potential to transform the clinical trial industry.

PRESENT CHALLENGES IN EDIT CHECK CONFIGURATION

In the present scenario, whenever the clinical trial managers need to setup edit checks, they may proceed with one of the two following approaches:

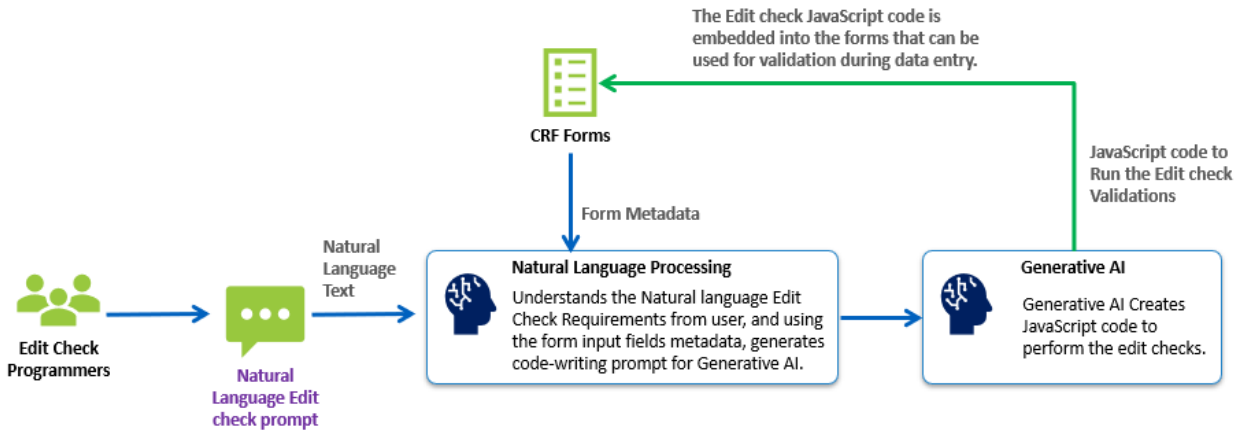
1. They communicate the custom edit check requirements to the application support team. The application support team in turn communicate the requirements to the developers. The developers get involved in discussion with the business owners to finalize the requirements, and they develop the new edit check algorithms in the system, which is made live after a series of testing and approvals.
2. If the application provides a mechanism for building a custom edit check, and the end user is familiar with the approach, they may build the edit check using the low-code no-code builder interface provided by the application. Often it requires special skills in that tool, and experience in low-code-no-code edit check building.

Neither of the above two approaches are ideal and have a quick turn-around time. The developer-based approach can take long time depending on the complexity and development lifecycle of the vendor. The edit check builder tool requires specialized skills and training that may not be always readily available to the end user.

PROPOSED APPROACH

In the proposed approach, the new solution will help to overcome the existing challenges using the following key principles.

1. Accept custom edit check requirement from user in plain natural text.
2. Leverage natural language processing to understand the requirement and context.
3. Use Generative AI to create algorithm / code that can perform the edit check.
4. Embed the machine- generated code with the eCRF forms so that when data entry operators submit the form, the edit checks are validated.



These key principles ensure that the user do not need to have specialized skills, and the turnaround time for edit check setup is quick and instantaneous.

The following sections elaborate the key principles in detail.

USER INTERACTION AND INPUT

In the proposed approach, the end users need to provide the custom edit check requirement in plain English text in either (1) a text input box, or (2) a chatbot style conversation interface.

In the text prompt, the users need to specify which fields they want to validate, and what will be the validation condition.

So, instead of interacting with a complex edit check builder interface, Users can just provide a plain simple natural English prompt like:

```
Subject Age must be between 18 and 60, Heart Rate should not be blank, and
Visit date must not be older than 1 week from present date.
```

This approach eliminates the need to have specialized low-code-no-code training or programing skills for end users.

NATURAL LANGUAGE PROCESSING AND REQUIREMENT UNDERSTANDING

Once the natural language input is received in the application, the machine learning model will analyze the input and will perform the following steps:

1. The model will identify the field names and the edit check conditions that the user has requested for.
2. The machine learning model will be pre-trained with the form metadata (Field List) of the application. Using this knowledge base, the model will be able to automatically map the requested field names and edit check conditions with the corresponding form field names.
3. Once the natural language input is received in the application and the context is understood by the application, The user request will be validated against few safety checks to prevent unauthorized operations – like script injection or cross-site scripting attacks.

GENERATIVE AI TO BUILD FORM VALIDATION ALGORITHM

Once the requested conditions are mapped with actual form field names, the system will perform the following steps:

1. a Generative AI model will generate a JavaScript code which, after few sanity checks, will be embedded with the e-CRF form.
2. When user submits the form, the JavaScript code will execute and will perform the field validation checks and will show appropriate validation error message to the user if validation fails.

CONTEXT UNDERSTANDING AND SMART RECOMENDATIONS

While user creates the edit checks using Natural language, the solution can analyze the eCRF form and will show context-aware smart recommendations to the user to suggest edit check that might be useful for that eCRF form.

Some Examples of the smart recommendations are:

1. Mandatory Field Validations

2. Start date and End date validation.
3. Date or Birth / Age Validation

Example of Smart Suggestions

For example, if the Form has a Patient ID field, and the machine learning algorithm is trained with historical forms that usually have this field set as mandatory, then the system will suggest the edit check programmer to set this field as mandatory.

Suggestions (Example):

Set "Patient ID" as mandatory field

Another example is, if the form has two fields as Start date, and End Date. Typically, the end date should not be less than the start date. In that case the application can show smart suggestions like:

Check if "End date" is greater than or equal to "Start Date"

ARCHITECTURE AND UNDERLAYING TECHNOLOGY

EDIT CHECK GENERATION ENGINE

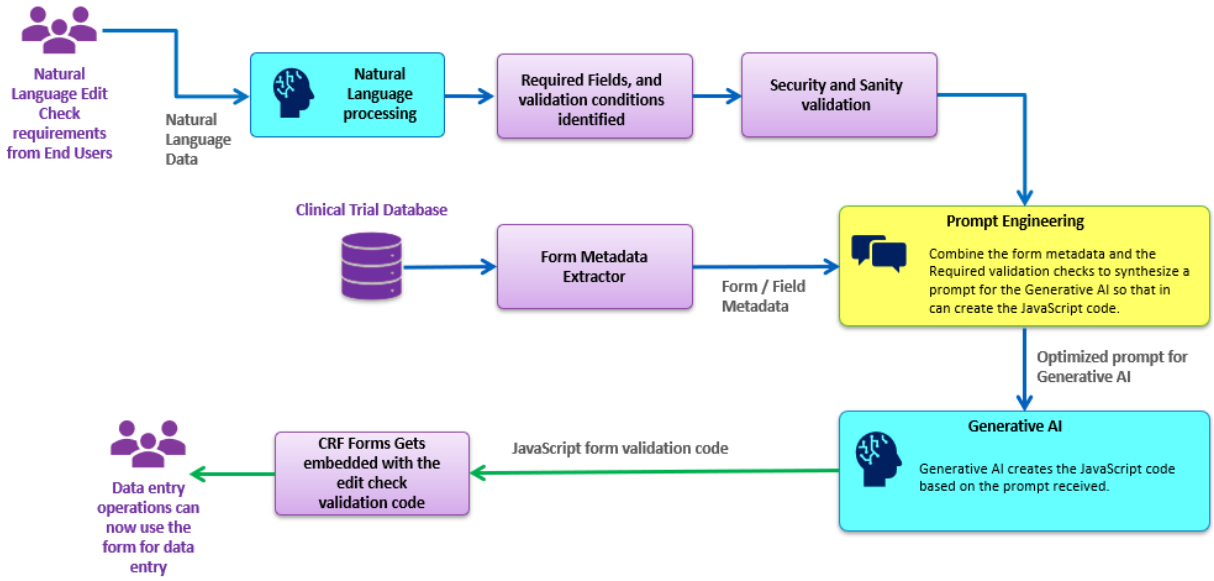


Figure 2. High level architecture of the Edit check generation engine

The proposed solution will implement the following key components as depicted in the below high-level architecture diagram.

1. **Natural language processing engine**: responsible for processing the initial user input. This will identify the fields and edit check conditions the user needs.
2. **Security validation block**: This block will ensure the natural language request does not contain any malicious intent like script injection.
3. **Form metadata extractor**: This module will scan the eCRF form metadata and will help to map the user requested validation checks along with the actual form fields.
4. **Prompt engineering block**: this module will use prompt engineering the synthesize an optimize prompt which will be fed into the generative AI to create the JavaScript code for form validation.
5. **Generative AI**: This module will be responsible for creating the JavaScript code from the engineered prompt.
6. **Updated eCRF forms**: the final module will embed the generated JavaScript with the eCRF forms so that the edit check validation can execute on submission of the forms.

SMART RECOMMENDATION ENGINE

The recommendation engine analyzes the original eCRF from metadata and generates few suggestions for creating new edit check conditions that the user might be interested in.

The recommendation engine uses a combination of the following method to prepare the recommendations:

1. **Collaborative filtering**: this method uses the data from other edit check programmers of similar therapeutic area forms to understand what edit checks are most used by users for similar forms.
2. **Rule based recommendations:** in this approach, the engine has pre-programed rules that is used to show recommendation if user is working with a specific form. For example, if user is working with a demography form, the system can show a suggestion to create edit check that makes the Age and Gender fields mandatory.
3. **Pre-Trained models**: using historical data from past user interactions with the edit check setup engine, a machine learning model is trained which can be used to show suggestions for similar relevant edit checks to the user.

BENEFITS OF THE PROPOSED APPROACH

The AI-based proposed solution that can understand Natural language input and translate the same into edit check validation code can solve most of the problems the clinical trial industry is facing for edit check configuration.

The following table depicts the advantages of the solution over the traditional approach.

Measurement Criteria	Traditional Approach (Low Code Method)	Traditional Approach (Programing Method)	Proposed Approach (AI-Based Method)
Method for configuring edit checks	By Selecting fields and edit check conditions in a low code platform.	By writing software code in the CRF forms.	By typing requirements in plain Natural language English text.
Is Specialized training required?	Yes – On the tool used to configure edit checks.	Yes – on programing languages like JavaScript	No specialized training needed
Complexity of setting up edit checks	Moderate	Difficult	Easy
Flexibility to set up Custom Edit checks	Not Flexible	Flexible	Flexible
Turn-around time	Few days to few weeks	Few weeks to few months	Few minutes to few hours

CONCLUSION

The AI-based proposed solution that can understand Natural language input and translate the same into edit check validation code can solve most of the problems the clinical trial industry is facing for edit check configuration.

This solution does not need the user to have Low-code-no-code edit check configuration skills, or specialized coding skills. On an average, a person needs to spend several days to master a low-code-no-code platform and learning programing may take months to master. With this proposed solution no upskilling is necessary.

On an average an edit check development takes weeks to months in existing clinical trial landscape. This proposed solution can cut down that turnaround time to minutes or hours, which has the potential to transform the clinical trial edit check configuration landscape.

REFERENCES

OpenAI “OpenAI Codex”.

Available at https://openai.com/blog/openai-codex

Google “Prompt Engineering for Generative AI”.

Available at https://developers.google.com/machine-learning/resources/prompt-eng

CONTACT INFORMATION

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

Vandita Tripathi
Tata Consultancy Services
vandita.tripathi@tcs.com
https://www.linkedin.com/in/vandita-tripathi-3154a358/

Manas Saha
Tata Consultancy Services
saha.manas1@tcs.com
https://www.linkedin.com/in/sahamanas/