PharmaSUG China 2023 - Paper AA-169

Metadata recycling - Open the door to AI programming in clinical trials

Zhiping Yan, Dizal Pharma
Huadan Li, Dizal Pharma

ABSTRACT

With the advancement of AI technology, more and more people are exploring the potential applications of AI in statistical programming. However, current implementations of artificial intelligence lack self-feedback and data/process learning capabilities across various scenarios. This paper proposes a metadata recycling method to digitize the statistical programming process, theoretically reducing the human intervention required for global metadata updates and enhancing the ability of systems to learn from data.

INTRODUCTION

AI technology is also more and more popular in statistical programming. The statistical programmers are investigating the implementation of artificial intelligence (AI) in end-to-end programming scenarios. While many believe that AI technologies rely solely on vast amounts of data, the collection and analysis of programmer behavior and metadata during the programming process are also crucial for training AI models in statistical programming.

There are two pain points in study level metadata collection. 1) How to automate recycling study level metadata and enhance global level metadata? 2) How to capture and analyze the programmer behavior? This paper demonstrates example of incorporating these two important technologies. Moreover, how to automate whole process and even form a self-learning system will also be described.

AUTOMATED METADATA-EXCHANGE PROCESS

Metadata driven processing of submission-ready artifacts becomes more common in recent years. Figure 1 shows the Automated Metadata-Exchange Process. There are two types of metadata: corporate level metadata and study level metadata. Corporate level metadata including SDTM metadata, ADaM metadata, Reporting metadata, Controlled Terminology metadata and other types of metadata are placed in Metadata Repository (MDR). They can be applied to automatically generate study level metadata which are placed in corresponding study level folders.

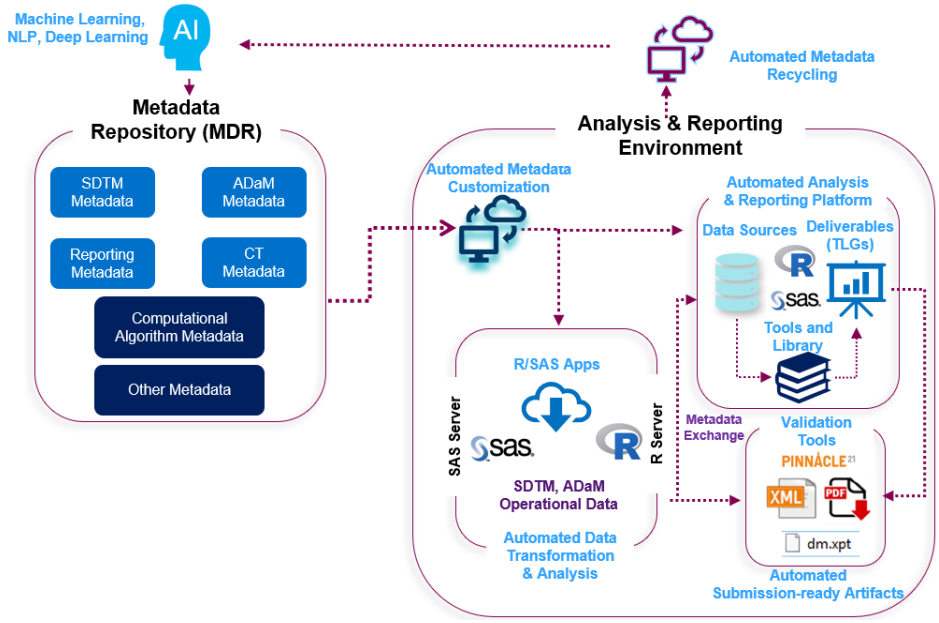


Figure 1. Automated Metadata-Exchange Process

Study level metadata can be used to automate generation of SDTM, ADaM, TLGs and other submission-ready artifacts. These metadata can be transmitted automatically among three main programming activities: Automated Data Transformation and Analysis, Automated Data Analysis and Reporting, Automated Preparation of Submission-Ready artifacts. Moreover, these study level metadata can also be stored automatically in another repository while above programming activities are going on.

The stored study level metadata can serve as Machine Learning input data to produce corporate level metadata. The new corporate level metadata are expected to improve accuracy of automatically generated study level metadata. With more and more data are accumulated, this kind of input output process cycle can make whole process increasingly automated and intelligent.

DIZAL-ISCP (INTELLIGENT GUI) BRING AUTOMATION FORWARD

Managing and processing of different types of metadata files as well as submission-ready artifacts is time-consuming and boring. To reduce manual work and increase work efficiency, a seamless and interactive platform consolidating SAS EG, Linux SAS, Python and other software like Adobe PDF, Microsoft Word, Microsoft Excel is urgently needed. At Dizal, we build more than 25 independent applications covering Basic, EDC, SDTM ADaM and Reporting related tasks per clinical data analysis flow. These applications are integrated into one platform (Dizal-iSCP) through which users can access, generate, and process different type of files by clicking on GUI buttons. In addition, the programmer behavior can be collected when click each button in standalone app.

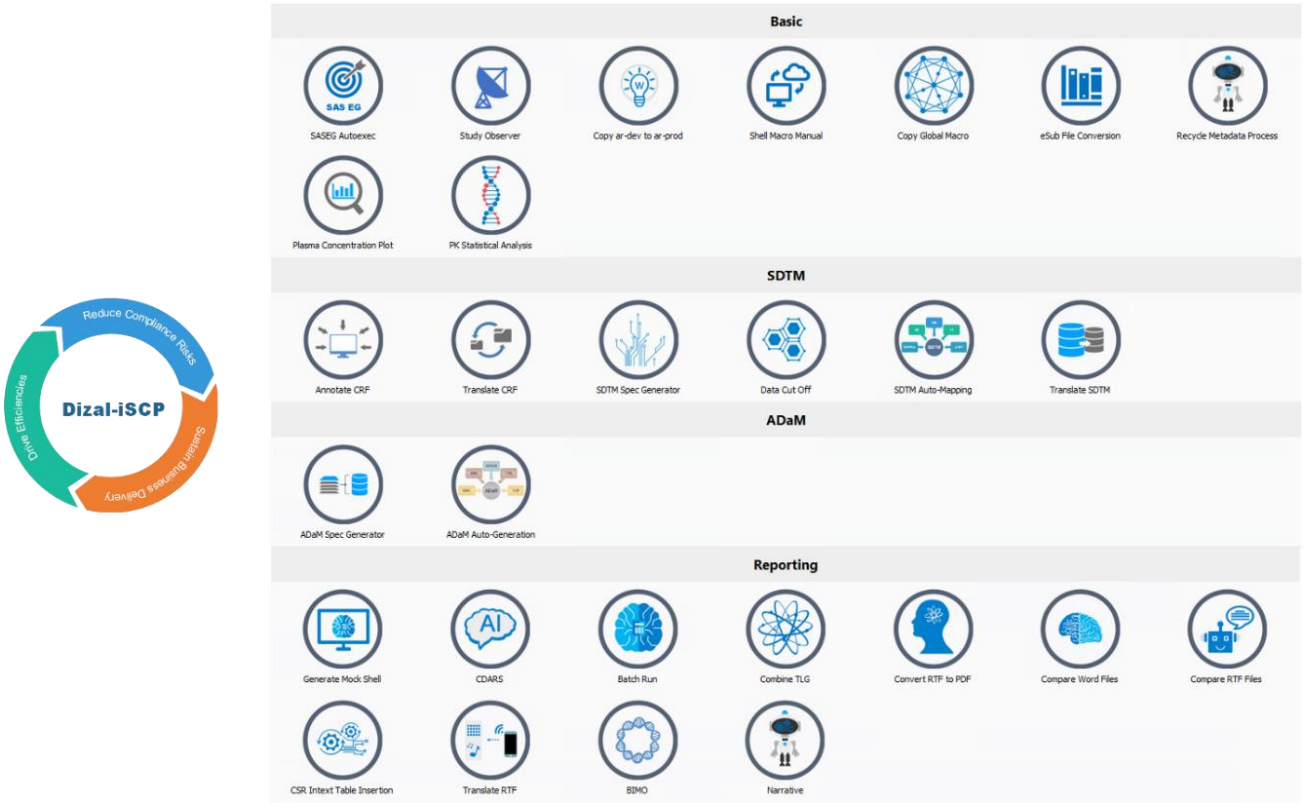


Figure 2. Dizal-iSCP Intelligent GUI Bring Automation Forward

AUTOMATED METADATA RECYCLING

To form the closed control system between corporate level metadata and study level metadata, another repository for placing recycled study level metadata is also required. By applying different types of independent applications, self-learning and self-processing metadata system can be built.

Each application consists of many interface elements like dropdown boxes and buttons. By interacting with specified interface element (e.g., operation x in Figure 3), initial study level metadata can be automatically generated from corporate level metadata. When interacting with another interface element (e.g., operation y in Figure 3), final study level metadata can be collected and placed in recycled metadata folder for centralization and further manipulation.

Through Recycle Metadata Process application, all recycled study level metadata can be pooled together. With required manual process as well as AI technology if needed, corporate level metadata can be updated automatically from pooled metadata.

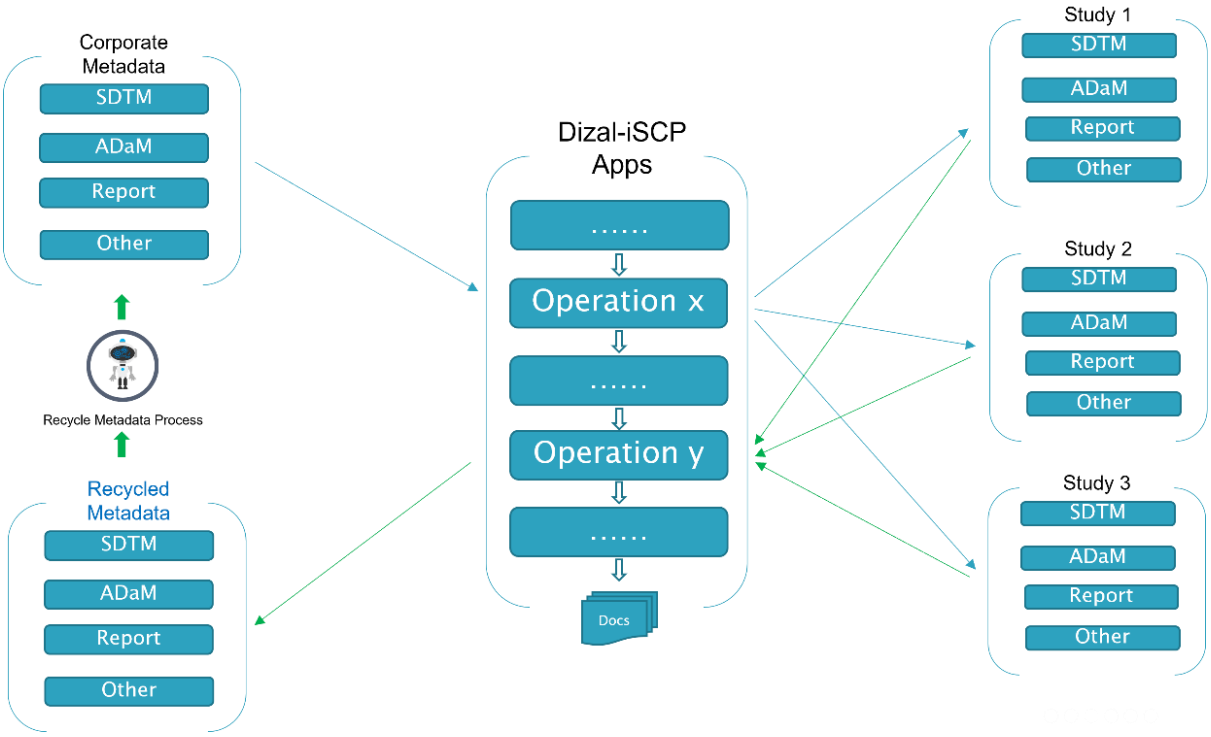


Figure 3. Automated Metadata Recycling

In addition to collecting study level metadata, these apps also automatically capture programming behaviors at the study level, allowing us to gain insight into all programming actions during the course of study. This data can be leveraged for behavior analysis and identification of patterns at the study level.

SIMPLE EXAMPLE - ANNOTATE CRF

WORKFLOW AND PROCESS

Figure 4 shows whole process for automated annotation of CRF. Essential inputs include edc2sdtm metadata file, ALS file and blank CRF from Data Management function. With blank CRF, study level edc2sdtm file and intermediate files deriving from the input files, bookmark can be added into blank CRF. Study level edc2sdtm can be served as input file to generate FDF file. By importing FDF file into CRF containing bookmark, generation of annotated CRF can be achieved.

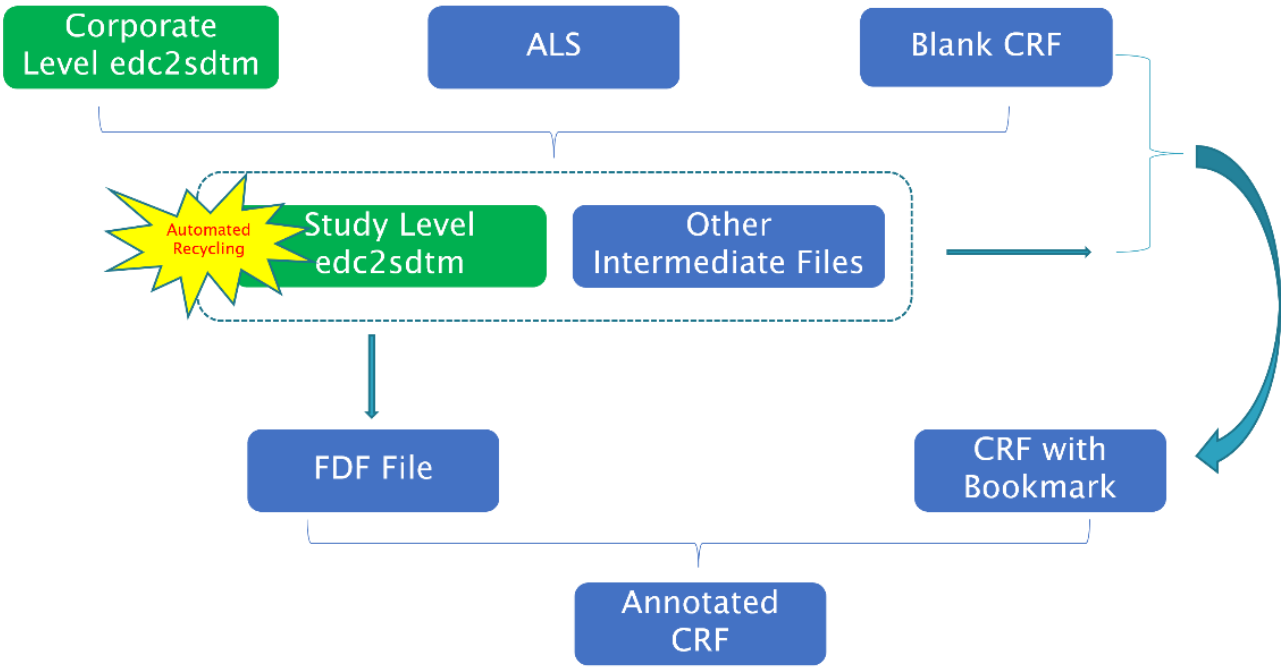


Figure 4. Annotate CRF Workflow and Process

AUTOMATED RECYCLING AND PROCESSING OF EDC2SDTM

From above introduction, it is not hard to find that edc2sdtm is a key and required metadata for CRF annotation. Figure 5 and Figure 6 shows recycling and processing of edc2sdtm and how to automate the process with Annotate CRF and Recycle Metadata Process applications.

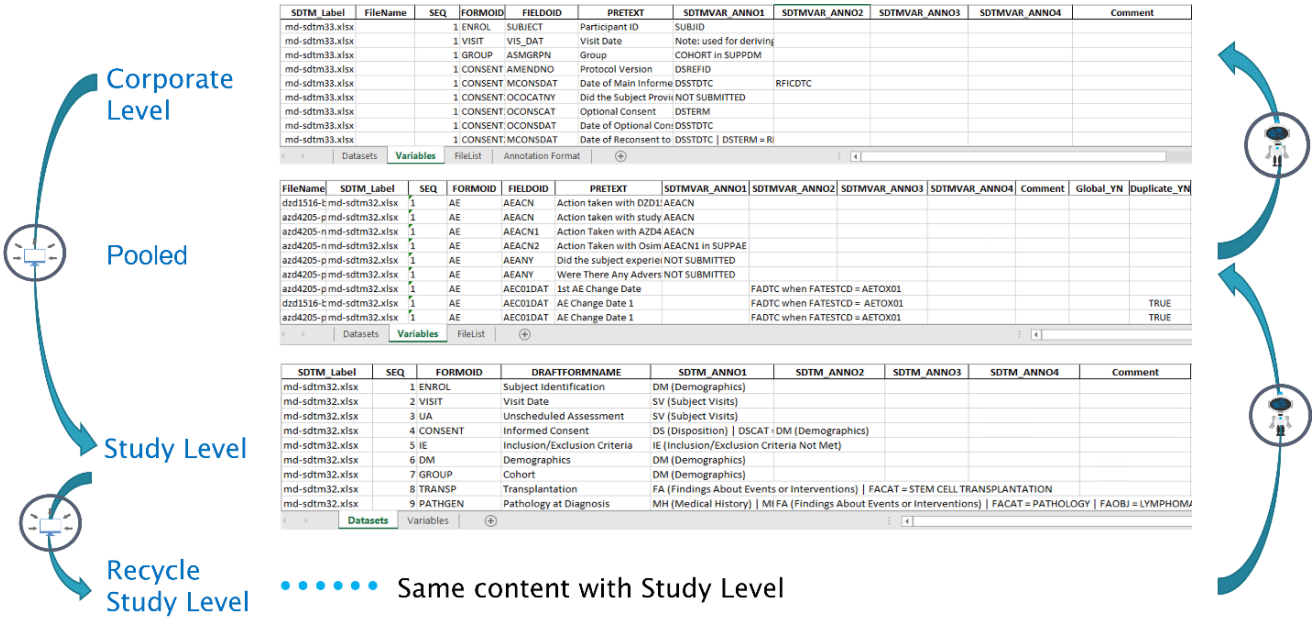


Figure 5. Automated Recycling and Processing of edc2sdtm

By clicking on button ‘Generate edc2sdtm template with global template’ from Annotate CRF application, study level edc2sdtm can be produced from blank CRF, ALS and corporate level edc2sdtm. When clicking on button ‘Link edc2sdtm and Position’ to add text coordinates, study level edc2sdtm can be recycled quietly. Users know nothing about the recycling.

Button ‘Pool’ from Recycle Metadata Process application can help pool all recycled study level metadata. Information like whether the annotations already exist in corporate level or whether the annotations are duplicate can be added into pooled metadata for manual check and process.

Once processing of pooled edc2sdtm is done, ‘Update Old Global’ button can help replace corporate level edc2sdtm with pooled edc2sdtm.

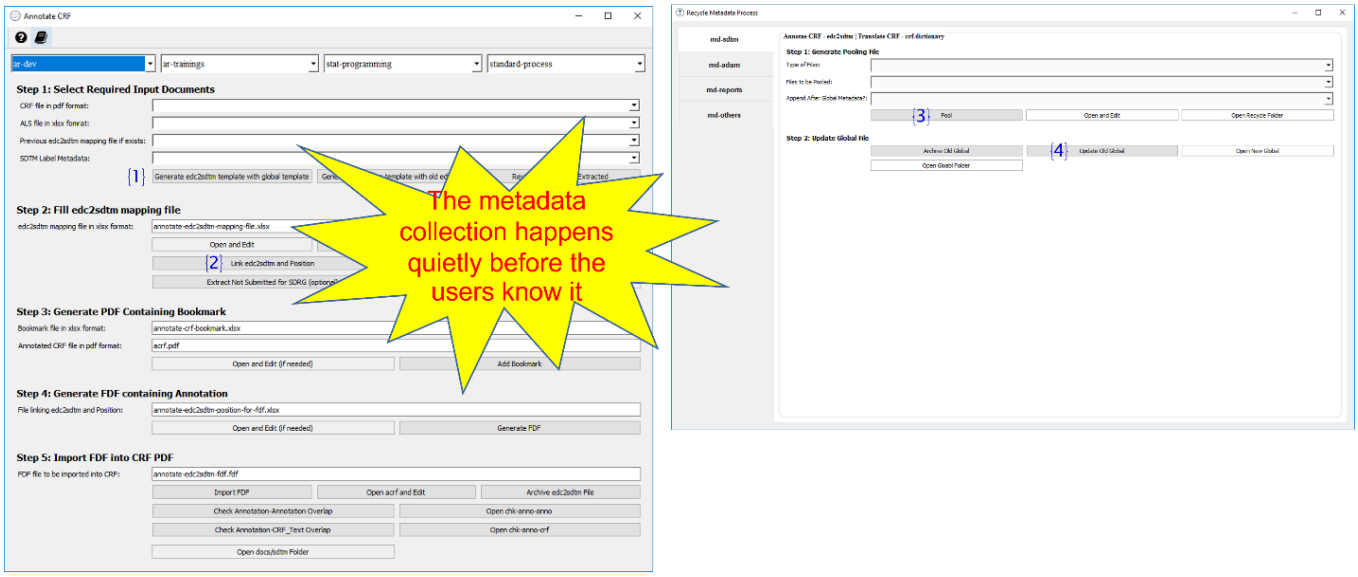


Figure 6. Annotate CRF and Recycle Metadata Process Applications

In fact, all these actions as the programming behaviors data during the course of CRF annotation are captured by the app “annotate CRF”. This vast amount of data can be analyzed to improve both the app and programming process.

COMPLICATED EXAMPLE – TLG NER

WORKFLOW AND PROCESS

At Dizal, both TLG mock shell and TLG outputs are produced based on standard shell ID. According to these shell ID, a set of SAS reporting macros are generated (Figure 7). For example, SAS macro m_ar_ae_socpt was intended to develop AE summary table having shell ID like AE0201, AE0202, AE0301 and AE0302. Based on these reporting SAS macros, TLG SAS program templates are generated.

Given shell ID and TLG SAS program template, automated generation of TLG SAS program file is half the battle. The remaining task is to fill macro parameters. This can be achieved by extracting NER from TLG title and other information from TLG mock shell. For details, please refer to our previous paper *Interlocking and Mutually Reinforcing System - From Shells to TLGs Automatically Generation* [1].

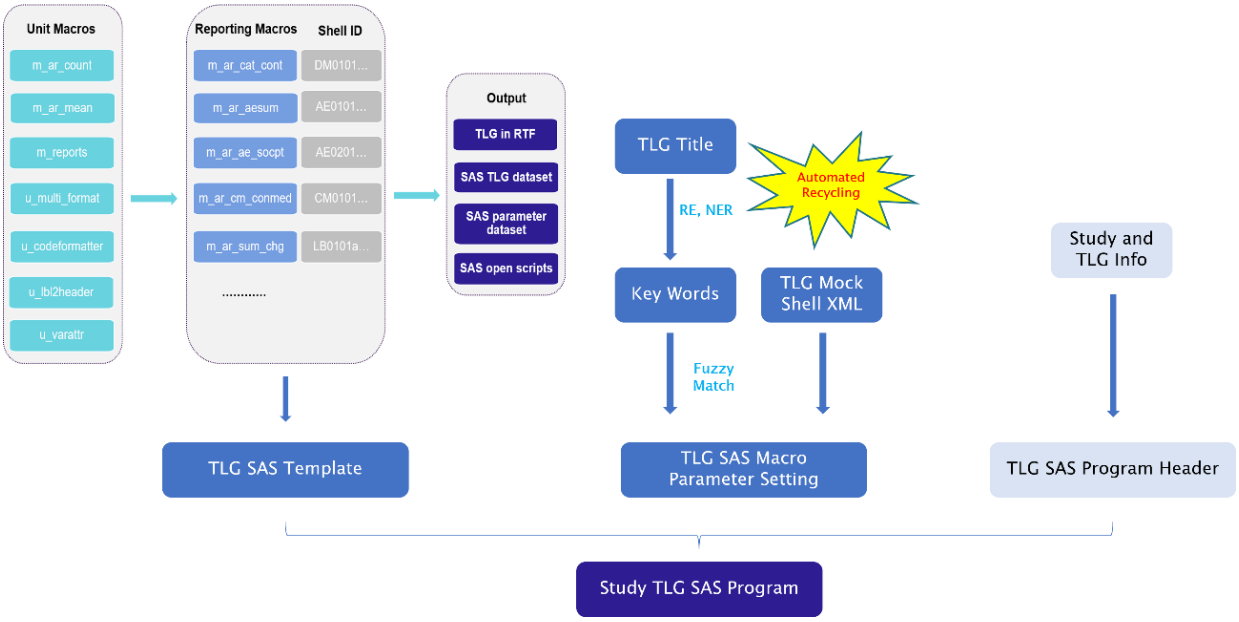


Figure 7. TLG SAS Program Generation Workflow and Process

Left part of Figure 8 shows an example of TLG SAS program template. Information required by program header and macro parameter value are broken into small units enclosed within brackets for replacement. With NER extracted from TLG title and mapping Text as well as mapping code, specific TLG SAS program file can be generated [1].

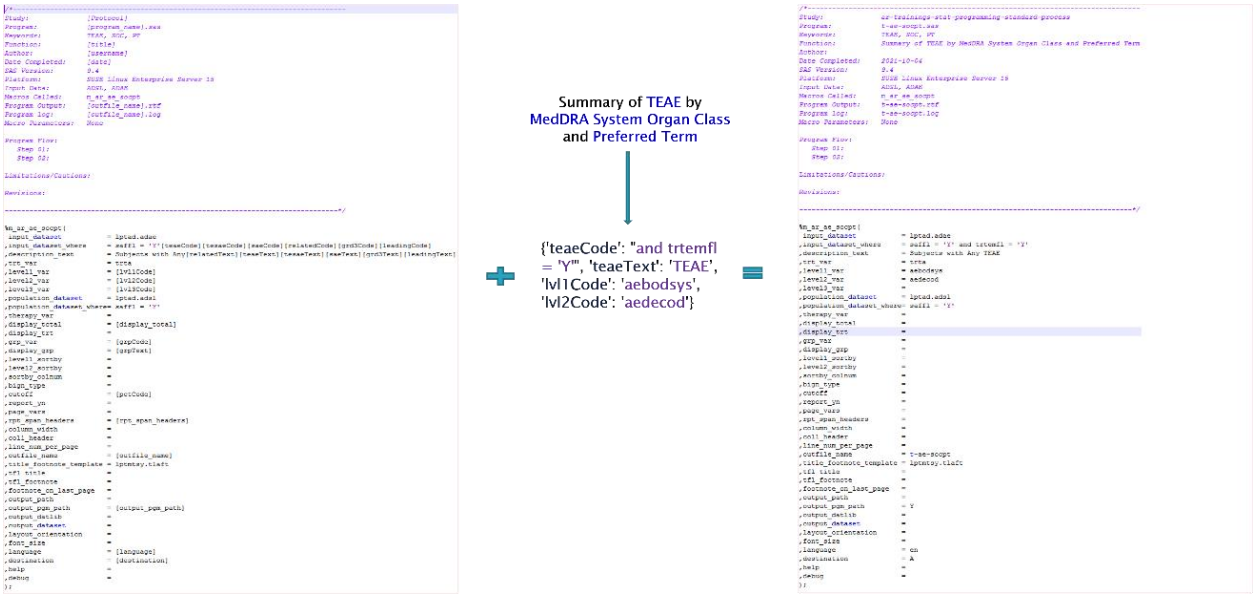


Figure 8. TLG SAS Program from TLG SAS Program Template and TLG NER

AUTOMATED RECYCLING AND PROCESSING OF NER

Unlike edc2sdtm metadata, there is no global NER model for initialization. The possible method is to use regular expression and named entity recognition to extract initial key words from each TLG in real studies as described above. This can be done by using CDARS application [1] when clicking on 'Generate Program File' right click menu. While at the same time, extracted NER will be saved under recycled metadata repository.

Recycle Metadata Process application can pool all study level metadata for manual check and process. After transformation of final pooled NER, the application can train a corporate level NER model. Once the number of accumulated NER is large enough, corporate NER Model can be used to generate study level NER which can be used for TLG production.

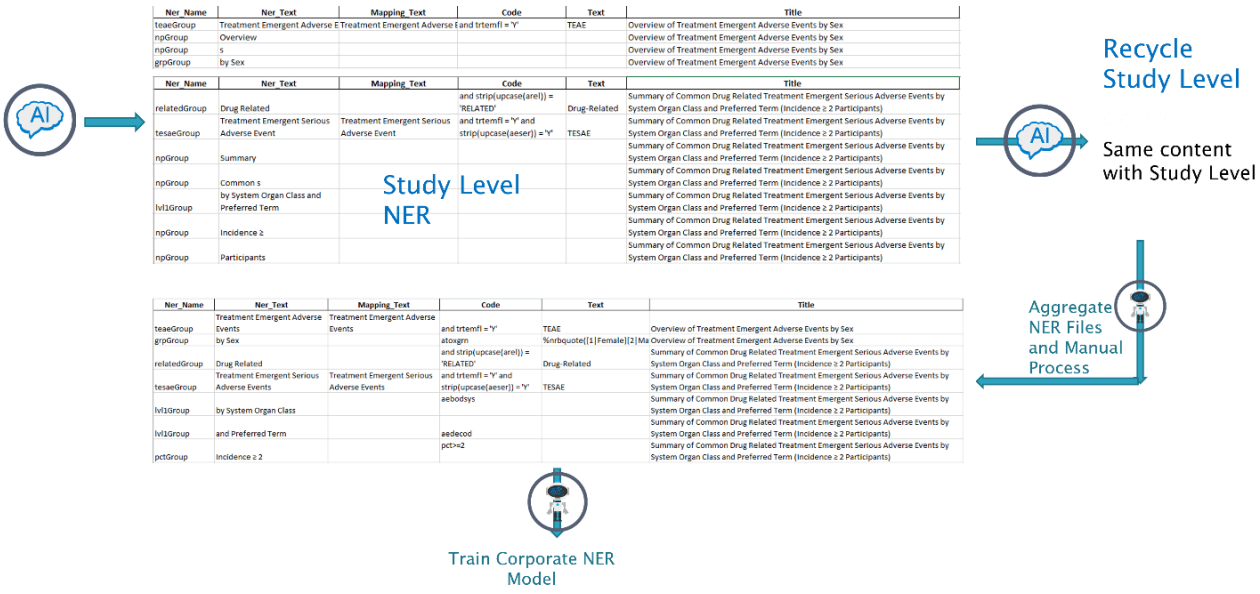


Figure 9. Automated Recycling and Processing of NER

Figure 10 presents Python sample code for training a NER model. By clicking on 'Train Model', the corporate level model can be built based on available NER.

```
with open(pkl_file, 'rb') as f:
        TRAIN_DATA = pickle.load(f)

    # Adding labels to the 'ner'
    for _, annotations in TRAIN_DATA:
        for ent in annotations.get("entities"):
            ner.add_label(ent[2])

    nlp.begin_training()

    pipe_exceptions = ["ner", "trf_wordpiecer", "trf_tok2vec"]
    unaffected_pipes = [pipe for pipe in nlp.pipe_names if pipe
not in pipe_exceptions]

    with nlp.disable_pipes(*unaffected_pipes):
        for iteration in range(30):
            random.shuffle(TRAIN_DATA)
            losses = {}
            batches = minibatch(TRAIN_DATA,
size=compounding(4.0, 32.0, 1.001))
            for batch in batches:
                texts, annotations = zip(*batch)
                print(texts)
                print(annotations)
                nlp.update(texts,
                        annotations,
                        drop=0.5,
                        losses=losses,
                        )
                print("Losses", losses)

    # Save the model to directory
    output_dir = Path(r'E:\test\03 model')
    nlp.to_disk(output_dir)
```

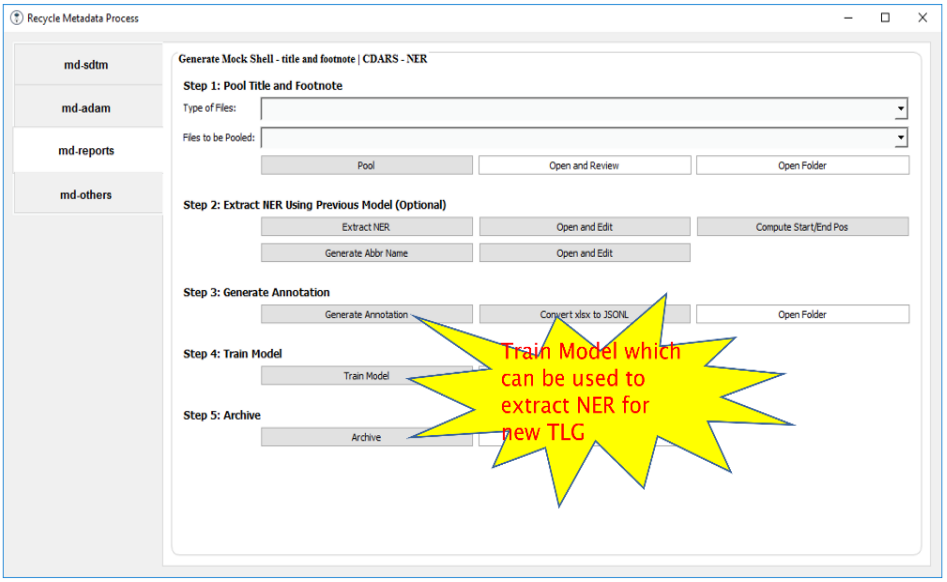


Figure 10. Code to Train NER Model and Recycle Metadata Process Application

AI REALIZATION IN STATISTICAL PROGRAMMING

The collection of extensive study-level metadata and behavioral data is integrated as a training set for AI models, enabling the identification of patterns specific to each study via machine learning technology. By matching our new study with this schema, we can obtain new study-level metadata for computer self-

programming. The AI scenarios in statistical programming are illustrated in Figure 11.

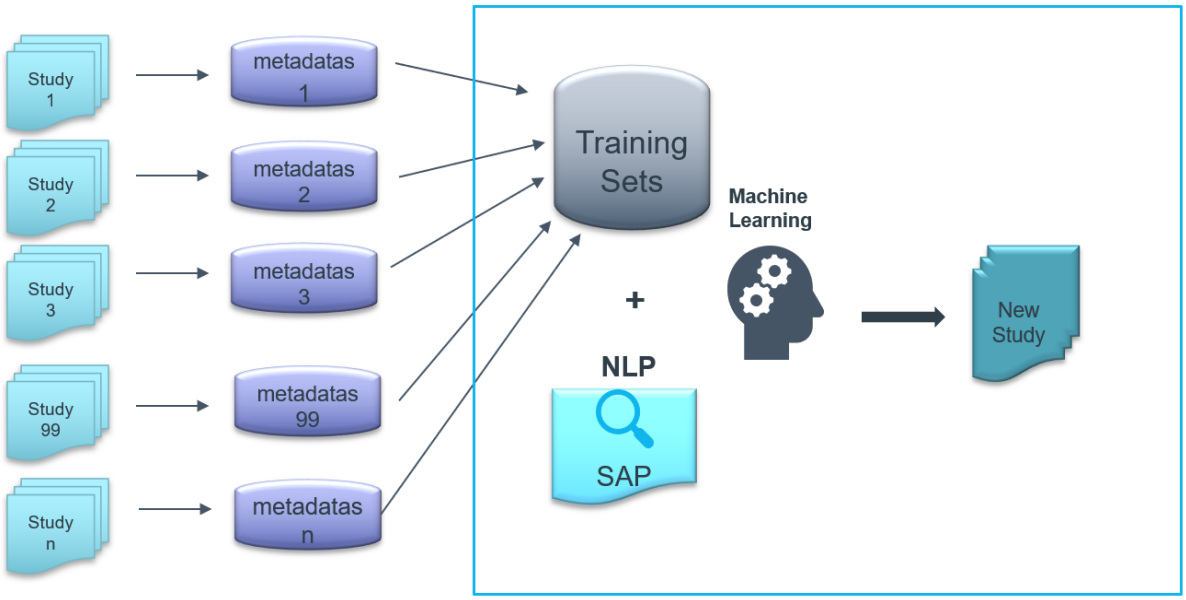
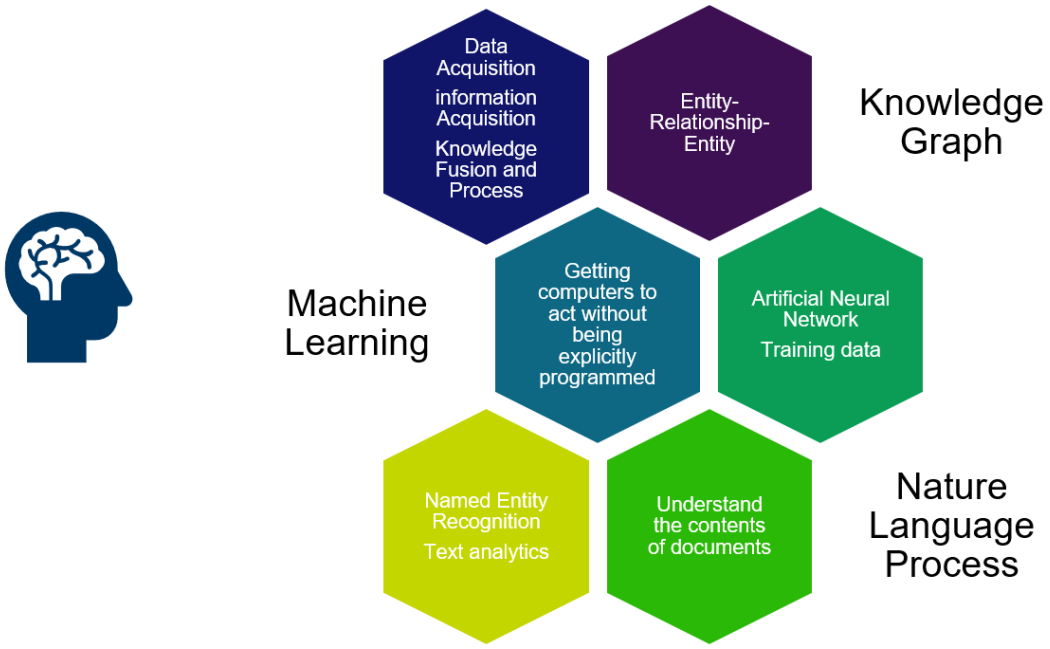


Figure 11. AI Scenarios in Statistical Programming

The three technologies, namely ML, NLP, and Knowledge Graph, can be effectively utilized for the analysis of aggregated study-level metadata and programming behaviors data as well as text mining.



CONCLUSION

This paper introduced how to build automated self-recycling and self-learning metadata system for clinical trial programming. With more and more metadata including programming behavior recycled from studies, we believe that the productivity and working efficiency can be improved further.

REFERENCES

1. Zhiping Yan. "*Interlocking and Mutually Reinforcing System - From Shells to TLGs Automatically Generation*", PharmaSUG China 2022.
2. Zhiping Yan, Huadan Li. "*The Concept of IoT – The Next Generation of Statistical Computing Platform"*, PhUSE EU Connect 2021

ACKNOWLEDGMENTS

I would like to thank Huadan Li and all statistical programming team members at Dizal for their advice and feedback so that the process can be improved continuously.

CONTACT INFORMATION

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

Zhiping Yan
Dizal Pharma
+86-13691423469
zhiping.yan@dizalpharma.com