Paper ET17

Recommender System for Study Build: Using the Protocol to Select CRFs from Standard Libraries

Komal Govil, Bristol Myers Squibb, Lawrenceville, NJ, USA
Karan Pande, Bristol Myers Squibb, Lawrenceville, NJ, USA

ABSTRACT

The goal is to build a machine learning model that will enhance data management capabilities and lead to faster and more standardized study builds. This will assist sponsors in providing draft CRFs to health authorities in an expeditious manner as well as decrease time to build a study. Here we discuss the methods used in the first attempt at the implementation of such a recommender system for CRF selection. Challenges include converting a pdf to text, data harmonization, natural language processing, feature selection, dimension reduction, continuous learning, and the cold-start problem. Future implementation to improve the model and recommendations, such as use of preexisting coding libraries, adjusting decision tree weights and using the ensemble method, will also be discussed. Welcome to the journey of using machine learning during study startup to promote operational efficiency.

INTRODUCTION

Imagine a computer program that can take in a protocol pdf, read it, and create an ALS that can easily be used to build a CRF within a few weeks. What if this was possible?

Per the Declaration of Helsinki, a study protocol is required prior to conducting research such as a clinical trial. It is typically a pdf document that outlines the design of the study. It includes sections outlining the purpose, on-trial procedures, primary endpoints and objectives, and sponsor information. The protocol is submitted to the IRB for each investigational site and used to design the database. Some health authorities may also request a draft CRF to be submitted alongside the protocol for consideration. Currently an extremely preliminary draft is being provided which is usually created by modifying the pdf of a similar study when available.

A major component of the database for a clinical trial is the CRF. The case report form (CRF) captures each patient's data within an electronic database such as Oracle Clinical or iMedidata Rave. The sponsor aims to design the forms so that all relevant information is captured without redundancy. Any time information is captured in 2 places it must be reconciled between the two locations to ensure consistency. In order to help maintain consistency across the portfolio standard libraries are created by sponsors.

One way to structure a standard library is to have a single library superset that contains every possible form, field and dictionary value that is then broken into different possible subsets by therapeutic area and disease indication. The benefits of such a library include standardization across all compounds and indications throughout the company. As each library is a subset of the universal, forms can be used as is from the subset libraries with very few allowed modifications. Any additional modifications require change requests that the cross-functional standards committee must deliberate. This library design leads to a lot of standardization that can then be used to program global reports and quicker study builds since less modifications have to be made. However, even with this library design a typical CRF will take at least 12 weeks to build.

The ALS or Architect Loader Spreadsheet can be used to build a RAVE database. This is an excel file that outlines the forms fields and dictionaries. A combined file with all of the studies, subset libraries and other projects on the URL is also available and can be used easily and quickly look across studies. Furthermore, a database specification tool already exists that can quickly and easily build an ALS if provided the number of instances of each form and the applicable fields for each form.

Our goal is to build a machine learning program that can take in a protocol pdf and output a recommended ALS for the study. This could potentially cut build time in half and will have benefits both upstream and downstream.

CHALLENGES WITH ATTEMPT 1: A RECOMMENDER SYSTEM

In 2006 Netflix launched a million-dollar competition to improve the Netflix recommendation algorithm. This led to increased research and development into recommender systems. Recommender systems can be categorized into 3 types – collaborative filtering, content-based filtering and hybrid models.

A collaborative model looks at other similarities between users, in this case studies, and makes recommendations for CRFs based on user similarities and differences. A content-based model would look at the similarities between CRFs and makes suggestions based on this. For example, if a randomization CRF is selected, an enrollment CRF may not be needed. A hybrid model, like the one used by Netflix, combines both approaches to provide better recommendations. This is a type of ensemble method where multiple algorithms are combined to give better predictability. In this case our item profiles (ALS) are structured, however our user profiles (protocols) are unstructured.

This was our first challenge. Dealing with a pdf is very different from dealing with a table or a database. In order to work with the protocol, the pdf forms must first be converted to text files. Not every pdf may be converted to text in the same way and footers and page numberings may need to be removed or conventions must be established to harmonize the data. Once that is completed this becomes a natural language processing task to intelligently read the protocol text file and extract the necessary information.

Protocol documents are .pdf files that are many pages long and contain lots of formatted text per page. Since protocol are blobs of text, they contain lot of stop words which increase the size of text thereby increasing the size of the input data. To overcome this problem, stemming, a data engineering feature was considered. However, with the proposed solution this step, along with tokenization is no longer necessary.

This is more difficult than it seems since protocols contain a lot of jargon that may not be easily identified by the program. For example, a reader knows that the medical monitor, physician, MD, doctor all could potentially refer to the same person, but this may not easily be recognized by the program. Similarly, cancer and oncology have similar meanings but simply changing the word to cancers instead of cancer may mean the program will not pick it up. Therefore, it was prudent to look into preexisting coding libraries to help mitigate this problem. These libraries are complex themselves and are not easily compatible with current NLP packages.

Even after conversion to text, the protocol is still unstructured data. Determining which features add value and which don't, or feature selection, in this data is a difficult resource intensive task. Determining a strategy for structuring the data is difficult. Even if the data is structured the protocol is long and selecting the best features that lead to better predictions, also known as dimensionality reduction, is difficult.

In order to determine the forms, the idea was to use a machine learning decision tree to find a solution. One of the problems with this solution is that each decision point is not explicitly written. For example, we cannot tell the program to include this form when this condition is met and not otherwise. The decisions made by the algorithm are a black box that cannot be controlled, manipulated, or even viewed. Without visibility into the program, it is difficult to weight the nodes to lead to better recommendations.

This leads to further problems depending on the current user database. A protocol is created for a completely new disease indication or compound would have no other protocols to refer to in order to provide suggestions. This is known as the cold-start problem. Similarly, newly created standard fields may never get selected since most older protocols may not use them. The program needs to be able to continuously learn and update its suggestions.

BACK TO THE DRAWING BOARD WITH ATTEMPT 2

Every failure brings you one step closer to success. With all of these challenges in implementation in mind we went back to the drawing board. Sometimes the solution is "Elementary, my dear Watson". We realized that sometimes Occam's razor is the path forward. Instead of displaying functional fixed on using machine learning to solve this challenge perhaps there was an even simpler solution. With this in mind, we came up with the following steps.

1) Convert protocol pdf to be readable
2) Use a pre-existing NLP Model
3) Select CRFs using a simple decision tree
4) Interface with an excel macro to build an ALS

STEP 1: CONVERT PROTOCOL TO BE READABLE

Most protocols are PDF files. PDF files retain their formatting and therefore are not easy interpretable as text. To extract text from PDF, the PyPDF2 python library was used. For simplicity, instead of converting from .pdf to .txt this library was used to interpret the protocol as text.

STEP 2: SELECTING CRFS USING A SIMPLE DECISION TREE

We've discussed the challenges with using machine learning to recommend forms. The biggest challenge is that all decisions will be a black box that cannot even be viewed. Therefore, we suggest using a simple decision tree instead of another machine learning algorithm.

Each node of the tree would be a very simple question, such as is this a randomized study if yes include the randomization form. The decision would be extracted from the protocol by the next step of the program. Additional user input such as connections to other databases, safety, IRT, etc could also easily be added if a simple tree is used.

STEP 3: USE OF A PRE-EXISTING NLP MODEL

We decided to use a pre-trained NLP model. Huggingface uses a combination BERT & T5 encoder-decoder based model which is a self-supervised transformer model. Self-supervised learning is a type of training in which the objective is automatically computed from the inputs of the model. Since the model is pre-trained additional resources to train the NLP model are not required, this also helps us in solving cold-start problem which many NLP models face. Since HuggingFace is trained on Generic Raw data, NLP Model developed using such datasets would work on any Document blob or Protocol in our case. A combination of Exact-Match and F1 score helps the model to predict appropriate answers for questions asked.

We used the question-answering pipeline from huggingface. Huggingface NLP models help to retrieve answers for questions provided context. The advantage of this pipeline lies in it's ability to encode distance between words in order to determine context. In our case the context provided to the model is the protocol itself. Each question answer pair will be a node in a decision tree and depending on the answers the CRFs will be selected.

STEP 4: USE THE SELECTIONS TO CREATE AN ALS

Once the question-answering program determines the nodes of each branch of the decision it can interface with an excel based macro to create the ALS. The program should then be able to select the correct subset libraries and the correct forms and fields from each of those libraries to build a CRF. The program should be programmed to be over conservative and include more forms than necessary, so that all potential options are presented to the cross-functional teams during review.

With this final step the program will be able to take the protocol and create an ALS that can then be uploaded and built in an interactive environment for cross-functional team review in less time than the build takes today.

CONCLUSION

CHATGPT WILL NOT TAKE YOUR JOB

Although HuggingFace is based on the same sort of technology as ChatGPT, known as BERT transformers, it will not take you job. A human is still required to discuss the rough draft of the CRFs with the cross-functional team and reach a consensus for database build. What this method does is speed up the development of the first draft of the CRF leading to a faster and more standard build. Teams can therefore focus on the nuances of their protocol and utilize standards wherever possible.

NEXT STEPS

As this program is implemented a greater focus can be placed on developing standard libraries and programs that can be used for multiple studies. The first step would be to build a simple decision tree for the existing libraries and forms. This is partially already completed by the standards team in their training and user guides and conditional comments provided. It would simply need to be modified into a question format to easily be programmed.

Using this method cross-functional teams can spend more time reviewing the actual database and understanding its functionality instead of a pdf of CRF forms that may differ from the online user's experience. The build time could easily be cut in half and those long CRF review meetings can be significantly shortened.

Programming must be completed to ensure the program is able to interface with existing tools and libraries. Additional challenges may be faced during actual implementation.

BENEFITS

Time is money. Decreasing the time to build a customized rough draft of the ALS not only saves time and resources but also helps deliver better drafts for health authority approval. The algorithms used are open source. The cost of implementation for the company would be negligible when the savings are considered. Having said that, the building of such a program requires an initial investment of resources and testing before it can be rolled out on a larger scale.

Use of an algorithm to develop first drafts will lead to better adherence to standards. This not only decreases the time for study build but can also reduce the number of costly migrations. It can also reduce time and resources required for report development for cleaning data as reports can then be created globally rather than on a study specific level. It also allows GBDS to reliably pool data from different studies. This will increase the scalability of the industry and can lead to offering more clinical trials to patients.

CALL TO ACTION

Artificial intelligence is everywhere; we use it in our lives every day. The challenge is to incorporate the technological advances in our work so that we can deliver medicine to patients in need faster and with quality. Be curious about the technological advances happening and be willing to experiment and apply your domain knowledge to come up with new solutions.

RECOMMENDED READING

Karia, S., & Taylor, B. (2021). *Using AI to drive automation for clinical data, from protocol to submission*. Retrieved February 24, 2023, from https://www2.deloitte.com/content/dam/Deloitte/us/Documents/technology/us-deloitte%E2%80%99s-automated-clinical-life-cycle-management-product.pdf

Govil, K. & Vundela, S. (2019). *Utilizing Artificial Intelligence for Efficient CRF Design.* Retrieved February 24, 2023, from https://www2.deloitte.com/content/dam/Deloitte/us/Documents/technology/us-deloitte%E2%80%99s-automated-clinical-life-cycle-management-product.pdf

Wolf, T. Debut, L. et.al. (2020). *Transformers: State-of-the-Art Natural Language Processing.* Retrieved February 24, 2023, from https://github.com/huggingface/transformer

CONTACT INFORMATION

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

Author Name: Komal Govil & Karan Pande
Company: Bristol Meyer Squibb
Address: 3551 Lawrenceville Rd, Lawrence Township, NJ
City / Postcode: 08648
Work Phone: (800) 332-2056
Email: komal.govil@bms.com, karan.pande@bms.com