The LeafAI query engine: a data model-agnostic query generator for cohort discovery and multi-hop biomedical reasoning using natural language

Nicholas J Dobbins[+1,2], Bin Han[4], Weipeng Zhou[1], Kristine Lan[3], H. Nina Kim[3], Robert Harrington[3], Özlem Uzuner[5], Meliha Yetisgen[1]

[1]*Department of Biomedical Informatics & Medical Education, University of Washington, Seattle, WA, USA*
[2]*Department of Research IT, UW Medicine, University of Washington, Seattle, WA, USA*
[3]*Department of Medicine, University of Washington, Seattle, WA, USA*
[4]*Information School, University of Washington, Seattle, WA, USA*
[5]*Department of Information Sciences and Technology, George Mason University, Fairfax, VA, USA*

[+]Corresponding author: Nicholas Dobbins, MLIS, Department of Biomedical Informatics and Medical Education, University of Washington, 1851 NE Grant Ln, Seattle, WA 98195, USA; ndobb@uw.edu.

Word count: 3925

Keywords: clinical trials, natural language processing, machine learning, electronic health records, cohort definition

Abstract

Objective: Finding patients within clinical databases meeting certain criteria is a critical step for clinical trials and biomedical research. Yet accurate query generation often requires extensive technical- and biomedical domain-specific expertise and knowledge of data models and terminologies. We sought to create a system capable of generating data model-agnostic queries while also providing novel logical reasoning capabilities for complex clinical trial eligibility criteria.

Materials and Methods: The LeafAI query engine utilizes a micro-service-based pipeline approach for query generation using named entity recognition and relation extraction, sequence to sequence transformation into logical intermediate representations, normalization, reasoning using a knowledge base of the UMLS and linked ontologies, and finally query generation using database schema semantic metadata. To evaluate our system, we compared patients found by queries generated by LeafAI and a human database programmer to patients enrolled in actual clinical trials at our institution. We measured each system by the number of actual enrolled patients matched in their queries.

Results: LeafAI matched a mean 43.5% of enrolled patients across 8 clinical trials, compared to 27.2% matched in queries by a human database programmer.

Conclusions: Our work contributes a state-of-the-art data model-agnostic query generation system capable of conditional reasoning using a knowledge base. We demonstrate that results of patients found by the LeafAI query engine can rival that of a human programmer in finding patients eligible for clinical trials.
Keywords: social determinants of health, natural language processing, machine learning, electronic health records, data mining

INTRODUCTION

Identifying groups of patients meeting a given set of eligibility criteria is a critical step for recruitment into randomized controlled trials (RCTs). Yet many trials fall short of recruitment goals, leading to time and cost overruns while creating challenges in ensuring adequate statistical power [1, 2]. Failures in adequate recruitment may result from a variety of factors, but can often stem from difficulties in translating complex eligibility criteria into queries using data collected in the electronic medical record (EHR) [3]. Despite these difficulties, RCTs increasingly rely on EHR data as a useful and more expedient means of identifying potential patients versus manual chart or case report form review [4]. At the same time, EHRs increasingly capture and store patient health and outcomes data in greater volume and variety, creating additional challenges and opportunities for patient recruitment [5]. While more granular - and potentially useful - data may be captured and stored in EHRs than in the past, the process of accessing and leveraging those data often require extensive technical expertise and knowledge of biomedical terminologies and data models.

Cohort discovery tools such as Leaf [6] and i2b2 [7] may be used in many cases, as they offer relatively simple drag-and-drop interfaces capable of querying EHR data to find patients meeting given criteria [8]. Yet these tools are not a panacea, as they often have significant learning curves and may be unable to represent particularly complex nested or temporal eligibility criteria [9]. Moreover, existing cohort discovery tools lack functionality to dynamically reason upon non-specific criteria that frequently appear in real-world eligibility criteria. For example, a criterion may require patients "indicated for bariatric surgery", but translating such non-specific criteria into a query (e.g., patients with a diagnosis of morbid obesity or body mass index greater than 40) must be performed manually by a researcher, even in cases where constructing an exhaustive list of such criteria may be time-intensive, subjective, and error-prone.

In recent years, alternatives to web-based cohort discovery tools have been explored. In particular, various methods using Natural Language Processing (NLP) have been put forth by the research community [10–18]. NLP-based cohort discovery methods hold unique potential and appeal, as they are hypothetically able to leverage existing eligibility criteria described in natural language, a medium researchers and investigators already use and are comfortable with.

BACKGROUND AND SIGNIFICANCE

Various methods for cohort discovery using natural language processing have been explored. Exploring database query generation methods, Yuan *et al* developed Criteria2Query, a hybrid information extraction (IE) pipeline and application which uses both rules and machine learning to generate database queries on an OMOP [19] (Observation Medical Outcomes Partnership) database, later expanded by Fang *et al* [12]. Other research has used encoder-decoder neural architectures for transforming clinical natural language questions into SQL queries [16, 20–23]. These studies have included exploration of cross-domain transformations, where systems much generalize to unseen database schema[21], handling of typos and abbreviations[20], and the generation and leveraging of intermediate representations between a natural language utterance and final SQL database query.[23]

Beyond database query generation, other methods explored to date include document ranking and classification[11, 14], where clinical notes are summarized, ranked and classified as relevant to a given eligibility criterion, and embedding projections for entailment prediction[13, 16], where predicting that a patient can be inferred from a given eligibility criteria equates to eligibility. Further studies have also explored the use of ontologies and OWL-based reasoning in determining eligiblilty[15, 24–27].

Gaps and opportunities

Among methods for database query generation, which we focus on in this study, most efforts to date are capable of generating database queries on only a single database schema, such as OMOP or MIMIC. While the OMOP database schema is widely used in research, this lack of flexibility and adaptability toward other data models limits potential utility, in particular given the widely documented necessity to change and extend the standard OMOP schema to accommodate real-world project needs [28–34]. Moreover, most methods, particularly those using encoder-decoder architectures, tend to generate relatively simple SQL statements, with few JOINs or nested sub-queries and typically no support for UNION operators and so on. This relative simplicity contrasts with the complexity of real-world clinical databases and it is not clear whether such methods alone are yet viable in a real-world RCT context. Additionally, few of the methods described provide support for complex logic, and none support reasoning on non-specific criteria (e.g., "diseases that affect respiratory function"), two phenomena common to eligibility criteria [3, 35].

Perhaps most importantly, to the best of our knowledge, only one previous work has been tested in terms of matching patients enrolled in actual clinical trials [13], and none have been directly compared to the capabilities of a human database programmer.

Key Contributions

We introduce the LeafAI query engine, an application capable of generating database queries for cohort discovery from free-text eligibility descriptions. This work contributes the following:

1. A novel database schema annotation and mapping method to enable **data model-agnostic** query generation from natural language.
2. Methods for transforming and leveraging **intermediate logical representations** of eligibility criteria.
3. A **corpus of human-annotated eligibility criteria and corresponding logical representations** available to the research community[1].
4. Methods for dynamically **reasoning upon non-specific criteria** using an integrated knowledge base of biomedical concepts.

We measure performance of the LeafAI query engine by comparing patients found by generated queries to those enrolled in actual past clinical trials using LeafAI and a human database programmer.

MATERIALS AND METHODS

System Architecture

The LeafAI query engine was designed using a modular, micro service-based architecture with a central API (Application Program Interface) which orchestrates end-to-end query generation. Inter-module communication is performed using gRPC [36], a robust open-source remote procedure call framework which enables language-agnostic service integration. This allows individual modules to be implemented (and substituted) in programming languages and using libraries well-suited to a given task. A diagram of the LeafAI query engine architecture is shown in Figure 1.

At a high level, query generation is performed in the following steps:

[1] Will be made available upon article acceptance

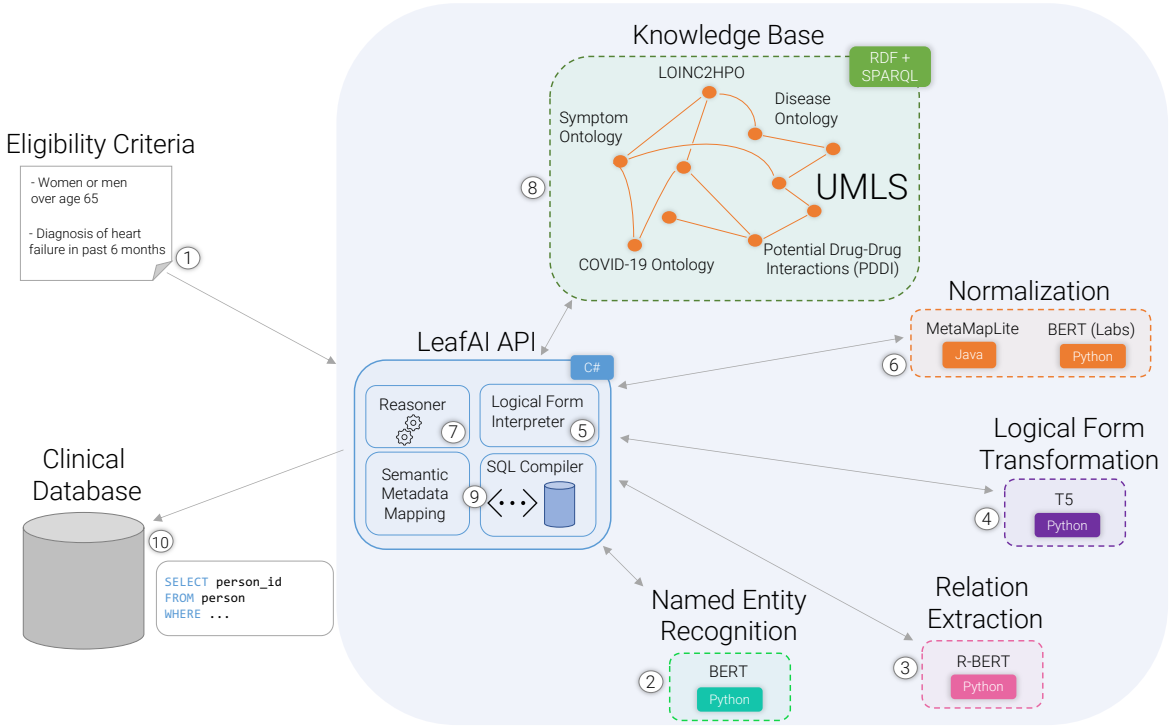

Figure 1: LeafAI query architecture. Inter-module communication is performed using the gRPC framework. Individual modules are deployed as Docker [37] containers and communicate solely with the central API, which orchestrates query generation and handles query generation requests.

1. A query request is received by the API in the form of inclusion and exclusion criteria as free-text strings.

2. The input texts are tokenized and named entity recognition is performed to determine spans of text representing conditions, procedures, and so on.

3. Relation extraction is performed to determine relations between the entities, such as *Caused-By* or *Numeric-Filter*.

4. The input texts are transformed by replacing spans of "raw" text with logical form names. For example, "Diagnosed with diabetes" would become "Diagnosed with cond("diabetes")." The resulting input texts are in turn transformed into an output logical representation using a Sequence to Sequence (Seq2Seq) architecture, in the form of a string.

5. A logical form interpreter module implemented as a recursive descent parser [38] reads the logical form string input and instantiates it as an abstract syntax tree (AST) of nested in-memory logical form objects.

6. "Named" logical form objects (i.e., specified with quoted text, such as "lab("hemoglobin A1c")") are normalized into one or more corresponding UMLS concepts.

7. Working recursively inside-to-outside the AST structure, each logical form object calls a *Reason()* method which executes various rules depending on context.

8. Each reasoning rule is performed as one or more pre-defined SPARQL queries to the knowledge base (KB), concept by concept.

9. The final normalized, reasoned, logical form AST is thus a nested structure of UMLS concepts. Each AST criterion is mapped to zero or more corresponding entries in the semantic metadata mapping (SMM), which in turn lists meanings, roles, and relations of a database schema in the form of UMLS concepts.

10. The final mapped AST object is transformed into a series of database queries, one per line of eligibility criteria text. The output SQL query can either be executed directly on a database or returned to the API caller.

Figure 2 illustrates an example of this process. In the following subsections we examine these steps in detail.

Named entity recognition and relation extraction

Named entity recognition (NER) refers to the segmentation and identification of tokens within an input sentence as "entities", such as conditions or procedures. We used the Leaf Clinical Trials (LCT) corpus [39] to train two BERT-based [40] NER extractors, one each for LCT general- and fine-grained-entities (see [39] for more information on LCT entity types). Next, we perform relation extraction between named entity pairs similarly using a BERT-based model also trained on the LCT corpus.

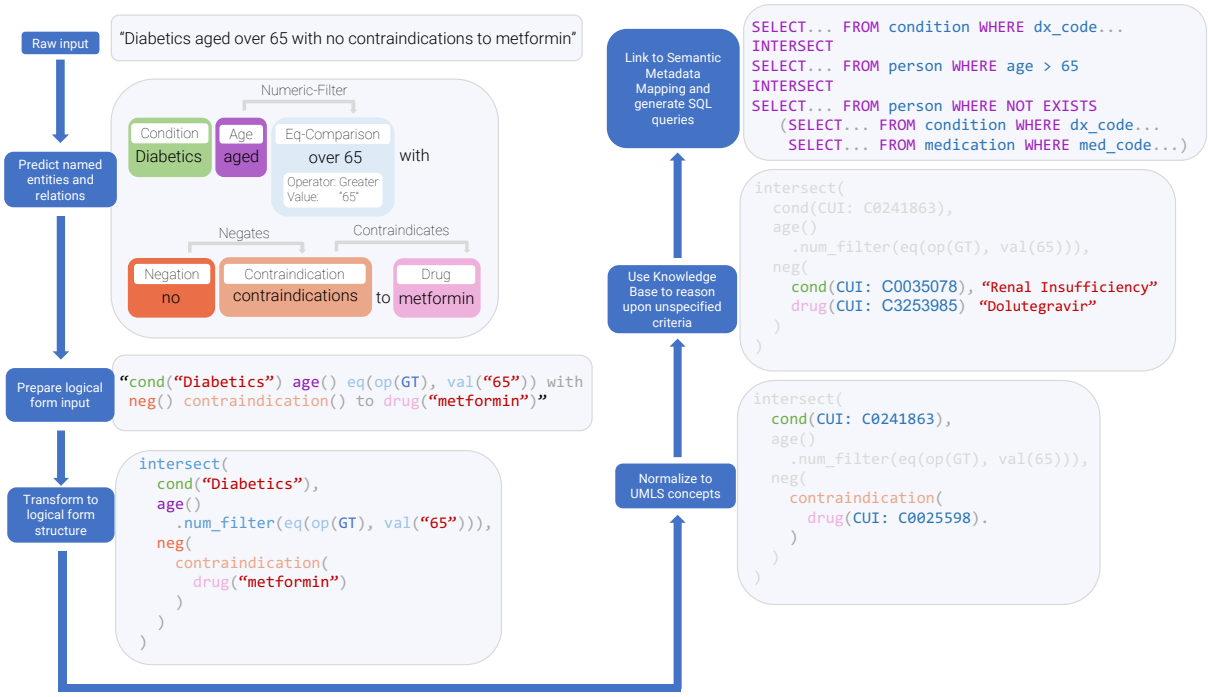

Figure 2: LeafAI query generation processes

Logical form transformation

One of the core challenges of generating queries for eligibility criteria is the problem of logical representation. Generating queries directly based on named entities and relations alone, while perhaps practical, may perform poorly in cases of nested or particulary complex logic. An alternative to this approach is to use a so-called intermediate representation (IR), which transforms the original natural language input by removing "noise" unnecessary to a given task and which more logically represents underlying semantics (see Herzig *et al* [41] for an examination of IR-based SQL generation approaches). Similar to earlier discussed work using Description Logics, Roberts and Demner-Fushman [42] proposed a representation of questions on EHR databases using a comparatively compact but flexible format using first order logic expressions, for example, representing "Is she wheezing this morning?" as

$$\delta(\lambda x.has\_problem(x, C0043144, status) \wedge time\_within(x, \text{"this morning"}))$$

This style of representation is powerfully generalizable, but also difficult to translate directly into SQL

statements as multiple predicates (e.g., *has_problem* and *time_within*) may actually correspond to one or many SQL statements, depending on context, complicating direct transformation into queries.

We thus chose a similar intermediate representation (hereafter simply "logical forms") as proposed by Roberts and Demner-Fushman but more closely resembling a nested functional structure in programming languages such as Python or JavaScript and more amenable to SQL generation. A criterion such as "Diabetic women and men over age 65" would be represented by our logical forms as

$$
\begin{aligned}
&intersect(\\
&\quad cond(\text{"}Diabetic\text{"}),\\
&\quad union(female(), male()),\\
&\quad age().num\_filter(eq(op(GT), val(\text{"}65\text{"})))\\
&)
\end{aligned}
$$

A description of our logical forms annotation schema, corpus, called the Leaf Logical Forms (LLF) corpus, annotation process, and performance metrics can be found in Appendix A.

After named entity recognition and relation extraction are performed, we leverage T5 [43], a state-of-the-art Seq2Seq architecture we fine-tuned for predicting logical forms on the LLF corpus. As inputs to the Seq2Seq model we use the original eligibility criteria with named entity spans replaced by logical form representations, as we found this to significantly improve performance compared to training with raw inputs without named entities. Thus the above example would be transformed to the input string

"cond("Diabetic") female() and male() over age() eq(op(GT), val("65"))"

The returned logical form string is then instantiated into an abstract syntax tree (AST) of nested in-memory logical form objects using a recursive descent parser [38] within our API.

Concept normalization

Normalization refers to the mapping of free-text string values (e.g., "diabetes mellitus") to coded representations (e.g., UMLS, ICD-10, SNOMED, or LOINC). We normalize "named" logical forms to UMLS concepts using MetaMapLite [44, 45]. We consider a logical form "named" if it contains a free-text value

surrounding by quotes. For example, *cond()* is unnamed and refers to any condition or disease, while *cond("hypertension")* is named as it refers to a specific condition.

Normalization using MetaMapLite can often result in high recall but low precision, as MetaMapLite has no NER component and tends to return UMLS concepts which match a given phrase syntactically but refer to abstract concepts not of interest (e.g., a search for "BMI" may return "body mass index" (C1305855), but also organic chemical "BMI 60" (C0910133), likely unhelpful for query generation). To improve normalization precision, we employ two strategies. First, we compare term frequency-inverse document frequency (tf-idf) on MetaMapLite predictions, dropping UMLS concepts whose matched spans have a tf-idf score lower than that of unmatched spans in a given named entity. For example, for the string "covid-19 infection", MetaMapLite predicts both "COVID-19" (C5203670) as well as several concepts related to general infections. We pre-computed term-frequency across the entirety of the UMLS. Using our tf-idf strategy removes the erroneous infection concepts. Next, our NER component also us to further improve precision by filtering the predicted UMLS concepts to only those of specific semantic types. For example, we limit condition concepts to only those which include semantic types of signs or symptoms, diseases or syndromes, and so on.

Laboratory values present a particular challenge, as LeafAI expects predicted lab concepts to have directly associated LOINC codes, while MetaMapLite typically normalizes lab test strings to UMLS concepts of semantic type "laboratory test or finding", but which do not have direct mappings to LOINC codes. For example, a search for "platelet count" with MetaMapLite returns the concept "Platelet Count Measurement" (C0032181), but not the needed concept of "Platelet # Bld Auto" (C0362994). Thus similar to Lee and Uzuner with medications [46], we trained a BERT model specifically for normalization of lab tests.

Reasoning using an integrated knowledge base

For reasoning and derivation of ICD-10, LOINC, and other codes for UMLS concepts, we designed a knowledge base (KB) accessible via SPARQL queries and stored as RDF triples. The core of our KB is the UMLS, derived using a variation of techniques created for ontologies in BioPortal [47]. To further augment the UMLS, we mapped and integrated the Disease Ontology [48], Symptom Ontology [49], COVID-19 Ontology [50], Potential Drug-Drug Interactions [51], LOINC2HPO [52], and the Disease-Symptom Knowledge Base [53]. We then developed SPARQL queries parameterized by UMLS concepts for various

scenarios which leveraged our KB, such as contraindications to treatments, symptoms of diseases, and so on. Using LOINC2HPO mappings further allows us to infer phenotypes by lab test results rather than only ICD-10 or SNOMED codes, for example.

Our KB, nested logical forms, and inside-to-outside normalization methods enable "multi-hop" reasoning on eligibility criteria over several steps. For example, given the non-specific criterion "Contraindications to drugs for conditions which affect respiratory function", our system successfully reasons that (among other results),

1. **Asthma** causes changes to **respiratory function**
2. **Methylprednisolone** can be used to treat **asthma**
3. **Mycosis** (fungal infection) is a contraindication to **methylprednisolone**

These features allow LeafAI to reason upon fairly complex non-specific criteria.

Query generation using semantic metadata mapping

To enable data model-agnostic query generation, we leveraged a subset of codes within the UMLS in what we define as a semantic metadata mapping, or SMM. An includes a listing of available databases, tables, columns, and so on within a given database schema. Critically, these database artifacts are "tagged" using UMLS concepts. An example of this can be seen in Figure 3, which shows strategies by which a given criterion can be used to generate schema-specific queries by leveraging different SMMs. In cases where the LeafAI query engine finds more than one means of querying a concept (e.g., two SQL tables for diagnosis codes), the queries are combined in a UNION statement.

Evaluation

An NLP-based system for finding patients based on eligibility criteria should be reasonably expected to find many or most patients enrolled in a real clinical trial - with the assumption that patients enrolled in those trials correctly met the necessary criteria as determined by study investigators. While there are caveats to this approach (for example, certain structured diagnosis codes may be missing for some patients, etc.), we aimed to establish a new baseline by which tools such as ours are evaluated in their ability to handle real-world eligibility criteria and clinical data.

Figure 3: The LeafAI query engine's SQL query generation process using two hypothetical database schema to generate queries for platelet counts (shown in logical form after normalization). This example illustrates the flexibility of LeafAI's semantic metadata mapping system (represented here in JSON format) in adapting to virtually any data model. On the left, "Tall Table Structure", platelet counts must be filtered from within a general purpose "labs" table. The LeafAI KB recognizes that labs may be stored as LOINC codes, and the corresponding SMM indicates that records in this table can be filtered to LOINC values. On the right, "Pivoted Table Structure", platelet counts are stored as a specific column in a "complete_blood_counts" table, and thus can be directly queried without further filtering. Additional metadata, columns, tables, types and so on needed in SMMs are omitted for brevity.

We compared LeafAI's results to that of a human database programmer experienced in the use of clinical databases and data extraction. We considered evaluating Criteria2Query as well, but ultimately determine it inappropriate for our analysis, as we aimed to review results found longitudinally (i.e., line by line of criteria), a function which Criteria2Query does not do. Our evaluation was performed as follows:

1. We extracted metadata on 165 clinical trials from our EHR between January 2010 and December 2021 where at least 10 patients were indicated as enrolled and not withdrawn and the total number of raw lines within the eligibility criteria (besides the phrases "Inclusion Criteria" and "Exclusion Criteria") were less than or equal to 30.

2. By manual review, we excluded 41 trials with multiple sub-groups, as it would not be possible to know

which eligibility criteria applied to which sub-group of enrolled patients.

3. Using the "condition" field for each trial within metadata from `https://clinicaltrials.gov`, we filtered and grouped the remaining 124 trials into only those related to predetermined groups: Cardiology, COVID-19, Crohn's Disease, Multiple Sclerosis, Diabetes Mellitus, Hepatitis C, and Oncology.

4. We randomly chose 1 trial from each group, with the exception of Oncology, where we chose 2 trials.

5. Both the LeafAI query engine and human programmer created queries to find patients for each eligibility criteria, which we executed on an OMOP database derived from our EHR of our institution's entire research-eligible patient population.

6. In order to ensure results returned would be limited to only data available during the time of each trial, we replaced references to the SQL function for generating a current timestamp (*GETDATE()*) with that of each trial's end date, and similarly replaced OMOP table references with SQL views filtering data to only that existing prior to the end of a trial.

7. The human programmer was instructed to (1) ignore criteria which cannot be computed, (2) make a best-effort to reason upon non-specific criteria (e.g., symptoms for a condition), (3) not check whether patients found by a human query enrolled within a trial, and (4) skip criteria which cause an overall query to find no eligible patients.

RESULTS

Results of the query generation experiment are shown in Table 1. Overall, LeafAI matched 212 of 427 (49%) total enrolled patients across 8 clinical trials compared to 180 (42%) found by queries of the human programmer. The mean per-trial percent of patients matched was 43.5% for LeafAI and 27.2% for the human programmer. LeafAI had a greater number of patients deemed eligible across trials, for a total of 27,225 eligible compared to 14,587 found by the human programmer.

Table 2 shows the number of criteria which were skipped by LeafAI. Of the 103 total criteria across all 8 studies, LeafAI executed queries for 61 (59.3%) and skipped 5 (4.8%) as it found no patients and 42 (40.7%) because no computable concepts were found.

				LeafAI		Human		
Condition	**ID**	**# Crit.**	**Enrolled**	**Matched**	**Eligible**	**Matched**	**Eligible**	**Time (hrs)**
CL Lymphoma	NCT04852822	4	83	80 (96%)	3,252	77 (92%)	2,382	1
Hepatitis C	NCT02786537	8	42	33 (78%)	9,529	32 (76%)	9,372	4
Crohn's Disease	NCT03782376	9	16	0 (0%)	113	1 (6%)	9	2
Cardiac Arrest	NCT04217551	12	27	12 (44%)	4,792	0 (0%)	598	5
COVID-19	NCT04501952	13	41	0 (0%)	0	0 (0%)	98	2
Multiple Sclerosis	NCT03621761	14	196	77 (39%)	4,891	69 (35%)	1,016	3
Type 1 Diabetes	NCT03335371	18	11	0 (0%)	1,006	1 (9%)	1,104	4
Ovarian Cancer	NCT03029611	25	11	10 (91%)	1,667	0 (0%)	8	5
Mean				43.5%		27.2%		
Total		103	427	212 (49%)	27,225	180 (42%)	14,587	26

Table 1: Statistics for each clinical trial evaluated by the LeafAI query engine and human programmer. The number of enrolled and matched patients were determined by cross-matching enrollments listed within our EHR. The *Time* column indicates the number of hours the human programmer spent developing queries for each trial.

Condition	**# Criteria**	**# No Patients**	**# Not Computable**	**# Fully Executed**
Cl Lymphoma	4	0 (0%)	0 (0%)	4 (100%)
Hepatitis C	8	0 (0%)	4 (50%)	4 (50%)
Crohn's Disease	9	0 (0%)	4 (44.4%)	5 (55.5%)
Cardiac Arrest	12	0 (0%)	8 (66.6%)	4 (33.3%)
COVID-19	13	0 (0%)	6 (46.1%)	7 (53.8%)
Multiple Sclerosis	14	1 (7.1%)	3 (21.4%)	10 (71.4%)
Type 1 Diabetes	18	2 (11.1%)	8 (44.4)	8 (44.4%)
Ovarian Cancer	25	2 (8%)	9 (36%)	14 (56%)
Total	103	5 (4.8%)	42 (40.7%)	61 (59.3%)

Table 2: The LeafAI query engine's handling of eligibility criteria for each trial. The column *No Patients* indicates the count of criteria which would, if executed, cause no patients to be eligible. The column *Not Computable* indicates the count of criteria which LeafAI could not generate a query for, for various reasons. Both of these types of criteria were ignored by the system.

Figure 4 shows longitudinal results from four trials with commentary indicating differences in query strategy by LeafAI and the human programmer.

Figure 4: Longitudinal results listing results of patients found at each step in the query process for four representative trials. The blue line indicates % recall for LeafAI and orange that of the human programmer. The X axis represents the line number within the free-text eligibility criteria. Dots indicate that a query was executed for a given line. On the right, boxes represent the text of a given eligibility criteria, with comments below discussing strategies of LeafAI and the human programmer and findings related to available data.

DISCUSSION

Our results demonstrate that under certain circumstances, LeafAI is capable of rivaling the ability of a human programmer in matching patients eligible to clinical trials. Indeed, in numerous cases we found LeafAI and the human programmer executing similar queries, such as for Hepatitis C (NCT04852822), Chronic Lymphocytic Leukemia (NCT04852822), Multiple Sclerosis (NCT03621761), and Diabetes Mellitus (NCT03029611), where both ultimately matched a similar number of patients.

One notable pattern we found is that LeafAI consistently finds a higher number of potentially eligible patients. As we have not done manual chart review of the patients found, it is difficult to determine the proportion of true positive versus false negative eligible patients compared to eligible patients found by the human programmer. We hypothesize that in many cases, LeafAI's Knowledge Base played a key role both finding additional eligible patients, but also sometimes in unnecessarily excluding otherwise eligible patients. For example, in the Multiple Sclerosis (MS) trial, LeafAI searched for 11 different SNOMED codes related to MS (including MS of the spinal cord, MS of the brain stem, acute relapsing MS, etc.), while the human programmer searched for only one, and ultimately LeafAI found nearly 5 times the number of potentially eligible patients (4,891 versus 1,016). We hypothesize that the the human programmer likely had a lower rate of false positives (and thus higher precision), though we leave an analysis of that to future work. On the other hand, in the same trial, as can be seen in 4, given the exclusion criteria: "Current shift work sleep disorder, or narcolepsy diagnosed with polysomnography and multiple sleep latency", LeafAI's KB included diagnosis codes for drowsiness, snoring, and so on, as within the UMLS they appear as child concepts of sleep disorder (C0851578). The exclusion of these patients likely resulted in an approximately 40% drop in recall at that stage compared to the human programmer, though ultimately both achieved similar recall (39% versus 35%).

Beyond performance as measured by recall, it is notable that in total, the human programmer spent approximately 26 hours in crafting queries for the 8 trials while LeafAI took to only several minutes running on a single laptop. We believe that the time saved by used automated means such as LeafAI for cohort discovery may save health organizations significant time and resources.

Limitations

This project has a number of limitations. First, while the 8 clinical trials we evaluated were randomly chosen, we specifically restricted the categories of diseases to select trials among, and thus our results should not be presumed to generalize to other kinds of clinical trial. Next, we evaluated our queries using an OMOP-based extract of data from our EHR. Our OMOP database does not contain the full breadth of data within our EHR. Had our experiments instead been conducted using data directly from our enterprise data warehouse (populated by our EHR), it is very possible the human programmer would have achieved greater results than LeafAI due to knowledge and experience using granular source data. For example, in the Cardiac Arrest clinical trial, the human programmer noted that data for use of cooling blankets is available in our EHR, but not in OMOP. LeafAI would likely not have been able to easily utilize that data, while the human perhaps would have been.

Future work

We are actively developing a web-based user interface for LeafAI, shown in Figure 5. In future work, we will deploy a prototype of the tool and evaluate user feedback and system performance. The LeafAI web application will provide rapid feedback to users explaining its search strategies, and allow users to override system-reasoned concepts and edit or add their own. Additionally, we intend to explore the adaptation of our logical form-based query generation methods to general-purpose question answering.

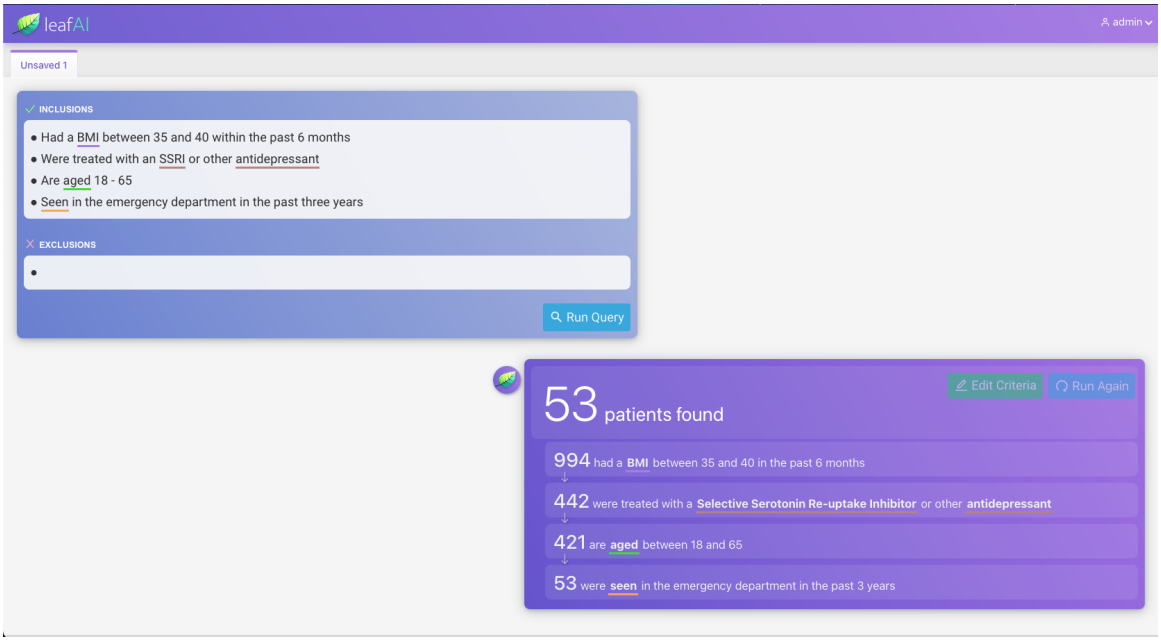

Figure 5: Example screenshot of the LeafAI web application, which is currently in development.

CONCLUSION

The growing volume and breadth of clinical data available in EHRs offers great potential sources of information for clinical trial recruitment, but also challenges in sorting, filtering, and making sense of. Automated applications to do so, such as LeafAI, represent a promising direction in expanding potential pools of trial candidates in a fast, scalable, and cost-saving means.

This study introduced LeafAI, a NLP-based system leveraging deep learning and an integrated Knowledge Base which can automatically generate queries for cohort discovery on virtually any clinical data model. Using an OMOP database representing the entire patient population of our institution, we demonstrated that LeafAI is capable of rivaling the performance of a human programmer in identifying eligible patients. We look forward in future work to deploying LeafAI for researchers and collaborating with the research community to expand and improve the tool.

ACKNOWLEDGEMENTS

This study was supported in part by the National Library of Medicine under Award Number R15LM013209 and by the National Center for Advancing Translational Sciences of National Institutes of Health under Award Number UL1TR002319. Experiments were run on computational resources generously provided by the UW Department of Radiology. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

AUTHOR CONTRIBUTIONS STATEMENT

ND is the developer of LeafAI and wrote the majority of the manuscript. BH and WZ annotated the LLF dataset and contributed to the annotation schema. KL served as the human database programmer, and NK and RH advised on study design. OU and MY advised on strategies for query generation and NLP architectures. All authors contributed to the interpretation of the data, manuscript revisions, and intellectual value to the manuscript.

COMPETING INTERESTS

The authors declare no competing interests.

REFERENCES

[1] Gul RB, Ali PA. Clinical trials: the challenge of recruitment and retention of participants. Journal of clinical nursing. 2010;19(1-2):227-33.

[2] Adams M, Caffrey L, McKevitt C. Barriers and opportunities for enhancing patient recruitment and retention in clinical research: findings from an interview study in an NHS academic health science centre. Health research policy and systems. 2015;13(1):1-9.

[3] Wang AY, Lancaster WJ, Wyatt MC, Rasmussen LV, Fort DG, Cimino JJ. Classifying clinical trial eligibility criteria to facilitate phased cohort identification using clinical data repositories. In: AMIA Annual Symposium Proceedings. vol. 2017. American Medical Informatics Association; 2017. p. 1754.

[4] Cowie MR, Blomster JI, Curtis LH, Duclaux S, Ford I, Fritz F, et al. Electronic health records to facilitate clinical research. Clinical Research in Cardiology. 2017;106(1):1-9.

[5] Lee CH, Yoon HJ. Medical big data: promise and challenges. Kidney research and clinical practice. 2017;36(1):3.

[6] Dobbins NJ, Spital CH, Black RA, Morrison JM, de Veer B, Zampino E, et al. Leaf: an open-source, model-agnostic, data-driven web application for cohort discovery and translational biomedical research. Journal of the American Medical Informatics Association. 2019 10;27(1):109-18.

[7] Murphy SN, Weber G, Mendis M, Gainer V, Chueh HC, Churchill S, et al. Serving the enterprise and beyond with informatics for integrating biology and the bedside (i2b2). Journal of the American Medical Informatics Association. 2010;17(2):124-30.

[8] Johnson EK, Broder-Fingert S, Tanpowpong P, Bickel J, Lightdale JR, Nelson CP. Use of the i2b2 research query tool to conduct a matched case–control clinical research study: advantages, disadvantages and methodological considerations. BMC medical research methodology. 2014;14(1):1-6.

[9] Deshmukh VG, Meystre SM, Mitchell JA. Evaluating the informatics for integrating biology and the bedside system for clinical research. BMC medical research methodology. 2009;9(1):1-12.

[10] Yuan C, Ryan PB, Ta C, Guo Y, Li Z, Hardin J, et al. Criteria2Query: a natural language interface to clinical databases for cohort definition. Journal of the American Medical Informatics Association. 2019;26(4):294-305.

[11] Soni S, Roberts K. Patient cohort retrieval using transformer language models. In: AMIA annual symposium proceedings. vol. 2020. American Medical Informatics Association; 2020. p. 1150.

[12] Fang Y, Idnay B, Sun Y, Liu H, Chen Z, Marder K, et al. Combining human and machine intelligence for clinical trial eligibility querying. Journal of the American Medical Informatics Association. 2022.

[13] Zhang X, Xiao C, Glass LM, Sun J. DeepEnroll: patient-trial matching with deep embedding and entailment prediction. In: Proceedings of The Web Conference 2020; 2020. p. 1029-37.

[14] Chen L, Gu Y, Ji X, Lou C, Sun Z, Li H, et al. Clinical trial cohort selection based on multi-level rule-based natural language processing system. Journal of the American Medical Informatics Association. 2019;26(11):1218-26.

[15] Patrão DF, Oleynik M, Massicano F, Morassi Sasso A. Recruit-An Ontology Based Information Retrieval System for Clinical Trials Recruitment. In: MEDINFO 2015: eHealth-enabled Health. IOS Press; 2015. p. 534-8.

[16] Dhayne H, Kilany R, Haque R, Taher Y. EMR2vec: Bridging the gap between patient data and clinical trial. Computers & Industrial Engineering. 2021;156:107236.

[17] Liu R, Rizzo S, Whipple S, Pal N, Pineda AL, Lu M, et al. Evaluating eligibility criteria of oncology trials using real-world data and AI. Nature. 2021;592(7855):629-33.

[18] Xiong Y, Shi X, Chen S, Jiang D, Tang B, Wang X, et al. Cohort selection for clinical trials using hierarchical neural network. Journal of the American Medical Informatics Association. 2019;26(11):1203-8.

[19] Hripcsak G, Duke JD, Shah NH, Reich CG, Huser V, Schuemie MJ, et al. Observational Health Data Sciences and Informatics (OHDSI): opportunities for observational researchers. Studies in health technology and informatics. 2015;216:574.

[20] Bae S, Kim D, Kim J, Choi E. Question Answering for Complex Electronic Health Records Database using Unified Encoder-Decoder Architecture. In: Machine Learning for Health. PMLR; 2021. p. 13-25.

[21] Park J, Cho Y, Lee H, Choo J, Choi E. Knowledge graph-based question answering with electronic health records. In: Machine Learning for Healthcare Conference. PMLR; 2021. p. 36-53.

[22] Wang P, Shi T, Reddy CK. Text-to-SQL generation for question answering on electronic medical records. In: Proceedings of The Web Conference 2020; 2020. p. 350-61.

[23] Pan Y, Wang C, Hu B, Xiang Y, Wang X, Chen Q, et al. A BERT-Based Generation Model to Transform Medical Texts to SQL Queries for Electronic Medical Records: Model Development and Validation. JMIR Medical Informatics. 2021;9(12):e32698.

[24] Patel C, Cimino J, Dolby J, Fokoue A, Kalyanpur A, Kershenbaum A, et al. Matching patient records to clinical trials using ontologies. In: The Semantic Web. Springer; 2007. p. 816-29.

[25] Huang Z, Teije At, Harmelen Fv. SemanticCT: a semantically-enabled system for clinical trials. In: Process Support and Knowledge Representation in Health Care. Springer; 2013. p. 11-25.

[26] Baader F, Borgwardt S, Forkel W. Patient selection for clinical trials using temporalized ontology-mediated query answering. In: Companion Proceedings of the The Web Conference 2018; 2018. p. 1069-74.

[27] Johnson AE, Pollard TJ, Shen L, Lehman LwH, Feng M, Ghassemi M, et al. MIMIC-III, a freely accessible critical care database. Scientific data. 2016;3(1):1-9.

[28] Belenkaya R, Gurley MJ, Golozar A, Dymshyts D, Miller RT, Williams AE, et al. Extending the OMOP common data model and standardized vocabularies to support observational cancer research. JCO Clinical Cancer Informatics. 2021;5.

[29] Peng Y, Nassirian A, Ahmadi N, Sedlmayr M, Bathelt F. Towards the Representation of Genomic Data in HL7 FHIR and OMOP CDM. In: GMDS; 2021. p. 86-94.

[30] Zoch M, Gierschner C, Peng Y, Gruhl M, Leutner LA, Sedlmayr M, et al. Adaption of the OMOP CDM for Rare Diseases. In: MIE; 2021. p. 138-42.

[31] Warner JL, Dymshyts D, Reich CG, Gurley MJ, Hochheiser H, Moldwin ZH, et al. HemOnc: A new standard vocabulary for chemotherapy regimen representation in the OMOP common data model. Journal of biomedical informatics. 2019;96:103239.

[32] Zhou X, Murugesan S, Bhullar H, Liu Q, Cai B, Wentworth C, et al. An evaluation of the THIN database in the OMOP Common Data Model for active drug safety surveillance. Drug safety. 2013;36(2):119-34.

[33] Shin SJ, You SC, Park YR, Roh J, Kim JH, Haam S, et al. Genomic common data model for seamless interoperation of biomedical data in clinical practice: retrospective study. Journal of medical Internet research. 2019;21(3):e13249.

[34] Kwon E, Jeong CW, Kang D, Kim Y, Lee Y, Yoon KH, et al. Development of common data module extension for radiology data (R-CDM): A pilot study to predict outcome of liver cirrhosis with using portal phase abdominal computed tomography data. European Congress of Radiology-ECR 2019; 2019. .

[35] Ross J, Tu S, Carini S, Sim I. Analysis of eligibility criteria complexity in clinical trials. Summit on translational bioinformatics. 2010;2010:46.

[36] gRPC: A high performance, open source universal RPC framework;. Accessed: 2022-08-16. https://grpc.io//.

[37] Docker;. Accessed: 2022-08-16. https://www.docker.com.

[38] Johnstone A, Scott E. Generalised recursive descent parsing and follow-determinism. In: International Conference on Compiler Construction. Springer; 1998. p. 16-30.

[39] Dobbins NJ, Mullen T, Uzuner Ö, Yetisgen M. The Leaf Clinical Trials Corpus: a new resource for query generation from clinical trial eligibility criteria. Scientific Data. 2022;9(1):1-15.

[40] Devlin J, Chang MW, Lee K, Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:181004805. 2018.

[41] Herzig J, Shaw P, Chang MW, Guu K, Pasupat P, Zhang Y. Unlocking compositional generalization in pre-trained models using intermediate representations. arXiv preprint arXiv:210407478. 2021.

[42] Roberts K, Demner-Fushman D. Annotating logical forms for EHR questions. In: LREC... International Conference on Language Resources & Evaluation:[proceedings]. International Conference on Language Resources and Evaluation. vol. 2016. NIH Public Access; 2016. p. 3772.

[43] Raffel C, Shazeer N, Roberts A, Lee K, Narang S, Matena M, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. J Mach Learn Res. 2020;21(140):1-67.

[44] Aronson AR. Effective mapping of biomedical text to the UMLS Metathesaurus: the MetaMap program. In: Proceedings of the AMIA Symposium. American Medical Informatics Association; 2001. p. 17.

[45] Demner-Fushman D, Rogers WJ, Aronson AR. MetaMap Lite: an evaluation of a new Java implementation of MetaMap. Journal of the American Medical Informatics Association. 2017;24(4):841-4.

[46] Lee K, Uzuner Ö. Normalizing Adverse Events using Recurrent Neural Networks with Attention. AMIA Summits on Translational Science Proceedings. 2020;2020:345.

[47] Noy NF, Shah NH, Whetzel PL, Dai B, Dorf M, Griffith N, et al. BioPortal: ontologies and integrated data resources at the click of a mouse. Nucleic acids research. 2009;37(suppl_2):W170-3.

[48] Schriml LM, Arze C, Nadendla S, Chang YWW, Mazaitis M, Felix V, et al. Disease Ontology: a backbone for disease semantic integration. Nucleic acids research. 2012;40(D1):D940-6.

[49] Sayers EW, Barrett T, Benson DA, Bolton E, Bryant SH, Canese K, et al. Database resources of the national center for biotechnology information. Nucleic acids research. 2010;39(suppl_1):D38-51.

[50] Sargsyan A, Kodamullil AT, Baksi S, Darms J, Madan S, Gebel S, et al. The COVID-19 ontology. Bioinformatics. 2020;36(24):5703-5.

[51] Ayvaz S, Horn J, Hassanzadeh O, Zhu Q, Stan J, Tatonetti NP, et al. Toward a complete dataset of drug–drug interaction information from publicly available sources. Journal of biomedical informatics. 2015;55:206-17.

[52] Zhang XA, Yates A, Vasilevsky N, Gourdine J, Callahan TJ, Carmody LC, et al. Semantic integration of clinical laboratory tests from electronic health records for deep phenotyping and biomarker discovery. NPJ digital medicine. 2019;2(1):1-9.

[53] Wang X, Chused A, Elhadad N, Friedman C, Markatou M. Automated knowledge acquisition from clinical narrative reports. In: AMIA Annual Symposium Proceedings. vol. 2008. American Medical Informatics Association; 2008. p. 783.

[54] Einolghozati A, Pasupat P, Gupta S, Shah R, Mohit M, Lewis M, et al. Improving semantic parsing for task oriented dialog. arXiv preprint arXiv:190206000. 2019.

[55] Lin CY. Rouge: A package for automatic evaluation of summaries. In: Text summarization branches out; 2004. p. 74-81.

[56] Callison-Burch C, Osborne M, Koehn P. Re-evaluating the role of BLEU in machine translation research. In: 11th conference of the european chapter of the association for computational linguistics; 2006. p. 249-56.

APPENDIX

Leaf Logical Forms corpus

We developed annotation guidelines for the LLF corpus using a simplification of entities and relations from the preceding LCT corpus[39]. Generally speaking, LCT *entities* correspond to logical form *functions*, while LCT *relations* correspond to logical form *predicates*. For example, the LCT *Condition* entity has a corresponding *cond()* function, while the *Num-Filter* relation has a corresponding *.num_filter()* predicate. The LLF annotation guidelines can be found at `https://github.com/ndobb/clinical-trials-seq2seq-annotation/wiki`.

We also hypothesized that the performance of predicting logical forms could likely be improved by replacing "raw" tokens in each eligibility criteria with corresponding logical form names derived from named entities from the LCT corpus. For example, given the eligibility criterion:

"*Diabetics who smoke*",

we would replace the named entities for "Diabetics" and "smoke":

cond("Diabetics") who obs("smoke")

using *Condition* and *Observation* annotations in the LCT corpus. We call this substituted text an "augmented" eligibility criteria. The augmented criteria syntax reshapes named entities to more closely resemble expected logical form syntax and allows us to leverage the LCT corpus for logical form transformation.

Creation and annotation of the LLF corpus proceeded in the following steps:

1. We randomly chose 2,000 lines of eligibility criteria from the LCT corpus, limited to only criteria which included at least one named entity and which were not annotated as hypothetical criteria. 30%

of the 2,000 lines (600) were randomly chosen among lines with particularly complex entity and relation types, such as *If-Then*, *Before-After*, *Contraindication*, etc.

2. Each annotation file consists of the text "EXC" if exclusion or "INC" if inclusion (line 1), an original "raw" eligibility criteria (line 3), an augmented eligibility criteria (line 5), and an (initially blank) expected logical form equivalent to annotate (line 7). An example annotation is shown in Figure 6.

3. 3 informatics graduate students met weekly for 2 months to review annotations. Annotators were initially trained on 20 triple-annotated training annotations.

4. After training, each annotator was assigned a batch of 100 sentences (one per file) and tasked with writing a logical form version of each.

5. After each batch was completed, we executed a quality control script to parse each logical form annotation to ensure consistency. Any syntax errors were reported to and corrected by the annotators.

6. Annotators received additional batches of files to annotate until all 2,000 single-annotated annotations had been completed.

```
'INC'

'-  women age 20 - 34 years ;'

'-  female() age() eq(val("20"), op(BETWEEN), val("34"), temporal_unit(YEAR)) ;'

intersect(
    female(),
    age()
        .num_filter(
            eq(val("20"), op(BETWEEN), val("34"), temporal_unit(YEAR))
        )
)
```

Figure 6: A example LLF corpus annotation. The annotation file is saved in JavaScript (.js) format, which enables syntax highlighting and validation to assist annotators. Whether a given criterion was an inclusion or exclusion criteria is indicated at the top, followed by the original raw text, then augmented text. The final annotated logical forms are shown last.

The pair-wise mean inter-annotator agreement by BLEU score was 82.4%. After annotations were completed, we experimented with predicting logical forms by fine-tuning T5 [43] Seq2Seq models. The T5

architecture and pre-trained models are widely used for and achieve at or near state-of-the-art for many machine translation and semantic parsing tasks.

Following earlier work on task-oriented dialog semantic parsing structures in the domain of digital assistants, we also experimented using various alternative input-output syntax styles from our original logical forms:

1. **Shift-Reduce**. Einolghozatic *et al.* [54] used square brackets instead of parentheses and blank spaces instead of commas. We followed Rongali *et al's* suggestion to add a trailing repeat of function names to improve performance.

2. **Pointer**. Rongali *et al.* found that replacing input tokens with "@ptr_{index}", where *index* corresponds to a token's sequential position in the input text improved performance in their semantic parsing task. We modified this approach by omitting the characters "ptr" and using the sequential position of the quoted span as our index rather than individual token positions.

We used a randomly sorted a 70/20/10 train/test/validation split of the LFF corpus to fine-tune the pre-trained T5$_{base}$ model using combinations of these syntax styles. We call our gold standard annotated logical form syntax "Standard" style. Example inputs, outputs, and training results are shown in Table 3.

We found that our Standard logical forms achieved the highest performance using both BLEU [55] and ROUGE-L [56] scores, two commonly used metrics in measuring Seq2Seq performance. Replacing raw tokens with function names corresponding to named entities also significantly improved performance (+14.7%, comparing raw text to Standard input styles), demonstrating that leveraging the LCT corpus to generate augmented text achieved relatively high performance (> 93% BLEU score) for this task. As it was the highest-performing syntax style and also the most straightforward to parse, we chose to use the Standard logical form style as our IR for LeafAI.

Syntax Style	Example Input	Example Logical Form	BLEU	ROUGE-L
Raw-text→ Standard	Diabetics who smoke	$intersect($ $cond(" Diabetics"),$ $obs(" smoke")$ $)$	78.7	79.1
Standard	cond("Diabetics") who obs("smoke")	$intersect($ $cond(" Diabetics"),$ $obs(" smoke")$ $)$	**93.5**	**92.3**
Standard+ Pointer	cond(@1) who obs(@2)	$intersect($ $cond(@1),$ $obs(@2)$ $)$	93.3	91.2
Shift-Reduce	[cond "Diabetics" cond] who [obs "smoke" obs]	$[intersect$ $[cond\ "Diabetics"\ cond]$ $[obs\ "smoke"\ obs]$ $intersect]$	89.8	91.7
Shift-Reduce+ Pointer	[cond @1 cond] who [obs @2 obs]	$[intersect$ $[cond\ @1\ cond]$ $[obs\ @2\ obs]$ $intersect]$	89.4	90.4

Table 3: Example inputs and logical form syntax styles with fine-tuning performance results using the $T5_{base}$ model.