Dictation Evaluation Reddit Parser

Garcia, Benjamin Lembke, Logan MacMillan, Kyle Smith, Christopher
Stelter, Andrew

September 21, 2018

Contents

1 Introduction to DERP

DERP, the **D**ictation **E**valuation **R**eddit **P**arser, is a DSL(Domain Specific Language) for finding news information from multiple web sources simultaneously. The system was designed as part of a group project for South Dakota School of Mines & Technology's Compilers course.

1.1 What is DERP: Human Queries For a Digital World

Time is a resource that people regularly find themselves lacking, and many would agree it would be convenient to have an easy way to consume online news while doing daily tasks – preparing and eating breakfast, taking a shower, getting dressed, or driving to work. We should be able to say to a phone, "What's going on in the world today?" and get a response containing relevant information that we care about; not just "I'm sorry, I don't understand, 'What's going on in the world today' ". If we want news on a specific topic, such as Tesla, getting that information should be as easy as saying, "What's in the news about Tesla?".

The formats used to present news are largely unchanged in recent years; sure, the internet provides a way for news to be shared, but we still have to hunt down the stories we want, whether we get them through a radio or television broadcast, an individual online or print article, or a newspaper. These formats are often quite rigid, giving us little control over what news we get, or so full of information that doesn't concern us that it can be difficult to know where to begin. One common solution used as a middle ground to these problems is aggregation sites such as Reddit, steemit, band, and voat to name a few. While these sites in this category go a long way toward solving the problems presented, they often fall into the same traps: Lacking information users want or presenting so much information that it's difficult to wade through it all.

1.2 DERP Origins

In the fall of 2018 we were tasked with making a DSL for a class in compiler theory. A number of applications discussed by our team follow:

- Describe Constraints to Generate something
  - Floor Plans
  - Workout Assistant
  - Computer Part Picker
- Perform a Task
  - Robot "AI" Proposal by Dr. Hinker
  - Simple Image Processing
- Configure a Task
  - News Reader
  - SQL Helper

We decided against a Constrain/Generate model because the ideas we came up with all required labeled data which we did not have access to, and we were not sure that we would be able to find or make sufficient data in a timely manner.

Creating a language to help perform a task seemed like a good idea at first, but we realized being reliant on some external item (a robot, in the case of Dr. Hinker's proposal) could hinder progress unnecessarily. We were also wary of working with a new, possibly unfinished, project and didn't know if we'd be walking into a well-documented, fully-defined product or a hack. Going down the route of something purely software, as we would have with our image processing concept, was a more viable option for this category of project, but we ultimately decided against it.

Lastly, we discussed configuration-style tasks. The two proposals from team members were for a News Reader and for a SQL Helper. The core concept of the News Reader was that a user should be able to

create custom queries that will obtain only the news that they are interested in, regardless of where the information was originally from. The idea behind the SQL Helper was to make SQL more accessible for the general public to use in everyday business situations; for example, a secretary with no prior SQL knowledge should be able to use it to get information from the company database to generate a report. We deemed the headache of building such a system while also making it robust enough to translate human queries into SQL to be too much work for the amount of time given for the project; so, we settled on the News Reader.

Every project needs a name; having selected our project, this was the most important task left to us. We settled on **D**ictation **E**valuation **R**eddit **P**arser because, if we have enough time, we'd like it to take spoken words as input to evaluate. The project will focus on interfacing with Reddit for a minimum viable product; however, the language is designed with other sources in mind.

1.3 Why DERP

DERP allows a developer to create an interface between a person and information feed sites such as Reddit, steemit, BAND, and Voat, as described in the What is DERP section. This interface allows for a programmer to develop an application that cleanly links an end-user to the feed site(s) of their choice, without the user even being aware that they are using DERP. While developers who want to access new news sites will still need to know how to get information from the source site of their choosing, DERP is designed to do the heavy lifting of finding and sorting relevant data, allowing less technically-minded users to build their own information filters from existing sources. It is a free solution which can provide a consistent starting point for developers so that they do not need to worry about the details of determining what should be searched and how results should be organized.

2 DERP Design Goals

2.1 Consolidated Information

The DERP project provides an intuitive way to multiplex groups of news sources into a personalized stream of data. Using DERP, a user can specify as many different sources as they want to (provided language plugins are available), and then all of those sources are accessed through the same set of language keywords, regardless of if the source is a subreddit, an arbitrary news website, or even just a file on the user's device. Furthermore, DERP facilitates naming groups of sources and queries to create criteria and new sources composed of other sources that can be used together, allowing users to further personalize what they get out of DERP.

Online news sources regularly expose the same categories of information - date, author, title, etc. DERP acts as an interface for all different article types, allowing users to work only with this high-level information about the articles they are finding. Hiding the details of what exactly is required to find an article that matches a specific criterion allows users to focus more on what they want to read, and less on how they obtain that information.

2.2 Easy-to-Use Natural English Interface

The second goal of DERP is to design a language that would not feel awkward if spoken. All of the DERP keywords and syntax follow simple but natural English speech patterns. This allows for easy adaptation of DERP into speech-recognition tools such as Google Assistant and Amazon Alexa. Using the natural form of the language, users with one of these devices could speak their program to the interpreter and receive results immediately.

In a similar vein, DERP provides output that also feels natural. DERP has a set of phrases available to it for reporting errors or results from user queries. The main output from DERP, articles the user has requested, are outputted as full text so that, should devices reading them use a text-to-speech system, they will sound as natural as possible.

2.3 Search More, Faster

By merging news sources into a personalized feed, DERP allows users to find the information they want with fewer operations. Rather than check each of their favorite subreddits, news sites, and RSS feeds, a user can simply load those sources into DERP and make a general query that applies all of them at once. DERP will do the hard work of finding things the user might be interested in, which means the user will be able to spend more of their time actually consuming the information provided and deciding on new topics to get information about.

Because it is an extensible language, DERP's users are limited only by the language plugins they use. While some keywords are understood specially by the language, such as those pertaining to dates and times, any language plugin can provide additional fields and keywords, making the language flexible and powerful while keeping complexity out of it's core.

3 The DERP Execution Model

The DERP query language provides users the ability to select the content they want while handling the nuances of arbitrary Internet data. In order to make this language as simple as possible in the face of these goals, the DERP execution model inherits a fair amount of complexity.

3.1 Architecture

3.1.1 Interactive

The execution of statements from the DERP query language will be done through a Read-Eval-Print-Loop (REPL). This REPL will evaluate language statements sequentially from its input stream, which can be any valid source of text. For the initial prototype, the DERP REPL will provide a text interface similar to the interactive environments of Python and Bash. If time permits, we will implement a second interface layer to accept speech as the input - Such an interpreter would be similar to the REPL, but we will refer to it as the Hear-Eval-Speak-Loop (HESL).

The DERP query language is a declarative language, so the main focus of a programmer will be the logic of the statements to perform queries, otherwise known as selections. This means that the task of organizing queries and their results is performed by the interpreter executing the REPL. The REPL will accept statements which define sources and search criteria, as well as statements which execute the resulting queries. The interpreter itself will verify that each statement is valid and conforms to the standards of the current mode before executing it, taking into account the set of modules loaded for news sources.

Upon execution of a statement, the interpreter may be expected to change the set of registered news sources, change the registered knowledge modules (see below), save or delete part of a search criterion, or execute a query.

When executing a query, the interpreter will retrieve only enough information to identify an article's defining information for the user. The defining information of articles will be stored in the form of tags; they will contain things like its title, date published, and location of origin. Statements following the execution of a query may perform summaries of the query results, start new queries, give details about each article, or otherwise perform some operation on the retrieved data.

3.1.2 Modes

The REPL will have three modes of evaluation; the mode used to interpret a given statement will be dependent upon the language keywords provided in the statement. These modes are named Main Mode (or Top-Level Mode), Selection Mode, and Criteria Mode.

Main Mode

The DERP interpreter will always start in the main mode. This mode allows users to change the registered modules, execute and delete saved queries (selections), and as switch to other modes. From this state, it is valid to switch the interpreter to either the selection mode or the criteria creation mode. In this mode, language keywords which end a program should indicate that the REPL is to exit.

Selection Builder Mode

The selection mode will be used to build full queries that can be executed through a later statement. Notably, selections must select some data from a loaded news source in order to produce a set of results. In this mode, language keywords which end a program should indicate that the REPL is to return to the top-level mode. Examples of using this mode are found in section 4.1.1 Composable .

Criteria Builder Mode

This mode allows users to create criteria that can be applied to sources as part of a query. While selections must reference a loaded news source, criteria do not have the same constraint. In this mode, language

keywords which end a program should indicate that the REPL is to return to the top-level mode. Details can be found in section 4.1.1 Composable .

3.1.3 Modules

The interpreter will also allow for language modules to be loaded during run-time. These modules will add language keywords which allow users to use new kinds of news sources or query criteria. Modules are not loaded by default; users must choose to activate them within a session of the REPL. Examples of modules can be found in section 4.1.3 Extensible

News Source Module

Each news source will be a modular component of the interpreter that can be selected from. News source modules will be defined in a package which indicates how the interpreter will interface with the source, what the keywords are associated with it, what fields are contained on each post, and what sorting options are allowed for results from the source. For example, the Reddit news source module for the Python implementation of the DERP REPL uses the Reddit Python API and provides the "subreddit" and "posts" keywords. It also defines the name and data type of each searchable field for each post.

Knowledge Module

Knowledge modules provide more search-able fields for query results on demand. For example, a knowledge module may wrap a prepackaged sentiment analysis engine in order to provide the numeric field "sentiment" to query against.

3.2 Language Constraints

There are a few constraints on the interpreter implementation for the DERP-Query Language. These are mainly based around the keywords defined in section 4.7 Keywords .

The following must be defined for the main selection mode:

- create
- clear
- read
- exit

The following must be defined for the selection and criteria modes:

- add
- remove
- clear
- recall
- save as
- read
- stop

Also all modules loaded must must not define their own keywords that are the same as interpreter keywords.

4 The DERP Query Language

The DERP Query Language (DERP-QL) makes it easy to write complex queries for a variety of Internet sources. In DERP, queries are represented by *selections*. Selections find postings from a particular set of sources that match a corresponding set of criteria. The following is an example of a DERP-QL selection.

```
Add posts from subreddit "nba" or subreddit "basketball" on "Lakers"
Remove posts with under 1000 upvotes
Remove posts on "Lebron James"
```

This selection does exactly what it says. It finds posts from `reddit.com/r/nba` or `reddit.com/r/basketball` that don't involve Lebron James with 1000 upvotes or more.

4.1 Main Features

DERP's main features revolve around being composable, interactive, and extensible.

4.1.1 Composable

DERP-QL is composable on several levels. First of all, selections are composed of several *selection statements*. Selections always start with an `Add...From` statement. `Add...From` statements supply at least one source to pull postings from. Additionally, they may restrict the postings to a set of criteria. While selections always start with an `Add` statement, the selection can be further altered by any number of subsequent `Add` or `Remove` statements.

Once a selection has been built, it can be saved under a given name. Any saved selection can then be used as a source for another selection. For example, the above example could be split into multiple selections.

```
Create a new selection
    Add posts from subreddit "nba" or subreddit "basketball" on "Lakers"
    Save as "Lakers news"
Stop
Create a new selection
    Add posts from "Lakers news"
    Remove posts with under 1000 upvotes
    Remove posts on "Lebron James"
    Save as "Lakers news without Lebron"
Stop
```

Composing selections makes it easy to assign a name to a number of sources. For example, the first selection in the example immediately above could be broken into two selections.

```
Create a new selection
    Add posts from subreddit "nba" or subreddit "basketball"
    Save as "basketball subreddits"
Stop
Create a new selection
    Add posts from "basketball subreddits" on "Lakers"
    Save as "Lakers news"
Stop
Create a new selection
    Add posts from "Lakers news"
    Remove posts with under 1000 upvotes
    Remove posts on "Lebron James"
    Save as "Lakers news without Lebron"
Stop
```

While reusing selections is handy, it is even more useful to reuse the same filter criteria across multiple sources. While selections must pull from at least one source with an `Add...From` statement, *criteria* are

purely logical constructs. *Criteria* provide a means to compose search criteria without pulling posts from any individual source. The selection above can be rewritten in terms of criteria.

```
Create new criteria
    Add posts on "Lakers"
    Remove posts with under 1000 upvotes
    Remove posts on "Lebron James"
    Save as "Lakers without Lebron"
Stop
Create new selection
    Add posts from subbredit "nba" or subreddit "basketball" \
        matching "Lakers without Lebron"
    Save as "Lakers news without Lebron"
Stop
```

Note: "\" is not a part of DERP-QL. However, "\" here means the statement continues on the next line.

4.1.2 Interactive

DERP-QL is meant to be used in an interactive manner either through a REPL (Read-Eval-Print-Loop) or a HESL (Hear-Eval-Speak-Loop). DERP-QL specifies keywords and semantics that a DERP-QL REPL/HESL must support. Of particular note are the `Read` and `Recall` keywords. `Read` signals the interpreter to execute a selection. For example, `Read "Lakers news without Lebron"` pauses the interpretation of the containing DERP-QL program, and begins the interpreter dependent handling of the stored selection "Lakers news without Lebron". The `Read` command may also be used while creating a new selection to check the output of the current selection being built.

`Recall`, on the other hand, makes programming easier for voice users. `Recall` causes the interpreter to list out the statements which make up a given selection. This command is especially helpful in the selection and criteria creation environments when programming by voice.

In order to promote human interaction, several words are parsed as filler including lone articles `a, an, the` and the word `with`.

4.1.3 Extensible

Source modules may define their own grammars which are concatenated into the overall language. For example, the reddit source module defines the `subreddit` keyword. The subreddit keyword takes the next string and uses it to select a portion of reddit to use as a source.

4.2 Primitives

1. Datetime - field type used for criteria based on date fields
2. Number - field type used for criteria on numeric fields
3. Boolean - field type used for criteria on true/false fields
4. String - field type used for criteria on text fields

4.3 Higher Order Types

1. Selection - Composed of one or more selection statements. Contains at least one `Add...From` statement.
2. Criteria - Composed of one or more criteria statements (Boolean expressions). Represents a predicate for whittling down the results from a selection.

4.4 Memory Management

Memory management is automatic, and handled by the Python runtime. Selections, and criteria are persisted to files, and loaded into their representative objects at runtime. When selections or criteria are composed, they are composed by value rather than reference. This prevents dependency issues when criteria or selections are removed from the system.

4.5 Criteria Matching Operators

4.5.1 String Operators

- with the exact - String comparison equality operator. Includes results where the specified field is an exact match for the provided string.
- like - String comparison fuzzy equality operator. Includes results where the specified field is an approximate match for the provided string.
- in the - Sub-String comparison equality operator. Includes results where a sub-string of the specified field is an exact match for the provided string.

4.5.2 Datetime Operators

- date on - Date comparison equality operator. Includes results where the specified field is exactly the same date as the provided date.
- date after - Date comparison greater-than operator. Includes results where the specified field is strictly after the provided date.
- date before - Date comparison less-than operator. Includes results where the specified field is strictly before the provided date.

4.5.3 Boolean Operators

- which are | are - Boolean comparison equality operator. Includes results where the specified field contains the Boolean value of 'true'.
- which are not | are not - Boolean comparison non-equality operator. Includes results where the specified field contains the Boolean value of 'false'.

4.5.4 Numeric Operators

- with exactly | exactly - Number comparison equality operator. Includes results where the specified field contains the same numeric value as the provided numeric value.
- with over | over - Number comparison greater-than operator. Includes results where the specified field contains a value strictly greater than the provided numeric value.
- with under | under - Number comparison less-than operator. Includes results where the specified field contains a value strictly less than the provided numeric value.
- with roughly | roughly - Number comparison epsilon equality operator. Includes results where the specified field contains a value within an epsilon value greater or less than the provided numeric value.

4.5.5 Misc Operators

- matching - Criteria composition operator. The criteria corresponding to the provided name will be textually included into the current criteria or selection.
- on - Substring search on "topical" fields as defined by a source module.
- and | or - Combine the results from the statements on either side of the operator.

4.6 Selection Matching Operators

- from - Source selection operator/ Selection composition operator. Pulls posts from a registered source or a previously saved selection.
- All criteria matching operators.

4.7 Keywords

4.7.1 Main Mode

- exit - Exits the program, and may only be used in the mode selection mode.
- clear - Deletes a specified selection or criteria.
- load/ unload - Loads a news source module or a knowledge module.
- recall - Read out the selection statements or criteria statements which make up a specified selection or criterion.
- read - Execute the specified selection and present the results.
- create - Creates a new selection or criterion. Enters the respective builder mode.

4.7.2 Selection and Criteria Builder Mode

- stop - End mode keyword. Ends selection or criteria creation mode and clears the active state.
- clear - Reset mode state keyword. Without leaving the current creation mode, clears the active state.
- recall - Read out the selection statements or criteria statements which make up the object that is currently being built.
- read - (selection builder only) Execute the selection currently being built and present the results.
- save as - Store the current selection or criteria with a specified name. A saved selection or criteria may be used in the creation of other selections and criteria.
- add - Select posts which match a set of criteria. `From` is disallowed in in add statements in the criteria builder mode.
- remove - Remove posts which match a set of criteria.