# The Dictation Evaluation Reddit Parser Language Tutorial

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

December 5, 2018

# Contents

# Part I
# Language Tutorial

## 1 Introduction

This tutorial aims to quickly teach those skilled in the art how to use DERP to find, filter, and consume internet content. DERP is a Domain Specific Language (DSL) suited to information retrieval using natural English. For the purpose of this tutorial we will assume the existence of a 'Reddit' submodule which allows for retrieving data from www.reddit.com.

### 1.1 Target Systems

DERP is written in Python 3. As such, DERP will run wherever Python 3 is available. The DERP language is intended to be adapted to multiple platforms such as smart home devices, desktop computers, and mobile phones; however, this document specifically covers the desktop implementation of DERP. Because DERP is for querying online locations, most plugin modules will likely require a network connection.

## 2 Getting Started

Using DERP is easy. The simplest query one can write with DERP is one which fetches posts from the front page of reddit.

First, we must start the interpreter. After starting DERP, we are presented with an empty prompt.

```
$ python3 DERP.py
>>>
```

The first command entered into most DERP sessions is the `load` command. The load command tells DERP to add the language extensions needed to interact with a given resource. In the case of this tutorial, the first command will load the extensions needed to interact with Reddit.

```
$ python3 DERP.py
>>> load reddit
>>>
```

The next command tells DERP to start building a new query called a selection.

```
$ python3 DERP.py
>>> load reddit
>>> create a new selection
>>>
```

Selections contain plans for gathering lists of postings. In this case, we want to simply add the latest posts from the front page of Reddit.

```
$ python3 DERP.py
>>> load reddit
>>> create a new selection
>>> add posts from reddit
>>>
```

Selections can be executed immediately while they are being built. In order to fetch the posts specified by the current selection, we simply need to call `read`.

```
$ python3 DERP.py
>>> load reddit
>>> create a new selection
>>> add posts from reddit
>>> read
```

This will bring up an interface for browsing through the latest posts from the front page of Reddit. Once we have finished viewing the postings and close the reading interface, we are dropped back into the DERP interpreter.

```
$ python3 DERP.py
>>> load reddit
>>> create a new selection
>>> add posts from reddit
>>> read
>>>
```

From here, we could execute various commands. We could save the selection so we can read through the front page of Reddit later, we could add more posts from different subreddits, or we could filter out posts we don't want to see. For now, let's just end the selection. `stop` and `exit` can be used to end the current DERP operation, so entering `stop` right now will let DERP know we want to stop building a selection, and then entering `stop` or `exit` will stop the DERP interpreter entirely.

```
$ python3 DERP.py
>>> load reddit
>>> create a new selection
>>> add posts from reddit
>>> read
>>> stop
>>> exit
```

# 3 Basic Usage

While the above example is easy to understand, it doesn't accomplish much. After all, we could just open up an internet browser and head to www.reddit.com. In the next section, we will explore ways to join multiple sources of information, remove unwanted posts, and reuse and compose queries.

## 3.1 Fetching Posts from Various Sources

DERP handles fetching posts from various sources as provided by the loaded modules. The Reddit module provides a source for each subreddit. For example, the following snippet fetches the latest posts from the "worldnews" subreddit.

```
>>> create a new selection
>>> add posts from subreddit "worldnews"
```

We can combine multiple sources in a couple ways. The simplest way to do so is to use the `or` keyword. The `or` keyword may be repeatedly used to add posts from as many sources as needed.

```
>>> create a new selection
>>> add posts from subreddit "worldnews" or subreddit "news"
```

As DERP strives to fit natural English, the word `and` can be used in place of `or` in this case. The phrase "add posts from subreddit 'worldnews' and subreddit 'news'" is commonly understood as "add posts from subreddit 'worldnews', and add posts from subreddit 'news'", rather than "add posts that are in both subreddit 'worldnews' and subreddit 'news'".

Alternatively, we can use multiple add statements to construct an equivalent query.

```
>>> create a new selection
>>> add posts from subreddit "worldnews"
>>> add posts from subreddit "news"
```

## 3.2 Removing Posts from Various Sources

If we add posts from a given source, we can remove those posts later on using `remove`.

```
>>> create a new selection
>>> add posts from subreddit "worldnews"
>>> add posts from subreddit "news"
>>> remove posts from subreddit "worldnews"
```

# 4 Qualifiers

While straightforward, the simple `add` and `remove` statements determine which results we receive with broad strokes. We can obtain a finer degree of control using *`qualifiers`*. A *`qualifier`* allows us to match results

with certain traits. We can use qualifiers based on phrases, tags, dates, and more.

## 4.1 The `Which` Keyword

Some posts may be marked with special flags such as the "verified" tag. We can check for the presence of these flags using 'which'.

```
>>> create a new selection
>>> add posts from subreddit "finance" which are "verified"
```

This selection only returns trustworthy, verified posts. We can also negate this expression by adding not.

```
>>> create a new selection
>>> add posts from subreddit "finance" which are not "verified"
```

This could get us posts that we would miss using the preceding selection.

## 4.2 The `With` Keyword

The `with` keyword allows us to define a qualifier which matches results with certain fields such as strings, numbers, and dates.

## 4.3 Query by Contained Phrase

If we want to query for certain phrases in our posts, we can use `with ... in the` expressions. These expressions specify substring matches which are useful for filtering away unrelated topics.

```
>>> create a new selection
>>> add posts from subreddit "finance" with "stocks" in the body
```

This selection will retrieve posts in which the word 'stocks' occurs, allowing us to view posts that relate to the stock market rather than, say, savings accounts.

### 4.3.1 Query by Exact Phrase

If we know the exact title of an article we can find it in a given source using a `with the exact` expression.

```
>>> create a new selection
>>> add posts from subreddit "finance" with the exact title "CEO says part of
    ↪ October's market sell-off driven by programmatic trading"
```

More generally, if we know the exact value of a given string field, we can use a with the exact expression to match against it.

### 4.3.2 Query by Numeric Qualifier

If our source supports it, we can add qualifiers on the numeric properties of posts. For example, we can make sure to receive popular information by restricting results from Reddit based on their upvote counts.

Here is a basic example.

```
>>> create a new selection
>>> add posts from subreddit "finance" with exactly 500 upvotes
```

This makes sure that our results have 500 upvotes, but it will retrieve results with only 500 upvotes. Usually, we would want to define a range of acceptable values for a numeric field. To accomplish this, we can use the 'over' and 'under' qualifiers.

The `over` modifier requires that the returned posts have a numeric value `greater than` the value we asked for. In fact, the phrase greater than may be used instead of `over`.

```
>>> create a new selection
>>> add posts from subreddit "finance" with over 500 upvotes
```

Now we will only see posts that have over 500 upvotes. Similarly, the under or less than qualifier allows us to retrieve posts with a numeric value less than a given number.

Sometimes we want to find posts with a field around certain value. We can use the roughly modifier to find posts which have a numeric value close to a given number.

```
>>> create a new selection
>>> add posts from subreddit "finance" with roughly 500 comments
```

The desktop implementation of DERP finds values within +/- 10% of the given value.

#### 4.3.3 Query by Date Qualifier

We can set what time period we want posts from by making a query on the date the post was made. For example, we may want to limit our selection to recent topics, because out of date topics may not be useful.

```
>>> create a new selection
>>> add posts from subreddit "finance" with a post date after october 20 2018
```

Assuming the current date was October 21 2018 this would get us posts in the last day. We can also use `before` and `on` to limit ourselves to dates before and equal to the given date, respectively.

#### 4.3.4 Negative Qualifiers Using `Without`

We can also invert the meaning of a `with` statement by using `without`. For example, if we want finance posts without savings account information we could use the following selection.

```
>>> create a new selection
>>> add posts from subreddit "finance" without "savings accounts" in the body
```

`Without` is a good way to express negative logic in a qualifier, and it can allow for specifying selections without needing a remove statement.

### 4.4 The `On` Keyword

The `on` keyword makes it easy to perform a topic search. Another phrase that can be used in place of `on` is `about`. The actual functionality of `on` is implementation dependent. The original desktop implementation of `on` performs substring searches over the "topical" fields as defined by a given module. In this case, we can think about the the `on` as shorthand for multiple `with ... in the` qualifiers. Let's use the same example as from the `with ... in the` section above.

```
>>> create a new selection
>>> add posts from subreddit "finance" on "stocks"
```

We can negate `on` by placing the `not` modifier in front of it. The previous selection is equivalent to the following.

```
>>> create a new selection
>>> add posts from subreddit "finance"
>>> remove posts not on "stocks"
```

### 4.5 Combining Qualifiers Using `And` and `Or`

We may want to use multiple qualifiers when creating a selection. Consider the case in which we want to control what posts young children will see. We could prevent questionable content from showing up using the following query with `and`.

```
>>> create a new selection
>>> add posts from subreddit "all" which are "verified" and which are not "
    ↪ NSFW"
```

This statement will check for the presence of the verified flag and the absence of the NSFW tag. Alternatively, if we wanted at least one of our conditions to hold true we could replace the `and` with an `or`.

When we use `and` and `or` together we need to make sure to understand how our qualifiers will be interpreted by the runtime. The `and` keyword has higher precedence than the `or` keyword, which has implications for how we chain statements. For instance, the following selection likely has an error.

```
>>> create a new selection
>>> add posts from subreddit "all" which are "verified" or which are not "NSFW
    ↪ " and with a post date after October 14 2018
```

We can infer that we tried to find recent trustworthy posts, but, in reality, older posts may slip through if they are marked "verified". As `and` is higher precedence than `or`, the date check is anded with the "NSFW" check before getting ored with the "verified" check. To write this selection correctly we can split it up into multiple add and remove statements.

```
>>> create a new selection
>>> add posts from subreddit "all" which are "verified" or which are not "NSFW
    ↪ "
>>> remove posts without a post date after October 14 2018
```

An additional point to note is that when writing a selection, if `and` or `or` are not specified, the qualifiers are implied to have the `and` keyword between them.

```
>>> create a new selection
>>> add posts from subreddit "all" which are "verified" which are not "NSFW"
```

Is the same as

```
>>> create a new selection
>>> add posts from subreddit "all" which are "verified" and which are not "
    ↪ NSFW"
```

Creating Complex Queries Now that we know more about how to construct a variety of queries, we can construct more realistic examples. For instance, let's say we want the latest news within the last week, on a few of our favorite topics, while making sure to get only trustworthy sources. For the purpose of this example, the current date will be October 21 2018.

```
>>> create a new selection
>>> add posts from subreddit "worldnews" which are "verified" and which are
    ↪ not "NSFW" and with "stocks" in the body
>>> add posts from subreddit "worldnews" with "sports" in the body or with "
    ↪ lebron"
>>> remove posts from subreddit "worldnews" with "sports" in the body and with
    ↪  under 1000 upvotes
>>> add posts from subreddit "worldnews" with "Brexit" in the title or with "
    ↪ France" in the body
>>> remove posts with a post date before October 14 2018
```

This complex query will get us recent world news from trustworthy sources on stocks, that are popular on sports, and about the 'Brexit' or France from US sources. While this is a fairly diverse selection, it provides a glimpse into the potential for creating arbitrarily complex queries.

Sometimes it takes a while to work out what exactly you want in a given selection. In fact, the selection may become so long that we've forgot what we asked for in the first place! Thankfully in the desktop version of DERP, we can simply scroll back up through our terminal. In order to support alternative interfaces to DERP, each implementation must implement the `recall` statement. The `recall` statement simply lists out each of the add and remove statements entered so far.

## 5 Reusing Selections

So far we've made complex queries which allow us to quickly sort through Reddit postings. However, we needed to create a new selection each time we wanted to execute a query. Thankfully, DERP allows us to save our queries for later use. Consider building up the following query which we use to keep up on finance news:

```
$ python3 DERP.py
>>> load "reddit"
>>> create a new selection
>>> add posts from subreddit "finance" or subreddit "wallstreetbets"
>>> remove posts with under 2000 upvotes
>>> remove posts on "penny stocks"
>>>
```

We certainly don't want to re-enter the entire selection every time we want to fetch new data. In order to save this selection for later use, we use the `save` keyword. We can then exit out of the criteria creation mode and use the `read` keyword in the main mode to execute the query.

```
$ python3 DERP.py
>>> load "reddit"
>>> create a new selection
>>> add posts from subreddit "finance" or subreddit "wallstreetbets"
>>> remove posts with under 2000 upvotes
>>> remove posts on "penny stocks"
>>> save as "finance news"
>>> stop
>>> read "finance news"
```

If we'd like to remove a previously saved selection, we can use the `clear` keyword.

```
$ python3 DERP.py
>>> load "reddit"
>>> create a new selection
>>> ...
>>> save as "finance news"
>>> stop
>>> clear "finance news"
```

Finally, if we want to know which statements were used to create a selection we can use the `recall` keyword.

```
$ python3 DERP.py
>>> load "reddit"
>>> create a new selection
>>> ...
>>> save as "finance news"
>>> stop
>>> recall "finance news"
```

## 5.1 Composing with Criteria

It is common when querying for content that we might want to apply the same qualifiers to a variety of selections. We can save ourselves the effort of retyping the same qualifiers multiple times by defining criteria. Criteria, singular criterion, are saved groups of qualifiers that can be used in selections or when defining other criteria.

We can save criteria in much the same way as we save selections. Any saved criteria can be added to the current selection or criteria using the `matching` qualifier. Take the following criterion.

```
>>> create a new criteria
>>> add posts with "stocks" in the body and without "penny stocks"
>>> save as "finance news"
```

We could compose this into a new criteria named "recent finance news" as follows.

```
>>> create a new criteria
>>> add posts matching "finance news" and with a post date after October 14
    ↪ 2018
>>> save as "recent finance news"
```

And we can use this composed criteria in a selection as shown in the following example.

```
>>> create a new selection
>>> add posts from subreddit "worldnews" or subreddit "news" matching "recent
    ↪ finance news"
>>> read
```

By composing criteria with each other, we can create usefully complex queries without having to retype many lines of qualifiers. Even better, we can use a criterion with as many selections or other criteria as we like.

Note that when we use the `matching` keyword, it acts as a qualifier with the meaning of the criterion specified – it doesn't retain any reference to the source criterion. This means that if a criterion is destroyed or overwritten, any place it was used will not be changed.

As with selections, `clear` and `recall` work on criteria in main mode. This can be useful when considering how to combine a criterion with a new selection, or when you want to remove an outdated criterion.

Because they are very similar, it is important to keep in mind the differences between selections and criteria. Selections must contain at least one source to pull posts from, but criteria cannot specify any

sources - they contain only qualifiers. As a result, selections can be read, but criteria cannot. On the flip side, sources cannot be used with the `matching` qualifier - only criteria can.