Slackwire Code Documentation

1. Overview

The UIUC MCS Slack has a variety of course specific channels. These channels are used year over year, resulting in a buildup of relevant information for a given course. This information can be vital to students in future semesters. Campuswire is another great resource for information on the content of a course's current semester, however students must be invited to a course, and only have access to that semester's content. As a result, Campuswire does not have the rich historical history as Slack does. This disconnect between these two data sources inhibits current students from leveraging previous students' experiences in learning new content. In addition, students are inundated with a variety of posts, many of which are duplicated (e.g. a slack question that was also posted to Campuswire). Our project will aim to merge historical Slack and semesterly Campuswire whilst removing duplicated answers, providing an efficient and concise querying service for UIUC course questions.

2. Implementation

2.1 Code Overview

To retrieve data from Campuswire and Slack, their APIs will be utilized. That data is then de-duplicated before being saved onto the database for later use. With the appropriate datasets available, MetaPy is then used to rank documents based on given user queries. The query results are then evaluated using the Cranfield Evaluation Methodology to fine tune the ranker parameters.

The software is packaged using Python’s *setuptools* for easy installation. To facilitate usage, a CLI is available for users to provide commands for retrieving data, querying the datasets, and evaluating the query results.

All code within this project is fully MyPy (strict) and autopep8 compliant. Additionally, all methods have full type-hints to enable easy navigation for further development.

2.2 Slack API (slack.py)

The file *slack.py* is used to retrieve data from the Slack API. In order to access the API, the Slack channel ID and Slack authorization token needs to be provided. The Slack channel ID is hard coded in with the CS410 Slack channel, but the Slack authorization token needs to be provided as an environment variable. Information on how to set this authorization token is located in the Appendix of this document.

1. get_all_threads():
Uses the Slack API's conversation_history method, to obtain all threads from the Slack channel. It returns a list of dataclasses containing the thread_ts and messages. This method paginates all threads to prevent overloading the SlackAPI and caches results in a LRU-Cache to reduce duplicative calls to Slack.

2. get_thread_replies(thread_ts):
Uses the Slack API's conversation_replies method to retrieve a thread's replies. It returns a list of dataclasses containing the ts and reply messages.

2.3 Campuswire API (campuswire.py)

The file campuswire.py is used to retrieve data from the Campuswire API. In order to access the API, the channel and authorization token needs to be provided. The channel is hard coded in as the Fall 2021 CS410 channel, but the authorization token needs to be provided. Information on how to set this authorization token is located in the Appendix of this document.

1. get_all_threads():
Sends requests to the Campuswire API, paginating all results.

2. get_thread_comments(thread_id):
Uses the Slack API's *conversation_replies* method to retrieve a thread's replies. It returns a list of dataclasses containing the ts and reply messages.

2.4 CLI (cli.py)

The command line interface (CLI) has five commands of interest. Each of the commands are tied to a function within cli.py using the *click* package.

1. initialize_combined():
This function retrieves all data from Slack and Campuswire (reusing the methods 2 and 3 below), combines them, de-duplicates, then saves it to datasets/*combined/dataset.dat* for later usage.

2. initialize_slack():
Initializes the slack dataset via methods in dataset.py for querying usage. The documents are saved to *datasets/slack/dataset.dat* for later use.

3. initialize_campuswire():

Initializes the campuswire dataset via methods in dataset.py for querying usage. The documents are saved to *datasets/campuswire/dataset.dat* for later use.

4. search(only_slack: bool, only_campuswire: bool):
This function uses MetaPy to query the database of the user's choosing (slack only, campuswire only, or both) and prints the top 10 relevant documents in a reader-friendly format. The OkapiBM25 ranking algorithm is used, with its parameters tuned to best fit this use case. It uses the stopwords in *stopwords.txt*.

5. search_eval(only_slack: bool, only_campuswire: bool):
This function uses the Cranfield Evaluation Methodology to evaluate the ranker based on the queries and query relevancies provided in **queries.txt* and **qrels.txt*, respectively. We used this function to fine tune the parameters in our ranker.

2.5 Deduplicate (deduplicate.py)

Deduplication is an important part of our application to ensure users don't receive repetitions of similar documents (e.g. a user posting identical questions in Campuswire and Slack). To achieve this deduplication, we perform clustering on a VSM representation of the documents and select a single document from each cluster.

1. _get_alphabet(documents):
Constructs the entire alphabet of words used in all of our documents by decomposing all documents into a set of words.

2. _encode_documents(documents):
Encodes all documents into a VSM representation by decomposing each document into a bag of words, accounting for all words in the alphabet (retrieved by *_get_alphabet*) Each documents VSM is stored into a Pandas dataframe for easier processing later.

3. _get_best_cluster(documents):
Given a Pandas dataframe of encoded documents (VSM representation), this method iterates over a variety of number of clusters (K), and attempts to find the best K to optimize document deduplication. The method uses Scikit-learns Agglomerative Clustering, and computes the Silhouette Score and Calinski-Harabaz Score for each K-clustered label set. The method then uses these metrics to determine the best K to optimize deduplication, without clustering unique documents. The method then returns this set of unique clustered labels for each document.

4. deduplicate_docs(documents):
This method acts as our main entrypoint into document deduplication. Given a list of documents, it uses the above describe private methods to encode documents and determine the best clustering. The method then enumerates through the discovered labels, and selects the first document for each cluster.

2.6 Datasets (dataset.py)

Datasets.py is a helper class for managing our individual datasets. This class maintains the hardcoded paths to our individual dataset directories and configs.

1. _safe_open_w(path):
Safely opens the specified path for usage later, making any directories needed to ensure the write is performed successfully.

2. write_dataset(path):
Writes a dataset file to a specified path using the _safe_open_w from above.

3. retrieve_slack_dataset():
Using the methods in slack.py, all thread_ts are collected from the Slack API, then a nested for loop is used to combine the replies for each thread into their respective documents. One thread = one document.

4. retrieve_campuswire_dataset():
Using the methods in campuswire.py, all threads are collected from the Campuswire API, then a nested for loop is used to combine the replies for each thread into their respective documents. One thread = one document. Each of the posts' endorsement status and number of votes are also included in document text for the user to see, but they are included in *stopwords.txt* so they don't influence the query results.

5. get_dataset_paths(only_slack, only_campuswire):
This function is a helper function to easily access specific files of a dataset. Returns a tuple of paths leading to the specified datasets config and dataset files, respectively.

2.6 Setuptools (setup.py)

Python's *setuptools* package was used to facilitate easy installation. This allows the user to install the package easily by entering the following command:

```
pip install .
```

The re*quirements.txt* file is used to tell setuptools which packages to install. Entry points are also provided in *setup.py* to link CLI commands to their respective functions.

3. Usage

Because the program is packaged using *setuptools*, the program and its dependencies are installed by entering the following command from the repo root. All requirements to execute are located within the package. External data files may be generated if requested.

```
pip install .
```

Environment variables also need to be set for the Slack and Campuswire authorization tokens (refer back to Section 2.2 and 2.3 for more information).

After installation, the following commands can be used in the terminal:

1. To retrieve data from both CampusWire and Slack and store it locally:

```
slackwire initialize-combined
```

2. To retrieve data from Slack only and store it locally:

```
slackwire initialize-slack
```

3. To retrieve data from CampusWire only and store it locally:

```
slackwire initialize-campuswire
```

4. To query for relevant threads in both databases, the Slack database only, or the CampusWire database only:

```
slackwire search
slackwire search --only-slack
slackwire search --only-campuswire
```

5. To evaluate queries based on the queries.txt and qrels.txt files:

```
slackwire search-eval
slackwire search-eval --only-slack
slackwire search-eval --only-campuswire
```

4. Team Contributions

Hung Nguyen:

1) Creation of dataset from Slack data
2) Creation of dataset from Campuswire data
3) Ranking relevant documents given a query
4) Evaluation of rankings
5) Provided relevancy scores for documents with various queries

Blake Jones:

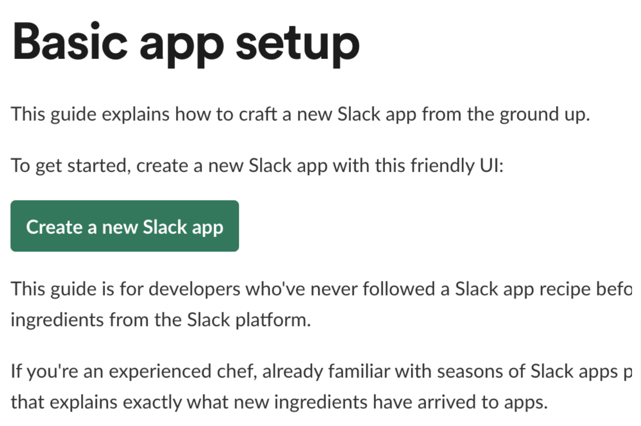
1) Retrieval of data from Slack API
2) Retrieval of data from Campuswire API
3) Creation of utilities (CLI and datasets management)
4) Deduplication of documents
5) Provided relevancy scores for documents with various queries

Video Documentation:
https://youtu.be/XS7kyNheJMg

5. Appendix

Slack Auth Token Creation

To setup Slack API access, you will need to create an app integration. Go to this link: https://api.slack.com/authentication/basics and select 'Create a new slack app', then 'create a new app'.



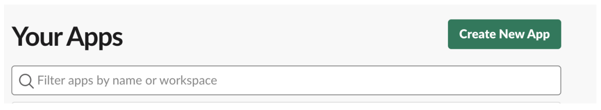
Basic app setup

This guide explains how to craft a new Slack app from the ground up.

To get started, create a new Slack app with this friendly UI:

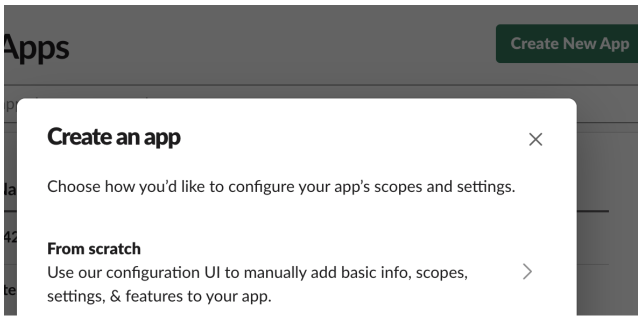
This guide is for developers who've never followed a Slack app recipe befo
ingredients from the Slack platform.

If you're an experienced chef, already familiar with seasons of Slack apps p
that explains exactly what new ingredients have arrived to apps.



Your Apps

A menu will pop up, select 'From scratch'. Then select the UIUC MCS workspace for 'Pick a workspace to develop your app in:'.



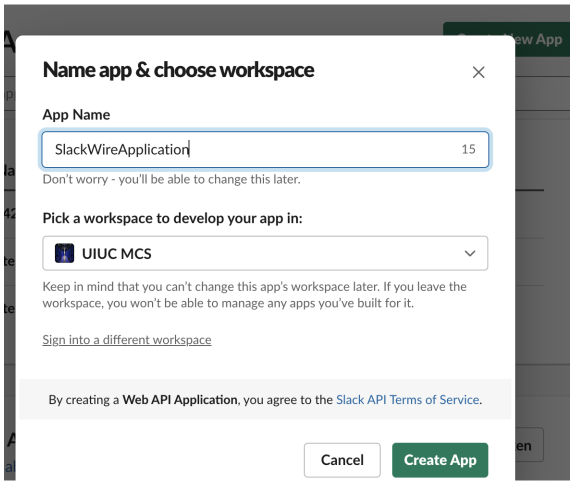
Apps

Create New App

Create an app

Choose how you'd like to configure your app's scopes and settings.

From scratch
Use our configuration UI to manually add basic info, scopes, settings, & features to your app.



Name app & choose workspace

App Name

SlackWireApplication

Don't worry - you'll be able to change this later.

Pick a workspace to develop your app in:

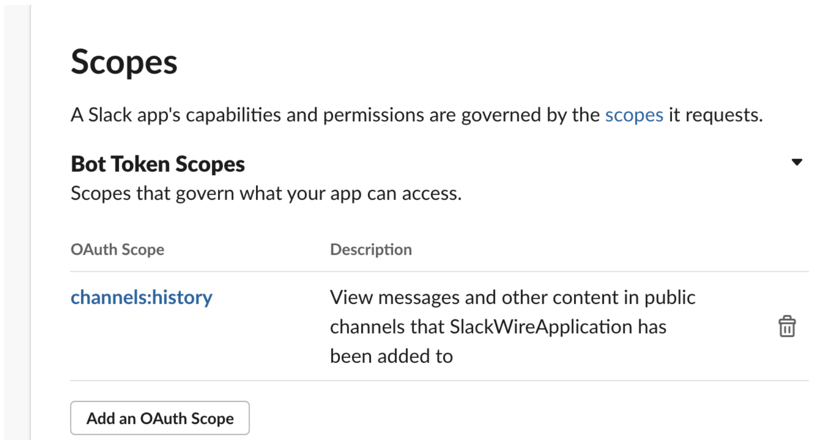
UIUC MCS

Keep in mind that you can't change this app's workspace later. If you leave the workspace, you won't be able to manage any apps you've built for it.

Sign into a different workspace

By creating a **Web API Application**, you agree to the Slack API Terms of Service.

Now that your App has been created, click on "Oauth and Permissions" on the left. Under 'User Token Scopes', add a new Oauth Scope 'channels:history'.



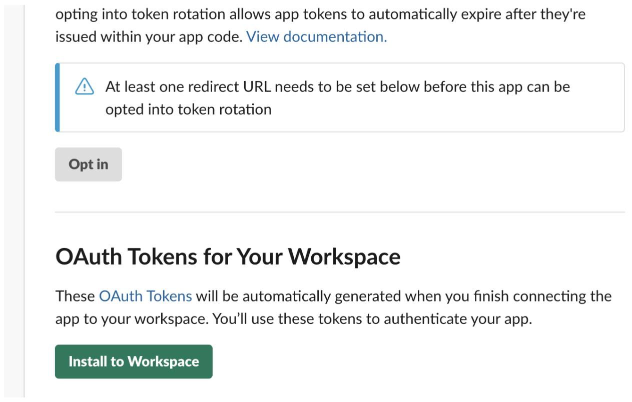
Scopes

A Slack app's capabilities and permissions are governed by the scopes it requests.

Bot Token Scopes
Scopes that govern what your app can access.

OAuth Scope	Description
channels:history	View messages and other content in public channels that SlackWireApplication has been added to

Scroll to the top of the 'Oauth and Permissions' page and select "Install to workspace" and hit allow.



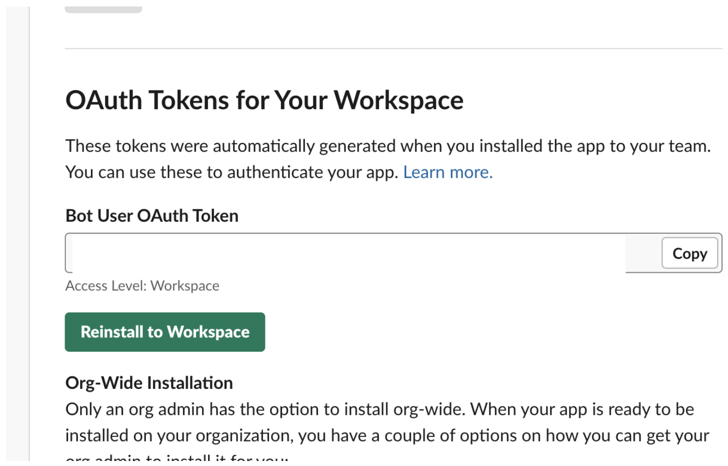
opting into token rotation allows app tokens to automatically expire after they're issued within your app code. View documentation.

At least one redirect URL needs to be set below before this app can be opted into token rotation

OAuth Tokens for Your Workspace

These OAuth Tokens will be automatically generated when you finish connecting the app to your workspace. You'll use these tokens to authenticate your app.

You will then see a "User OAuth Token". Copy this token.



OAuth Tokens for Your Workspace

These tokens were automatically generated when you installed the app to your team. You can use these to authenticate your app. Learn more.

Bot User OAuth Token

Access Level: Workspace

Org-Wide Installation

Only an org admin has the option to install org-wide. When your app is ready to be installed on your organization, you have a couple of options on how you can get your

Mac:
```
export SLACK_TOKEN=<oauth_token>
```

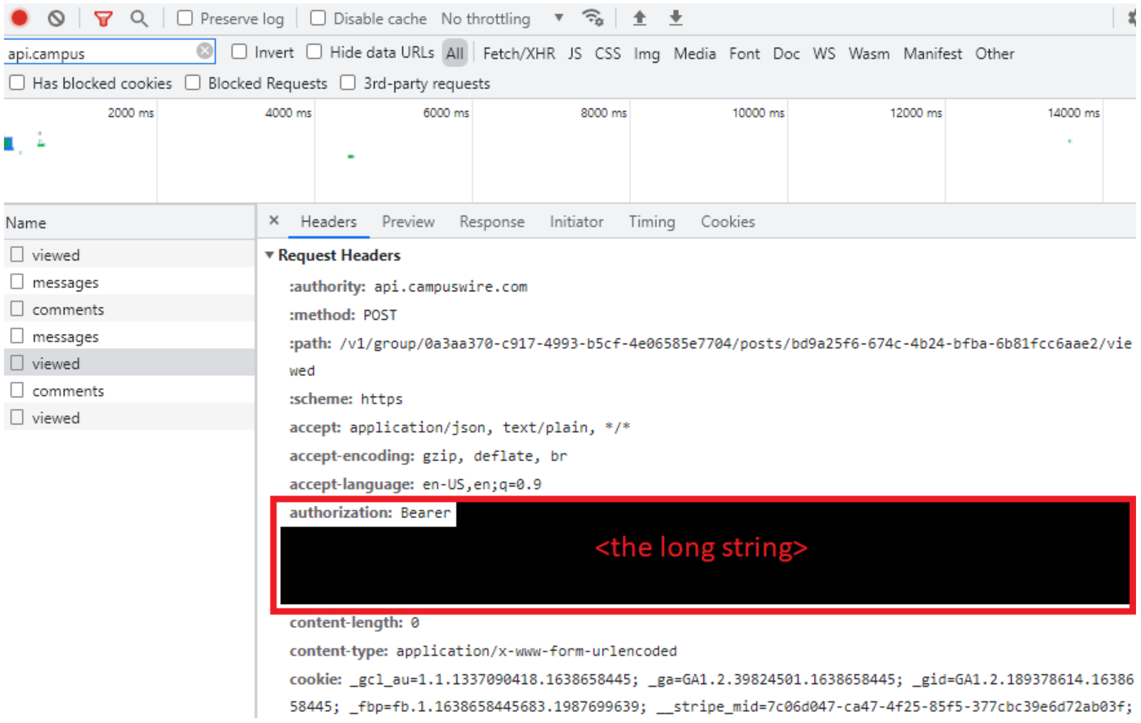
Windows:
```
setx SLACK_TOKEN <oauth_token>
```

Campuswire Auth Token Creation

To find your Campuswire authorization token, open a browser and navigate to CampusWire. Authenticate and login, going to the CS410 Campuswire Page. Open Developer Tools and navigate to the Network Tab. Search for a request to the posts endpoint (e.g. https://api.campuswire.com/v1/group/0a3aa370-c917-4993-b5cf-4e06585e7704/posts?number=20). You may need to refresh this page. Look in the request headers for this request, and you will see an authorization header with value Bearer <some_long_string>. Copy the long string and run the following command:

Mac:
```
export CAMPUSWIRE_TOKEN=<the_long_string>
```

Windows:
```
setx CAMPUSWIRE_TOKEN <the_long_string>
```



api.campus

```
viewed
messages
comments
messages
viewed
comments
viewed
```

Request Headers

```
:authority: api.campuswire.com
:method: POST
:path: /v1/group/0a3aa370-c917-4993-b5cf-4e06585e7704/posts/bd9a25f6-674c-4b24-bfba-6b81fcc6aae2/vie
wed
:scheme: https
accept: application/json, text/plain, */*
accept-encoding: gzip, deflate, br
accept-language: en-US,en;q=0.9
authorization: Bearer
```

[redacted] <the long string>

```
content-length: 0
content-type: application/x-www-form-urlencoded
cookie: _gcl_au=1.1.1337090418.1638658445; _ga=GA1.2.39824501.1638658445; _gid=GA1.2.189378614.16386
58445; _fbp=fb.1.1638658445683.1987699639; __stripe_mid=7c06d047-ca47-4f25-85f5-377cbc39e6d72ab03f;
```