Intro

Tweepy is a Python library to access the Twitter API. The Tweepy library provides functions to easily retrieve text from Twitter. Given the extensive use of social media and social media's increasing relevance in today's world for all kinds of data, Tweepy makes it easy to mine the text from the platform and provides many useful features.

Getting Started and API

The main start is using authentication through your own Twitter profile using tokens. Authentication is easy as it uses OAuth 1a with the tokens simply given to you in your account and then can be applied directly with the following two lines of code.

```
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
```

This is straight forward as a user just needs a Twitter account and "apply" for a developer account which just consists of some basic questions as to what you want to do with the API.

Tweepy provides several interesting features including being able to interact through a user profile such as posting tweets, retweeting, interacting with other users, etc. The most basic feature seems to be interacting with your own timeline. The first example that the documentation uses is to print the first 20 tweets from your home timeline. However, given the context of this class, we are most interested in the "search" features.

Search Features

An important note is that the search methods of Tweepy are limited based on what type of API account you have – ranging from a free account to enterprise / premium account. For a full historical search, an enterprise / premium account is needed which allows a user to search through the entirety of the history of Twitter from 2006. Unfortunately, in a free search, not all tweets will be made available – however, for most general purposes in analyzing trends or sentiment, this should be good enough.

The search function takes in a total of ten arguments which the user can declare.

```
API.search(q[, geocode][, lang][, locale][, result_type][, count][, until][, since_id][, max_id][, include_entities])
```

The query can be up 500 characters maximum. For example, we can use an array of search terms if interested in several queries. The rest of the arguments act as filters. We can filter for things such as location or language or a time interval. Unless using a premium account, tweet history is limited to just the past seven days.

Upon completing a search, Tweepy returns what is called a *SearchResults object*. The best way to interact with this object is through calling the search function in Tweepy's pagination method.

Pagination

Tweepy also provides an easy to iterate through different objects in Twitter using something called a *Cursor* object.

```
for status in tweepy.Cursor(api.user_timeline).items():
```

By using a cursor object, we can treat each tweet as an "item" which allows us to specify exactly how many tweets we want to use for analysis per search. This makes iterating a lot easier. This feature can also be used for other features of Tweepy that requires iteration.

Additional Features

Streaming may be of interest for users that want to keep interact or analyze in real time.

Conclusion

Tweepy provides a very easy to use package for those interested in mining tweets from Twitter. With regards to text retrieval, Tweepy makes it quite easy.

If combined with another type of text package such as TextBlob, there are a lot of possibilities in what we can do.