Twitter TF-IDF with Spark

Steps:

To stream live tweets from Twitter, we use the Twitter Streaming API. To stream live tweets with Python, `tweepy` is the de facto library used that provides a Python wrapper around the Twitter Streaming API. To stream tweets, the following classes need to be imported.

```
from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream
```

The `OAuthHandler` class authenticates a user's access keys and the application's access keys, which can be obtained from the Twitter Developers website. The `StreamListener` class has to be overridden with an implementation to read the tweets and store them in files. Using a list of keywords and the `track=keywords` option to filter tweets in the stream, we get tweets which only contain strings from that list of keywords.
In this case, we use the company names as keywords to filter on.

```
keywords = ['facebook', 'google', 'amazon', 'microsoft', 'twitter', 'linkedin', 'ap
ple']
stream.filter(track = keywords)
```

Each Tweet is further filtered based on language and then appended to a file `compan_name.txt` based on which companies the tweet matches.

```
companies = [keyword for keyword in keywords if keyword in text]
for company in companies:
  print json_tweet['text'], 'Company: ', company
  filename = 'twitter-docs/' + company + '.txt'
  f = open(filename, 'a') # append option
  f.write(text + '\n')
```

Now that we have the tweets, separated into files based on companies, we can compute TF-IDF from these tweets.
To do so, we first have to load each of these files into separate Spark Resilient Distributed Datasets (RDDs).

```
from pyspark import SparkContext
sc = SparkContext(appName='TwitterTFIDF')
company_tweets = sc.wholeTextFiles('twitter-docs/').values().map(lambda doc : re.
split('\W+',doc))
```

The last line reads text files from the directory `twitter-docs`, converts them into RDDs, and then splits the contents of each RDD into its component words. To do this, we use regular expressions to split the input on all characters except those that constitute words. This is pattern is encoded by the regular expression `\W+`.

Once we have the words of the tweets we have collected, we are ready to compute the TF-IDF values of each of the words. This is done with the following code block.

```
tf = HashingTF().transform(company_tweets)
    idf = IDF().fit(tf)
    tfidf = idf.transform(tf)
```

This first hashes the words into keys for TF-IDF and computes the term frequencies.
Then we make 1 pass over the transformed TF output to compute the IDF and another pass to compute the final TF-IDF values.

Last but not least, we store the TF-IDF scores into text files in the `twitter-tfidf-output/` folder.

```
tfidf.saveAsTextFile('twitter-tfidf-output')
```