TF-IDF of Wikipedia webpages

Steps

To collect a large number of related pages from Wikipedia, I have used the `goose-extractor` Python package.
The approach followed is:
Provide a base URL to Goose (https://en.wikipedia.org/wiki/List_of_NP-complete_problems (https://en.wikipedia.org/wiki/List_of_NP-complete_problems)) and get the contents of the web page identified by this base URL.

```
from goose import Goose
g = Goose()
base_url = 'https://en.wikipedia.org/wiki/List_of_NP-complete_problems'
base_article = g.extract(url=base_url)
```

Using the `base_article`, we can get all the links going out of this page using `base_article.links`. These links are then filtered to only include links to other wiki-pages and curated to restrict the number of links to only related pages.

```
links = filter(lambda s: ('/wiki/' in s) and (':' not in s), base_article.links)
upto_idx = links.index('/wiki/Karp%27s_21_NP-complete_problems')
links = links[:upto_idx]
```

Now for each url in the wiki links, we have to get the contents of its web page and store it a file stored in the directory `wiki-pages/`. In this way, we get one `.txt` file per url.

```
for link in links:
    url = 'https://en.wikipedia.org' + link
    print link[6:]
    article = g.extract(url=url)

    filename = 'wiki-pages/' + link[6:]
    f = open(filename, 'w')
    f.write(article.cleaned_text.encode('ascii', 'ignore'))
    f.close()
```

Now that we have the content of the wiki page for each Wikipedia article, we can compute TF-IDF from these pages. 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='WikipediaTFIDF')
npc_problems = sc.wholeTextFiles('wiki-pages/').values().map(lambda doc : re.spli
t('\W+', doc))
```

The last line reads text files from the directory `wiki-pages`, 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 all the words of each of the wiki pages 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 `wikipedia-tfidf-output/` folder.

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