

Information Retrieval

COMP 479 Project 4 DEMO

Lin Ling (40153877)

A report submitted in partial fulfilment of the requirements of Comp479.

Concordia University

If you run the concordia_crawler.py at terminal: scrapy runspider concordia_crawler.py

And the terminal will show what happening as follows:

```
Terminal: Local +
Microsoft Windows [Version 10.0.19045.2251]
(c) Microsoft Corporation. All rights reserved.

(base) D:\0.Information Retrieval\Project\P4\P4>scrapy runspider concordia_crawler.py
2022-12-06 21:12:48 [scrapy.utils.log] INFO: Scrapy 2.6.1 started (bot: scrapybot)
2022-12-06 21:12:48 [scrapy.utils.log] INFO: Versions: lxml 4.8.0.0, libxml2 2.9.12, cssselect 1.1.0, parsel 1.6.0, w3lib 1.21.0, Twisted 22.2.0, Python 3.9.12 (main, Apr 4 2022, 05:22:27)
[MSC v.1916 64 bit (AMD64)], pyOpenSSL 21.0.0 (OpenSSL 1.1.1n 15 Mar 2022), cryptography 3.4.8, Platform Windows-10-10.0.19045-SP0
2022-12-06 21:12:48 [scrapy.crawler] INFO: Overridden settings:
{'ROBOTSTXT_OBEY': True, 'SPIDER_LOADER_WARN_ONLY': True}
2022-12-06 21:12:48 [scrapy.utils.log] DEBUG: Using reactor: twisted.internet.selectreactor.SelectReactor
2022-12-06 21:12:48 [scrapy.extensions.telnet] INFO: Telnet Password: 9a3820c71e08fee
2022-12-06 21:12:48 [scrapy.middleware] INFO: Enabled extensions:
['scrapy.extensions.corestats.CoreStats',
 'scrapy.extensions.telnet.TelnetConsole',
 'scrapy.extensions.logstats.LogStats']
2022-12-06 21:12:49 [scrapy.middleware] INFO: Enabled downloader middlewares:
['scrapy.downloadermiddlewares.robotstxt.RobotsTxtMiddleware',
 'scrapy.downloadermiddlewares.httpauth.HttpAuthMiddleware',
 'scrapy.downloadermiddlewares.downloadtimeout.DownloadTimeoutMiddleware',
 'scrapy.downloadermiddlewares.defaultheaders.DefaultHeadersMiddleware',
 'scrapy.downloadermiddlewares.useragent.UserAgentMiddleware',
 'scrapy.downloadermiddlewares.retry.RetryMiddleware',
 'scrapy.downloadermiddlewares.redirect.MetaRefreshMiddleware',
 'scrapy.downloadermiddlewares.httpcompression.HttpCompressionMiddleware',
 'scrapy.downloadermiddlewares.redirect.RedirectMiddleware',
 'scrapy.downloadermiddlewares.cookies.CookiesMiddleware',
 'scrapy.downloadermiddlewares.httpproxy.HttpProxyMiddleware',
 'scrapy.downloadermiddlewares.stats.DownloaderStats']
2022-12-06 21:12:49 [scrapy.middleware] INFO: Enabled spider middlewares:
```

```
2022-12-06 21:12:49 [scrapy.middleware] INFO: Enabled item pipelines:
[]
2022-12-06 21:12:49 [scrapy.core.engine] INFO: Spider opened
2022-12-06 21:12:50 [scrapy.extensions.logstats] INFO: Crawled 0 pages (at 0 pages/min), scraped 0 items (at 0 items/min)
2022-12-06 21:12:50 [scrapy.extensions.telnet] INFO: Telnet console listening on 127.0.0.1:6023
2022-12-06 21:12:50 [scrapy.core.engine] DEBUG: Crawled (404) <GET https://www.concordia.ca/robots.txt> (referer: None)
```

Start to crawl and download the web pages, start from doc#1 and will end at doc#100 as files limit set up to 100.

```
2022-12-06 21:12:50 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.concordia.ca/ginacody/students.html> (referer: https://www.concordia.ca/ginacody.html)
2022-12-06 21:12:50 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.concordia.ca/summer/ginacody.html> (referer: https://www.concordia.ca/ginacody.html)
2022-12-06 21:12:50 [concordia] INFO: Scrapping doc #1: https://www.concordia.ca/ginacody.html
2022-12-06 21:12:50 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.concordia.ca/web/terms.html> (referer: https://www.concordia.ca/ginacody.html)
2022-12-06 21:12:50 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.concordia.ca/web/accessibility.html> (referer: https://www.concordia.ca/ginacody.html)
2022-12-06 21:12:50 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.concordia.ca/ginacody/programs/co-op.html> (referer: https://www.concordia.ca/ginacody.html)
2022-12-06 21:12:50 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.concordia.ca/web/feedback-forms.html> (referer: https://www.concordia.ca/ginacody.html)
2022-12-06 21:12:50 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.concordia.ca/contact.html> (referer: https://www.concordia.ca/ginacody.html)
2022-12-06 21:12:50 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.concordia.ca/web/privacy.html> (referer: https://www.concordia.ca/ginacody.html)
2022-12-06 21:12:50 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.concordia.ca/ginacody/programs/graduate.html> (referer: https://www.concordia.ca/ginacody.html)
2022-12-06 21:12:50 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.concordia.ca/indigenous/resources/territorial-acknowledgement.html> (referer: https://www.concordia.ca/ginacody.html)
2022-12-06 21:12:50 [concordia] INFO: Scrapping doc #2: https://www.concordia.ca/ginacody/students/services.html
2022-12-06 21:12:51 [concordia] INFO: Scrapping doc #3: https://www.concordia.ca/ginacody/research/centres.html
2022-12-06 21:12:51 [concordia] INFO: Scrapping doc #4: https://www.concordia.ca/ginacody/research/chairs.html
2022-12-06 21:12:51 [concordia] INFO: Scrapping doc #5: https://www.concordia.ca/ginacody/research.html
2022-12-06 21:12:51 [concordia] INFO: Scrapping doc #6: https://www.concordia.ca/ginacody/research/expertise.html
2022-12-06 21:12:51 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.concordia.ca/ginacody/students/academic-services/undergraduate/course-sequences.html> (referer: https://www.concordia.ca/ginacody.html)
2022-12-06 21:12:51 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.concordia.ca/ginacody/events.html> (referer: https://www.concordia.ca/ginacody.html)
2022-12-06 21:12:51 [concordia] INFO: Scrapping doc #7: https://www.concordia.ca/ginacody/research/fellowships.html
2022-12-06 21:12:51 [concordia] INFO: Scrapping doc #8: https://www.concordia.ca/ginacody/students/associations.html
2022-12-06 21:12:51 [concordia] INFO: Scrapping doc #9: https://www.concordia.ca/ginacody/students/future-students.html
2022-12-06 21:12:51 [concordia] INFO: Scrapping doc #10: https://www.concordia.ca/ginacody/students/visiting-exchange.html
2022-12-06 21:12:51 [concordia] INFO: Scrapping doc #11: https://www.concordia.ca/ginacody/students/funding-research-opportunities.html
2022-12-06 21:12:51 [concordia] INFO: Scrapping doc #12: https://www.concordia.ca/ginacody/students/new-students.html
```

The spider will be closed as limit reached.

```

2022-12-06 21:12:55 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.concordia.ca/cunews/main/stories/2020/07/14/new-classrooms-and-study-locations-coming-to-concordia-building.html?rootnav=news/stories> (referer: https://www.concordia.ca/cunews/main/stories/2022/04/11/concordia-indoor-faubourg-building-parking-reopens.html?c=/hospitality)
2022-12-06 21:12:55 [concordia] INFO: Scrapping doc #100: https://www.concordia.ca/news/stories/2022/10/28/concordia-releases-its-final-report-of-the-presidents-task-force-on-ism.html?c=/hospitality
2022-12-06 21:12:55 [scrapy.core.engine] INFO: Closing spider (Limit reached.)

```

If you run the `extract_html_text.py`, it will print out the result of two different clustering runs.

```

P4 / extract_html_text.py
concordia_crawler.py
extract_html_text.py
Project
P4 [Test1] D:\0.Information Retrieval\Project\P4\P4\extract_html_text.py
testData
2_clusters.txt
3_clusters.txt
5_clusters.txt
6_clusters.txt
concordia_crawler.py
extract_html_text.py
Figure_1.png
P4_DEMO.docx
P4_DEMO.pdf
P4_REPORT.docx
P4_REPORT.pdf
README.txt
~$4_DEMO.docx
~$5_REPORT.docx
External Libraries
Scratches and Consoles

31 stop_words = stopwords.words('english')
32 text_lower = text.lower()
33 tokens = word_tokenize(text_lower)
34 tokens = list(filter(lambda token: token not in string.punctuation, tokens))
35 words = [ps.stem(token) for token in tokens if token not in stop_words]
36 return words
37
38
39 # use sklearn learn to cluster the resulting document collection
40 vec = TfidfVectorizer(tokenizer=my_tokenizer)
41 X = vec.fit_transform(documents)
42 print(f"n_samples: {X.shape[0]}, n_features: {X.shape[1]}")
43
44 # uses silhouette coefficient to measure the quality of the clusters
45 sil_avg = []
46 range_n_clusters = [2, 3, 4, 5, 6, 7, 8]
47
48 # for k in range_n_clusters:
49 #     kmeans = KMeans(n_clusters=k, init='k-means++', max_iter=100, n_init=5).fit(X)
50 #     labels = kmeans.labels_
51 #     sil_avg.append(silhouette_score(X, labels, metric='euclidean'))
52 #
53 # plt.plot(range_n_clusters, sil_avg, 'bx-')
54 # plt.xlabel('Values of K')
55 # plt.ylabel('Silhouette score')
56 # plt.title('Silhouette analysis For Optimal k')

Run: extract_html_text.py
D:\Users\janel\anaconda3\python.exe "D:\0.Information Retrieval\Project\P4\P4\extract_html_text.py"
n_samples: 53, n_features: 4254
Top terms per cluster:
Cluster 0: school concordia servic student calendar academ graduat scienc univers studi art class colleg event comput gina codi campu engin health
Cluster 1: cybersecur concordia institut research ' school " " citil le ai univers student servic de scienc et calendar system secur
Cluster 2: concordia school student 2022 servic calendar campu univers academ studi 2021 graduat art scienc new colleg class ' media news
(1929.0, 60.28125, [45.0, 49.0, 69.0, 54.0, 52.0, 52.0, 81.0, 78.0, 51.0, 184.0, 58.0, 75.0, 47.0, 76.0, 64.0, 36.0, 67.0, 93.0, 75.0, 48.0, 71.0, 62.0, 55.0, 51.0, 34.0, 54.0, 51.0,
(337.0, 67.4, [60.0, 110.0, 111.0, -15.0, 71.0])
(1258.0, 78.125, [58.0, 56.0, 82.0, 111.0, 42.0, 50.0, 60.0, 86.0, 155.0, 28.0, 61.0, 48.0, 43.0, 197.0, 114.0, 53.0])

Process finished with exit code 0

```

Print out affinn scores is First is total affinn score, second is average affinn score and rest is the list of each document' affinn scores.

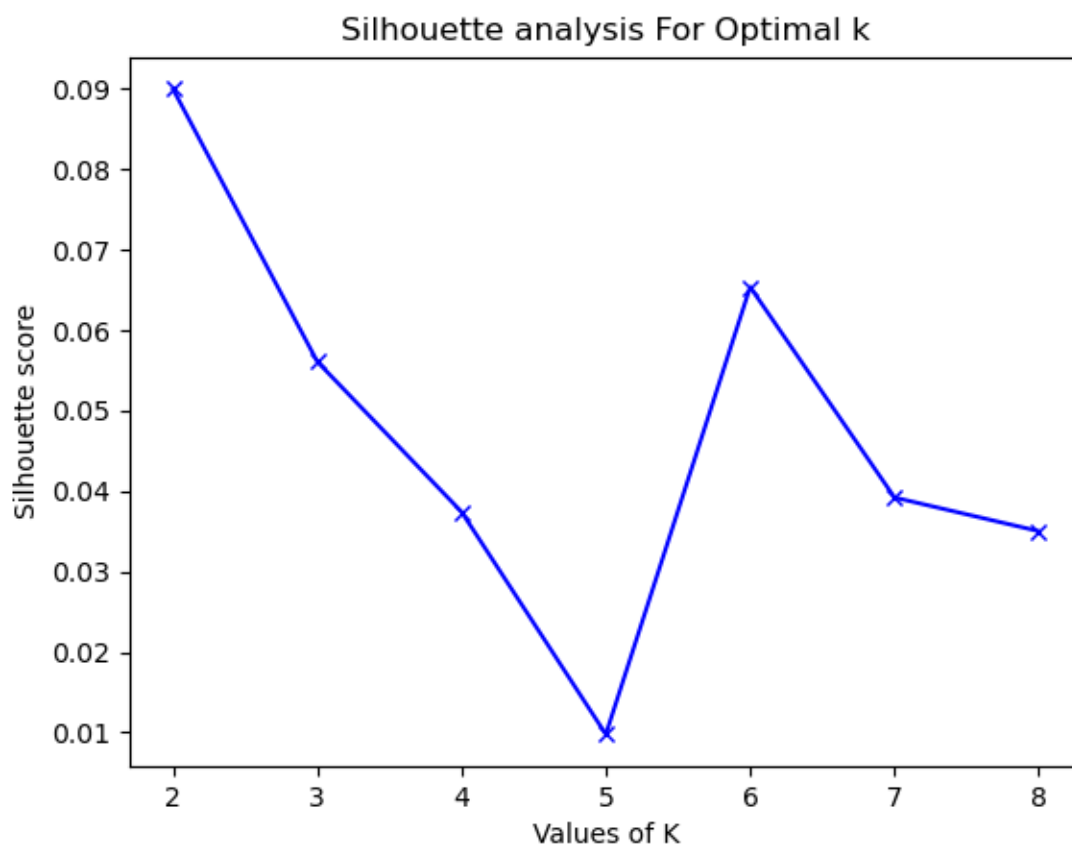
If you want to check more details, please check [report](#) here. If you want to see the output of them, please check [k=3 output](#) and [k=6 output](#). And in the outputs, you could find the sentiment scores for each cluster.

uses silhouette coefficient to measure the quality of the clusters

```

44 # uses silhouette coefficient to measure the quality of the clusters
45 sil_avg = []
46 range_n_clusters = [2, 3, 4, 5, 6, 7, 8]
47
48 for k in range_n_clusters:
49     kmeans = KMeans(n_clusters=k, init='k-means++', max_iter=100, n_init=5).fit(X)
50     labels = kmeans.labels_
51     sil_avg.append(silhouette_score(X, labels, metric='euclidean'))
52
53 plt.plot(range_n_clusters, sil_avg, 'bx-')
54 plt.xlabel('Values of K')
55 plt.ylabel('Silhouette score')
56 plt.title('Silhouette analysis For Optimal k')
57 plt.show()

```



To find an optimal value for the number of clusters K, we use a silhouette plot to display a measure of how close each point in one cluster is to a point in the neighboring clusters and thus provide a way to assess parameters like the number of clusters visually. Let's see how it works.

- Compute K-means clustering algorithm for a range of values.
- For each value of K, find the average silhouette score of data points:
- Plot the collection of silhouette scores for each value of K
- Select the number of clusters when the silhouette score is maximum:

It shows k=2 is optimal value and here is the [output k = 2.](#)

It shows k=5 is optimal value and here is the [output k = 5.](#)