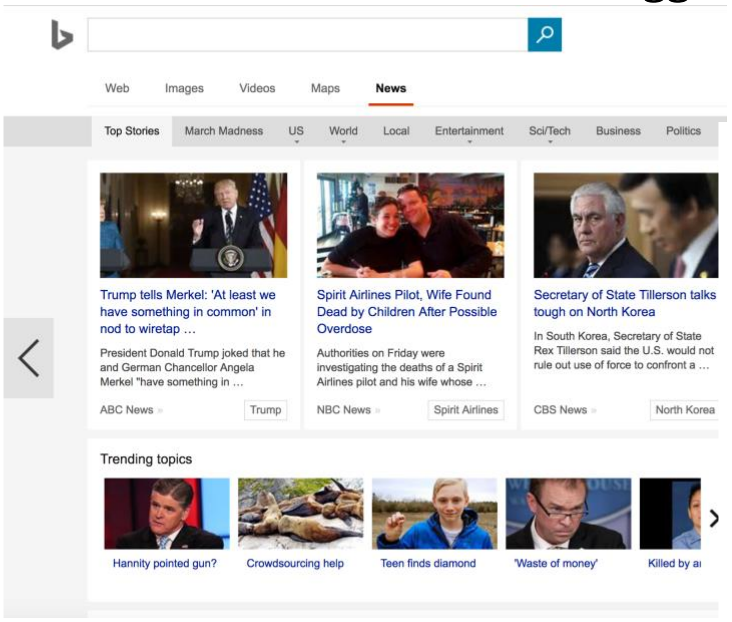
CT594 homework 6:
News aggregator



Trump tells Merkel: 'At least we have something in common' in nod to wiretap ...

President Donald Trump joked that he and German Chancellor Angela Merkel "have something in ...

ABC News · Trump

Spirit Airlines Pilot, Wife Found Dead by Children After Possible Overdose

Authorities on Friday were investigating the deaths of a Spirit Airlines pilot and his wife whose ...

NBC News · Spirit Airlines

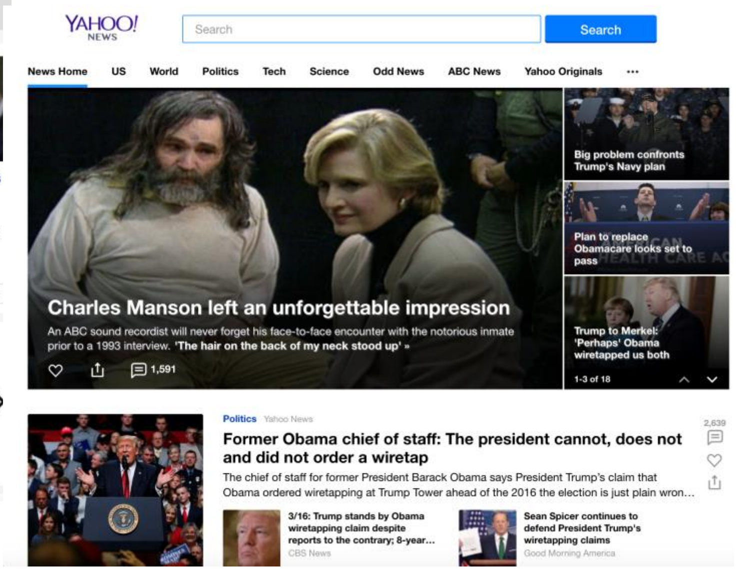
Secretary of State Tillerson talks tough on North Korea

In South Korea, Secretary of State Rex Tillerson said the U.S. would not rule out use of force to confront a ...

CBS News · North Korea

Trending topics

Hannity pointed gun? · Crowdsourcing help · Teen finds diamond · 'Waste of money' · Killed by ai



Charles Manson left an unforgettable impression

An ABC sound recordist will never forget his face-to-face encounter with the notorious inmate prior to a 1993 interview. 'The hair on the back of my neck stood up' »

Big problem confronts Trump's Navy plan

Plan to replace Obamacare looks set to pass

Trump to Merkel: 'Perhaps' Obama wiretapped us both

Politics · Yahoo News

Former Obama chief of staff: The president cannot, does not and did not order a wiretap

The chief of staff for former President Barack Obama says President Trump's claim that Obama ordered wiretapping at Trump Tower ahead of the 2016 the election is just plain wron...

3/16: Trump stands by Obama wiretapping claim despite reports to the contrary; 8-year... · CBS News

Sean Spicer continues to defend President Trump's wiretapping claims · Good Morning America

Many people get their news from online news aggregators and search engines. Those sites can return very different results as shown in the above pictures. The pictures show yahoo news and bing.com news homepages on Friday, March 17 2017 at 8:44 pm. The discrepancies in the results stem from:
- News outlets: The websites they get their content from. News aggregators sites get their content from media outlets websites and often different news sites use different sources.
- Algorithms: The algorithms used to rank news articles. Also, news sites can automatically collect data about the user (like the location) and use that information to customize what is presented to users.
- Personalization: User personalization as many news sites allow the user to specify their interests.

In this assignment, you will write a program that will reproduce the functionality of a news aggregator. A news aggregator is a type of search engine and its implementation follows the same steps.

1. The first component of a web search engine is the ***crawler***. This is a program that downloads web page content that we wish to search for.
2. The second component is the ***indexer***, which will take these downloaded pages and create an inverted index.
3. The third component is ***retrieval***, which answers a user's query by talking to the user's browser. The browser will show the search results and allow the user to interact with the web.

Our news aggregator will have a crawler (sort of), an *indexer* and a *retrieval* component. Also, our news aggregator will have a default page and will provide autocomplete capabilities (the revenge of hw5).

Topics: Indexing, information retrieval, Ethical reasoning, Algorithm analysis, Comparing record, Sorted and Ordered structures, Map structures.

Task I: Social impact/Ethical reasoning

Before we start building our news aggregator, let's think about the social impact of such technology.

We will use the following paper as a starting point of our discussion "How Technology Encourages Political Selective Exposure".
In this part of the assignment, you will write a paper discussing the social impact of internet technology on political selective exposure. Your paper must be at least 1 page (500 words minimum) long and should not be more than 1 and a half pages (800 words) long.
Your paper must be organized as follow.

1- A summary of the paper (listed above): the question(s) being investigated, the method(s) used to answer the questions and the results/findings of the study.
2- The social impacts of political selective exposure: You should discuss
   - The consequences of political selective exposure.
   - The duties of the stakeholders (anyone affected by the technology: technology creators, journalists, users of the internet news outlets, etc.)
   - The actions that can be taken (especially by computer scientists) to prevent and address the negative impact (if any) of political selective exposure

Some useful definitions from the Internet Encyclopedia of Philosophy
Ethics: systematizing, defending, and recommending concepts of right and wrong behavior
Normative ethics: moral standards that regulate right and wrong conduct.
An ethical issue: when the actions of one party pursuing their goal affects the ability of another party to pursue its goal.
Consequence: an action is morally right if the consequences of that action are more favorable than unfavorable.
Duty: acting to fulfill a fundamental obligation to ourselves or others including treating a person as an end and never a means to an end (Kant), rights (a justified claim on another's behavior), and justice (the duty to recognize merit).
Virtue: acting in ways that foster good habits of character (courage, generosity, sincerity...)

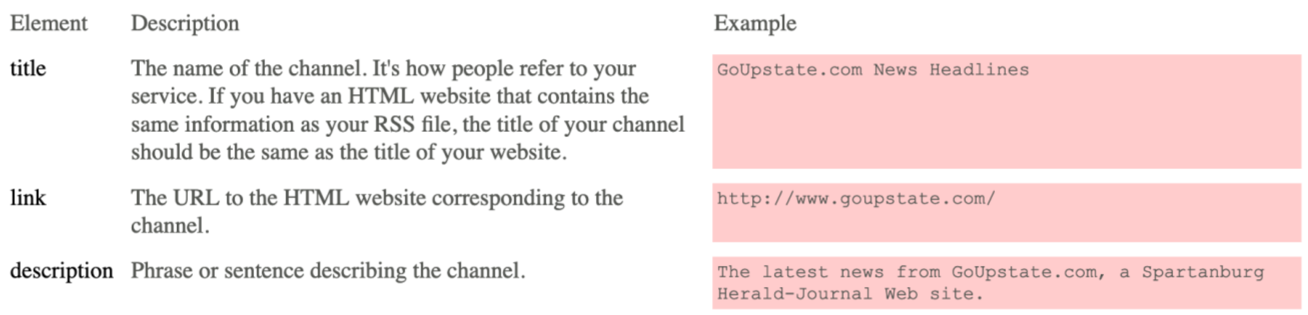
Task I is due on Wednesday, April 29, 2020 at 11:59PM on Gradescope.

Task II: Parsing RSS and HTML documents (the crawler)

RSS:

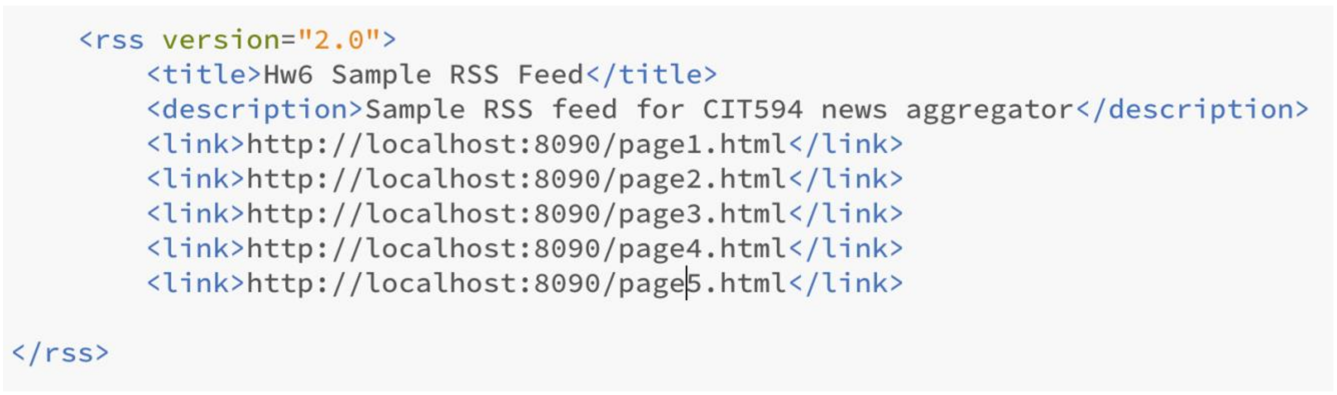
- RSS is a Web content syndication format and it stands for ***R**eally **S**imple **S**yndication.*
- Provides a summary of a website using a syndication format
- News organizations use RSS to provide information about the content of their website
- RSS is a subdialect of XML and is a markup language
- **RSS elements** are the building blocks of RSS feeds (files)
- RSS elements are represented by tags
- **RSS tags** annotate/label pieces of content such as "title", "link", "description", and so on

Required Elements:



Element	Description	Example
title	The name of the channel. It's how people refer to your service. If you have an HTML website that contains the same information as your RSS file, the title of your channel should be the same as the title of your website.	`GoUpstate.com News Headlines`
link	The URL to the HTML website corresponding to the channel.	`http://www.goupstate.com/`
description	Phrase or sentence describing the channel.	`The latest news from GoUpstate.com, a Spartanburg Herald-Journal Web site.`

Sample RSS file:



```
    <rss version="2.0">
        <title>Hw6 Sample RSS Feed</title>
        <description>Sample RSS feed for CIT594 news aggregator</description>
        <link>http://localhost:8090/page1.html</link>
        <link>http://localhost:8090/page2.html</link>
        <link>http://localhost:8090/page3.html</link>
        <link>http://localhost:8090/page4.html</link>
        <link>http://localhost:8090/page5.html</link>

</rss>
```

Parsing RSS feeds:

We will use JSOUP to scrape and parse documents (RSS and HTML).

Download the JSOUP jar and add it to your project classpath

(right-click on the project in eclipse -> properties -> java build path -> Libraries ->Add External Jars…)

Parsing an RSS feed means retrieving the URLs stored in (all) the link tags.

To open a document (RSS or HTML) using JSoup do:

- `Document doc = Jsoup.connect(url_of_the_document).get();`

Use the Document variable to get all the elements in the documents representd by a specific tag do:

- `Elements elements = doc.getElementsByTag(tag_name);`
- To get all the links elements in an RSS feed do
  `Elements links = doc.getElementsByTag("link");`

Elements is actually an ArrayList of Element objects. You can use the For-each loop to iterate through all the links in the collection.

```
for (Element link : links){
//do something with link
}
```

To retrieve the text of a specific link Element, do

- `String linkText = link.text();`

`linkText` will contain the url inside the Link tag.
You can retrieve the content of the HTML document located at the URL you just retrieved.

Parsing HTML documents:

The procedure to parse an HTML document is similar to the one of RSS documents. The content of a website is surrounded by "body" tags. We will retrieve the text contained inside the "body" tag using the steps described above.

Note that it might be useful to remove punctuation (and maybe HTML tags) before storing the content of a webpage.

Create a class named `IndexBuilder` that implements `IIndexBuilder`.

Task 2 is performed by the method
`Map<String, List<String>> parseFeed(List<String> feeds);`

Testing:
Create a new package in Eclipse called ‘test’. You will put your test class(es) here.
A directory with an RSS feed and files is provided for testing. Watch the video provided with the assignment to see how to use it.
Test that:

- Your map has the correct number of files
- Your map contains the names of the documents (URLs/keys)
- Your map contains the correct number of terms in the lists (values)

Task III: Create the forward index (indexer)

A *forward index* is used for applications such as topic modeling and most classification tasks. In this program, it will be used for classification (tagging) of news articles.
A forward index maps documents to a list of terms that occur in them.
To identify the "relevant terms" a document, we will use an algorithm called TF-IDF which stands for Term Frequency-Inverse Document Frequency.
The term frequency represents the count of a term in a document and the inverse document frequency penalizes words that appear in many documents (like articles, pronouns, etc.).
TF-IDF is computed for each term in each document (for all the documents) using the following formulas:

```
TF(t) = (Number of times term t appears in a document) / (Total number of terms in
the document)
IDF(t) = log_e(Total number of documents / Number of documents with term t in it).
TF-IDF(t) = TF * IDF
```

You can read more about TF-IDF here.

In your program, the forward index will be represented as a map of a document (key) and a map (value) of the tag terms and their TFIDF values (Map<String, Double>). The value maps are sorted by lexicographic order on the key (tag term).

Task 3 is performed by the method

```
Map<String, Map<String, Double>> buildIndex(Map<String,
                                             List<String>> docs);
```

Testing:
Below is a list of some TFIDF values from the test files that you can use for testing.

Term	**Document**	**TFIDF**
data	page1.html	0.183
structures	page1.html	0.183
linkedlist	page1.html	0.091
stacks	page1.html	0.091
lists	page1.html	0.051
data	page2.html	0.0832
structures	page2.html	0.0666
search	page2.html	0.0585
binary	page2.html	0.0499
queues	page2.html	0.0166
implements	page3.html	0.0502
java	page3.html	0.0502
trees	page3.html	0.0832
treeset	page3.html	0.0487
binary	page3.html	0.0277

mallarme	page4.html	0.0731
poem	page4.html	0.0731
do	page4.html	0.1463
think	page4.html	0.0731
others	page4.html	0.0731
categorization	page5.html	0.0894
random	page5.html	0.0894
topics	page5.html	0.0894
files	page5.html	0.0509
completely	page5.html	0.0894

Task IV: Create the inverted index (indexer)

An inverted index is the main data structure used in a search engine. It allows for a quick lookup of documents that contain any given term. An inverted index is a map of a tag term (the key) and a Collection (value) of entries consisting of a document and the TFIDF value of the term (in that document). The collection is sorted by reverse tag term TFIDF value (the document in which a term has the highest TFIDF should be visited first).

Task 4 is performed by the method
`Map<?,?> buildInvertedIndex(Map<String, Map<String, Double>> index);`

Wildcard generics (<?>) are used here to allow you to practice data structure selection. Think of wildcard as "unknown types". The general rules of subtyping and Java generics also apply to wildcards. In this case, your data structure must be a subtype of Map and contains unknown types (at compile time) for the key and the value. Your goal here is to decide what implementation of Map is appropriate and what is the appropriate type for the Keys and the Values.

The Java `Entry` interface allows you to manipulate key-value pairs (similar to a tuple). You can use the `AbstractMap.SimpleEntry<K,V>` class that implements Entry everywhere you need to store key-value pairs in a data structure that is not a subtype of Map.

Choosing a collection: some implementations are sorted, and some are not. Some collections allow duplicates and others do not. If your collection is not sorted, you can create a Comparator to sort it (you can also use it with sorted collections) using the `Collections.sort()` static method. Examples of Comparator were provided in homework five, go over them if you are unsure about how to implement a Comparator.

Testing:
Test that:

- Your map is of the correct type (of Map)
- Your map associates the correct files to a term
- Your map stores the documents in the correct order

Task V: Generate the home page

The homepage displays the tag terms and their associated articles. Tag terms are sorted by the number of articles. If two terms have the same number of articles, then they should be sorted by reverse lexicographic order.
Before displaying the homepage, we must remove “stop words” (and their articles).

Stop words:
They are “words which are filtered out before or after processing of natural language text”.
Removing stop words gets rid of the noisy high-frequency words that don’t give any information about the content of the document. You will remove all stop words when creating the collection of entries (Entry objects) that will be displayed on the home page.
A list (HashSet) of stop words is provided in IIndexBuilder (STOPWORDS)

Task 5 is performed by the method
```
public Collection<Entry<String, List<String>>> buildHomePage(
        Map<?, ?> invertedIndex)
```
Testing:
Test that:
- Your Collection is of the correct type
- Your collection stores the entries are in the correct order

Task VI: Searching (retrieval)

The users should be able to enter a query term and our news aggregator will return all the articles related (tagged) to that term. The relevant articles are retrieved from the inverted index.

Task 6 is performed by the method
```
public List<String> searchArticles(
        String queryTerm,
        Map<?, ?> invertedIndex);
```

Testing:
Test that:
- Your list contains the correct number of articles
- Your list contains the correct of articles

Task VII: Autocomplete integration

All good search engines include autocomplete capabilities. We will use the autocomplete system built in the previous homework.
Create a package called `autocomplete`.
Copy your hw5 files (you do not need to include test files) and paste them into the autocomplete package.
Our news aggregator needs to generate the file of tag terms that will be used to initialize our autocomplete module. By default, the autocomplete file is named autocomplete.txt and is located at the root of the assignment folder.

- autocomplete.txt format:
  - The first line contains the number of words in the file
  - The following lines are formatted as follow (without the single quotes): `'spaces 0 word'.` We are ignoring the weight (of a term) in this application so it is perfectly fine to set it to 0 (zero) for all the terms

Potential error:
If after generating the autocomplete file the new list (of the suggested terms) is not updated, just restart the application, select the same RSS feed but do not rebuild the autocomplete file. The suggestions should be accurate after that

Task 6 is performed by the method
```
public Collection<?> createAutocompleteFile(Collection<Entry<String,
                                            List<String>>> homepage);
```
Testing:
Test that:
- Your collection is of the correct type
- Your collection contains the correct number of words

Task VIII: Big-O Estimate

Provide a Big-O estimate of all the methods in *IIndexbuilder*.
Put your answers with a short justification inside an (ASCII) file named Algorithm_analysis.txt file.

Part IX: Reflective assessment (Readme file)
Complete the readme file

Polish your code and be proud of your work. You have just built a search engine from scratch!!!

Grading: This assignment is worth 300 points

Description	**Points**
Autograder	150 points
Manual grading	20 points
Testing (correctness + code coverage)	35 points
Algorithm analysis	35 points
Social impact paper	55 points
Readme file	5 points

This assignment was created by Eric Fouh, based on Jerry Cain assignment.

References:
ChengXiang Zhai and Sean Massung. 2016. Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining. Association for Computing Machinery and Morgan & Claypool, New York, NY, USA.