

Please note that some of the resources used in this assignment require a Stanford Network Account and therefore may not be accessible.

CS107  
Spring 2008

Handout 17  
April 23, 2008

## Assignment 4: RSS News Feed Aggregation

---

Virtually all major newspapers and television news stations have bought into Al Gore's most famous invention ever: the Internet. What you may **not** know is that all of these media corporations serve up RSS feeds summarizing the news stories that've aired or gone to press in the preceding 24 hours. RSS news feeds are XML documents with information about online news articles. If we can get the feeds, we can get the articles, and if we can get the articles, we can build a database of information similar to that held by **news.google.com**. That's precisely what you'll be doing for Assignment 4.

**Due: Thursday, May 1<sup>st</sup> at 11:59 p.m.**

This week's assignment has you index a few hundred online news articles. Indexing a news article amounts to little more than breaking the content down into the individual words, and noting how many times each word appears. If a particular word appears a good number of times and it isn't so common that it appear in virtually every other web page, then said word is probably a good indicator as to what the web page is all about. Once everything's been indexed, you can talk to the database and ask for a list of stories about a specific person, place, or thing. If you're curious what bipartisan issues are surfacing over the war in Iraq, you can just ask your friendly neighborhood database and it's sure to come back with a lot:

```
Please enter a single search term [enter to break]: Iraq1  
We found 196 articles with the word "Iraq". [We'll just list 10, though.]
```

- 1.) "Iraq fears action 'may escalate'" [search term occurs 26 times]  
"news.bbc.co.uk/2/hi/middle\_east/7046765.stm"
- 2.) "Blackwater boss grilled over Iraq" [search term occurs 20 times]  
"news.bbc.co.uk/2/hi/middle\_east/7024370.stm"
- 3.) "Minister seeks Blackwater trials" [search term occurs 19 times]  
"news.bbc.co.uk/2/hi/middle\_east/7046272.stm"
- 4.) "Iraqi blogs " [search term occurs 17 times]  
"news.bbc.co.uk/2/hi/talking\_point/6940384.stm"
- 5.) " Turkey eyes Iraq border incursion" [search term occurs 16 times]  
"seattletimes.nwsourc.com/html/iraq/turkey16.html"
- 6.) "Iraq call over UK military help" [search term occurs 15 times]  
"news.bbc.co.uk/2/hi/uk\_news/7047342.stm"
- 7.) "Blackwater's U.S. complex: mini war zone" [search term occurs 15 times]  
"seattletimes.nwsourc.com/html/nationworld/blackwater14.html"
- 8.) "Navy protects Iraq in Persian Gulf" [search term occurs 13 times]  
"www.boston.com/news/world/navy\_protects\_iraq\_in\_persian\_gulf"
- 9.) "\$4.5 million for a boat nobody wanted" [search term occurs 11 times]  
"seattletimes.nwsourc.com/html/nationworld/favorfactory14m.html"
- 10.) "Saddam's US jailer goes on trial" [search term occurs 11 times]  
"news.bbc.co.uk/2/hi/middle\_east/7045990.stm"

---

<sup>1</sup> I format the sample output a differently than the sample application does, just because I have less space here. You're free to format the output however you want, though.

If you're contemplating a semester abroad, you might see what Paris is up to these days:

Please enter a single search term [enter to break]: **Paris**  
Nice! We found 10 articles that include the word "Paris".

- 1.) "England v France as it happened" [search term occurs 30 times]  
"news.bbc.co.uk/sport2/hi/rugby\_union/7043225.stm"
- 2.) "Diana jury hears of crash horror" [search term occurs 4 times]  
"news.bbc.co.uk/2/hi/uk\_news/7047121.stm"
- 3.) "Bob Denard, at 78; fought communism" [search term occurs 4 times]  
"www.boston.com/news/globe/obituaries/bob\_denard\_at\_78\_fought\_communism"
- 4.) "Bob Denard, 78, staged coups across Africa" [search term occurs 4 times]  
"seattletimes.nwsourc.com/html/nationworld/denardobit16.html"
- 5.) "Obituary: Bob Denard" [search term occurs 2 times]  
"news.bbc.co.uk/2/hi/europe/7044019.stm"
- 6.) "Witness: Motorcyclists caused Diana crash" [search term occurs 2 times]  
"www.philly.com/philly/news/Motorcyclists\_caused\_Diana\_crash.html"
- 7.) "Rapper T.I. still in jail " [search term occurs 2 times]  
"www.boston.com/ae/celebrity/articles/2007/10/16/ti\_still\_in\_jail/"
- 8.) "Airbus delivers first superjumbo" [search term occurs 1 time]  
"news.bbc.co.uk/2/hi/business/7043812.stm"
- 9.) "Chad state of emergency imposed" [search term occurs 1 time]  
"news.bbc.co.uk/2/hi/africa/7047472.stm"
- 10.) "Interpol IDs a suspected pedophile" [search term occurs 1 time]  
"www.boston.com/news/world/europe/articles/2007/10/16/interpol"

If the word is so common that it's useless, the application will tell you about it:

Please enter a single search term [enter to break]: **the**  
Too common a word to be taken seriously. Try something more specific.  
Please enter a single search term [enter to break]: **whatever**  
Too common a word to be taken seriously. Try something more specific.  
Please enter a single search term [enter to break]: **without**  
Too common a word to be taken seriously. Try something more specific.  
Please enter a single search term [enter to break]: **Microsoft**  
Too common a word to be taken seriously. Try something more specific.

Sometimes a perfectly wonderful thing just doesn't get mentioned:

Please enter a single search term [enter to break]: **CS107**  
None of today's news articles contain the word "CS107".

## Starter Code

Once you copy over the Assignment 4 files, you'll see how much is already done for you. All of the networking needed to find and pull online news articles is there. The starter code compiles, runs, and parses web pages from all over the planet. However, it does not build the indices and allow you to do meaningful queries like those illustrated above. Your job for the next several days is to augment the existing code base and integrate in a **hashset** or two (or three) to store everything you might need into order to replicate the functionality of my sample application. The focus of the assignment isn't networking. Assignment 4 is all about taking the client role and using your **hashset** and **vector** along with a few other well-

documented data types in order to build a scalable, efficient search engine. We just happen to index news articles instead of the entire web, but in principle what we do here could easily be extended to index every last web page on Earth.

### Implementation Strategies

Here's how I would tackle the assignment if I were you:

- I would load the list of stop words into a **hashset** of dynamically allocated C strings, and be prepared to pass that **hashset** through the entire code tree. "stop list" is terminology for a list of words that don't say much. We never want to insert a word like that into our set of indices, and we want to inform the client when they enter a stop word during the querying phase. Conceptually, this task is pretty easy, but it'll force you to deal with all of the plumbing required to construct and otherwise manipulate an instance of the generic **hashset**. Check out the supplied **README** file for a **StringHash** function. (There's a stop word list in the **assn-4-rss-news-search-data** directory.)
- Figure out how you're going to store all of the information needed to imitate the functionality of my sample application. You can't possibly build a database mapping keywords to relevant documents if you don't have a clear picture of how everything will be laid out. The **hashset** and **vector** are exactly what you want here.
- Introduce another **hashset** and construct it to just store those words that appear in one or more of the news articles without appearing in the stop list. Pass this **hashset** down through the code hierarchy. Change **BuildIndices** to add all of the words, and update **QueryIndices** to see if the user-supplied search term is in the **hashset** somewhere. The starter code already divvies up each stream into tokens, filtering out HTML tags and stop words, leaving only the words we care to store.
- Never index the same article twice. Consider two online news articles to be the same if they have the same URL (even if the titles are different), **or** if they have the same title and come from the same server (i.e., "**Clay Aiken Joins Spamalot**" might be syndicated twice by **www.nytimes.com**: once by the Front Page feed, and again by the Entertainment feed).

### Implementation Hints

- Copy over the assignment files using **cp -r**. The assignment directory is **assn-4-rss-news-search**. There's a parallel directory (you needn't copy this one over) called **assn-4-rss-news-search-data** that contains the RSS feeds and the stop words list you should be using.
- Read all of the interface files—particularly the **streamtokenizer.h** file, which explains how the **streamtokenizer** type can be used to break down the text of a web

page into a series of words. The `streamtokenizer` is already used quite a bit by the starter application, so between interface and client you should be able to figure out how it works.

- You don't need to bring over your own `vector.c` and `hashset.c` files. I've compiled and archived my own implementations in with the implementations of the `url`, the `urlconnection`, and the `streamtokenizer` functions.
- We want the interactive phase of our application to be case-insensitive. That's easy to support, but it requires we use `tolower` (in `StringHash`) and `strcasecmp` (in a lot of places.) Type `man tolower` and `man strcasecmp` at the command prompt for the lowdown on each.
- Choose the number of buckets to be some large prime number. I used 1009 and 10007 in my own solution.
- Be careful to close all files, close all network connections, and free all dynamically allocated memory. All strings that persist beyond the lifetime of a function should be dynamically allocated character arrays of just the right length. Otherwise, you should keep your own dynamic memory allocation to a minimum.

### Extension Ideas

- Use a real XML parser! If you're interested in working with an open source XML parser called `expat`, then let me know and I'll hook you up with a version of the starter code that uses it. The current version uses my handwritten XML parser which is much more special-purpose and brittle compared to the real one. I contemplated using the `expat` version this time around, but decided it would require too much explanation and would blur the focus of the assignment, which is to understand how C strings and our `vector` and `hashset` generics work.
- Provide support for multiple word and phrase search. The specification just requires that single search terms be supported, but we all know that search engines are much more robust and intelligent than that. Go ahead and research what types of heuristics simple search engines use to index and query mass quantities of information.
- The web pages we index and dynamically populated with text from advertisements, so you'll note that irrelevant words like `vacuum` and `gardening` appear in web pages about burglaries and nuclear weapon truces. It would be nice if you were to come up with a place that somehow filters out the fluff from the real content so that the index only includes meaningful information.