

Corpus Linguistics (L615)

Web as Corpus

Markus Dickinson

Department of Linguistics, Indiana University
Spring 2009

The web provides unprecedented opportunities for gathering data

- ▶ Viable source of “disposable corpora”, built ad hoc for specific purposes
- ▶ Essential for working with specialized languages

Need to automatically extract web corpus data: manual extraction is time-consuming

You can use Perl's libwww (LWP) module to access webpages

(<http://search.cpan.org/~gaas/libwww-perl-5.800/lib/LWP.pm>)

To obtain this, you can download a whole package

(<http://search.cpan.org/dist/libwww-perl/lib/Bundle/LWP.pm>):

- ▶ Type the following at a terminal:
`perl -MCPAN -e 'install Bundle::LWP'`
- ▶ When it asks, “Are you ready for manual configuration?” type 'no' in order to have the installation configure your files

Fetch a webpage

web.pl

```
# See: http://www.perl.com/pub/a/2002/08/20/perlandlwp.html

# import the LWP::Simple module
use LWP::Simple;

# define the webpage to access:
$url = "http://jones.ling.indiana.edu/~mdickinson/";

# get the webpage content
$content = get $url;
die "Couldn't get $url" unless defined $content;

print $content;
```

With this knowledge, you can:

- ▶ Learn more about LWP from Sean Burke's book, *Perl & LWP*, available at: <http://lwp.interglacial.com/>
- ▶ Write regular expressions to extract data of interest (assuming you know what the HTML looks like)
- ▶ Learn more about HTML::Parse and the like

Or: use pre-built software such as BootCaT or the Web as Corpus toolkit (<http://www.drni.de/wac-tk/HomePage>)

BootCaT

Baroni and Bernardini (2004)

BootCaT (Bootstrapping Corpora and Terms) works as follows:

1. Automatically search Google using a small set of seed terms
2. Extract new terms from this initial corpus
3. Use these terms to build a new corpus
4. Extract new terms

From this, you can extract a list of multi-word terms

Extraction of corpora & unigram terms

Bootstrapping terms starts with a small list of seeds

- ▶ Typically, only need 5-15 seed terms for a specialized domain
- ▶ Seed terms are randomly combined & combinations are used as Google queries
 - ▶ Top n pages are retrieved

From this corpus, new unigram seed terms are extracted

- ▶ Compare frequencies of terms to frequencies in a reference corpus
- ▶ Then, the random combination is done again

Process is then repeated

User defines the following parameters:

- ▶ Number of queries issued for each iteration
- ▶ Number of seeds used in a single query
- ▶ Number of pages to be retrieved

BootCaT also aims to extract multi-word terms from a particular domain of interest

- ▶ Extract one & two-word connectors from the corpus: terms frequently occurring between single-word terms
- ▶ Extract a list of stop words (high document frequency)
- ▶ Look for multi-word terms
 - ▶ contain at least one unigram term
 - ▶ do not contain stop words
 - ▶ connectors do not appear at the edges
 - ▶ have a high enough frequency
 - ▶ are not part of longer frequent multi-word terms
 - ▶ do not contain shorter frequent multi-word terms

Acquiring BootCaT

Download the toolkit from:

<http://sslmit.unibo.it/~baroni/bootcat.html>

- ▶ It is a suite of Perl programs
- ▶ “Each program should do only one thing, but do it well”
(unix adage)
- ▶ Also, be sure to download the updates

For documentation about installation & use, see the
`readme.txt` file

Using BootCaT

From the newboot download, look at `web_corpus_how_to.txt`: this is an extremely useful file to get you started

1. (if desired) Obtain an API key from Yahoo (developer.yahoo.com/wsregapp/)
2. Create a set of seeds
3. Run `build_random_tuples.pl` to create tuples
4. Run `collect_urls_from_yahoo.pl` to create url list
 - ▶ You may have to install `Yahoo::Search` package
5. Run `retrieve_and_clean_pages_from_url_list.pl` to get the actual corpus data
 - ▶ This (somewhat untested) program cleans the files
6. Run `discard_duplicates.txt` to remove duplicate files

After that, you can iterate & do it all over again with a larger set of seeds

Making my own corpus (1)

Just for fun, I created a small web corpus of pages about the movie *Teen Wolf*

First, I created a list of seed words:

teen wolf
michael j. fox
styles
boof
basketball

Making my own corpus (2)

Then, I randomly generated tuples:

```
./build_random_tuples.pl teenwolf/seeds.txt \  
> teenwolf/tuples.txt
```

```
"teen wolf" styles basketball  
"michael j. fox" basketball styles  
"michael j. fox" basketball boof  
basketball "teen wolf" "michael j. fox"  
basketball styles boof  
styles "teen wolf" boof  
basketball "teen wolf" boof  
"michael j. fox" "teen wolf" styles  
boof "michael j. fox" "teen wolf"  
"michael j. fox" boof styles
```

Making my own corpus (3)

From this, I got the appropriate urls & cleaned it up:

```
./collect_urls_from_yahoo.pl -k MY_YAHOO_API \  
-l English teenwolf/tuples.txt \  
> teenwolf/url_list.txt
```

```
grep -v CURRENT_QUERY teenwolf/url_list.txt \  
| grep -v NO_RESULTS_FOUND | sort | uniq \  
> teenwolf/cleaned_url_list.txt
```

```
...  
http://en.wikipedia.org/wiki/Teen\_Wolf  
http://furry.wikia.com/wiki/Teen\_Wolf  
http://imdb.com/Title?0090142  
http://members.tripod.com/donignacio/movteenwolf.html  
...
```

Making my own corpus (4)

Finally, we get the corpus (& remove duplicates)

```
./retrieve_and_clean_pages_from_url_list.pl \  
teenwolf/cleaned_url_list.txt > teenwolf/corpus.txt
```

```
./discard_duplicates.pl -d "CURRENT URL" \  
teenwolf/corpus.txt > teenwolf/cleaned_corpus.txt
```

```
CURRENT URL http://allthings80s.blogspot.com/...
```

He always wanted to be special... but he never expected this!
taken me a bit to get over Russ Abbott and that jumper but no
moving onto Teen Wolf. Basically this film is only ever popular
because Michael J Fox stars in it, thats why the people behind
film delayed its release until after Back to the Future was r
(Teen Wolf was made pre BTTF!) to take advantage of the any f
Fox's role in Back to the Future. Cheeky huh! And so it is.

Sketch Engine

Baroni and Kilgarrieff (2006)

The Sketch Engine (<http://www.sketchengine.co.uk/>) has many of the same motivations as BootCaT

- ▶ More general purpose corpus, however

Downside: you can use it for free for 30 days ... but then you have to pay to use it

Upside: some of the tools they use are available for free:

- ▶ <http://www.drni.de/wac-tk/HomePage>
- ▶ <http://sslmitdev-online.sslmit.unibo.it/wac/wac.php>
- ▶ <http://wacky.sslmit.unibo.it/doku.php>

And some seem to have been modified & incorporated into BootCaT (e.g., boilerplate stripping)

Crawl seeding & crawling

Same idea as BootCaT: crawl the web with a few different seeds, combined into two-word queries

- ▶ single word queries seemed to give too many inappropriate pages
- ▶ queries with 3 or more words seemed to give lists of words

Goal was to get both “public sphere” pages (e.g., newspapers) as well as more personal pages (e.g., blogs)

- ▶ sampled mid-frequency words from traditional written domain
- ▶ used basic vocabulary

Limited search to .de and .at domains; kept only one URL from a given domain

Filtered documents by size

- ▶ Small documents (<5KB) contain very little real text
- ▶ Large documents (>200KB) tend to be indices, catalogues, lists, etc.

Removed perfect duplicates

- ▶ Actually, they removed both the original & the duplicate: tend to be warning messages & the like

Boilerplate stripping

boilerplate = HTML markup, javascript, & other non-linguistic material

- ▶ Removing boilerplate information is crucial to obtaining linguistic data only

Heuristic:

- ▶ Content-rich sections of a document will have a low html tag density
- ▶ Boilerplate sections have a wealth of html

This heuristic is “relatively independent of language and crawling strategy”

Function word & pornography filtering

If a text does not have enough function words, it is likely non-linguistic material (e.g., a list)

- ▶ Require at least 10 function word types & 30 tokens on a page ... which must make up at least 25% of the total words
- ▶ With a function word list from a language, this can also serve as a language identifier

Also remove pages which match a stop list of words likely to appear in pornography

- ▶ Tend to contain randomly-generated keyword lists
- ▶ Needs some work, as it includes otherwise harmless words like *girl*

Near-duplicate detection

BootCaT will automatically remove duplicates; the Sketch Engine also removes near duplicates

- ▶ Remove function words
- ▶ Take “fingerprints” of a fixed number of randomly-selected n-grams
 - ▶ e.g., extract 25 5-grams from each document
- ▶ Near-duplicates have a high overlap
 - ▶ e.g., at least 2 5-grams in common

With this data, the Sketch Engine then prepares it better for searching:

- ▶ Run a POS tagger over it (TreeTagger)
- ▶ Clean the documents further, using POS tags
 - ▶ By noting where the POS tag distribution is unusual, they perform another round of anomalous document finding
 - ▶ They look for problematic (erroneous) POS tags and remove those documents
 - ▶ Use cues such as number of unrecognized words, proportion of words with upper-case initial letters, ...

Benefits of a web corpus

This corpus can:

- ▶ Help address data sparseness issues
- ▶ Provide more interpersonal material
- ▶ Check the claims made with other corpora

General purpose web corpora

Sharoff (2006)

Corpus Linguistics

Web as Corpus

LWP

BootCaT

Sketch Engine

Sharoff 2006

References

Additional motivation for general-purpose web corpora

- ▶ Expensive to build corpora otherwise, yet they are needed for under-resourced languages
- ▶ Current corpora are often restricted in size and/or variety
- ▶ News corpora do not represent general language
 - ▶ Need a variety of text types

Can search through corpora described here at:

<http://corpus.leeds.ac.uk/internet.html>

Limitations of web search interfaces

To answer some questions, we could just search the web, but ...

- ▶ Search engines only provide limited context
- ▶ Search engines do not allow for linguistically complex queries (as was a motivation for the LSE)
- ▶ Results are organized according to relevance to the topic, not to left/right context
- ▶ Search engine counts cannot generally be trusted

Corpus of web URLs

One strategy for releasing a corpus is to organize a list of appropriate URLs

- ▶ Need to check that every page has real, connected text
- ▶ Need to develop a BNC-style (representative) corpus from the web

We'll also talk about page attrition later

To get a representative corpus, we need a sufficiently general word list

Issues:

- ▶ Function words shouldn't be included: they often occur with incomplete sentences
- ▶ Polysemous words should be good, in terms of not biasing the corpus towards one particular topic (e.g., *word*, *room*)
- ▶ Lemmatization would be good to use, to handle languages with elaborate morphology
 - ▶ *high* in English and *vysokyi* in Russian have similar counts/ranks with lemmas, but not full word forms

BNC-style corpus

Word selection (2)

Approach: select 500 frequent word forms from a language

- ▶ words which start with lower-case letters (to avoid proper nouns)
- ▶ not specific with respect to a topic

More words starts to get into specific topics & increases efforts on developing query list

BNC-style corpus

Query generation

The queries need to get representative content, with a minimal of noisy pages

Approach: use 4-word queries (i.e., 4-word random combinations of 500 words), totaling 5000 queries

- ▶ Fewer words could lead to unconnected text (e.g., *work & room*)
- ▶ More words returns smaller number of pages: not a random snapshot of the web
- ▶ Four words leads to:
 - ▶ Connected prose
 - ▶ A variety of domains

To ensure that a particular language is obtained, can add a language-specific function word to the query

BNC-style corpus

Downloading

5,000 queries led to 50,000 URLs, with about 3,000-4,000 words per query

- ▶ Generally enough to get 100 million word corpus (though, depends on if links have died, etc.)
 - ▶ Top 25,000 words have at least 100 occurrences each
 - ▶ Seems to be sufficient for lexicography

Corpus size potentially limited by tools that can process them

BNC-style corpus

Post-processing

1. Unify page encodings (e.g., all in UTF-8)
2. Convert HTML into plain text (e.g., using lynx)
3. Filter out identical/near-identical pages
 - ▶ Near-duplicate detection done by looking at shared n -grams (shingling algorithm)

The retrieved corpora

| | I-EN | I-DE | I-RU |
|--------------------------------|-------------|-------------|-------------|
| # of tokens | 126,643,151 | 126,117,984 | 156,534,391 |
| # of word forms | 2,003,056 | 3,384,491 | 2,036,503 |
| # of lemmas | 1,608,425 | 3,081,197 | 791,311 |
| # of URLs | 42,133 | 31,195 | 33,811 |
| Avg. doc. length (in words) | 3,006 | 4,043 | 4,630 |

Text assessment

Authorship

To determine whether the corpora is balanced like the BNC, Sharoff assesses a variety of factors

Authorship:

- ▶ Single
- ▶ Multiple
- ▶ Corporate: 44% for I-EN, 18% for BNC
- ▶ Unknown

Also, female writers are underrepresented: 23%/3% male/female split in I-EN vs. 28%/13% for BNC

- ▶ Written
- ▶ Spoken: 0-1% for web corpora, 10% for BNC
- ▶ Electronic: 16% for Russian, 13% for English, 9% for German; 0% for BNC
 - ▶ Important type of data, as is similar to speech in some ways, similar to writing in some

Text assessment

Audience

Test the level of knowledge expected from the audience (size & level are harder to gauge)

- ▶ General: 33% in I-EN
- ▶ Informed: 45% in I-EN
- ▶ Professional: 22% in I-EN

Overall, I-EN seems somewhat balanced w.r.t. this classification (similar to BNC)

Text assessment

Aims of text production

Not a classification used in the BNC ...

- ▶ Discussion: 45% in I-EN
- ▶ Recommendation
 - ▶ Hard to tell apart from discussion sometimes
- ▶ Recreation: fiction=17% in BNC vs. recreation=4% in I-EN
- ▶ Instruction
- ▶ Information

Text assessment

Domain

- ▶ natsci (natural sciences)
- ▶ appsci (applied sciences): 7% in BNC vs. 29% in I-EN
- ▶ socsci (social sciences): 16% in RRC vs. 5% in I-RU
 - ▶ Would help to subclassify, as most such English web texts are legal
 - ▶ Less than 5% of Reuters texts are any type of science
- ▶ politics
- ▶ business
- ▶ life
- ▶ arts
- ▶ leisure

Word list comparison

To further understand the differences between corpora, Sharoff compares their word frequency lists

- ▶ This can be done quickly, and without predefining categories
- ▶ The log likelihood statistic can be used
 - ▶ See p. 88 of the paper for details on calculating the statistic

Compared both word forms & lemmas

Word list comparison (cont.)

Results are given in the 4 tables on p. 89-90

Financial data tends to:

- ▶ overuse financial terms & specific institutions
- ▶ show greater use of temporal markers for date & time
- ▶ overuse reported speech, i.e., verbs of saying
- ▶ underuse first & second person pronouns, question words, modals, & mundane verbs (e.g., *go*)

If a corpus consists of a list of URLs and associated software for extracting them, how stable is such a corpus?

- ▶ We can measure a corpus's *half-life* by seeing how many pages are left after a certain amount of time
- ▶ Initial experiments show that some links are gone after a few months
 - ▶ February 2005 → August 2005: 934/1000 remaining
 - ▶ June 2005 → August 2005: 982/1000 remaining
- ▶ Need longer term studies and studies testing different parameters

References

- Baroni, Marco and Silvia Bernardini (2004). BootCaT: Bootstrapping corpora and terms from the web. In *Proceedings of LREC 2004*.
http://sslmit.unibo.it/~baroni/publications/lrec2004/bootcat_lrec_2004.pdf.
- Baroni, Marco and Adam Kilgarriff (2006). Large linguistically-processed Web corpora for multiple languages. In *Proceedings of EACL-06, Demonstration Session*. Trento, Italy. <http://aclweb.org/anthology-new/E/E06/E06-2001.pdf>.
- Sharoff, Serge (2006). Creating general-purpose corpora using automated search engine queries. In Marco Baroni and Silvia Bernardini (eds.), *WaCky! Working papers on the Web as Corpus*, Gedit, Bologna.
<http://wackybook.sslmit.unibo.it/pdfs/sharoff.pdf>.