

# HG3051 Corpus Linguistics

## DIY Corpora, Processing Raw Text, SQL

Francis Bond

**Division of Linguistics and Multilingual Studies**

<http://www3.ntu.edu.sg/home/fcbond/>

[bond@ieee.org](mailto:bond@ieee.org)

Lecture 6

<https://github.com/bond-lab/Corpus-Linguistics>

HG3051 (2018)

# Overview

---

- Revision of Statistics
  - Frequency
  - Corpus Statistics
  - Collocations
- DIY Corpora
- Processing Raw Text
- Structured Query Language

---

# Revision of Statistics

## Lexical statistics

---

- Statistical study of the frequency distribution of types (words or other linguistic units) in texts
- Try to gauge how confident we are in our results
- Try to measure if things are more common than chance

# Zipf's law

---

➤ Zipf's (1949, 1965) famous law:

$$f(w) = \frac{C}{r(w)^a} \quad \text{or} \quad f(w) \propto \frac{1}{r(w)} \quad (1)$$

➤ With  $a = 1$  and  $C = 60,000$ , Zipf's law predicts that:

- \* most frequent word occurs 60,000 times
- \* second most frequent word occurs 30,000 times
- \* third most frequent word occurs 20,000 times
- \* and there is a long tail of 80,000 words with frequencies
- \* between 1.5 and 0.5 occurrences(!)

# Applications of word frequency distributions

---

- Most important application: extrapolation of vocabulary size and frequency spectrum to larger sample sizes
  - productivity (in morphology, syntax, ...)
  - lexical richness  
(in stylometry, language acquisition, clinical linguistics, ...)
  - practical NLP (est. proportion of OOV words, typos, ...)
- Direct applications of Zipf's law in NLP
  - Population model for Good-Turing smoothing
  - Realistic prior for Bayesian language modelling

# Statistics & language

---

- Apply statistical procedure to linguistic problem
  - take random sample from (extensional) language
  - What are the objects in our population?
    - \* words? sentences? texts? ...
  - Objects = whatever proportions are based on → unit of measurement
- We want to take a random sample of these units

# Inference from a sample

---

- Principle of inferential statistics
  - if a sample is picked at random, proportions should be roughly the same in the sample and in the population
- Take a sample of, say, 100 VPs
  - observe 19 passives  $\rightarrow p = 19\% = .19$
  - style guide  $\rightarrow$  population proportion  $\pi = 15\%$
  - $p > \pi \rightarrow$  reject claim of style guide?
- Take another sample, just to be sure
  - observe 13 passives  $\rightarrow p = 13\% = .13$
  - $p < \pi \rightarrow$  claim of style guide confirmed?



# Sampling variation

---

- random choice of sample ensures proportions are the same on average in sample and in population
- but it also means that for every sample we will get a different value because of chance effects → sampling variation
- The main purpose of statistical methods is to estimate & correct for sampling variation

# Estimating sampling variation

---

- We don't need an infinite number of monkeys (or corpus linguists) to answer these questions
  - Statistical hypothesis tests
    - \* define a rejection criterion for refuting  $H_0$
    - \* control the risk of false rejection (type I error) to a “socially acceptable level” (significance level)
    - \* p-value = risk of false rejection for observation
    - \* p-value interpreted as amount of evidence against  $H_0$
  - Two-sided vs. one-sided tests
    - \* in general, two-sided tests should be preferred
    - \* one-sided test is plausible in our example

# Hypothesis tests in practice

---

➤ Easy: use online wizard

- <http://sigil.collocations.de/wizard.html>
- <http://faculty.vassar.edu/lowry/VassarStats.html>
- open-source software <http://www.r-project.org/>

➤ Or Python

- One-tail test  
`scipy.stats.binom.sf(51-1, 235, 1.0/6)`
- Two-tail test  
`scipy.stats.binom_test(51, 235, 1.0/6)`

# Confidence interval

---

- What if we do not have an obvious null hypothesis to start with?
  - this is typically the case in (computational) linguistics
- We can estimate the true population proportion from the sample data (relative frequency)
  - sampling variation → range of plausible values
  - such a confidence interval can be constructed by inverting hypothesis tests (e.g. binomial test)
- Size of confidence interval depends on sample size and significance level
- <http://sigil.collocations.de/wizard.html>

## Frequency comparison

---

- Many linguistic research questions can be operationalised as a frequency comparison
  - Are split infinitives more frequent in AmE than BrE?
  - Are there more definite articles in texts written by Chinese learners of English than native speakers?
  - Does *meow* occur more often in the vicinity of *cat* than elsewhere in the text?
  - Do speakers prefer *I couldn't agree more* over alternative compositional realisations?
- Compare observed frequencies in two samples

## Frequency comparison

---

$k_1$	$k_2$	19	25
$n_1 - k_1$	$n_2 - k_2$	81	175

- Contingency table for frequency comparison
  - e.g. samples of sizes  $n_1 = 100$  and  $n_2 = 200$ , containing 19 and 25 passives
  - $H_0$ : same proportion in both underlying populations
- Chi-squared  $X^2$ , likelihood ratio  $G^2$ , Fisher's test
  - based on same principles as binomial test
- Many possible scores

# What is a collocation?

---

- Words tend to appear in typical, recurrent combinations
- such pairs are called collocations (Firth 1957)
- the meaning of a word is in part determined by its characteristic collocations

“You shall know a word by the company it keeps!”
- Native speakers have strong and widely shared intuitions about such collocations
  - Collocational knowledge is essential for non-native speakers in order to sound natural
  - This is part of “idiomatic English”

## An important distinction

---

- **Collocations** are an empirical linguistic phenomenon
  - can be observed in corpora and quantified
  - provide a window to lexical meaning and word usage
  - applications in language description (Firth 1957) and computational lexicography (Sinclair 1966, 1991)
- **Multiword expressions** = lexicalised word combinations
  - MWE need to be lexicalised (i.e., stored as units) because of certain idiosyncratic properties
  - non-compositionality, non-substitutability, non-modifiability (Manning and Schütze, 1999)
  - not directly observable, defined by linguistic tests (e.g. substitution test) and native speaker intuitions
  - Sometimes called **Collocations**



---

# DIY Corpora

# Why build your own?

---

1. Decide what you want to study  
see if you can do it with existing resources  
No ☹️
2. Collect data that fits your needs
  - Speech (expensive)
  - Text (easy to get, hard to do legally)
3. Process it
  - Clean up
  - Mark up (what do we know about the data originally)
  - Annotation (what will we add to it)

# Collecting Text

---

Some ways to collect text:

- Word-processed texts: save as a text only file
- Keyboard entry: speech transcription, students' handwritten essays, etc.
- Scanning: copyright protected novels, latest magazines, etc.
- CD-ROM: newspaper, encyclopaedia, ICAME, etc.
- Internet resources: email, chat, public documents, newspapers, magazines etc.
- Text archives: copyright free (old) novels, essays, etc.
- Copying from a large corpus: e.g. using sections of the BNC

## Recent Trends

---

- Government Documents (proceedings, publications)
- Open source documents (manuals, wikipedia)
- Social Media: Blogs and twitter

# Web as Corpus

---

## Two Approaches to using the Web as a Corpus

---

- **Direct Query:** Search Engine as Query tool and WWW as corpus?  
(Objection: Results are not reliable)
  - Population and exact hit counts are unknown → no statistics possible.
  - Indexing does not allow to draw conclusions on the data.
  - ⊗ Google is missing functionalities that linguists / lexicographers would like to have.
- **Web Sample:** Use search engine to download data from the net and build a corpus from it.
  - known size and exact hit counts → statistics possible.
  - people can draw conclusions over the included text types.
  - (limited) control over the content.
  - ⊗ sparser data

# Direct Query

---

- Accessible through search engines (Google API, Yahoo API, Scripts)
- Document counts are shown to correlate directly with “real” frequencies (Keller 2003), so search engines can help - but...
  - lots of repetitions of the same text (not representative)
  - very limited query precision (no upper/lower case, no punctuation...)
  - only estimated counts, often hard to reproduce exactly
  - different queries give wildly different numbers

## Web Sample

---

- Extracting and filtering web documents to create linguistically annotated corpora (Kilgarriff 2006)
  - gather documents for different topics (balance!)
  - exclude documents which cannot be preprocessed with available tools (here taggers and lemmatizers)
  - exclude documents which seem irrelevant for a corpus (too short or too long, word lists,...)
  - do this for several languages and make the corpora available



# Building Internet Corpora: Outline

---

1. Select Seed Words (500)
2. Combine to form multiple queries (6,000)
3. Query a search engine and retrieve the URLs (50,000)
4. Download the files from the URLs (100,000,000 words)
5. Postprocess the data (encoding; cleanup; tagging and parsing)

Sharoff, S (2006) Creating general-purpose corpora using automated search engine queries. In M. Baroni, S. Bernardini (eds.) *WaCky! Working papers on the Web as Corpus*, Bologna, 2006.

## Post-processing

---

- Filter documents by size
  - Small documents ( $< 5KB$ ) contain very little real text
  - Large documents ( $> 200KB$ ) tend to be indices, catalogues, lists, etc.
- Remove perfect duplicates
  - Actually, removed both the original & the duplicate:  
... tend to be warning messages

# Boilerplate stripping

---

- **Boilerplate**: HTML markup, javascript, other non-linguistic material
- Removing boilerplate information is crucial to obtaining linguistic data only
  - 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”
- 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 and 30 tokens on a page  
... which must make up at least 25% of the total words

## Near-duplicate detection

---

- Take **fingerprints** of a fixed number of randomly-selected  $n$ -grams (ignoring function words)
  - e.g., extract 25 5-grams from each document
- Near-duplicates have a high overlap
  - e.g., at least 2 5-grams in common

# Linguistic Post-processing

---

- Prepare the data for searching:
  - Run a POS tagger over it
  - Clean the documents further, using POS tags
    - \* Where the POS tag distribution is unusual,  
... perform another round of anomalous document finding
    - \* 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, ...
- Index the document by word, POS and lemma

# Internet Corpora Summary

---

- The web can be used as a corpus
  - Direct access
    - \* Fast and convenient
    - \* Huge amounts of data
    - ⊗ unreliable counts
  - Web sample
    - \* Control over the sample
    - \* Some setup costs (semi-automated)
    - ⊗ Less data
- Richer data than a compiled corpus
  - ⊗ Less balanced, less markup

---

# Processing Raw Text

# Language Identification

---

- Given a document and a list of possible languages, in what language was the document written? (e.g. English, German, Japanese, Uyghur, ...)
- Language identification provides us with the means to automatically “discover” web data to convert into a corpus over which to learn linguistic (lexical) properties
- Main Approaches
  - Linguistically-grounded methods
    - \* Diacritics/Characters
    - \* Character  $n$ -grams
    - \* Stop words
  - Statistical Methods
  - Context (under `.jp` or `.ko`?)



# Normalization

---

- Extracting text from various documents
- Segmenting continuous text
- Number Normalization: *\$700K, \$700,000, 0.7 million dollars, ...*
- Date Normalization: *2000AD, 1421AH, Heisei 12, ...*
- Stripping stop words
- Lemmatization: *produces* → *produce*
- Stemming: *producer* → *produc*; *produces* → *produc*
- Decompounding: *zonnecel* → *zon cel*

---

# Relational Databases and SQL

## Many corpora are stored as DBs

---

- In relational databases
- Data is stored in **Tables** (or relations)
  - **attributes** are **columns**
  - **records** are **rows**
- Tables can be joined together
  - If they share a common column
- It can be retrieved with **Queries**
  - Properly written these can be very fast

**Structured Query Language** (SQL) is a special-purpose programming language designed for managing data held in a relational database management system (RDBMS)

## Example tables

---

TABLE: sent

sid	sent
1	Does this work?
2	I hope so.

TABLE: word

sid	wid	word	pos	lemma
1	1	Does	VBZ	do
1	2	this	DT	this
1	3	work	NN	work
1	4	?	.	?

## Select-From-Where

---

- The simplest query
  - SELECT desired attributes (columns)
  - FROM one or more tables
  - WHERE condition applies about the records in the tables
- What lemmas are there associated with the word *does*?

```
SELECT word, lemma
FROM word
WHERE word = 'does'
```

<b>word</b>	<b>lemma</b>
does	do
does	do
does	doe
...	

## How does it work?

---

- Begin with the table in the FROM clause.
- Apply the selection indicated by the WHERE clause.
- Select only those parts indicated by the SELECT clause.

## \* in SELECT

---

- When there is one relation in the FROM clause, \* in the SELECT clause stands for “all attributes of this relation.”
- What information is there associated with the word *does*?

```
SELECT *  
FROM word  
WHERE word = 'does'
```

sid	wid	word	pos	lemma ...
10	1	does	VBZ	do
11	7	does	VBZ	do
14	4	does	NNS	doe
...				

## You can rename things

---

- What information is there associated with the word *does*?

```
SELECT word as surface, lemma as indexform, pos
FROM word
WHERE word = 'does'
```

surface	indexform	pos
does	do	VBZ
does	do	VBZ
does	doe	NNS
...		



## You can sort things

---

- What information is there associated with the word *does*, sorted by POS?

```
SELECT word as surface, lemma as indexform, pos
FROM word
WHERE word = 'does'
ORDER BY pos ASC
```

surface	indexform	pos
does	do	VBZ
does	do	VBZ
does	doe	NNS
...		

- You can specify the order with DESC or ASC (default)
- You can order by multiple things: ORDER BY pos, LENGTH(word)

## And add new things

---

- What is the lemma, its length and pos associated with the word *does*?

```
SELECT lemma, length(lemma) as length, pos
FROM word
WHERE word = 'does'
```

lemma	length	pos
do	2	VBZ
do	2	VBZ
doe	3	NNS
...		

- strings: LENGTH(), LOWER(), UPPER(), TRIM(), LTRIM(), RTRIM(), SUBSTR(str, len), REPLACE(str, from, into)
- numbers: ROUND(num, digits=0), ABS(num)

# Task

---

- Download `eng.db` and `cmn.db`
- Start the sqlitebrowser
- Open *eng.db*
  - Find all surface forms of *leave*
  - Find all surface forms of *leave*, and show their length
  - Find all surface forms of *leave*, ordered from longest to shortest

## You can have complex conditions and limits

---

- Show 5 words that are not the same as their lemmas:

```
SELECT word, lemma, pos
FROM word
WHERE word != lemma
LIMIT 5
```

<b>word</b>	<b>lemma</b>	<b>pos</b>
does	do	VBZ
held	hold	VRN
does	does	NNS
Holmes	holmes	NNP
Holmes	holmes	NNP
...		

## You can even have simple regular expressions

---

- Show 5 words that include the string dog:

```
SELECT word, lemma, pos
FROM word
WHERE word GLOB '*dog*'
LIMIT 5
```

word	lemma	pos
dog-cart	dog-cart	NN
dog	dog	NN
dog-	dog	NN
dog-cart	dog-cart	NN
dogged	dog	VBD
...		

## Or slightly complicated ones

---

- Show 5 words starting with dog or Dog whose part of speech is not a noun

```
SELECT word, lemma, pos
FROM word
WHERE word GLOB '[dD]og*' and pos not GLOB 'N*'
LIMIT 5
```

<b>word</b>	<b>lemma</b>	<b>pos</b>
dogged	dog	VBN

\* matches anything

? matches one or one of anything

[] matches all characters listed (and allows ^ for negation and - for ranges)

## You can aggregate results

---

- Tell me more about the words

```
SELECT count(word), count(DISTINCT word),  
       MIN(LENGTH(word)),  
       MAX(LENGTH(word)), AVG(LENGTH(word))  
FROM word
```

<b>count</b>	<b>distinct</b>	<b>min</b>	<b>max</b>	<b>avg</b>
77820	9485	1	86	4.437

- MIN(num), AVG(num), MAX(num) are calculated over the whole set
- DISTINCT eliminates duplicate records thus fetches only unique records

## You can group things

---

- Tell me more about the words grouped into parts-of-speech

```
SELECT pos, word,  
       count(word), count(DISTINCT word),  
       MIN(LENGTH(word)),  
       MAX(LENGTH(word)), AVG(LENGTH(word))  
FROM word  
GROUP BY pos
```

pos	word	count	distinct	min	max	avg
CC	and	2264	13	2	7	2.98
DT	the	8256	38	1	7	2.74
EX	There	243	2	5	5	5.00
...						



## Task

---

- Which POS has the greatest difference between lemma and word length?
- Which preposition has a lemma not equal to its surface form?
- Find the number of words in each POS and sort from most to least frequent (for both English and Chinese)

## You can have a list in your condition

---

- Tell me more about common nouns

```
SELECT count(word), count(DISTINCT word),  
       MIN(LENGTH(word)),  
       MAX(LENGTH(word)), AVG(LENGTH(word))  
FROM word  
WHERE pos IN ('NN', 'NNS')
```

count	distinct	min	max	avg
14457	3593	1	21	6.74

- MIN(num), AVG(num), MAX(num) are calculated over the whole set
- DISTINCT eliminates duplicate records thus fetches only unique records

## This list can be the result of a select!

---

- Tell me more about frequent common nouns

```
SELECT count(word), count(DISTINCT word),  
       MIN(LENGTH(word)),  
       MAX(LENGTH(word)), AVG(LENGTH(word))  
FROM word  
WHERE word  
IN (SELECT word  
    FROM word  
    WHERE POS in ('NN', 'NNS')  
    GROUP BY word  
    ORDER BY COUNT(word) DESC  
    LIMIT 10)
```

---

<b>count</b>	<b>distinct</b>	<b>min</b>	<b>max</b>	<b>avg</b>
1096	10	3	10	5.13

- First find the ten most common words, then do things to them
  - the trick is to write simple queries first, and then combine them
- Note that more common words are shorter (as we would expect)

Now try to do some corpus queries yourself!

## You can Create Tables

---

You tell the database what the data will look like:

```
CREATE TABLE word (  
    sid INTEGER,  
    wid INTEGER,  
    word TEXT,  
    pos TEXT,  
    lemma TEXT,  
    cfrom INTEGER,  
    cto INTEGER,  
    comment TEXT,  
    username TEXT,  
    PRIMARY KEY (sid, wid),  
    FOREIGN KEY(sid) REFERENCES sent(sid)  
);
```

## You can Drop Tables

---

You tell the database what the data will look like:

```
DROP TABLE word;
```

But be careful, you can't get it back.

## You can Insert Data

---

```
INSERT INTO word (sid, wid, word, lemma, pos)
VALUES (1, 1, 'The', 'the', 'DT');
```

Normally you would add data using a program, or read it in from some other file, ...

## You can Update Data

---

```
UPDATE word SET lemma='a', word='A'  
WHERE sid=1,wid=1;
```

This changes the value permanently for all rows that match the condition.

```
UPDATE word SET pos='ART'  
WHERE pos = 'DT';
```

You can change a lot at a time.



## You can Delete Data

---

```
DELETE FROM word WHERE word='gannet';
```

But be careful, you can't get it back.

## You can Index Data

---

Indexes are special tables that the database can use to speed up data retrieval. An index is a pointer to data in a table, think of it as index in the back of a book. An index helps to speed up SELECT queries and WHERE clauses, but it slows down data input, with the UPDATE and the INSERT statements. Indexes can be created or dropped with no effect on the data.

```
CREATE INDEX word_word_idx ON word (word);  
CREATE INDEX word_lemma_idx ON word (lemma);
```

This makes it possible to look up words and lemmas very fast, but makes the database bigger. You normally add them after you have added the data. Whether they speed things up or not is an empirical question, and should be tested.

# SQL and NLP

---

- RDMS and SQL are the backbone of most data storage
- There are many implementations:
  - SQLite, PostgreSQL, ORACLE, MySQL, MS SQL Server, ...
  - typically share the same core
  - may have different extensions
- Text to SQL query is a popular task
  - *Which is the longest verb?*
    - ```
SELECT word, pos FROM word
WHERE POS GLOB 'V*' AND
LENGTH(word) =
(SELECT MAX(LENGTH(word)) FROM word
WHERE POS GLOB 'V*');
```