Text Statistics

M. Covington revised December 2008

Sources of information
Journal of Quantitative Linguistics
Literary and Linguistic Computing
Computers and the Humanities, now renamed *Lg. Resources and Evaluation*
Manning and Schütze, *Foundations of Statistical Natl. Lg. Processing* **(important book!)**
Manning et al., *Introduction to Information Retrieval* (useful)
Jackson and Moulinier, *Natural Language Processing for Online Apps.* (short, useful)

LIMITATIONS OF WHAT'S HERE

This handout describes shallow, statistical ways of looking at written text. This is not "natural language understanding." For instance, all techniques that work from word-frequency tables will fail to distinguish *dog bites man* from *man bites dog.* Do not be lulled into thinking that "powerful" statistical techniques or machine learning algorithms will be able to respond to information that has been withheld from them.

Some statistics that are easy to measure

Token count (number of words in text)
Type count (number of *different* words in text)
Type/token ratio (TTR): "A rose is a rose" has 5 tokens, 3 types, type/token ratio = 3/5 = 0.6.

Caution: Although widely used, *type/token ratio is a badly conceived statistic.*

A longer text *should* have more tokens than a short one, but *not* in proportion to the length. A 2000-word text does not have twice the vocabulary of a 1000-word text.

A way to compute a TTR independent of the text length is to take a **moving average** (Covington & McFall, work done in 2007). That is, find the TTR of:
- words 0 to 100
- words 1 to 101
- words 2 to 102

and so on, and at the end, average them.
This is called **moving-average type-token ratio (MATTR)**.

Advantages of MATTR:
- Not based on any statistical theory of how to adjust TTR for text length
- Truly independent of text length
- *Does* depend on window size, of course
- Can be computed quickly. You do not have to do a full count of everything every time the window moves along. You just have to note what has entered it and what has left it.
- Can be used to examine changes of style *within* a document, by plotting a graph of the moving averages.

What does type-token ratio tell us?

Things that can lower the TTR:
- Having a small vocabulary
- Repeating words and phrases a lot (stammering, perseveration)
- Sticking with a subject so that every idea is expressed more than once using similar words

Measuring the type-token ratio does not distinguish these!
Studies of the distance from each word to its next occurrence might be more revealing.

Richards, B. J. (1987). Type/Token Ratios: What do they Really Tell us? *Journal of Child Language* 14: 201-9.

Tweedie F.J., Baayen R.H. (1998) How variable May a Constant Be? Measures of Lexical Richness in Perspective. *Computers and the Humanities,* 32, pp. 323–352.

Words are not just things with a probability of occurring (like radioactive decay events). As Sir Anthony Kenny observes in *The Computation of Style*, the word *and* is frequent but you will almost never find *and and*.

Other things people measure:
Word length (average length; standard deviation; other descriptive stats.)
Sentence length (likewise)
How to you find the ends of sentences without parsing? A shortcut: Assume that you're at the end of a sentence when you hit !, ?, or . not followed by a digit or part of an abbreviation. This will lead to some mistakes, but they will be infrequent.

Word length and sentence length have an approximately **lognormal distribution**.
That is, you get a bell curve if you plot log(length) rather than length for the horizontal axis.
But we're not sure it's really lognormal rather than something else.

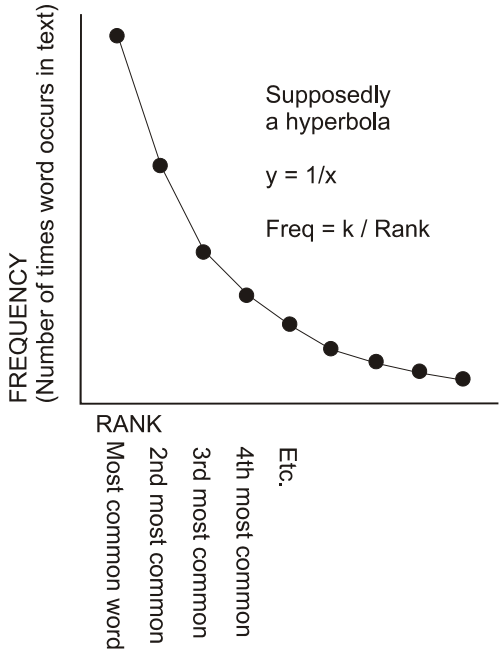
Word frequency tables are useful.
We need 2 kinds: alphabetical, and by frequency (most frequent first).

Alphabetical:

a	**50**
aardvark	**3**
abacus	**2**
acrid	**4**
etc.	

By frequency:

the	**105**
a	**88**
of	**76**
for	**50**
etc.	



Zipf's Law: Frequency of a word is (supposedly) inversely proportional to its rank
(e.g., the freq. of the 25th most common word is proportional to 1/25, etc.).
Zipf, G. *Human Behavior and the Principle of Least Effort* (1949).

Zipf's law is worth remembering, but ***I think it is has very little information content.***
–If the words are sorted by rank, then the graph *has* to slope downward to the right.
– If there is no limit to how rare a word can be, the graph *has* to be asymptotic on the right.
– Unlike a real hyperbola, a Zipf plot is *not* asymptotic at the top; the most common word has a finite frequency.

See also Wentian Li (1992) Random Texts Exhibit Zipf's-Law-Like Word Frequency Distribution, *IEEE Trans. Information Theory* 38:1842-45 (1992) for proof that **if you generate random "words" out of letters and spaces, they follow Zipf's Law.** So what does Zipf's Law prove? Simply that *we use words of all lengths, and the short ones are more common.*

The Porter Stemming Algorithm (a famous kluge)

For many purposes (vocabulary size determination, finding words in documents, etc.) you'll want to ignore inflectional morphology and some of the most common derivational morphology, so that (e.g.) *great, greater, greatness* all index the same.

Porter, M.F., 1980, An algorithm for suffix stripping, *Program*, **14**(3) :130-137.
Reprinted in Sparck Jones and Willet, 1997, *Readings in Information Retrieval.*

Web page: http://www.tartarus.org/~martin/PorterStemmer/
has the actual algorithm available in many prog. languages.
(Prolog was done by our own Philip Brooks. Want to do one and become famous?)

It removes suffixes ruthlessly, often with comical results:
The vastness of language surprises us ➔ *the vast of languag surpris us*
But it succeeds in its goal, which is to make related words look alike.

- ➔ **We could use a shallower (less ruthless) stemmer.**
- ➔ **Or we could use a lemmatizer, which has a dictionary built in and actually reduces each word to its dictionary form.**

Uses of text statistics

Author identification (stylometry): By studying habitual sentence length, vocab. size, presence of rare words, etc., you can distinguish different authors and in some cases identify the author of an anonymous text. Moving averages and cumulative sums can detect changes of style within a text, indicating inserted material.
Mosteller & Wallace, *The Federalist Papers* (Springer, 1984) is the classic study of this type.
Caution: Extravagant claims are often made. There are two or three workers in the field who often testify in court and claim a certainty that stylometry cannot provide.
Point of logic: Before you conclude 2 texts are by diff. authors, you must determine how much *one* author varies from one sample to another.

Readability indices: This is an old technology, but is still used with schoolbooks and the like. You can calculate the educational level supposedly required for reading the text.

"Fog Index"= 0.4 × (Avg. no. words per sentence + Percentage words over 2 syllables long)

When counting words over 2 syllables, do not include proper names, or words that exceed 2 syllables only because of an inflectional suffix (-ed, -es, -ing, etc.). Result is supposed to be a *grade level* (e.g., 9-12 for high school; 8 for easy adult reading; lower for children).

Fry readability index is based on number of words and of syllables in 100-word samples; involves a nonlinear function on a graph. See E. Fry, *Journal of Reading* 1968. Result is a grade level.

Flesch-Kincaid grade level index = (.39 × ASL) + (11.8 × ASW) – 15.59
where ASL = average sentence length in words; ASW = average syllables per word.

Flesch readability index = 206.835 – (1.015 × ASL) – (84.6 × ASW)
(This is *not* a grade level; higher is better; good writing should stay above 60.)

Microsoft Word computes both of these.

How do you count syllables *if you don't know how the word is pronounced?*
It's possible to construct a set of rules that nearly always get it right, by counting vowels or clusters of vowels surrounded by consonants. This would be an interesting project.

Syntactic studies: Frequency of certain words (e.g., subordinating conjunctions) reveals a lot about the author's habitual syntax.

Comparison of texts for information retrieval:

Two texts are **probably about the same subject** if they have similar word-frequency tables. This is a much better test than merely checking whether specific words are present in the text.

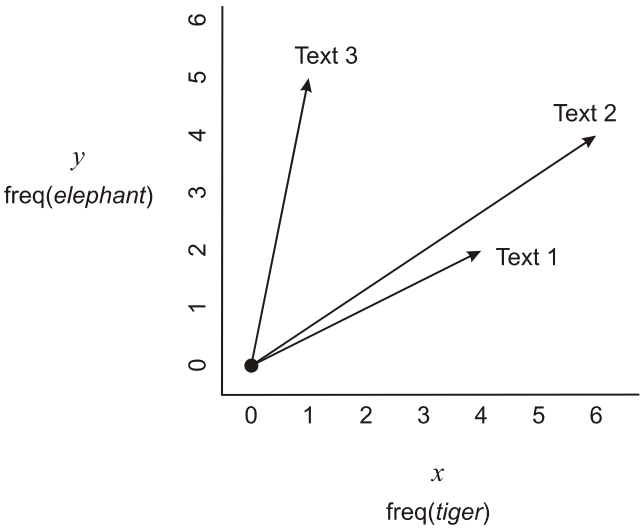
How do you compare word frequency tables? Here is a technique that uses **vectors**.

In mathematics, a vector is a *quantity plus a direction* (in 2 dimensions or more).
For example, a velocity is a speed plus a direction. Gravitational pull is a force plus a direction.

Suppose we're only comparing the frequency of 2 words, as follows:

	Text 1	Text 2	Text 3
Frequency of *tiger*	4	6	1
Frequency of *elephant*	2	4	5

Clearly, Texts 1 and 2 resemble each other more than text 3. How can we measure this?
Answer: Treat each *word frequency table* as a *vector* (list of numbers).
The freq. table of Text 1 is (4,2); that of Text 2, (6,4); that of Text 3, (1,5).
Each vector can be thought of as an arrow from point (0,0) to the specified point:



What matters is the *directions* of the vectors. (The *lengths* merely tell you the size of the text.)
Similar texts have vectors pointing in similar directions.

The *angle between two vectors* is easy to compute.
Let (a_1,a_2) and (b_1,b_2) be the two vectors and let θ be the angle. Then:

$$\cos\theta = \frac{a_1 b_1 + a_2 b_2}{\sqrt{{a_1}^2 + {a_2}^2}\sqrt{{b_1}^2 + {b_2}^2}}$$

You do not actually have to find θ – it is sufficient to know the cosine, because a high cosine (nearer 1.0) indicates a smaller angle and hence a closer match.

What if there are more than 2 words in the frequency table?
In the case, instead of 2-dimensional vectors, you have vectors in *n*-dimensional space (where *n* can be as large as you like); that is, you are comparing vectors that each contain *n* numbers. *The formula still works.* Here is how it generalizes:

$$\cos\theta = \frac{a_1 b_1 + a_2 b_2 + \cdots + a_n b_n}{\sqrt{{a_1}^2 + {a_2}^2 + \cdots + {a_n}^2}\sqrt{{b_1}^2 + {b_2}^2 + \cdots + {b_n}^2}}$$

This is the dot product of the two vectors (that is, the sum of the products of the corresponding entries in the table), divided by the lengths of both vectors.

It's quick to compute. No matter how many dimensions there are, you only have to take 2 square roots. The rest is just multiplication and addition.

An improvement on this: *tf*idf*

We actually don't want to give all the words equal importance in this process. The word *the* isn't good for distinguishing documents; the word *mastodon* is probably very good, since texts aren't likely to contain that word unless they are actually about mastodons.

Instead of building our vectors with **word counts (term frequencies)**, we should build them with **term frequency times logarithmic inverse document frequency.** That is:

each vector element =

no. of occurrences of this word in this document ×

log (total no. of documents / no. of documents containing this word)

Because log 1 = 0, the second factor goes to zero if a word occurs in all the documents.

Logarithms can be to any base (10, *e*, 2, or what you prefer) as long as you use the same base throughout the calculation.

Summary: *tf*idf* is vector comparison, not on raw word frequency tables, but on word frequency tables that have been weighted to emphasize words that are rarer across documents.

Latent semantic indexing (LSI) is a technique for reducing the amount of arithmetic and recognizing more subtle similarities.
"Official" LSI web site: http://www.cs.utk.edu/~lsi/
Key paper: Michael W. Berry, Susan T. Dumais, and Gavin W. O'Brien (1995)
Using linear algebra for intelligent information retrieval. *SIAM Review* 37:573-595.
(Available at the LSI web site.)

Key ideas:

(1) The set of word frequency tables (set of vectors) is a matrix.

(2) This matrix contains a *pattern of word frequencies.*
Is there a simpler set of numbers that preserves most of the pattern? Yes.

(3) Using *singular value decomposition,* this matrix can be decomposed into a three simpler matrices which, multiplied together, give the original matrix. One of the three matrices is *diagonal* (i.e., only has nonzero numbers on the main diagonal). The other two are rectangular but highly constrained.

Roughly: One rectangular matrix represents correlations or patterns between words; the other rect. matrix represents correlations or patterns between documents; the diagonal matrix tells how to mix them together.

(4) In the diagonal matrix, the biggest (most important) numbers come first. We can simply *throw away* the smaller numbers, replacing them with zero. We are then using only the first few rows of the first rect. matrix and the first few columns of the other. This is equivalent to throwing away the low-level "noise" in the original matrix while preserving most of the pattern.

(5) By searching the decomposition rather than the original matrix:

- We have a lot fewer numbers to work through.
- We can pick up *indirect similarities.* **Two documents will be recognized as similar to each other even if they don't share words, so long as they contain words which, *in the corpus as a whole,* tend to occur in the same documents.** These "latent" correlations between words are in the first rectangular matrix.
- We can ignore the second rectangular matrix, which describes the relative sizes of the documents.

This method is called "latent" because it deals with hidden patterns in word combinations rather than the word frequencies directly; "semantic" because it deals with words and the things they refer to.

LSA (LSI) was patented, not available for us to use freely. This patent has breathed new life into *tf*idf* and other techniques that are rivals of LSA. [Patent appears to have expired recently.]

Note that the abbreviation "LSA" also means "Linguistic Society of America," unrelated.