# Text Statistics Tool Box For Natural Language Processing

Cheng Hu
Artificial Intelligence Center
The University of Georgia
Athens, Georgia 30602, U.S.A.
http://www.ai.uga.edu/

2003 May 8

**Abstract**

Text statistics are frequently used in stylometry and cryptography studies. In this paper, some text statistics tools are developed in ISO Prolog for natural language processing. Details are given on the usage of 21 user-callable predicates. Logic and limitations of the program are also discussed.

## 1 Introduction

This paper describes the following text statistics tools written in ISO Prolog for natural language processing:

1. Average word length, standard deviation, skewness, and kurtosis;
2. Average sentence length, standard deviation, skewness, and kurtosis;
3. Token count (total number of words);
4. Type count(number of different words);
5. Type/token ratio;

6. Vocabulary richness;

7. Word frequencies;

8. First person form percentage;

9. Text similarity comparison;

10. Readability index.

Program inputs can be files, lists of tokens, lists of atoms, or lists of character lists while outputs are real values or lists. Statistics regarding sentence length and readability index are not valid for inputs of atom lists or lists of plain character lists, as there are no sentence marks in these two inputs.

## 2 Program logic

With the exception of text comparison predicates, all predicates must be run after the calling of their corresponding preprocessing predicates: stat_file/1, stat_tokens/1, stat_atoms/1, or stat_chars/1. The purpose is to avoid repeated computation. The preprocessing predicates call the tokenizer if the input is a file, preprocess a list of tokens, a list of atoms, or a list of character lists, and store relevant information. After that, other predicates do not need to compute the information again, instead just retrieving them from dynamically stored lists.

The vector space model is used in the comparison of text for information retrieval. Vector similarity is measured by the normalized correlation coefficient (Manning & Schütze, 1999).

## 3 User-callable predicates

The following predicates can be called by the user:

### go/0, help/0

Displays the usages of all predicates.

## stat_file(+FileName)

This predicate preprocesses a text file. First it calls the Efficient Tokenizer developed by Dr. Covington (Covington 2003), then transforms tokens into atoms and sorts them, calculates word frequencies, and stores corresponding information in dynamic lists.

```
?- stat_file(‘myfile.txt’).
```

## stat_tokens(+TokenList)

This predicate preprocesses a list of tokens. Tokens must be in the same format as the output of the Efficient Tokenizer. Covington (2003) has detailed description of the tokenizer. The predicate then transforms tokens into atoms and sorts them, calculates word frequencies, and stores corresponding information in dynamic lists.

```
?- stat_tokens([w([h,e]), w([h,a,s]), s('$'), n(['1','2']),
s('.'), ...]).
```

## stat_atoms(+AtomList)

This predicate preprocesses a list of atoms. It sorts the atom list, calculates word frequencies, and stores corresponding information in dynamic lists.

```
?- stat_atoms([input, is, prolog, atoms]).
```

## stat_chars(+TokenList)

This predicate preprocesses a list of plain character lists. It transforms character lists into prolog atoms, calculates word frequencies, and stores corresponding information in dynamic lists.

```
?- stat_chars([[h,e], [h,a,s], [a], [d,o,g]]).
```

## word_length(-WL, -SD, -Sk, -Ku)

Calculates the average word length, standard deviation, skewnewss, and kurtosis.

Skewness is a measure of symmetry, or more precisely, the lack of symmetry. A distribution or data set is symmetric if it is mirrored around the center point. The skewness for a normal distribution is zero and any symmetric data should also have a skewness near zero. Negative skewness values indicate data that are skewed left and positive values indicate data that are skewed right.

Kurtosis is a measure of whether the data are peaked or flat relative to a normal distribution. That is, data sets with high kurtosis tend to have a distinct peak near the mean, decline rather rapidly, and have heavy tails. Data sets with low kurtosis tend to have a flat top near the mean rather than a sharp peak. A uniform distribution would be the extreme case. The standard normal distribution has a kurtosis of zero. Positive kurtosis indicates a "peaked" distribution and negative kurtosis indicates a "flat" distribution.

```
?- word_length(WL, SD, Sk, Ku).
WL = 4.3125
SD = 2.57472
Sk = 0.731531
Ku = -0.596327
```

## sentence_length(-SL, -SD, -Sk, -Ku)

Calculates the average sentence length, standard deviation, skewness, and kurtosis. This predicate is only used after calling `stat_file` on a file or `stat_tokens` on a token list.

```
?- sentence_length(SL, SD, Sk, Ku).
SL = 19.4688
SD = 15.2611
Sk = 0
Ku = -2
```

## token_count(-TC)

Counts the total number of words in the text.

```
?- token_count(TC).
TC = 551
```

## type_count(-TC)

Counts the number of different words in the text.

```
?- type_count(T).
T = 274
```

## token_ratio(-R)

Calculates the type/token ratio.

```
?- token_ratio(TR).
TR = 0.497278
```

## richness(-R)

Calculates the vocabulary richness based on the formula in Těšitelová (1992).

```
?- richness(R).
R = 3.5
```

## word_freqA(-FA)

Prints word frequency table alphabetically.

```
?- word_freqA(FA).
Word Frequency Table in Alphabet Sequence:
a                    1
and                  3
cool                 2
does                 1
he                   1
..
FA = [[1, a], [3, and], [2, cool], [1, does], [1, he],
[...|...]|...]
```

## word_freqV(-FV)

Prints word frequency table by frequencies (high to low).

```
?- word_freqV(FV).
Word Frequency Table by frequency:
and                 3
cool                2
a                   2
does                1
he                  1
..
FV = [[3, and], [2, cool], [1, a], [1, does], [1, he],
[...|...]|...]
```

## freq(+Word, -Freq)

Finds the frequency of a word in the text.

```
?- freq(the, F).
F = 18
```

## first_person_percentage(-FP)

Calculates the percentage of first person forms.

```
?- first_person_percentage(FP).
FP = 18.5185
```

## read_index(-Index)

Calculates the Fog readability indices which are the educational levels supposedly required for reading the text. This predicate is only used after calling `stat_file` on a file or `stat_tokens` on a token list.

```
?- read_index(I).
I = 6.16296
```

### file_compare(+File1, +File2, -CosineTheta)

Compares the similarity of two text files by treating them as vectors in an $n$-dimensional space and computing $\cos\theta$, the cosine of the angle between the vectors.

```
?- file_compare('test1.txt', 'test2.txt', C).
C = 0.189092
```

### tokens_compare(+T1, +T2, -CosineTheta)

Compares the similarity of two lists of tokens by the value of $\cos\theta$.

```
?- tokens_compare([w([h,e]), w([h,a,s]), s('$'), n(['1','2']),
s('.')],[w([h,e]), w([h,a,s]), s('$'), n(['1','2']), s('.')], C).
C = 1
```

### atoms_compare(+A1, +A2, -CosineTheta)

Compares the similarity of two list of atoms by the value of $\cos\theta$.

```
?- atoms_compare([this, is , atom, comparison],[hello, world],C).
C = 0
```

### chars_compare(+C1, +C2, -CosineTheta)

Compares the similarity of two lists of plain character lists by the value of $\cos\theta$.

```
?- chars_compare([[t,h,i,s],[i,s],[a],[t,e,s,t]],
[[t,h,a,t],[i,s],[a],[d,o,g]],C).
C = 0.5
```

## 4 Limitations

To calculate the readability index, the auxiliary predicate `syl_x/2` counts the number of syllables in a character list based on English syllable and phonetics rules (Doyle, 2003; Kessler & Treiman, 1997). However, it cannot correctly count the syllables of a word that has silent vowels inside as it relies mainly on the consonant-vowel-consonant (CVC) syllable rule. For example, `syl_x/2` will count words like 'careful' and 'careless' and get three syllables, when in actuality they both only have two.

In order to eliminate the influence of inflectional suffixes that contain vowels (-ed, -es, -ing) on counting syllables, a simplified morphological analyzer is called to preprocess words. Without consulting the lexicon it will make mistakes in some rare cases. For example, the singular form of 'syllables' is 'syllable' which has three syllables. The morphological analyzer will remove the ending'-es', changing 'syllables' to 'syllabl' which would only have two syllables as counted by `syl_x/2`.

## References

Covington, M. (2003). ET: An efficient tokenizer in ISO Prolog. Available online at http://www.ai.uga.edu/mc/et/

Doyle, D. (2003). Phonics, syllable and accent rules. Available online at http://english.glendale.cc.ca.us/phonics.rules.html

Kessler, B., and Treiman, R. (1997). Syllable structure and the distribution of phonemes in English syllables. *Journal of memory and languages*, 37, 295-311.

Manning, C., and Schütze, H., (1999). *Foundations of statistical natural language processing*. Cambridge, Mass.: MIT Press.

Těšitelová, M. (1992). *Quantitative linguistics*. Amsterdam: Benjamins.