Simple syntactic tokens

Basic concepts

Characters, spaces, and *text*

All printable symbols of text processors are called characters.

Examples: Letters, digits, mathematical symbols, and subscripts are printable characters.

Spaces between characters and tabs are not printable.

Any finite sequence of characters and spaces is called a text.

Comment: You may think about a text as an unformatted text file.

Letter, *word,* and *alphabet*

These concepts are rather general. In a text processor, an alphabet is a set of characters called letters, and any finite sequence of these characters is called a word. But the same concepts are also used in describing human speech, different systems of animal communication, and the chemical structure of genomes of living organisms.

Syntactic and *semantic* properties

Syntax refers to the structure of a word, and semantics refers to its meaning.

Example: Consider the word W = mathematics.

1. The word W contains eleven letters.
2. The last letter of the word W is s.
3. The word W is an English word.
4. The word W is a noun.

Statements 1 and 2 describe syntactic properties of W, and statements 3 and 4 describe its semantic properties.

Extended alphabet

An alphabet that has n letters, where each letter has *two forms*, such as capital and lower case. And where every long word is partitioned into segments by a *break*, such as a *space*.

Comment: This is a concept we will use in the technical part of this paper. And it is based on the way that reading and writing are taught in many schools.

Simple words and cycles

Simple word: a word in which all letters are different
Cycle: a word in which all letters, except the first and the last, are different, and the first and last letters are the same

Examples

The words a, ab, abc, abcd are *simple*.
The words aa, aba, abca, abcda are *cycles*.

Notice the relation between *simple* words and *cycles*.
If you append to a *simple* word its first letter, it becomes a *cycle*.
If you remove the last letter from a *cycle,* you get a *simple* word.
So in any alphabet, the number of *simple* words of length r is equal to the number of *cycles* of length r+1.

Number of simple words

The number of simple words of length r in an alphabet having n letters is

$$nPr = n!/(n-r)!$$

r =	1	2	3	4	5	6
n						
1	1					
2	2	2				
3	3	6	6			
4	4	12	24	24		
5	5	20	60	120	120	
6`	6	30	120	360	720	720

10	10P5 = 3,024	10P10 = 3,628,800
20	20P10 = $6.7*10^{11}$	20P20 = $2.4*10^{18}$
	(20 is the number of different amino-acids)	
26	26P13 = $6.5*10^{16}$	26P26 = $4.0*10^{26}$
	(26 is the number of letters in the English alphabet)	
256	256P16 =$2.1*10^{38}$	256P32 =$1.5*10^{76}$
	(256 is the number of different bytes)	

Extended alphabets

Now we show how to divide all occurrences of letters in a word W into two categories, and how to insert breaks that partition W into segments that we call *simple syntactic tokens*.

All occurrences of letters in W are divided into two categories:
C: First and last letters of all cycles in W (*C* stands for Cyclic letter and is represented by lower case letters, so individual *C*-letters can be written as *c*)
S: All other letters (S stands for Single letter. They don't occur in pairs as C letters do. The S-letters are written as capital letters.)

Examples
W = alibabaandfortythieves

C CCCCC C C C C
alibabaandfortythieves
SS SSSSS S SS S S

aLIbabaaNDFORtYtHIeVeS

W = aquickbrownfoxjumpsoverthelazydog

C C C C C
aquickbrownfoxjumpsoverthelazydog

AQUICKBRoWNFoXJUMPSoVeRTHeLAZYDOG

Inserting a break _ between letters is more complex, and it has to be done from left to right. (Doing it from right to left provides a different result.)

Procedure:
1. Breaks are inserted only between letters of W, and not before or after W.
2. The word W is already partitioned into segments of *C*-letters and *S*-letters.
If a segment of *C*-letters is simple, insert a break after it and before the following segment of *S*-letters.
If a segment of *C*-letters is not simple, it contains at least one cycle.
Insert a break before the last letter of the first cycle.

Examples
Original word W = mathematics
In extended alphabet, W = matHEmatICS
Step by step procedure:
matHEmatICS => mat_HEmatICS => mat_HEmat_ICS

Original word W = tralalala
In extended alphabet, W = TRalalala
Step by step procedure:
TRalalala => TRal_alala => TRal_al_ala =>TRal_al_al_a
Comment
ala is a cycle. A break is inserted before the last letter of the cycle, al_a

Examples of *simple syntactic tokens*
mat, HEmat, ICS, TRa, al, a

Comment
So far we have shown how any word in any alphabet can be divided in one specific way into *simple syntactic tokens*, written in the *extended alphabet*. We know that there are other ways of doing it (for example, to enter breaks from right to left). But we will talk only about properties of the tokens described above.

Properties of tokens
Terminology:
C-letters *cyclic letters*, written as lower case letters
S-letters *single letters*, written as capitals

We divide tokens into three categories,
C-tokens tokens containing only *cyclic letters*
S-tokens tokens containing only *single letters*
SC-tokens tokens containing both *single* and *cyclic letters*

S-tokens occur only as *simple* words or at the ends of other words.
Examples
A, ABCD, DCBAE, and ICS at the end of mat_HEmat_ICS

Both *C*-tokens and SC-tokens can form long segments of a word W, but only C-tokens can have only one letter.
Example

GRoink_oink_oink_onk_onk_ok_ok_ok_o_o_o_o

SC-tokens, such as HEmat, GRoink and TRal, are the most complex, and they occur most often in any study of languages.
Example

a_LIba_ba_a_NDFORt_Yt_HIe_Ve_S

Look at each SC-token as having a root that consists of single letters and a suffix that consists of cyclic letters. But a cyclic letter never stands alone. So such a letter must also occur in one or two other tokens. We can look at these letters as links between tokens.

We call two tokens *connected* if each of them contains one of two *cyclic letters* of the same *cycle*.

a_LIba_ba_a_NDFORt_Yt_HIe_Ve_S
---------*-* *--* *---* connections between tokens

So this word consists of three groups of connected SC- and C-tokens, and one S-token at the end.

Examples
AQUICKBRo_WNFo_XJUMPSo_Ve_RTHe_LAZYDOG
-------------------* *-------* connections between tokens

GRoink_oink_oink_onk_onk_ok_ok_ok_o_o_o_o connections between tokens
****--****-****--***--***--**--**--**-*--*-*--*

Chains of tokens

Chain: a sequence of tokens such that every two consecutive tokens are connected by two ends of the same cycle.

Example

GRoink_oink_oink_onk_onk_ok_ok_ok_o_o_o_o
-----------*------*----*-----*---*---*---*--*-*--* chain connected by o

GRoink_oink_oink_onk_onk_ok_ok_ok
-----------*-----*-----*---*---*---* chain connected by k

GRoink_oink_oink_onk_onk
-----------*-----*-----* chain connected by n

GRoink_oink_oink
-----------* chain connected by i

Non-adjacent form of a language, Naf

In words with small alphabets, one-letter tokens occur so often that the difference between a sequence of letters and a sequence of tokens can be small.

But words in any language can be written in Naf form, in which any two consecutive letters are different.

How to write words in Naf form:

Extend the alphabet by a new letter, β. In every segment consisting of the same letter, x, replace every second letter by β. For example:

abcxxxxxxde => abcxβxβxβde

Tokens: ABCx_x_x_x_x_DE ABCxβ_xβ_xβ_DE

Example

A word in alphabet a, b, c: The sequence of tokens:

cbbacbbaabbcbbbcacbc Cb_b_ACb_ba_ab_b_Cb_b_bc_Ac_Bc

eleven tokens

The same word in Naf form:

cbβacbβaβbβcbβbcacbc cbβa_cbβa_βb_β_Cbβ_bc_Ac_Bc

eight tokens

Permutation of letters of an alphabet

Terminology

Pattern of letters: an abstract description of the word written already in the extended alphabet.

It is obtained by replacing any S-letter by capital letter *S.* And any C-letter by lower-case letter c.

Example

Word W in extended alphabet: mat_HEmat_ICS

Pattern of letters in W: *ccc_SSccc_SSS*

If a word U is obtained from W by a permutation of all letters of the alphabet (U is a cryptogram of W), then patterns of letters in W and U are the same.

Example
U =aquilabobons, and W = superstition, then both patterns are c_SSSSccc_cc_SS.

Intended applications

A statistical analysis of large samples of very long words is challenging. We think mainly about the primary structure of proteins, described as sequences of twenty amino-acids, and genomes of viruses and prokaryotes described in the four-letter alphabets of DNA and RNA.

One problem is to choose the unit of analysis. A statistic of single letters provides very little information, and looking at all sub-words is often not feasible, because even a one thousand letter word has a half million sub-words.

We think that using *simple syntactic tokens* as units for analysis would be a good choice in the case of proteins. But to be able to use *simple syntactic tokens* as the unit for study of DNA and RNA sequences, one may need to put these sequences in Naf form.

Final remark

We did not include any references because there are too many of them. Each concept used here has been used many times before, often under different names.

For more information, please contact Andrzej Ehrenfeucht, andrzej@cs.colorado.edu.