Computational lexicology, morphology and syntax

Laboratory #2 – Smart Tokenizer

Given a character sequence and a defined document unit, tokenization is the task of chopping it up into pieces, called *tokens*, at the same time throwing away certain characters, such as punctuation. Here is an example of tokenization:

Input: Paris is a nice city.

Output:

Paris

is

a

nice

city

.

These tokens are often loosely referred to as terms or words, but it is sometimes important to make a type/token distinction.

A *token* is an instance of a sequence of characters in some particular document that are grouped together as a useful semantic unit for processing.

A *type* is the class of all tokens containing the same character sequence.

A *term* is a (perhaps normalized) type that is included in a dictionary.

The major question of the tokenization phase is what are the correct tokens to use. In this example, it looks fairly trivial: you chop on whitespace and separate punctuation characters. This is a starting point, and your first assignment.

Task 1 (2 points):

Obtain some plain text data (e.g. visit a web-page and save it as plain text), and store it in a file 'corpus.txt', of about 2 pages. Then implement a tokenizer which splits the corpus on whitespace and separates punctuation characters.

Task 2 (4 points):

Communication and computer technologies have introduced new types of character sequences that a tokenizer should tokenize **as a single token**, including:

- email addresses: jblack@mail.yahoo.com
- web URLs: http://stuff.big.com/new/specials.html
- numeric IP addresses: 142.32.48.231
- and more.

Adjust your tokenizer to recognize only one token for emails, URLs etc.

Task 3 (3 points):

What about splitting “Mr. John is a doctor.”? The abbreviation tells us that the point should not be a split here, but come together with the preceding word: “Mr. John” in one token.

Adjust your tokenizer to treat abbreviations is this way.

Task4 (4 points):

Conceptually, splitting on white space can also split what should be regarded as a single token. This occurs most commonly with compound names (*San Francisco, Los Angeles*) but also with borrowed foreign phrases (*a priori*).

Other cases with internal spaces that we might wish to regard as a single token include phone numbers ((800) 234-2333. Other characters may also be considered for exceptions from splitting (in all cases?), such as for phone numbers: 0232 – 201090 or 0232/20-10-90 or 0232-20-10-90).

Adjust your tokenizer to treat also these cases.

Task 5 (5 points):

There are a number of tricky cases. For Romanian, the first example is the hyphen, such as the one in “s-a”, “i-am”, etc. It is though tricky because you shouldn’t merge all hypens, such as the hyphen with postposed clitic pronouns in imperatives and questions (e.g., *dați-i-l* should be 3 separate tokens: “dați”, „i” and „l”). However, you should keep one token for nouns merged as names (e.g. *Hewlett-Packard*).

Adjust your tokenizer to correctly treat the hyphens for Romanian.

Task 6: (2 points)

When word-processed documents are converted to plain text, the pieces are usually not recombined. It is easy to discover such texts by searching on the web for broken words, e.g. “depart- ment”. Handling hyphens automatically can thus be complex: it can either be done as a classification problem, or more commonly by some heuristic rules, such as checking the existence of the merged form in a dictionary.