Extracting measure units from files with medical texts and converting UK units to SI units when needed

Todor Arnaudov, 2007

By Todor Arnaudov at Research Group in Computational Linguistics, University of Wolverhampton

I solved the problem using Perl, because of its pattern-matching capabilities and becauase I have had done some programming with Perl an year ago. I had to learn and practice a lot, though, because I have used Perl for much simpler problems, and I have had used in practice only "toy" regular expressions.

A website with many example publications was given to me, that in the beginning I though will be the target texts: http://www.pubmedcentral.nih.gov/

My first goal was to study and practice Perl and regular expressions, and to explore the files, search for units and write code to recognize and convert them in SI, if needed.

Several questions arised soon, because there were a lot of units from biological sciences, like length of DNA ladder which has nothing to do with SI; and there weren't any non-SI measures like ft or in, so there was nothing to convert. Also, there were losses of information if we process the information as plain text that is copied from the browser or converted with simple program, where we face superscript, i.e. ^{...} tags and unicode characters.

The super scripts, e.g. 10 to minus 10-th, i.e. 10E-10 or 10^-10 changes to
10-10 in plain text. I have had ideas how to solve such ambiguities, but in some particular cases it was difficult to make decision automatically:

"Modelling the complexity of living beings should take into account the 10–12 order-of-magnitude span of timescales for events in biological systems, whether molecular (ion channel gating: 10-6 seconds), cellular (mitosis: 102-103 seconds), or physiological (cancer progression, ageing: 108 seconds)."

In HTML:

10-12 is ----> 10-12 (between 10 and 12)
10-6 is ------> 10E-6
102-103 ----> 10E+2 - 10E+3 (between ...)
108 is ----> 10E+8

I had ideas about guesses, e.g.. \s10-?\d+\d?\s could be supposed to be an exponent. Another option was just to work on the html and parse it for <sup> tags, but these cases were not very frequent and I've been told later that the target files will be different.

After I have created a working script for the example files that I have investigated -- with extracted uints and a file, and with concordance, -- I was given an example of a real XML-file with information that my script actually will be supposed to process. The task changed a bit, since a lot new measure units that were missing from the first examples appeared.

I had to investigate the file and excerpt the values by hand. I was also given a format for the output, which forced me to rewrite the style of the regular expressions in order to make the code simpler, i.e. I used grouping and generally improved the regexes. The format of the output was as follows:

<FILE NAME> <tab> <NUMBER> <tab> <UNIT> (followed by optional fields
<tab> <CONVERSION CO-EFFICIENT> <tab> <CONVERTED VALUE> <TARGET UNIT>).

Where <FILE NAME> was a tag in the XML.

The filename ws another new small task I solved like this: the script scans the file with appropriate regex and store in two arrays. The first one - @filePos - is for positions of the "files" in the real input file, which is read as a whole into memory and is represented as a string. The second array - @fileName - contains the names of the "files".

When the program matches a pattern of a measure unit, it searches through the array of positions for the first "file" which is in a position before the match. Here I tried to implement a binary search, but it got messed somewhere in some cases where the positions is very near the center of the string. I didn't manage to debug it quickly at the time and I switched to a simple "half-linear" search, that just searches to the left or to the right, starting from center. I've left in the code variants of subroutines which are taking into account that we pass through file linearly in the cycle of pattern-matching, thus we can use the last found position as a start of the search.

To make the concordance look better, I cleared up the most of XML tags - except the tags for filenames - and did substitute new lines and multispaces to one space. It is possible to be done if it's

critical, but it makes the script more complicated. My possible solution would be reading what's between the tags in a string array, then performing pattern matching through the strings in that array. In current structure of the script, we're passing through one string. Another solution for clearance of file tags is to substitute them directly from the file-string, but that will require experiments about how the positions found in the scan for filenames change after the substitutions.

Also, now there were several units to convert, although in the most of the cases the measures have appeared in both UK and SI formats - the second one in brackets.

My method for matching is with a two-dimensional array where we store a regex pattern, lenght of the context for extraction back and forth, name of the class of units recognized by the pattern, and a flag about does the unit require conversion. There's an associative array for conversions, with key - the unit name and value - conversion coefficient.

Actually, cases for conversion need also a special processing after matching.

1. Compound units; two types of "oz-es".

There were two compound units that required conversions: (ft in) and (lb oz), and they are postprocessed due to their different structure from the simple units. <value> <unit> <value < uint > with two conversion coefficients, instead of <value> <unit>' or <unit><value> in some rare cases, like for pH.

There was one ambiguilty: oz for weight - converted to kg, - and oz for capacity - converted to litres. The clue used for differenciation was that there shouldn't be an expression with lb and oz together for capacity, because lb is a sign only for weight. However, that pattern may classify incorrectly items if there are weights that are expressed just in weight oz.

Succesful examples:

MAE1612 6.0 lb oz lb*0.4535924+oz*0.02834952 2.7215544 kg ; He has had a 2.7-kg (6-lb) weight loss during this
MAE1638 6.13 lb oz lb*0.4535924+oz*0.02834952 3.09009816 kg aginal delivery of a 3090-g (6-lb 13-oz) newborn; .;
MAC3401 8 capacity oz 0.0338140227 0.2705121816 l .; He drinks four to five 8-oz bottles of milk daily;

2. Unknown units

There were tokens like the following (N=\d\d?\s\d\d?, i.e. N=5 9; N=1 10)

MAB5714	N=5 8	1		N=5 8	; - xylose 2.1 g/5 h (N=5 8); Schill	
MAB8055	N=1 10	1		N=1 10	wedge pressure of 23 mm Hg (N=1 10) , and markedly increa	
MAB8240	N=21 36	1		N=21 36	n time (activated) 30 sec (N=21 36);	

3. Others:

Units which begin with a word (not a number):

MAB6274	7.42	pH	1	7.42	pH	on room air :; pH 7.42; P 60 m
MAE1626	7.30	pH	1	7.30	pH	m air shows :; pH 7.30; P 52 m

They required simple post-processing.

These are not units, but I included them in the output as well:

MAC1575	50	th	1	50	th	.; Her length is at the 50th percentile for age , and
MAC1575	75	th	1	75	th	or age , and weight is at the 75th percentile .; Exami

...

Future improvements

The script does what it's supposed to do, but it could be improved in the following directions:

- More units.
- Faster search for files.
- Cleared tags of filenames in the concordance.
- Config-files with patterns for matching and conversion for easy addition of new units.

(Ambitious)

Calling the script

perl units.pl [inputFile] [contextBack] [contextForwardt]

Where context is number of characters back and forth, printed in the concordance.
If an input file is not given, the program tries to process file named sample100.xml.
If any of the contextNumbers is not given, a default value is used; if only one number is given - it's used for both directions.

Todor Arnaudov, 22.03.2007