Nov. 6, 20

Project #1: n-gram Algorithm Implementation in Intel x86 Assembly

Istanbul Technical University,
Faculty of Computer and Informatics
BLG413E- System Programming
2020-2021 Fall

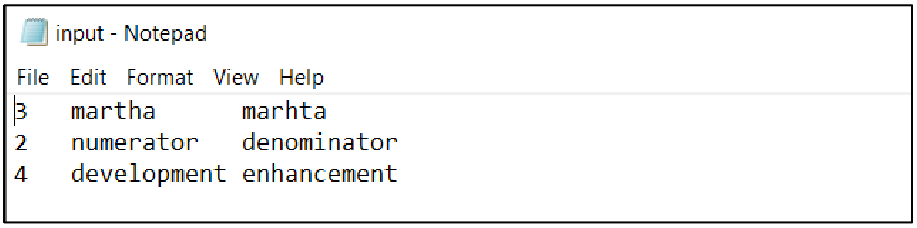
Important notes:

In this homework you will work with NASM on a 32-bit Linux, same as what you have used in class and recitations. Projects, which use another assembler or incompatible with 32-bit Linux platform will not be evaluated.

Description of Assignment

In this project, you need to implement n-gram algorithm using Intel assembly. N-gram defines a contagious sequence of *n* items (characters or samples) from any given information. It is commonly used in natural language processing (NLP) to derive probabilistic models.

In your project, your C program will use an input text file, from which you will read the value of *n* and two strings. This input file consists of three columns as shown in the following example. You can assume that the input file will contain at least 20 lines.



```
3   martha      marhta
2   numerator   denominator
4   development enhancement
```

Your main C file should read the input file, call the **n-gram** function written in Intel assembly for each line, and get the corresponding similarity value as the function output. Details of the n-gram algorithm, which you need to implement in assembly are as follows:

1. Your n-gram function will take two strings (might have different lengths) and value of *n* as its main inputs parameters and will measure the similarty of the given strings. The prototype of the function is given below:

```
int n_gram(char* str_1, int size_1, char* str_2, int size_2, int n);
```

where *size_1* and *size_2* specify the length of input strings 1 and 2.

2. The function will first determine the sequence sets of given string, listing the n-grams they contain. Let us use S' and S'' to specify sets beloning to the first and second input strings respectively.

3. Then, the similarity parameter will be determined with the following equation:

$$similarity = \frac{|S' \cap S''|}{|S' \cup S''|}$$

4. Lastly, your function will use the similarity parameters calculated as its return value and conclude its execution. It is important that you notice the return parameter of the function is declared as an integer. So, rather than utilizing floating points in your assembly code, you should simply return the similarty results in percentage as an integer.

An example[1] is given as follows:

Assume the inputs strings are “martha” and “marhta” and the given n is equal to 3.

3-grams (trigrams) set for the first string: $S' = \{\, mar, art, rth, tha \,\}$
3-grams (trigrams) set for the first string: $S'' = \{\, mar, arh, rht, hta \,\}$

Afterwards, we can derive $S' \cap S'' = \{mar\}$ and $S' \cup S'' = \{\, mar, art, rth, tha, arh, rht, hta \,\}$

Leading to a similarity parameter of $\mathbf{1/7} \; = \; \mathbf{14}\%$

Submission Details

- You are required to implement the given function in Intel assembly and main body of the program in C. The main program will handle all file read and standard I/O operations, so you do not need to read or write anything from the assembly.

- You should not write the requested n-gram codes in higher-level language (like C, C++) and then, transform it into an assembly code for your project submission. Accordingly, the codes generated by compiler won’t be accepted and this will be considered as cheating.

- Every group member is required to submit source code file(s) through the Ninova system as a zip file.

- Any form of cheating or plagiarism will not be tolerated. This includes actions such as, but not limited to, submitting the work of others as one's own (even if in part and even with modifications) and copy/pasting from other resources (even when attributed). Serious offenses will be reported to the administration for disciplinary measures.

[1] https://medium.com/@appaloosastore/string-similarity-algorithms-compared-3f7b4d12f0ff, retrieved on Nov. 2020.