Programming Assignment 6 (due Tuesday 30 April 2019)

For PA 6, you should build upon the QA system you started in PA 5. Your PA 5 system performed QA by re-expressing questions as "answer patterns" that could be searched for in order to find a direct match that would answer the question. In PA 6 you will extend this so your QA system does not have to rely on finding exact matches. Simply turning in a program that only has PA 5 functionality for PA 6 will result in no credit.

In PA 6 you should enhance the functionality of your QA system in three ways :

1. Your system should have at least one enhancements to query reformulation,
2. at least one enhancements to answer composition, and
3. Your system should decide between multiple candidate answers based on a confidence score you compute.

Details on each of these follow:

**(1)    You should have a minimum of two enhancements to query reformulation.**

In PA 5 your query reformulation created answer patterns that would locate exact matches to question answers (at least in some cases). In PA 6 you should generalize query reformulation so that it does not simply rely on answer patterns, and does more general searches when answer patterns do not succeed. You should take a back off strategy and search for fewer and fewer words in the question until you find matches.

You should also support query expansion, which will add words to your query that weren't found in the original question. For example, you could consider expanding the words in your query using x, or you could conduct a search of Wikipedia to find related terms that could be used in a query. These are just a few possibilities.

**(2)    You must also have a minimum of two enhancements to answer composition.**

Your program should not simply rely on finding an answer expressed in complete form in a search result (as you did with PA 5). Instead, your program should be able to ccombine smaller pieces of information into a single answer when no such exact answer is available (ie tiling). You should certainly take advantage of complete answers when you find them in the search results, but you should not assume that you will always find these for all questions.

**(3)    You should also have a method for assigning a confidence score to each of your possible candidate answers.**

The scores you assign should be shown in your log file, and your system should always report the answer with the highest score. In general your confidence scores should be tied to your query reformulation and pattern matching, where an exact match of a pattern should have a high confidence score, whereas a partial match that requires tiling to form an answer should have a much lower score. Your confidence scores should be real numbers in the range 0-1, where a score of 1.00 means your system is certain the answer is correct, while scores of 0.00 would mean there is no possibility the answer is correct. While these do not need to be computed as probabilities, you may wish to interpret your confidence scores as probabilities (where 0.80 means there is an 80% chance of being correct).

**These enhancements should be clearly explained in your introductory comments and clearly labeled as Enhancement 1 to Query Reformulation and  Enhancement 1 to Answer Composition (at a minimum, you may have more than these two enhancements if you like). You should also explain your scoring method in your introductory comments in a section labeled Confidence Scoring of Answers.**

You should use a Perl module from http://search.cpan.org to interact with Wikipedia. One possibility is WWW::Wikipedia, which we went over in class. If you would like to use other modules, please let me know although all modules that you use must be available from CPAN.

Your program should run interactively, and prompt the user for questions until the user says "exit". Then your program starts it should output a message with your name and a short description of what the system does (see below). Then, a user should enter their question and the system should respond with an answer that is a complete sentence, and should not contain any information beyond that which is asked by the question. Note that this is not a conversational agent, so your system does not need to remember previous questions or answers, nor does it need to engage in "small talk". You may assume that your user will cooperate and only ask well-formed questions.

You should also keep a log file that records the users question, the searches you actually execute, the raw results you obtain from Wikipedia, the confidence scores associated with your possible answers, and finally the answer you generate for the user. You may also include any additional information in your log file that you might think would be helpful in debugging your program. The name of this log file should be given on the command line.

perl qa-system.pl mylogfile.txt

\*\*\* This is a QA system by Bridget McInnes. It will try to answer questions that start with Who, What, When or Where. Enter "exit" to leave the program.

=?> When was George Washington born? (user)

=> George Washington was born on February 22, 1732. (system)

=?> What is a bicycle?

=> A bicycle has two wheels and is used for transportation.

=?> Where is Duluth, Minnesota?

=>I am sorry, I don't know the answer.

=?> exit

Thank you! Goodbye.

You may assume that the questions are well formed and grammatical, and that they are in fact questions. You may assume that all questions start with either Who, What, When or Where. Your system should recognize when it can't answer a question, and say something like "I am sorry I don't know the answer."  This is preferable to giving no answer or providing gibberish.

Your answers must be complete sentences. "George Washington was born on February 22, 1732" is an example of an acceptable response. "February 22, 1732" or "Your answer is February 22, 1732" would not be acceptable. Your answer must not contain any information beyond that which is asked for in the question, so "George Washington, the First President of the United States, was born on February 22, 1732" would not be acceptable (since we only asked for his birthday, not his most notable achievement or position). Each question should be assumed to be independent and separate from any other question asked (thus, your system will not be expected to make use of previous questions or answers when processing a question).

In addition to your program, please submit a plain text file called qa-samples.txt which includes 28 factual questions - seven each of the Who, What, When, and Where questions. You may re-use your questions from PA 5, although please makes ure you provide the answer given by your system after it has been modified for PA 6. You must also include answers that you have verified are correct, and the answers as given by your system. Include 14 examples where the system answer is correct, and 14 where the system answer is incorrect. The format you should use is as follows:

Question : When was George Washington born?

Correct Answer : George Washington was born on February 22, 1732.

My System Answer : George Washington was born to a poor family. (incorrect)

After you have listed your 28 questions (with correct and system answers) you should provide a single example that shows how your system generates an answer that is not found exactly as written in the Wikipedia search material. You should provide a step by step accounting of how your method handles this, and show what the search terms were, what candidate answers were considered and how they were scored, and the final answer of your system.

Note that I will use these examples to formulate my own questions to evaluate your systems. So if one of your examples that your system answers correctly is "Who was George Washington?" I will likely ask a related question like "Who was Thomas Jefferson" since the questions and answers are fairly similar (and it seems fair to expect your system to get a similar question with similar answer correct.) If your system can answer other types of questions, please show those too.

Please submit your program and all additional files to blackboard including your source code ( qa-system.(pl|py) ) and your samples file as described above (qa-samples.txt). I will run your program on a few examples based on your samples.