# Homework #8: Question Answering

**Deadline: 11:59pm Monday 9 December 2013**

*Please present a report with all your answers, explanations, and sample images or plots. Submit also a soft copy of the source code and binaries used to generate these results. Please note that copying of any results or source code will result in ZERO credit for the whole homework.*

**Acknowledgment:** This homework is adapted from Chris Manning and Dan Jurafsky's Coursera NLP class from 2012.

*Can you make a system to compete with IBM's Jeopardy-playing "Watson"?* This assignment contains two parts, the first making use of Wikipedia data, and the second marking use of Google query results.

## *Part 1: Question Answering with Wikipedia*

Your job in Part 1 is to perform SPOUSE-OF(x, y) question answering via *relation extraction* from Wikipedia.

You must use both of the following two methods:

1. Structured information from Wikipedia Infoboxes
2. String text patterns such as those of Hearst (1992) described in lectures

### Overview

Start by taking a look at the data, located in `data/small-wiki.xml`. The Infobox for Arnold Schwarzenegger tells you that he is married to Maria Shriver:

```
{{Infobox Governor
|name          = Arnold Schwarzenegger
...
|spouse        = {{nowrap|[[Maria Shriver]] (1986&ndash;present)}}
```

You should process this structured Infobox data (however you like; regular expressions are fine) to pull out these "spouse" relationships. For this portion of the assignment, you should **only** be using the Infoboxes and no other information from the remainder of the article.

Next, use patterns like those of Hearst (1992) described in lecture to find example patterns like **X is married to Y** that could help you extract the spouse relation from text data. For example, we could extract Ingrid Selberg and Mustapha Matura from this sentence:

```
Ingrid Selberg is married to playwright [[Mustapha Matura]].
```

Feel free to also use other properties of the document (such as the title of articles for instance) to help you with this part. However, you may not use any external information for this section (such has having seed sets of relationships).

For each of these two methods (the method using the Infoboxes and the method using the text data) you need to output your guess of the spouse for each of the people in the input file `data/wives.txt`. Note that the wife names may appear in a slightly different format in the `small-wiki.xml` file than they do in the `wives.txt` file, and thus you might need to think of creative methods for associating wives in the two files.

We will follow the Jeopardy format of returning a '**question**' for each input '**answer**'. For example, the first line of the input file wives.txt is "Maria Shriver", so the first line of your output should be:

```
Who is Arnold Schwarzenegger?
```

We have given you the correct output for the 10 sample wives in `gold.txt`. You will notice that most of the entries are divided by a "|" character, which seperates each of the valid answer choices. This is because some wives have multiple husbands or a particular husband's name is mentioned in different forms in the `small-wiki.xml` file. As you will notice in the `gold.txt` file, all valid answers include the first and last names of a husband and are written using only ASCII characters. You may thus return any form of a husband's name which appears in the `small-wiki.xml` file and follows those parameters. We have provided starter code in `python/Wiki.py`.

### Implementation Details

All your edits should be made in the method `processFile` which takes in the Wikipedia file along with a list of wives and a `useInfoBox` parameter. You will need to implement both methods of extracting relations in the `processFile` methods. Thus, when `useInfoBox` is `True`, the code should find the relations using Infoboxes, and when `useInfoBox` is `False`, your code should find the relations without the use of Infoboxes.

### Evaluation

Your question answering system will be given a score between -20 and 18 (as there are two articles without an Infobox in the training set). Each wrong guess will receive -1 points, a correct guess will receive +1 points and outputting 'No Answer' for any wife will receive 0 points. There will be 10 wives in the training set and you will have to find the corresponding husband once using Infoboxes and once without.

## *Part 2: Question Answering with Google*

### Overview

If you were looking for the answer to a factoid question, you might very well Google the question and look for the answer in the snippets. In this assignment you'll be attempting to automate that notion by extracting answers from the Google snippets. You will be responding to questions of the form, "**Where is the location of x?**", where x is a famous structure or landmark. In the assignment, we give you 10 values of x as a training set, which you can use to train your answer extractor.

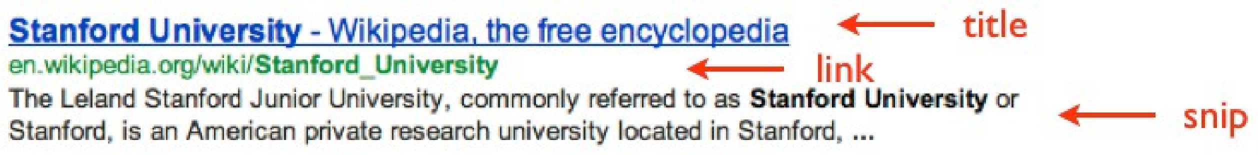
## Data

Data files can be found in the `data` folder, where you will find `landmarks.txt`, `googleResults_tagged.txt`.

- **landmarks.txt** contains sample query landmarks and their corresponding city/country answer. An example line is

  ```
  1\tstatue of liberty\tliberty island/new york harbor/new york city, new york
  ```

  This is a tab separated file where the **first column** represents the number for the landmark, the **second column** represents the landmark's name, and the **third column** represents the landmark's location. In the third column, the portion that comes before the comma represents the city of the landmark, and the portion after the comma represents the country or the state of the landmark, where a state is given if the landmark is located in the United States. Note that some times multiple cities/possible responses are given where each acceptable response is separated by a slash.
- **googleResults_tagged.txt** file contains the Google query results for Google queries of the form, "Where is the location of x?". The file contains a total of 10 queries for the landmarks listed in the landmarks.txt file and each query contains a list of the top 10 results returned by Google, with the exception of the query results for Saint Basil's Cathedral, which contain only 9 results. The results for each query is separated by an empty line and each result is clustered into a set of three lines. The first line contains the title of the result, the second line contains a snippet of text from the website, and the third line contains the link to the webpage of the result. See the illustration below for an overview of the meaning of each line:


Stanford University - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/Stanford_University
The Leland Stanford Junior University, commonly referred to as **Stanford University** or Stanford, is an American private research university located in Stanford, ...

  The text from the Google results were also run through the Stanford NER (Named Entity Recognition) tagger, which outputs three possible tags for each item: ORGANIZATION, LOCATION, and PERSON. The tags themselves have not been modified in any way, and thus are not guaranteed to be correct, however, you will likely still find them helpful. Finally, some phrases will also be marked with the `em` tag, which shows words that Google bolded in their results.

## Implementation Details

In this assignment, you will be modifying the method `guessLocation()`, located in `python/Googling.py`. The guessLocation() method takes a list of `GoogleQuery` objects for a specific query, where each GoogleQuery object encapsulates the information provided by a result in Google. Each Google query object contains 3 String fields: the title, snip, and link for that specific results. Using the information from the GoogleQuery objects, you are to return a `Location` object, which contains two fields, a city and country, where the country field represents a state if the landmark is located in the United States. Currently the starter code returns empty strings for both fields, which denotes "No Guess". In this part you will **not** have to return your queries in the form of a Jeopardy response. Note that the code only requires modifying the `guessLocation()` method (and adding any necessary helper methods). You can also optionally edit the `processQueries()` method or add any necessary global/class variables and import statements. Editing the `processQueries()` method, for instance, will allow you to use external information (such as gazatteers), which is only allowed for this part.

After running the code, the program will output a group of tab separated categories. The first column in each category represents the landmark for the guess, the second column represents the guess your program outputted for the **city** in which the landmark is located, the third column represents the acceptable list of cities where the landmark is located (or in some cases, regions), and fourth column represents the guess your program outputted for the **country** for in which the landmark is located, the fifth column represents the country or state in which the landmark is located. The CORRECT GUESSES section lists landmarks for which you correctly guessed both the city and country/state of the landmark, while the NO GUESSES and INCORRECT GUESSES portion lists landmarks for which either the city or country/state guess was incorrect or was not guessed at all. (It might just be easiest to run the code and look at the output instead of processing all of that).

**Evaluation**

In total, there are 10 landmarks for which 10 cities and 10 countries/states must be guessed in the training data. Correct guesses receive a score of +1, incorrect guesses receive a score of -1, and no guesses receive a score of 0. Thus, your program will output a score between -20 to 20 depending on how many of your guesses were correct.

## *Requirements*

You are required to implement part 1 (extracting spouse information) and part 2 (extracting location information for landmarks).

Please submit your code and report in one zip file, named **CMPN463.HW03.firstname.lastname.zip**. For example, if your name is Mohamed Aly, your file should be named **CMPN463.HW03.Mohamed.Aly.zip**.

## *Grading*

1 pts: report and submission file name
3 pts: correct implementation of part 1 with infoboxes
3 pts: correct implementation of part 1 without infoboxes
3 pts: correct implementation of part 2