Project Description: A simple web interface that allows a user to submit text, either a file or entered into a form, the application counts word frequencies and automatically generates a PyConText config file. Must be compliant with an apache license.

Requirements:  
  
1. An open source license.  
2. A new class defined and an derived class that inherits from this parent class. The classes can be simple, but you should have at least a useful ``\_\_str\_\_`` method.  
3. Data permanence using a database (e.g. SQLite, MySQL, or MongoDB) or some form of data serialization (e.g. JSON).  
4. A user interface. This could be a web application (such as Flask or Django); an application built with pyQT4 or wxWindows; an Jupyter notebook with widgets; or a command line interface.  
5. Some sort of visualization. This can be a static matplotlib-generated figure or an interactive plot with Bokeh or mpld3.  
6. A detailed README.md that describes the problem your project is addressing including any domain knowledge that will help someone (me) outside of your expertise appreciate the intent of the project, provides a description of the dependencies (any install instructions), instructions on how to use the project with example use cases.  
7. a separate git (e.g. Bitbucket or GitHub) repository for the project.

Deliverables:

Client side:

1. Web interface that allows the user to upload a text file, or enter text in a text box.
   1. Error checking – if its not a text file, alert the user with an error to submit text.
   2. Displays list of common words that will be removed automatically. Check boxes allow user to deselect. “Add new” allows users to add new words to common dictionary.
2. Web results page that has sliders to allow the user to define the cutoff point for results displayed using mpld3.
3. Allows user to select most frequent words (like a, the, but etc) and automatically filter results based on a stored list of common words.
4. Once the user selects the filter, the form is told to save the results file, using pickle.

Server side:

1. Receives free-form or uploaded file from client.
2. Writes to disk
3. Processes the file to determine word frequencies
4. Returns work frequency results for display and filtering
5. Once filters are applied, write the PyConText config file and allow user to download it as text.

Example of PyConText config file:

Comments: 4/29/2013

Direction: forward

Lex: atypical for

Regex: ''

Type: PROBABLE\_NEGATED\_EXISTENCE

---

Comments: ''

Direction: forward

Lex: be ruled out for

Regex: ''

Type: INDICATION

---

Comments: ''

Direction: forward

Lex: can be ruled out for

Regex: ''

Type: DEFINITE\_NEGATED\_EXISTENCE

---

Comments: ''

Direction: forward

Lex: can rule the patient out

Regex: can rule (her|him|the patient) out

Type: DEFINITE\_NEGATED\_EXISTENCE

---