A fast and accurate Command Line Suggstion feature

An introduction

My legal name is *P Sudeepam*, and I am a student of *Jaypee Institute of Information Technology,* Noida, India. I wish to add a *fast and accurate Command Line suggestion feature* to GNU Octave for Google Summer of Code, 2018. Please find my public application, current contributions to GNU Octave, and all other related biographical information here: https://wiki.octave.org/User:Sudeepam.

- **An idea of the time-frame I plan to be**
  - **working on my project**
    By default, it would be, 4:30pm - 9:30pm (UTC+0) and 6:30am - 11:30am (UTC+0) but honestly, my work schedule is flexible and I could work at any time of the day. Whatever fits the mentor best would in-fact be the best time.
  - **available on the IRC channel**
    By default, I'll be available on the IRC channel whenever I am working so the time-frame would either be, 4:30pm - 9:30pm (UTC+0) and 6:30am - 11:30am (UTC+0), or the time-frame which suits the mentor best. Again, since my schedule is flexible, I can be available on the IRC at any other time of the day also if required.

- **Other commitments for the summer period.**
  I'll have my Final exams during the first week of GSoC, i.e. between 14th May and 21st May. Other than that, I do not have any other commitments for the summer period.

- **A description of my written English.**
  My written English is good. I will be able to properly communicate and write reports in English.

Contact

- My Legal name is '*P Sudeepam*'
  My IRC channel nickname is '*peesu*'.
  My name on the Octave Wiki page is '*Sudeepam*'.
  The email address I check most regularly is <here, in the final proposal>

- **The time zone *(UTC+-x)* and country I live in**
  I live in India, UTC + 5:30 timezone. This will not change over the duration of GsoC.

- *<I'll add my phone number and Whatsapp contact here in the final proposal>*

Self-assessment

- I do like to give and receive constructive advice.
- I believe I am good at sorting useful criticisms from useless ones

My task

- **Why did I choose my particular task? What do I expect to gain from working on it?**
  I have chosen to work on the **command line suggestion feature** (savannah.gnu.org/bugs/?46881). My aim is to make something which would be very similar to what the 'Did you mean: ....' feature of the Google search does, i.e. suggest a correction to the user whenever she/he makes a typographic error while writing something. For the case of GNU Octave, this 'something' would be the inbuilt functions of Octave.

  I was able to think of this feature as a complex decision making problem, which directly implies that I could use Machine Learning, particularly, Neural Networks, to approach this problem. Machine learning is something that really gets me excited. A summer where I could spend most of my time working on a Machine Learning problem, under the guidance of a dedicated mentor, for an organization whose software has become an integrated part of my academic life, would be an ideal summer for me. This is the reason why I choose this particular task.

  Also, this project, promises to improve the user experience of Octave. Therefore, not only will it be beneficial for the current users, it could also help increase the popularity of Octave and expand its user base, which, also implies that more people would be bought into the world of Open Source. Therefore, this project, in my view, is beneficial for the organization and the open source world as well.

Project Proposal

I will try to train a neural network to understand the spellings of the inbuilt functions of GNU Octave. The neural network would essentially be a classifier and after learning the correct spellings, it would classify a misspelled word into a particular class, the 'class' being the correct spelling of the word. **I plan to use Octave itself (m-scripts) for this project.**

A particular function name is a sequence of characters. Each character has some ASCII value. Because an ASCII pattern 'completely defines' a word (or in our case, a function name), I will use the ASCII sequence of each function name as the input features of the neural network. The ASCII patterns of all the functions would then be fed to neural network, one at a time, and the network would be trained to understand the correct spelling of the functions.

The Neural Network could be trained with three different types of datasets. This would result in three different types of classifiers as follows :-

1) Training with only the correct spellings of the functions: This type of classifier would be very easy to make because only a list of all the existing functions of GNU Octave and no additional data would be required. With this approach, we would end up creating a Neural Network which would easily understand typographic errors caused due to **letter substitutions** and **transportation of adjacent letters** (Please refer to: https://savannah.gnu.org/bugs/?46881). In-fact, this network would understand multiple letter substitutions and transportations also and not only single letter substitutions or transportations. I say this with such confidence because I have already made a working neural network of this type (https://github.com/Sudeepam97/Did_You_Mean). This

network would however, perform poorly if an error is caused due to **accidental inclusion of letters** or due to **accidental deletion of letters.**

2) Training with the correct spellings of the functions and self created errors: This would be slightly harder to make but give us an improved performance. I will create some misspellings of all the functions, by additional inclusion, deletion, and substituting of one or two letters. Then I'll add all these self created misspellings to the dataset which will be used to train the network. Such a network **would understand what correct spellings and typographic errors look like.** It will easily understand substitutions and transportations like the first network but would also be more accurate while predicting errors caused due to additions/deletions. However, it is worth mentioning here that *we may create errors while creating errors.* Because our training data will be modified **randomly,** although the chances are rare, the Neural Network may show uncertain behaviour.

3) Training with the correct spellings of the functions and most common errors: This Neural Network will require the involvement of the entire Octave community, which, also implies that it will be the hardest and the most fun to make. By creating a script that would be able to catch typographical errors and asking the users of Octave to use this script and share the **most common spelling errors** with us, and training the network on the dataset thus created, we'll create a Neural Network which would understand what **correct spellings and the 'most common' typographic errors look like.** Such a network would give good results, almost every-time and with all kinds of errors. This is because when our network knows what common errors are like, most of the time it would '**know the answer'** beforehand. For the remaining times, the network would be able to '**predict the correct answer'.** At a later stage (possibly after GSoC), the data extraction script could also be merged with Octave so that the performance of the Network could be improved with time as Octave evolves. This could come with an easy disable feature, so that only the users who would like to share their spelling errors would do so.

Timeline/Milestones

- **Preparations for the project (pre-community bonding)**: While this application is being reviewed, I have already started working on a m-script which will be used for the extraction of the most common spelling errors. (I'm assuming that I'll be working on the third kind of classifier neural network). This script will catch the most common typographical errors that the users make. This list of errors could then be...
    - Uploaded to a secure server directly.
    - Stored as a text file and we can ask the users to share this file with us.

  I'd like to mention here that the data we receive would essentially be a list misspelled functions only. If there is no user meta data attached to it, we don't really need to hide it from anyone, all we would need is that no unauthorized person could modify/delete it. Nonetheless, how exactly we receive the data collected from the users would need some discussions with the community.
- **Community Bonding period:** I will use the community bonding period to...

        - persuade the community to use our data extraction script and help us collect training data. This can be done by discussing the benifits of a command line suggestion feature and sharing my current implementation of this feature (https://github.com/Sudeepam97/Did_You_Mean).
        - Ask the community to report issues with the m-script containing the current implementation. I'll shift the current implementation to mercurial if required.
        - Discuss how we should receive the data generated by the users, work on the approach, and start the collection of data.
        - Organize the data as it is received and divide it to create proper, training, cross-validation, and test sets for the Neural Network.

- **May, 14 – June, 10 (4 weeks)**
    - **Week 1 (May, 14 – May, 21):** I would not be able to do a lot of work in this week as I have my final examinations at this time. I'll take this week as an extension of the community bonding period and use it to collect issues, receive more data and divide it into proper datasets.
    - **Week 2 and Week 3 (May, 21 – June, 3)**: Most of the code of the Neural Network would be identical to my current implementation and so I'll start by making my current implementation bug free (Some known issues can be found here: **https://github.com/Sudeepam97/Did_You_Mean/issues**) and by coding it according to the Octave coding standards. I plan to keep the user data coming for these weeks also and so I'll leave room for network parameters such as the number of hidden layers and the number of neurons per hidden layer because these are data dependent parameters. If all this work gets completed before the expected time, I'll automatically move on to complete next week's work.
    - **Week 4 (June, 4 – June, 10)**: By now we will have sufficient data (data from approximately 6 weeks of extraction script's usage). I'll quickly give a final look to the data and start training the Neural Network with it. I will choose appropriate values of the data dependent network parameters which, while keeping the speed of the Neural Network fast, would fit the learning parameters (weights) of the Neural Network to our data with a high level of accuracy. I would then measure the accuracy of the Network on cross validation and test sets and see how our network generalizes to unknown typographic errors. I will also write some additional tests for various m-scripts used.

  Phase 1 evaluations goal: A working neural network, which could suggest corrections for typographic errors.

- **June, 11 – July, 8 (4 weeks)**
    - **Week 5 (June, 11 – June, 17):** I'd like to take this week to work in close connection with the community and perform tests on the newly created m-scripts. Essentially,

I'll be asking the community to try out our m-scripts and see how they work for them. I will work on the issues pointed out by the community and by the mentors as they are reported and would try to make the m-scripts perfect in this week itself.

    - **Week 6 (June, 17 – June, 24)**: I'll fix any remaining issues and proceed to discuss and understand how our Neural Network should be integrated with core Octave. I'll start working on integrating the network as soon as the approach is decided. It is worth mentioning here that we will **merge a trained network** with core Octave and therefore the chances of our code being slow are eliminated.
    - **Week 7 – Week 8 (June, 25 – June, 8)**: I will integrate our neural network with core Octave as discussed, and write, and perform tests to make sure that everything works the way it should. If this task gets completed earlier than expected, I'll automatically move on to the next task.

  Phase 2 evaluations goal: A development version of Octave which has a command line suggestion feature (currently there will be no mechanism available to easily select the corrections suggested and easily enable/disable this feature).

- **July, 9 – August, 5 (4 weeks):**
    - **Week 9 (July, 9 – July, 15):** The development version of Octave, with an inbuilt suggestion feature will be open for error reports. I'll work on the issues as they are reported and also discuss what an easy enable/disable mechanism and the mechanism to select the corrections suggested should be like.
    - **Week 10 (July, 16 – July, 22)**: I'll create the required mechanisms as discussed, write and perform tests, and push a development version **with a complete command line suggestion feature.**
    - **Week, 11 – Week, 12 (July, 23 – August, 5):** I'll work in close connection with the community, fix the issues that are reported, and ask for further suggestions on how the command line suggestion feature could be made better.

  Phase 3 evaluations goal: A development version of Octave with a complete and working command line suggestion feature, open to feedback and criticisms.

- **Last days and future deliverables:** During the last days of GsoC, I'll try to improve the command line suggestion feature, based on the feedback received. There are some other things also that I'd like to see in Octave. One is a signal package which is as complete as MATLABs signal toolbox. Another, is a function which could **suggest what functions to use for a particular task,** so for eg, if I type: hdi ("compute circular convolution"), Octave would tell me that you'll have to use the 'cconv' function. By the way, 'hdi' is a short form for 'how do I'. Over the time I have been using Octave, I have encountered situations where I knew that a function to perform a particular task surely exists but I did not know just what it was. The **help** function tells us **how** to use a particular function but I believe there is no function available which would tell us **what** function to use and I would love to add such a function. What I'm trying to convey here is that, interesting problems exist and they always will, and for this exact reason, I'll continue to associate myself with the Octave community, even after GsoC, 2018 will be over.