Read me

Software: Python
Libraries: nltk, scikit-learn, CRFsuite (http://www.chokkan.org/software/crfsuite/)

## *Part1- Issue sentence classification*

Steps to run the program.

***Run Part1.py from the folder Code/Part1***

```
Mohits-MacBook-Air:Part 1 mohitkaduskar$ python Part1.py
```

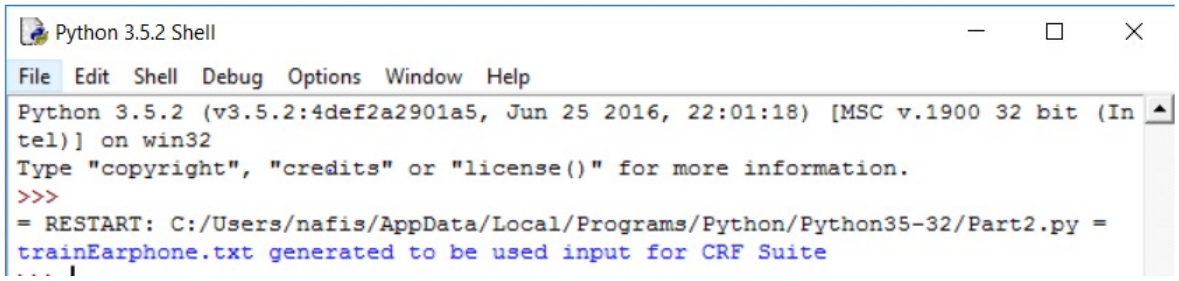
Code performs 10 fold Cross Validation.
You can change the product on which classification is to be done by modifying the variable named 'path'
path='../issue_sent_labeled_data/<<>>/*.txt'
<<>> can have following values depending on the domain

1. earphone
2. gps
3. keyboard
4. mouse
5. mp3_player
6. router

Average accuracy of each product along with classification algorithms used is mentioned in the file named "Result"

## *Part2- Issue Phrase Extraction*

Go To: Code/Part2
Steps to run the program.
***Step 1:***
Run Part2.py- Generates training file which is to be given to CRFSuite.




Sample file generated in this step is as follows:

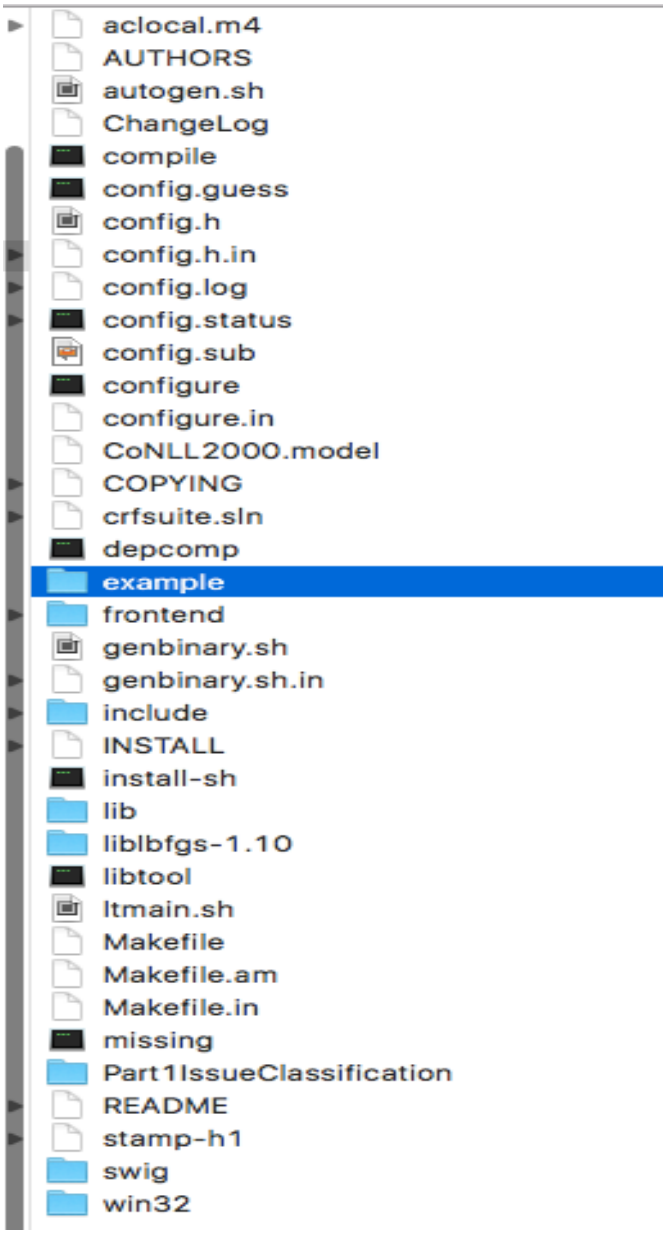
Read me

```
1 CD O AA AA Neutral O
) ) O AA AA Neutral O
It PRP O AA AA Neutral O
takes VBZ O es AA Neutral O
long JJ B-NP AA AA Neutral O
time NN I-NP AA AA Neutral O
to TO O AA AA Neutral O
acquire VB O AA AA Neutral O
satellite RB B-ADVP AA AA Neutral O
2 CD O AA AA Neutral O
) ) O AA AA Neutral O
Speaker NNP B-NP er AA Neutral O
quality NN B-NP ity AA Neutral O
is VBZ O s AA Neutral O
bad JJ O AA AA Negative O
3 CD O AA AA Neutral O
) ) O AA AA Neutral O
Starting NNP B-NP ing AA Neutral O
& CC O AA AA Neutral O
end VB O AA en Neutral O
direction NN B-NP tion AA Neutral B
are VBP O AA AA Neutral I
very RB B-ADVP y AA Neutral I
bad JJ O AA AA Negative I
. . O AA AA Neutral O
```

The first column is word followed by POS tag then Phrase chunk, Prefix, Suffix, Word Polarity and then the {B,I,O}.
B-Begin, I-Inside and O-Outside. These signify words in <i> tags of data set.
Sentences are separated by a blank.

***Step 2:***
This step requires CRF Suite
Instructions for installing the Suite are available on http://www.chokkan.org/software/crfsuite/
On installing the CRFSuite following Folders are created.

Read me




Kindly Copy the example folder from Code/Part2 folder on local machine
The chunking061216.py contains all the templates of the feature sets.
When data is processed using this file, it gives all the feature values of data in the output file

Below is a screenshot of a sample output file.

Read me

```
O       w[0]=Unfortunately      w[1]=,  w[2]=I  w[3]=am w[4]=sending    pos[0]=RB       pos[1]=,        pos[
polarity[1]=Neutral     polarity[2]=Neutral     polarity[3]=Neutral     polarity[4]=Neutral     w[0]|w[1]=Un
w[1]=sending|,  w[4]|w[2]=sending|I     w[4]|w[3]=sending|am    w[3]|w[0]=am|Unfortunately      w[3]|w[1]=am
w[1]|w[2]=,|I   w[1]|w[3]=,|am  w[1]|w[4]=,|sending     pos[0]|pos[1]=RB|,      pos[0]|pos[2]=RB|PRP    pos[
pos[3]|pos[1]=VBP|,     pos[3]|pos[2]=VBP|PRP   pos[3]|pos[4]=VBP|VBG   pos[2]|pos[0]=PRP|RB    pos[2]|pos[1
phrase[0]|phrase[1]=B-ADVP|O    phrase[0]|phrase[2]=B-ADVP|O    phrase[0]|phrase[3]=B-ADVP|O    phrase[0]|ph
phrase[3]|phrase[1]=O|O phrase[3]|phrase[2]=O|O phrase[3]|phrase[4]=O|O phrase[2]|phrase[0]=O|B-ADVP    phra
phrase[4]=O|O   polarity[0]|polarity[1]=Negative|Neutral        polarity[0]|polarity[2]=Negative|Neutral
polarity[1]=Neutral|Neutral     polarity[4]|polarity[2]=Neutral|Neutral polarity[4]|polarity[3]=Neutral|Neut
Neutral polarity[2]|polarity[0]=Neutral|Negative        polarity[2]|polarity[1]=Neutral|Neutral polarity[2]||
polarity[1]|polarity[3]=Neutral|Neutral polarity[1]|polarity[4]=Neutral|Neutral pos[0]|phrase[0]=RB|B-ADVP
phrase[2]=VBG|O pos[4]|phrase[3]=VBG|O  pos[4]|phrase[4]=VBG|O  pos[3]|phrase[0]=VBP|B-ADVP     pos[3]|phras
pos[2]|phrase[3]=PRP|O  pos[2]|phrase[4]=PRP|O  pos[1]|phrase[0]=,|B-ADVP       pos[1]|phrase[1]=,|O    pos[
phrase[2]=Negative|O    polarity[0]|phrase[3]=Negative|O        polarity[0]|phrase[4]=Negative|O        pola
polarity[3]|phrase[0]=Neutral|B-ADVP    polarity[3]|phrase[1]=Neutral|O polarity[3]|phrase[2]=Neutral|O pola
polarity[2]|phrase[3]=Neutral|O polarity[2]|phrase[4]=Neutral|O polarity[1]|phrase[0]=Neutral|B-ADVP    pola
polarity[1]=RB|Neutral  pos[0]|polarity[2]=RB|Neutral   pos[0]|polarity[3]=RB|Neutral   pos[0]|polarity[4]=R
Neutral pos[3]|polarity[0]=VBP|Negative pos[3]|polarity[1]=VBP|Neutral  pos[3]|polarity[2]=VBP|Neutral  pos[
polarity[3]=PRP|Neutral pos[2]|polarity[4]=PRP|Neutral  pos[1]|polarity[0]=,|Negative   pos[1]|polarity[1]=,
pos[1]=Unfortunately|,  w[0]|pos[2]=Unfortunately|PRP   w[0]|pos[3]=Unfortunately|VBP   w[0]|pos[4]=Unfortun
pos[1]=am|,     w[3]|pos[2]=am|PRP      w[3]|pos[3]=am|VBP      w[3]|pos[4]=am|VBG      w[2]|pos[0]=I|RB
pos[3]=,|VBP    w[1]|pos[4]=,|VBG       w[0]|polarity[0]=Unfortunately|Negative w[0]|polarity[1]=Unfortunate
Negative        w[4]|polarity[1]=sending|Neutral        w[4]|polarity[2]=sending|Neutral        w[4]|polarit
w[3]|polarity[3]=am|Neutral     w[3]|polarity[4]=am|Neutral     w[2]|polarity[0]=I|Negative     w[2]|polarit
polarity[1]=,|Neutral   w[1]|polarity[2]=,|Neutral      w[1]|polarity[3]=,|Neutral      w[1]|polarity[4]=,|N
phrase[4]=Unfortunately|O       w[4]|phrase[0]=sending|B-ADVP   w[4]|phrase[1]=sending|O        w[4]|phrase[
w[3]|phrase[3]=am|O     w[3]|phrase[4]=am|O     w[2]|phrase[0]=I|B-ADVP w[2]|phrase[1]=I|O      w[2]|phrase[
phrase[4]=,|O
```

Here for every word we can see the following features after the output tag
As we can see above O is the output tag.
Features upto second order are considered and the notations given are as follows:
w- current word and neighbours
pos- pos of words and their neighbours
phrase- phrase of words and their neighbours
polarity- polarity of words and their neighbours

To run command:
Go to example folder on your local machine and type the below command:

```
cat inputfilename | ./templatedefiner.py > outputfile.crfsuite.txt
```

```
>>cat testFFinal.txt | ./chunking061216.py > train1111.crfsuite.txt
```

SCREENSHOT

```
Mohits-MacBook-Air:example mohitkaduskar$ cat testFFinal.txt | ./chunking061216.
py > train1111.crfsuite.txt
```

**Step3:**
http://www.chokkan.org/software/crfsuite/tutorial.html has various ways of giving the input to the model. We have made use of 2 algorithms and mentioned the results in Result file.

Following is the command for 10 fold crossvalidation which involves passing the algorithm name as input

```
crfsuite learn [OPTIONS] [DATA1] [DATA2] ...
Trains a model using training data set(s).
```

Read me

```
  DATA    file(s) corresponding to data set(s) for training; if multiple N
files
          are specified, this utility assigns a group number (1...N) to the
          instances in each file; if a file name is '-', the utility reads a
          data set from STDIN

OPTIONS:
  -t, --type=TYPE       specify a graphical model (DEFAULT='1d'):
                        (this option is reserved for the future use)
      1d                    1st-order Markov CRF with state and transition
                            features; transition features are not
conditioned
                            on observations
  -a, --algorithm=NAME  specify a training algorithm (DEFAULT='lbfgs')
      lbfgs                 L-BFGS with L1/L2 regularization
      l2sgd                 SGD with L2-regularization
      ap                    Averaged Perceptron
      pa                    Passive Aggressive
      arow                  Adaptive Regularization of Weights (AROW)
  -p, --set=NAME=VALUE  set the algorithm-specific parameter NAME to VALUE;
                        use '-H' or '--help-parameters' with the algorithm
name
                        specified by '-a' or '--algorithm' and the graphical
                        model specified by '-t' or '--type' to see the list
of
                        algorithm-specific parameters
  -m, --model=FILE      store the model to FILE (DEFAULT=''); if the value
is
                        empty, this utility does not store the model
  -g, --split=N         split the instances into N groups; this option is
                        useful for holdout evaluation and cross validation
  -e, --holdout=M       use the M-th data for holdout evaluation and the
rest
                        for training
  -x, --cross-validate  repeat holdout evaluations for #i in {1, ..., N}
groups
                        (N-fold cross validation)
  -l, --log-to-file     write the training log to a file instead of to
STDOUT;
                        The filename is determined automatically by the
training
                        algorithm, parameters, and source files
  -L, --logbase=BASE    set the base name for a log file (used with -l
option)
  -h, --help            show the usage of this command and exit
  -H, --help-parameters show the help message of algorithm-specific
parameters;
```

Read me

```
                        specify an algorithm with '-a' or '--algorithm'
option,
                        and specify a graphical model with '-t' or '--type'
option
```

The following options are available for training.

-t, --type=TYPE
    Specify a graphical model used for feature generation. The default value is "1d".
    1d
    The 1st-order Markov CRF with state and transition features (dyad features). State features are conditioned on combinations of attributes and labels, and transition features are conditioned on label bigrams.
-a, --algorithm=NAME
    Specify a training algorithm. The default value is "lbfgs".
    lbfgs
    Gradient descent using the L-BFGS method
    l2sgd
    Stochastic Gradient Descent with L2 regularization term
    ap
    Averaged Perceptron
    pa
    Passive Aggressive (PA)
    arow
    Adaptive Regularization Of Weight Vector (AROW)
-p, --param=NAME=VALUE
    Configure a parameter for the training. CRFsuite sets the parameter (NAME) to VALUE. Available parameters depend on the graphical model and training algorithm selected. To see the help message of available parameters, use '-H' or '--help-parameters' with the algorithm name specified by '-a' or '--algorithm' and the graphical model specified by '-t' or '--type'.
-m, --model=MODEL
    Store the trained model to a file MODEL. The default value is "" (empty). CRFsuite does not store the model to a file when MODEL is empty.
-g, --split=N
    Split the instances into N groups, and assign a number in {1, ..., N} to each group. This option is mostly used to perform N-fold cross validation (with -x option). By default, CRFsuite does not split input data into groups.
-e, --holdout=M
    Use the instances of group number M for holdout evaluation. CRFsuite does not use the instances of group number M for training. By default, CRFsuite does not perform holdout evaluation.
-x, --cross-validate
    Perform N-fold cross validation. Specify the number of splits with -g option. By default, CRFsuite does not perform cross validation.
-l, --log-to-file

Read me

Write out the log message of training to a file. The file name is determined automatically from the command-line arguments (e.g., training algorithm, graphical model, parameters, source files). By default, CRFsuite writes out the log message to STDOUT.

-L, --logbase=BASE
Specify the base name for the log file (used with -l option). By default, the base name is "log.crfsuite".

-h, --help
Show the usage of this command and exit.

-H, --help-parameters
Show the list of parameters and their descriptions. Specify the graphical model and training algorithm with -t and -a options, respectively.

-p, --param=NAME=VALUE
Configure a parameter for the training. CRFsuite sets the parameter (NAME) to VALUE. To see the list of parameters and their descriptions, use -H (--help-parameters) option.

Move the crfsuite.txt output file of previous step to frontend folder of local machine and give it as input to crf suite
Then move the to the frontend folder from terminal.

Command Used by us:

```
./crfsuite learn -g10 -x -l crfsuitefilename
```

>>./crfsuite learn –g10 –x –l traingps.crfsuite.txt

For implementing default Algorithm (Gradient descent using the L-BFGS method)

```
Akshays-MacBook-Pro:frontend akshay$ ./crfsuite learn -g10 -x -l traingps.crfsuite.t
xt
```

>>./crfsuite learn a- l2sgd –g10 –x –l traingps.crfsuite.txt

For implementing Algorithm (Stochastic Gradient Descent with L2 regularization term)

```
Akshays-MacBook-Pro:frontend akshay$ ./crfsuite learn -a l2sgd -g10 -x -l traingps.c
rfsuite.txt
```

The detailed Results are posted in a word File named ‘Result ’ in the code folder