## Data

1. The data consists mostly of customer feedback
2. Feedback was annotated by customers
3. The data consists of:
   1. 5000 bugs
   2. 5000 crash
   3. About 89000 documents marked as other
   4. About 1000 as both bug and crash
4. After data cleansing, our dataset was reduced to about
   1. 2000 bugs
   2. 1900 crash
   3. 87000 other
5. To this end, we chose to model for types: bugs, crash and other. We omitted both because the set of documents annotated as other was too small.

## Model

1. Due to the disparity in data size, we generated random sets of content marked as other and built several training datasets consisting of all bugs, crash and other sets. You can find this training sets here <https://github.com/obinna240/Hack/tree/master/Hack/Final_Bug_Crash_TrainingSet>. Also observe that we have 7 training sets and 1 test set.
2. We implemented 7 different models which you can find here: <https://github.com/obinna240/Hack/tree/master/Hack/TestSavedModel>. The key models are labelled model1.model to model7.model. (Ignore all the other models).
3. Results of each model can be found in the html files.

# Using a trained model

### CLI

1. You can load and apply a trained model from your command prompt.
2. Make sure you have weka jar in your classpath including the saved model.
3. You can load the model and use it on a test set using:
   1. > >java <classifierClass> -l <modelName> -T <testfile Name>
   2. java weka.classifiers.meta.FilteredClassifier –l modelName.model –T testFileName.arff (Use this and set model file name and test file name).

Note that FilteredClassifier class consists of a pipeline of models and processors including converting String to Word Vectors, Attribute estimation and finally the perceptron algorithm.

### USING WEKA EXPLORER

See video.

Additional Resources

<https://weka.wikispaces.com/Making%20predictions>

https://weka.wikispaces.com/Saving+and+loading+models