**Project EPSILON**
Automatic JIRA triage
Yonatan Feleke, Gurkanwal Brar & Ashok Poothiyot

**PROBLEM STATEMENT**

The corporate world deals with task management in a variety of ways all having some form of triaging process to correctly assign tickets to developers. Automation of this task has proven elusive with less than 60% accuracy of latest ML solutions even for Web and SaaS companies that handle high volumes of tickets in the form of exceptions, support requests, user-reported bugs, and crash reports. Effective automation is essential to improve productivity and obviate the tedious work of manually triaging tickets.

Project aims to reduce this overhead by deploying a deep neural network classifier to assign tickets to a developer. The DNN model is trained by using the final assignee listed in the JIRA ticket as the label and predicts a previously seen developer on new tickets.

**DATASET & FEATURES**

The project utilized the generated jumble.expium.com dataset for developing the algorithm and then applies the architecture to train and test on the private LinkedIn Foundation team support dataset. The implementation uses a bag of words multinomial event model to vectorize the JIRA ticket. The text components of JIRA ticket namely: subject, body, and comments, are concatenated and featureized with a frequency threshold of $>5$.

The shape of the datasets after vectorizing were: LinkedIn m x n = <559, 2936> and Expium dataset m x n = <5435, 1970>.

**Features and Vectorization**

The input data described in the data set section was parsed into a bag of words, multinomial event model representation. The words are assumed to be independent and chosen with separate distributions at create time. During the preprocessing step we capture the list of frequent words at a threshold of >= 5 when building multinomial vector. While developing our solution we make the following assumptions:

1. A ticket $t$ in a set of tickets $\{t_1, \ldots, t_n\}$ assigned to a developer $a_1, \ldots, a_n$ is generated by a unique distribution modelled by $\Phi$:
   
   $$P(t_i|\Phi) = \prod_{d=1}^{D} p(x_i|a_i, \Theta)$$

2. Tickets are composed of independently and identically distributed words chosen at random (naive Bayes assumption) and from a multinomial distribution:

   $$\text{Labels: } \{a_1', \ldots, a_n'\} = \prod_{d=1}^{D} p(w_i|a_i, \Theta)$$

The labels are developers we have seen before represented as integers. We then create a matrix a multinomial input vector and an integer class label representing assignee to train and test. NB and SVM are not investigated by this paper but the paper presented only for reference.

The shape of the datasets after vectorizing were: LinkedIn m x n = <559, 2936> and Expium dataset m x n = <5435, 1970>.

**Network Model**

We utilize a DNN to approximate $\Phi$ in order to predict a designated assignee. As per the neural network design section, we use a 3x16 tanh neural network with a 0.005 learning rate and 1,000 backprop iterations. The final label (developer) selection is done via softmax followed by a cross entropy loss for backpropagation. The network is trained and tested using a Fold cross validation similar to the work by Bettenburg et al.

**Results**

Blow are the results of the experiments conducted on March 2018 tickets, larger compute/optimization is required to run full dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>LinkedIn Dataset (559/34383)</th>
<th>Expium Generated Dataset (~5500)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM Classifier (linear kernel, hinge loss)</td>
<td>1.0 w/ 70% train*</td>
<td>0.29166 w/o 30% test</td>
</tr>
<tr>
<td>Naïve Bayes Classifier</td>
<td>1.0 w/ 70% train*</td>
<td>0.223175 w/ 30% test</td>
</tr>
<tr>
<td>Deep Neural Network Classifier</td>
<td>0.65251 S</td>
<td>0.11613 S</td>
</tr>
<tr>
<td></td>
<td>0.058767833</td>
<td>2Fold Mean:</td>
</tr>
</tbody>
</table>

The architecture chosen is: a 3 wide, 16 high, and 32 deep layers using the Expium dataset. The dimensions investigated in the images below.

- **Remarks**
  - The highly non-linear nature of DNN has allowed almost perfect 5Fold mean accuracy on over set 3,000 LinkedIn tickets and 5,500 generated tickets, that same benefit has heavily overfit the data observed by the dismal ~7% test prediction accuracy.
  - Computational challenges with vectorizing text has prevented us from significantly scaling the number of trained samples, working on a batching solution to count words efficiently, store to a datastore and fetch on batched training and prediction stages.
  - A developer that has never been assigned a ticket will not be considered during prediction, this is normal in triaging.
  - The SVM classifier was only run with splitting up data once instead of the bagging like iFold cross validation completed for the others.

**Discussion & Results**

**Future Work:**
- Make the vectorization process cacheable to handle entire dataset.
- Utilize additional features such as: watchers, labels, reporter, hashed text exception and so on.
- Take advantage of MLP concepts to improve test accuracy, implementing stemming, lemmatization, Word2Vec and word embeddings.
- **Conclusions:**
The initial results of our experiments resulted in poor accuracy on the test set. The DNN overfit the small dataset and we hope that after running on full set (~500 more tickets) the test accuracy will dramatically increase, currently experiments have only been applied to a set of 5,000 tickets.

**References**


The Neural network parameters were chosen by searching through parameters using the Expium dataset. The dimensions investigated in the images on the left are:

- Activator Functions: (tanh, relu)
- Learning rates: [0.5, 0.05, 0.005]
- Depth: [3, 5]

The architecture chosen is: a 3 wide, 16 high, Relu activated network with a learning rate of 0.005 and 1,000 backpropagation iterations.

- The architecture is chosen due to the fast and smooth descent and relative simplicity of the network compared to the higher train but lower test accuracy architectures below.