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 with each 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. .

DATASET & FEATURES

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

A JIRA ticket has the following JSON structure:


```
{
   "summary": "Update success. 3.0 USB Card",
   "description": "OR memory.dmp folder, ....",
   "priority": "Minor",
   "reporter": "chantal.colman",
   "labels": [
     "Communication"
   ],
   "worklogs": [],
   "status": "Open",
   "issueType": "Epic",
   "created": "2018-09-13T14:22:15-07:00",
   "updated": "2018-11-16T16:00:00-08:00",
   "affectedVersions": [],
   "fixedVersions": [],
    watchers": [
     "kevin.mcwhorter"
    ],
   "components": [],
   "externalId": "OLDMOB-167",
   "comments": [
     {
       "body": "No results. 3. Replacing the next
would like help someone has efficient cooling.
Crashes only hardware problems. Even something or
removing the buzz is supposed to do anything.",
       "author": "sarah.clark",
       "created": "2018-10-11T17:00:00-07:00"
     },
   "customFieldValues": [
     {
       "fieldName": "Epic Name",
       "fieldType": "com.opper.jira:gh-epic-label",
       "value": "Update success. 3.0 USB Card"
     }
   },
"history": [
    {
      "author": "chantal.colman",
      "created": "2018-11-06T16:00:00-08:00",
      "items": [
        {
          "fieldType": "jira",
          "field": "status",
          "from": 1,
          "fromString": "Open",
          "to": 3,
          "toString": "In Progress"
        }
      ]
    },
]
```


Features, Models and Results

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 have a a list of important keywords that are added as features by default, a list of stop words that are ignored and a frequency threshold of >= 5 when building multinomial vector. While developing our solution we make the following assumptions:

/> A ticket t_i in a the set of tickets $\{t_1, \dots, t_{|T|}\}$ assigned to a developer $\{d_1, \dots, d_{|D|}\}$ is generated by a unique distribution modelled by θ:

$$P(\mathrm{t}_i|\theta) = \sum_{j=1}^{|\mathrm{d}|} P(\mathrm{t}_i|\mathrm{d}_j, \theta) P(\mathrm{t}_j|\theta)$$

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

$$P(t_i|d_j, \theta) = \prod_{i=1}^{n} p(\mathrm{w_t}|t_j, \theta)^{occurance}$$

Lastly, the labels are built based on developers we have seen before represented as ENUM integers. Based on these foundations, we create a matrix a multinomial input vector and an integer class label representing assignee.

Network Model

We utilize a DNN to approximate θ 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 back prop. The network is trained and tested with an 80% / 20% train to test split, initial attempts at 5 fold validation proved too time consuming.

Results

Blow we demonstrate the results of the experiments conducted

Model	Linkedin Dataset		Expium Generated Dataset (~5500)	
	Train Performance	Test Performance	Train Accuracy	Test Performance
SVM Classifier (linear kernel)				
Naive Bayes Classifier				
Deep Neural Network Classifier			5Fold (100%) : 1	

Remarks

One acceptable point is that a developer that has never been assigned a ticket will not be considered during prediction, this is similar to

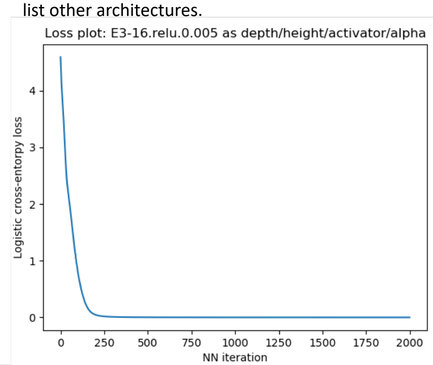
The highly non-l

Neural Network Design

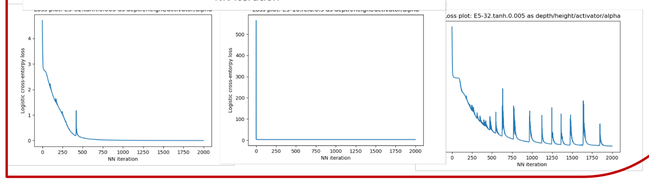
The Neural network parameters were chosen by searching through parameters using the Expium dataset. The dimensions investigated in the images on the left are:

- Activator Functions: [tanh, relu]
- Learning rates: [0.5, 0.05, 0.005]
- Hidden layers: [8, 16, 32]
- Depth : [3, 5]

The architecture chosen is: a 3 wide, 16 high neural, ReLu activated network with a learning rate of 0.005 and 1,000 backpropagation iterations. At the bottom we list other architectures.



The architecture is chosen due to the smooth descent and relative simplicity of the network compared to the higher train accuracy achieved by the architectures below.



Discussion Remarks

Conclusions: The initial results of our experiment resulted in poor accuracy on the test set. The DNN is able to model highly non-linear models but even with a dataset of <> tickets, the test set still performed poorly.

Future Work: Parameter search on the LinkedIn dataset though computationally expensive could improve our model capabilities

Additional features from JIRA ticket fields leaves a lot of room for improvement

- The current algorithm does not take advantage of NLP concepts, a great extension of this work would be to implement stemming, lemmatization, Word2Vec and word embeddings.
- Utilize additional features such as: watchers, labels, reporter, hashed exceptions and so on.

References

[1] Murphy, G., and D. Cubranic. "Automatic bug triage using text categorization." *Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering*. 2004.

[] –

[] –

[] –

[] –

[] – WUYUNTANA, D. and WANG, S. (2018). Distributed Representations of Mongolian Words and Its Efficient Estimation. DEStech Transactions on Computer Science and Engineering, (iceit).