MINING
STACK OVERFLOW

AN ACCOUNT OF EXPLORATORY RESEARCH

Created by Chris Foster / @chrisfosterelli

RESEARCH GOALS

1. Explore techniques for mining text and web data
2. Understand how to apply learning algorithms
3. Understand how to evaluate feature performance
4. Develop a program to automate record analysis
5. Compare and contrast literature work with results

INITIAL RESEARCH

We started with a literature review of nine papers:

- A Premliminary Psychometric Text Analysis of the Apache Developer Mailing List
- Techniques for Identifying the Country Origin of Mailing List Paricipants
- An Analysis of Newbies' First Interactions on Project Mailing Lists
- Content Classification of Developer Emails
- Communication in Open Source Software Development Mailing Lists
- Reviewer Recommendation to Expedite Crowd Collaboration
- Mining Developer Mailing List to Predict Software Defects
- Automatically Prioritizing Pull Requests
- Development Emails Content Analyzer: Intention Mining in Developer Discussions

KEY FINDINGS

1. Mailing lists are less important today than previously
2. Many papers had low or very low accuracy rates
3. Many papers had no clear pracitcal application

We wanted to find something with:

- A clear, useful application
- Substantial room for future work
- Datasets other than traditional mailing lists

We added two new papers to the list:

- Predicting Response Time in Stack Overflow
- Improving Low Quality Stack Overflow Post Detection

First step is to replicate the work done in the original paper

SOFTWARE USED

- Python
- Numpy
- Scipy
- Scikit-learn

DATA SOURCE

- Stack Overflow provides public data dumps
- Expanded dataset is a 39GB XML file
- This saved us substantial time!

INITIAL REPLICATION: FILTERING

- We had to reduce the dataset to the specified period
- Only posts between May 1st and August 1st, 2014
- Only posts that are a question or answer
- Converts post format from XML into JSON
- Reduces the dataset to **1,307,172 posts**

INITIAL REPLICATION: GENERATION

- The next step is to generate tag-based features
- At this point we have to filter out "unpopular tags"[1]
- The authors use three key features: RSR[2], ASR[3], and PR[4]

```
1: 15 unique contributors for that tag
2: users with avg response time below 2hr for that tag / total users
3: users with at least 10 answers for that tag / total users
4: number of tag occurances / total tag occurances
```

- We generate these using internal maps and caches
- The CSV output contains **266,482 questions**

INITIAL REPLICATION: ANALYSIS

- K-Means clustering to group response times
- K-Nearest-Neighbours classification engine
- K-Fold cross-validation to check accuracy
- Parameters: 25 bins, 10 neighbours, and 10 folds

We could successfully reproduce the paper's success rate!

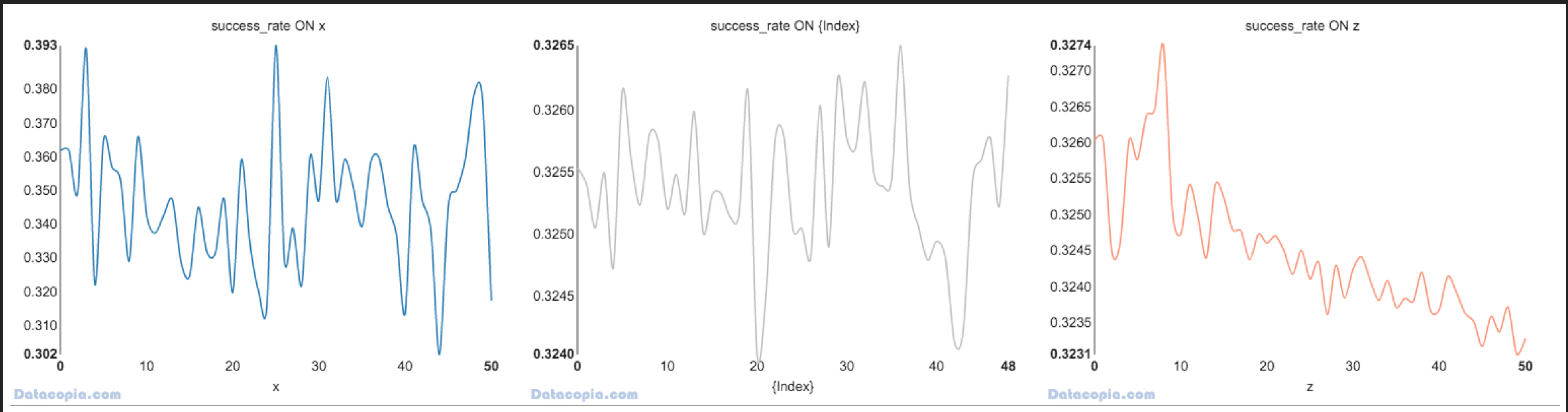
Success rate: 32.4%

So how can we improve this?

EXPERIMENT 1

- Features use many "magic numbers"
- *Can we vary unique contributor count? (X)*
- *Can we vary the avg. 2hr response cutoff? (Y)*
- *Can we vary the minimum answer count? (Z)*

RESULTS


success_rate ON x
0.393
0.380
0.370
0.360
0.350
0.340
0.330
0.320
0.310
0.302
0
10
20
30
40
50
x
Datacopia.com
success_rate ON {Index}
0.3265
0.3260
0.3255
0.3250
0.3245
0.3240
0
10
20
30
40
48
{Index}
Datacopia.com
success_rate ON z
0.3274
0.3270
0.3265
0.3260
0.3255
0.3250
0.3245
0.3240
0.3235
0.3231
0
10
20
30
40
50
z
Datacopia.com

INTEPRETATION

- Results indicate little variation in success
- No predictable pattern indicated by changing values
- Small changes accounted for by KMeans randomization
- Other parameters are likely "making up" for failures

EXPERIMENT 2

- *Do these results generalize to longer time periods?*
- *Do these results generalize to other time periods?*

RESULTS

Date From	Date To	Date Range	Performance	Notes
2014-05-01	2014-08-01	3 months	0.324749849661	Original range
2014-05-01	2014-11-01	6 months	0.34782602467887241	Extend to double length
2014-05-01	2015-05-01	12 months	0.35583501489290714	Extend to quadruple length
2015-05-01	2015-08-01	3 months	0.28158354151398396	Compare to period in 2015
2015-05-01	2015-11-01	6 months	0.351002756918	Compare to period in 2015

INTERPRETATION

- Results can generalize to different years
- Results can generalize to larger datasets
- Extending the dataset may slightly improve accuracy

EXPERIMENT 3

- *Do Neural Networks improve the success rate?*
- *Do Support Vector Machines improve the success rate?*

RESULTS

Algorithm	Accuracy
Nearest Neighbours	0.324731086386
Support Vector Machine	0.346657518841
Neural Network	0.346657518841

INTERPRETATION

- Both algorithms appear to only slightly increase accuracy
- Both algorithms also take substantially longer to run
- Both algorithms run with identical performance
- This suggests that *features* are the problem

CHALLENGES

- Working with very large files
- Replicating vague original work
- Finding an area to focus on

FUTURE WORK

- Experiment on value ranges with feature isolation
- Introduce new features for higher accuracy
- Apply this technique to other datasets

CONCLUSION

In this research we were able to:

- Compare and contrast current literature
- Explore techniques for data mining Stack Overflow
- Apply learning algorithms for response time prediction
- Evaluate feature performance of existing algorithm
- Evaluate different time ranges on existing algorithm
- Evaluate different algorithms on the dataset

THE END

- Chris Foster
- Thompson Rivers University