Ticket Tagger v2.0

Label Classification with Machine Learning

L.Fried, T. Moser, D. Steiger & C.Birchler

Ticket Tagger Introduction

- SMaE Project of Rafael Kallis (2018)
- GitHub App (NodeJS Server)
- Label GitHub Issues based on Title & Description
  - Bug, Enhancement, Question
- Helps organizing big projects automatically
- Uses Fasttext for text classification

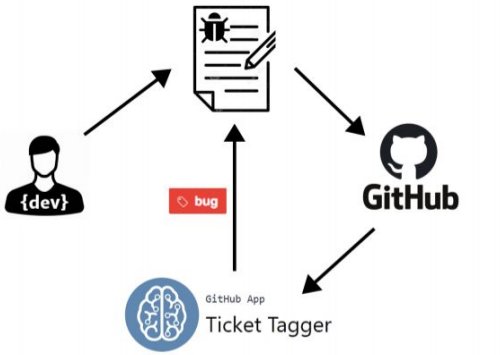


Figure 1: Ticket Tagger issue labeling process.

Ticket Tagger Limitations

- Uses only fasttext as its classifier
- Evaluation and training on different repositories
- No preprocessing
- Model size < 5MB

Research Questions

- To what extent can we increase performance (F1-score) with other classifiers?
- To what extent do preprocessing techniques affect the models for issue classification?

Study Methodology

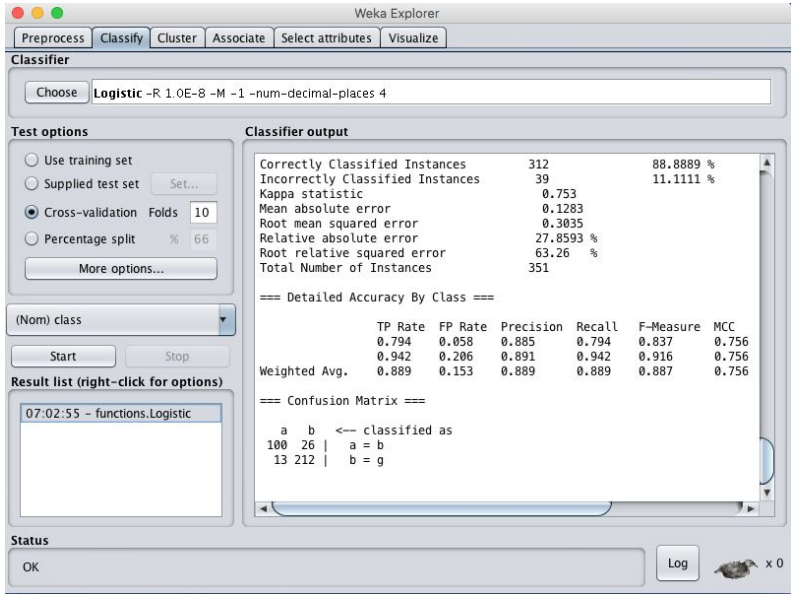
- Acquire appropriate Data Set
- WEKA / MEKA
- Multi Binary Label Classification
- Preprocessing
- Results
- Summary

A new Dataset: Pandas

- Limitations
  - >1000 issues / label (bug, question, enhancement)
  - One single Repository
  - All in English
  - Not in Java Ecosystem
- Find fitting repository
- Scrape GitHub
- Convert dump to Data Set format (fasttext & arff)
- Balance to equal label distribution (n=1230)

WEKA and MEKA: An introduction

- WEKA: **W**aikato **E**nvironment for **K**nowledge **A**nalysis
- MEKA: Based on Weka, multi-label learning



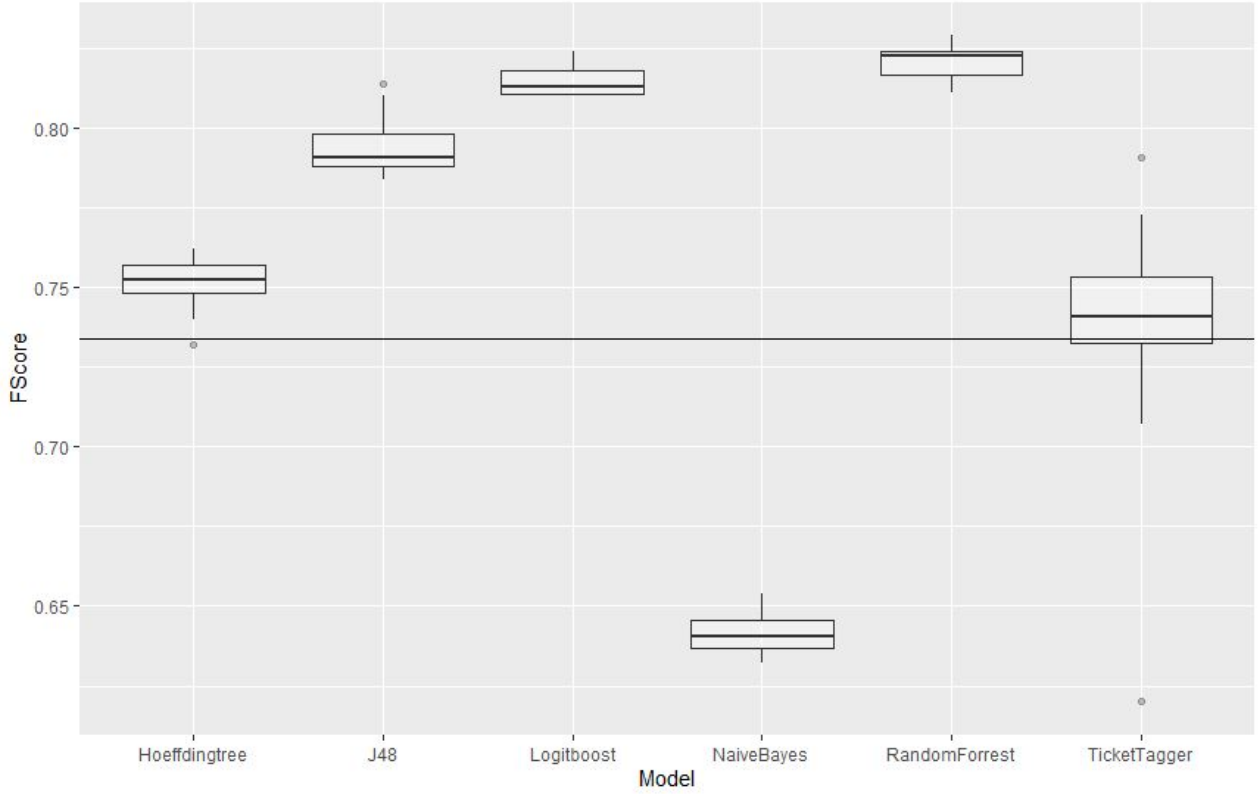
WEKA and MEKA - Approaches

- Different Models:
  - J48
  - RandomForest
  - LogitBoost
  - AdaBoost
  - NaiveBayes
  - HoeffdingTree
- Binary Relevance

Preprocessing

- Stemming
  - e.g., argue, argued, argues, arguing, and argus to the stem argu
  - Porter
  - Snowball
  - Stanford tool
- Stopword removal
  - ‘the’, ‘a’, ‘in’, ‘on’, etc.
- Combinations!
  - 12 different datasets with different preprocessing combinations

Results - Preprocessing



Results - MEKA & WEKA

- Every Model increased F1
  - except Naive Bayes
- Binary Relevance decreased F1
  - Transforms a multi-label classification problem with L labels into L single-label separate binary classification problems using the same base classifier
  - The prediction output is the union of all per label classifiers
- Boost models have best results
  - slow (chained models)

Multi Binary Label with Fasttext

- Train one model for each label
- Split data set in 3 for each label
- Test every case on 3 models, use the one with highest probability as answer
- Ten fold validation
- Calculate Statistics

Results - MBL in Fasttext

- Similar F1 to ticket-tagger but not an increase
- Not a good trade-off
  - more models
    - more memory
  - longer training
  - longer predictions

Limitations

- Only applied techniques to Pandas issues
  - Cannot generalize
- Model size not restricted
  - Ticket Tagger was restricted to 5MB
- Processing time not restricted
  - Better performance but longer calculations
  - Trade-off?
- Not integrated into Ticket Tagger itself
  - Maybe some approaches would not be feasible in NodeJs

Future Work

- Ticket Tagger integration
- Replication on other repos
- Try more preprocessing approaches
- More labels

Summary

- Ticket Tagger F1-score of 73.4 %
- Our project’s pipelines reached consistently over 75% (maximum of 82.5%)
- Preprocessing has a considerable effect on performance
    - fasttext varies between 66% and 79%

Questions and Discussion