Measuring Software Ticket Quality using Quantitative Data Analysis

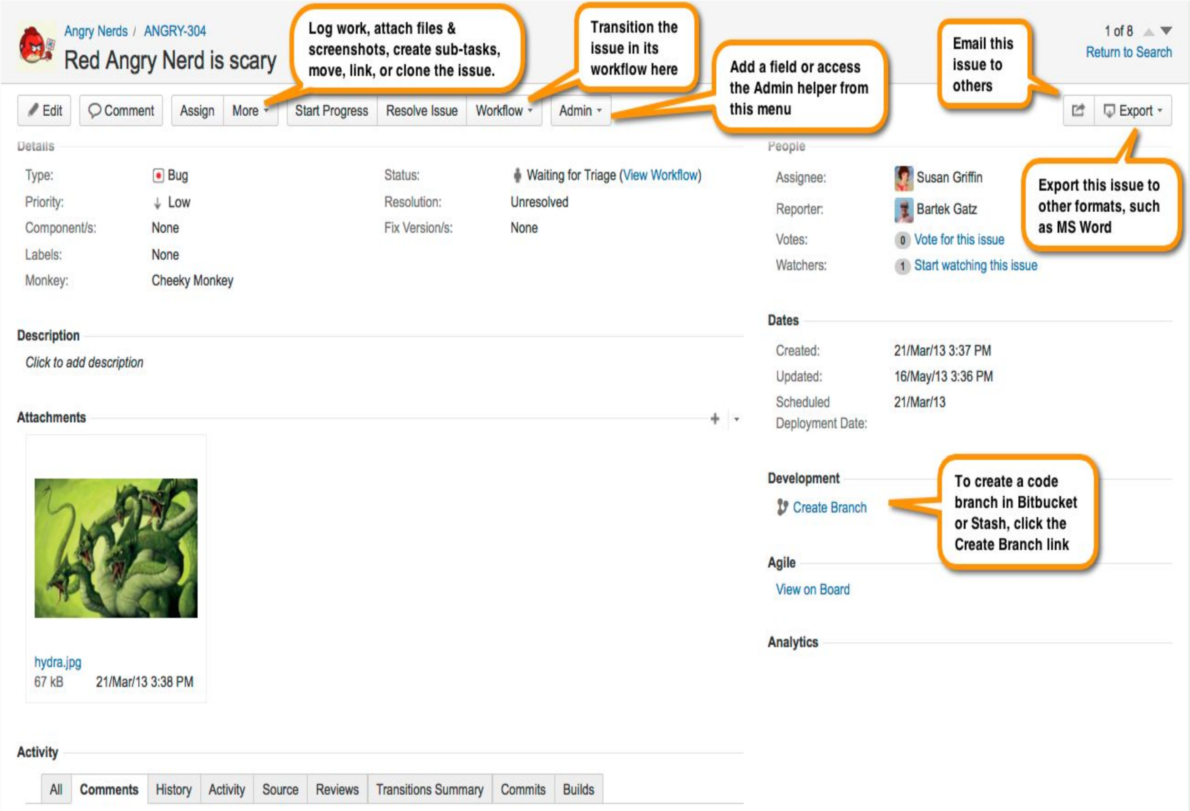
Andrei-Mihai Nicolae (2147392)
Dr. Timothy Storer

Introduction

- Software engineering is becoming more complex as technology has become ubiquitous
- Harder to plan, manage and track work during development lifecycle [8]
- Solution? **Issue tracking systems** (e.g. Jira, Bugzilla, Manuscript)
- **Software Tickets** are the core component of such systems

Software Tickets

- Many components: summary, description, attachments, comments…
- Usually come in one of 2 forms: feature requests and bug reports
- What makes for a **High Quality** software ticket?
- Our findings - contribution to the community



Angry Nerds / ANGRY-304

Red Angry Nerd is scary

Log work, attach files & screenshots, create sub-tasks, move, link, or clone the issue.

Transition the issue in its workflow here

Add a field or access the Admin helper from this menu

Email this issue to others

Export this issue to other formats, such as MS Word

Details

Type: Bug

Priority: Low

Component/s: None

Labels: None

Monkey: Cheeky Monkey

Status: Waiting for Triage (View Workflow)

Resolution: Unresolved

Fix Version/s: None

People

Assignee: Susan Griffin

Reporter: Bartek Gatz

Votes: 0 Vote for this issue

Watchers: 1 Start watching this issue

Description

Click to add description

Attachments

hydra.jpg
67 kB 21/Mar/13 3:38 PM

Dates

Created: 21/Mar/13 3:37 PM

Updated: 16/May/13 3:36 PM

Scheduled Deployment Date: 21/Mar/13

Development

Create Branch

To create a code branch in Bitbucket or Stash, click the Create Branch link

Agile

View on Board

Analytics

Activity

Contributions

1. Innovative Go tool built for providing efficient data collection and analysis; open sourced on GitHub [2]
2. One of the few studies in the field that performs a quantitative analysis rather than a qualitative one
3. One of the very few research projects that investigates such a large number of tickets (over 300,000) extracted from 38 different projects
4. To our knowledge, the first study to conduct sentiment and grammar correctness analyses on software tickets

Research Questions

- Does the presence of attachments and their type (e.g. code snippet, screenshot) influence the **Time-To-Close** for a ticket?
- Does the presence of stack traces reduce **Time-To-Close**?
- Does the presence of steps to reproduce reduce **Time-To-Close**?
- Is there a relationship between the number of words in comments and **Time-To-Close**?
- Does the total number of words in summary and description have an impact on **Time-To-Close**?
- Does the number of grammar errors in summary, description and comments have an effect on **Time-To- Close**?
- Does a positive or negative ticket influence its **Time- To-Close**?

Related Work

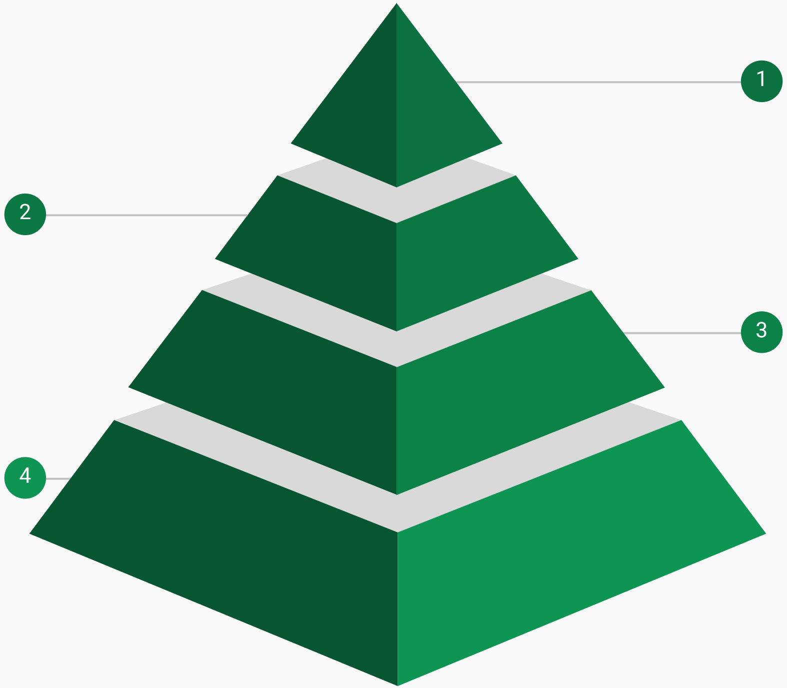
- Bettenburg et al. [1] - qualitative analysis through interviewing developers about what makes for high quality tickets; developed Cuezilla for predicting quality
- Hooimeijer et al. [3] - analyzed over 25,000 tickets; found that readability, attachments and comments can significantly increase the quality of a ticket
- Schroeter et al. [4] - investigated the effect stack traces have on ticket lifespan; around 60% of tickets with stack traces were fixed in one of the methods in the frame, 40% in the first frame
- Bettenburg et al. [5] and Prifti et al. [6] looked at bug report duplicates and their consequences; found that they actually bring value to the project, duplicates usually offering extra information not found in master report

Building the Data Set

- Needed a tool for fetching, storing, analyzing, plotting graphs and running statistical tests on tickets
- Should be fast
- Should have support for multiple databases
- Should have great HTTP support
- Should have plenty of libraries
- Solution? Go application called Ticket Guru (amazing mascot to the right 🙌)

The Almighty Ticket Guru

Ticket Guru Structure



Statistical Tests Command

Run Welch's T Test and Spearman Rank Correlation Coefficient on tickets.

Plot Command

Automatically plot bar charts and scatter plots using the tickets stored in the DB.

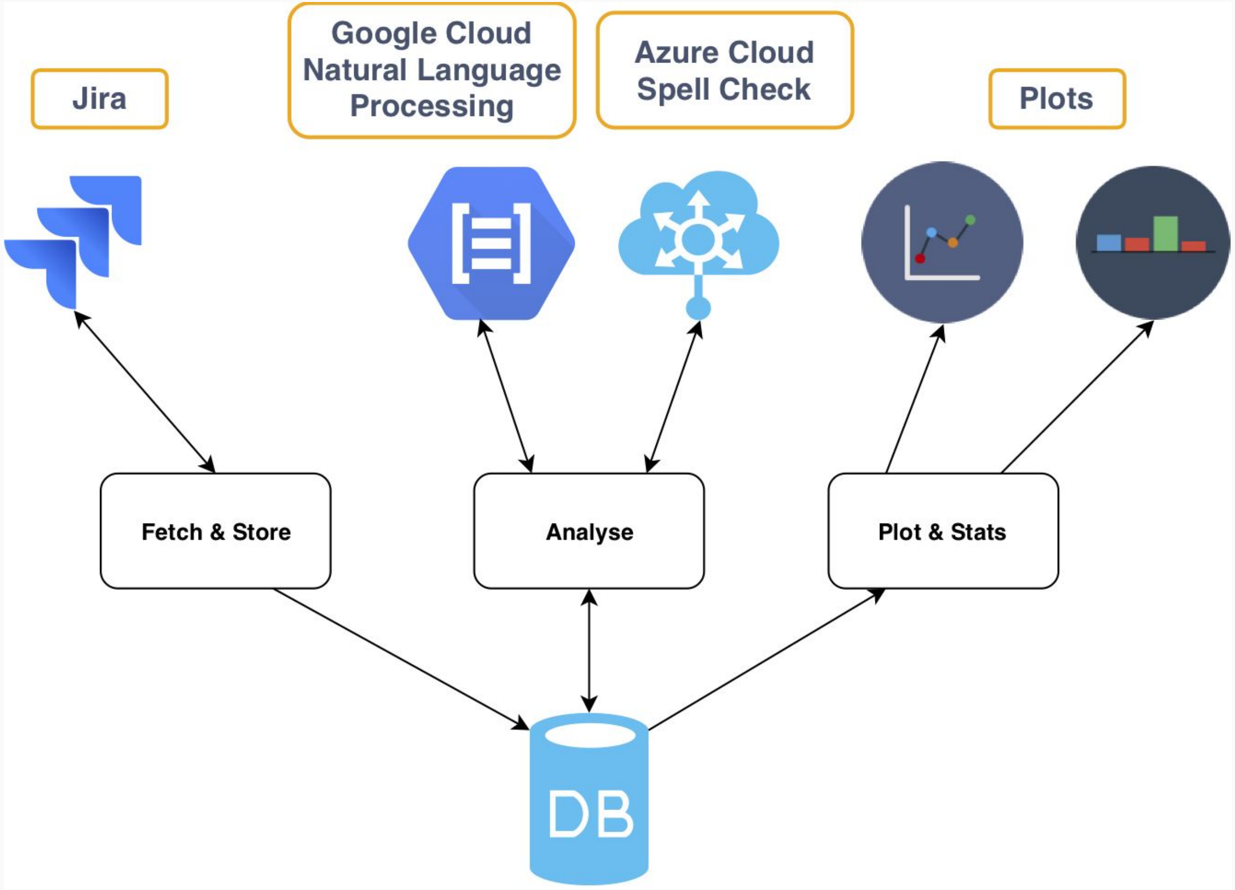
Analyze Command

Analyze tickets (e.g. calculate sentiment score) and store them back inside the DB.

Fetch & Store Command

Connect to Jira, fetch the tickets and concurrently store them in the DB.

Ticket Guru Flow



Characterizing the Data Set

303,138

Total number of tickets

- All collected from Apache Jira
- 38 different projects
- Different programming languages, ranging from Java to Python and Ruby

236,383

Closed tickets

Tickets marked **Closed, Resolved, Done** or **Completed** when they were fetched.

201,786

Tickets eligible for analysis

- Closed tickets
- High Priority (i.e. Critical, Blocker, Major, High)
- No outliers

Characterizing the Data Set

270,907

Tickets with comments

With at least one comment in the discussion thread of the ticket.

39,988

Tickets with Steps-To-Reproduce

Used complex regex [7] to determine whether the tickets had Steps-To-Reproduce in summary, description or comments.

1,942

Tickets with Java stack traces

Made use of technique outlined by Bettenburg et. al [7] to determine whether tickets had stack trace(s) in description or comments.

Characterizing the Data Set

157,047

Tickets with sentiment scores

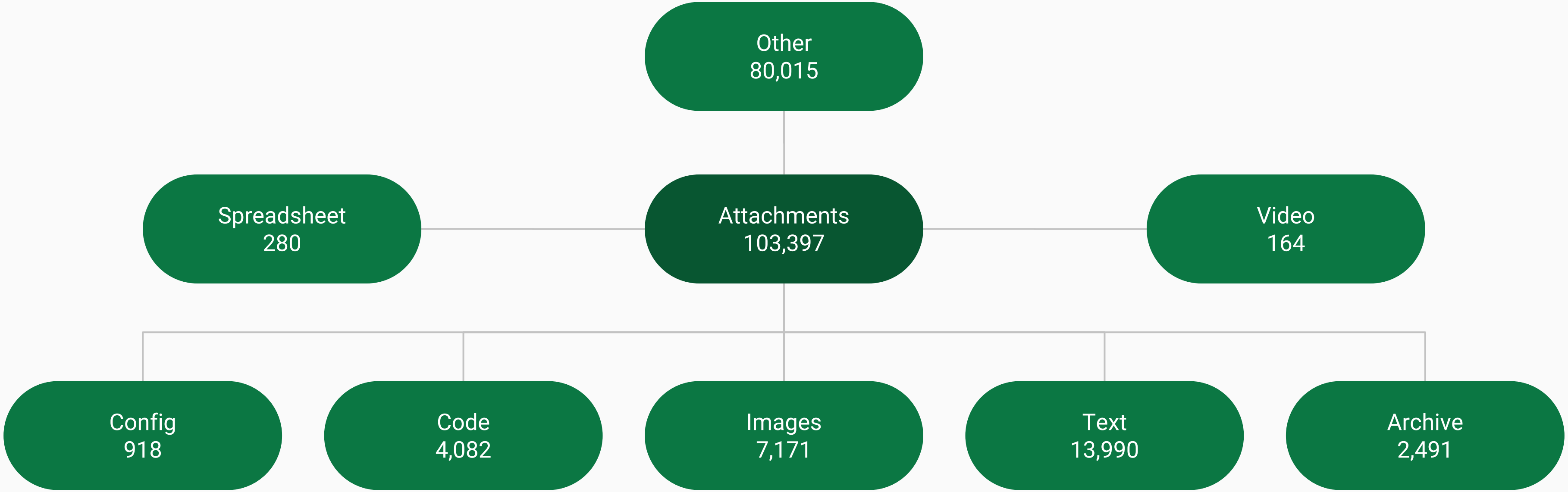
Sentiment scores were calculated using Google’s Natural Language Processing API.

133,689

Tickets with grammar correctness scores

Grammar correctness scores were calculated using Azure Cloud Bing Spell Check API.

Characterizing the Data Set


Other
80,015

Spreadsheet
280

Attachments
103,397

Video
164

Config
918

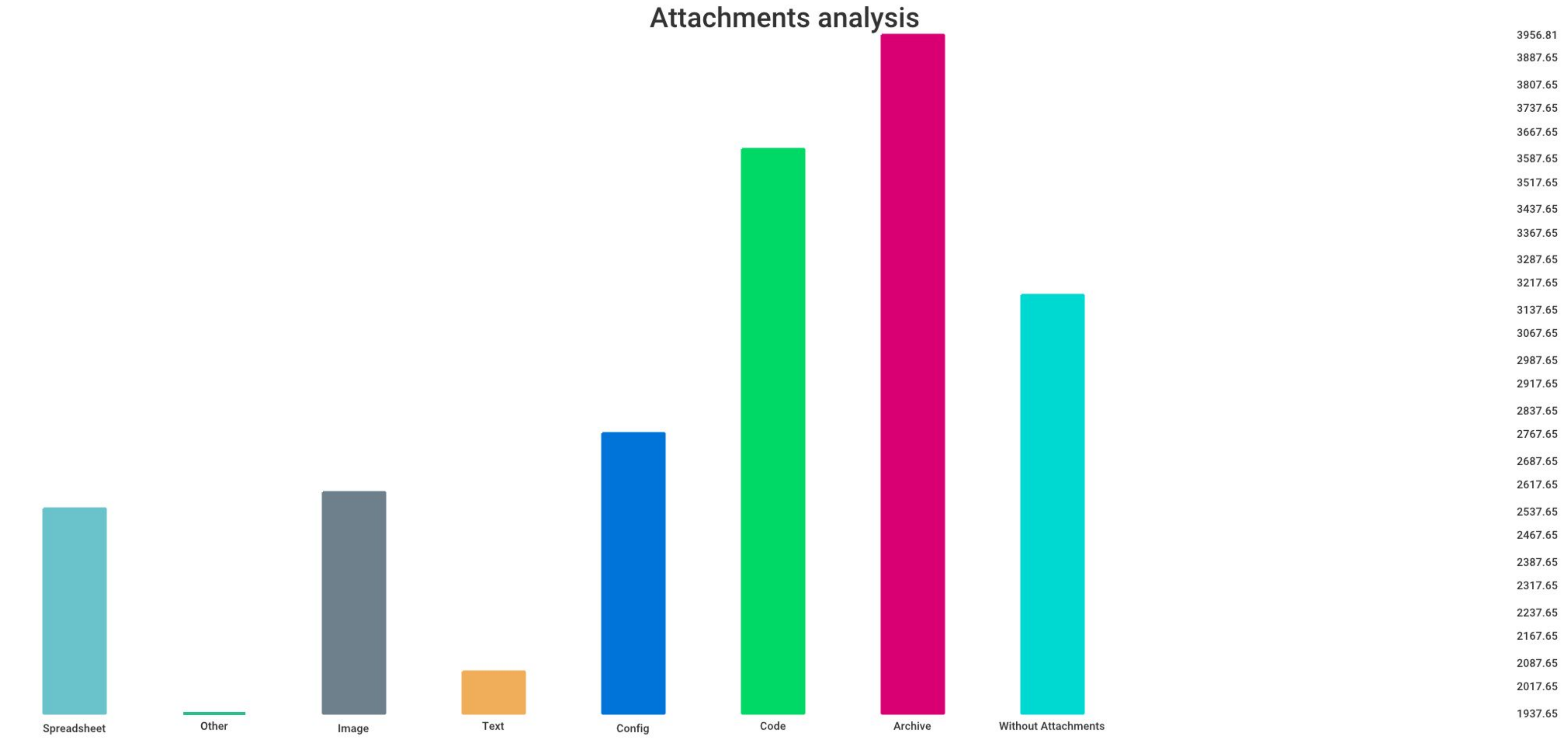
Code
4,082

Images
7,171

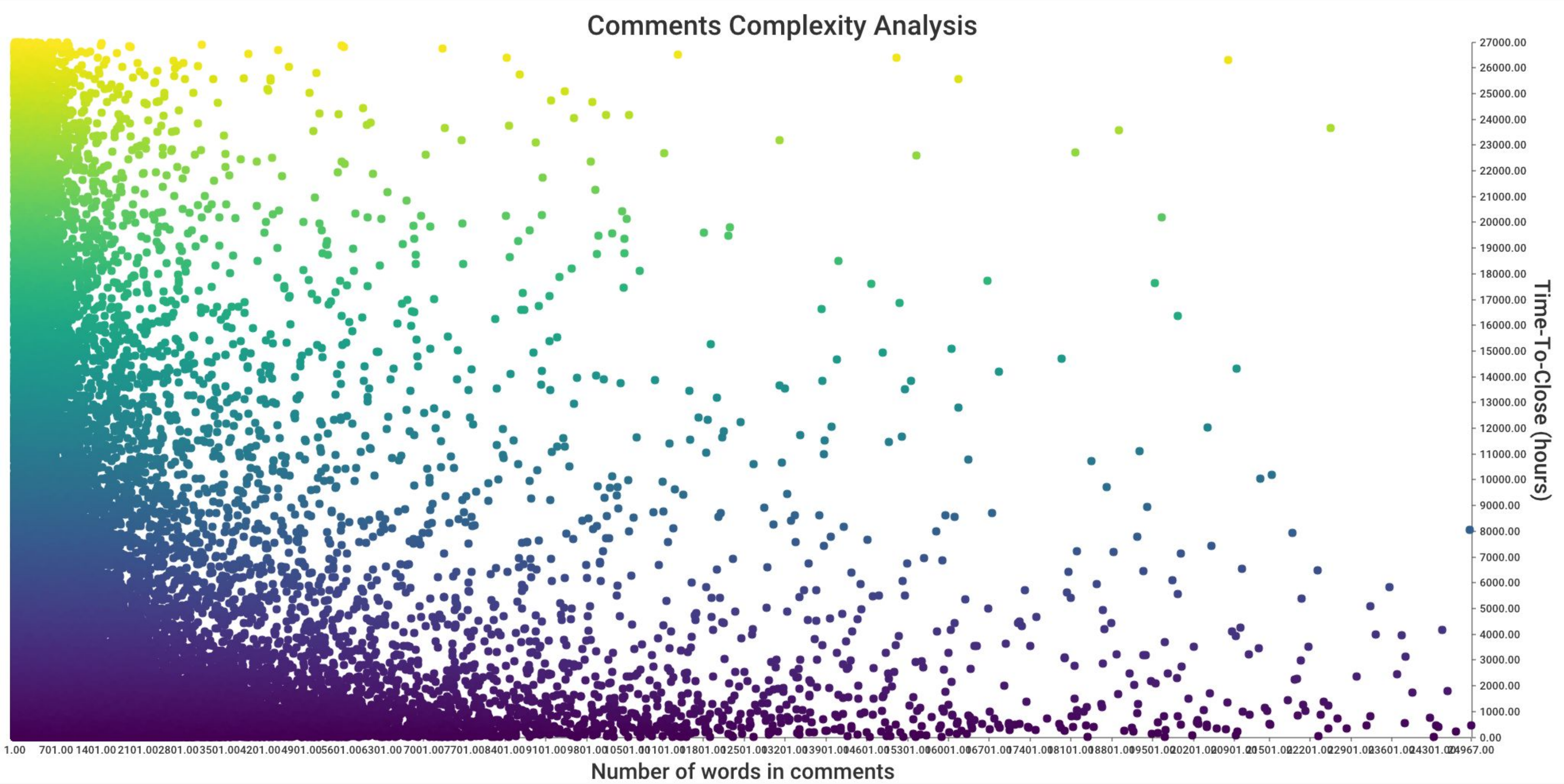
Text
13,990

Archive
2,491

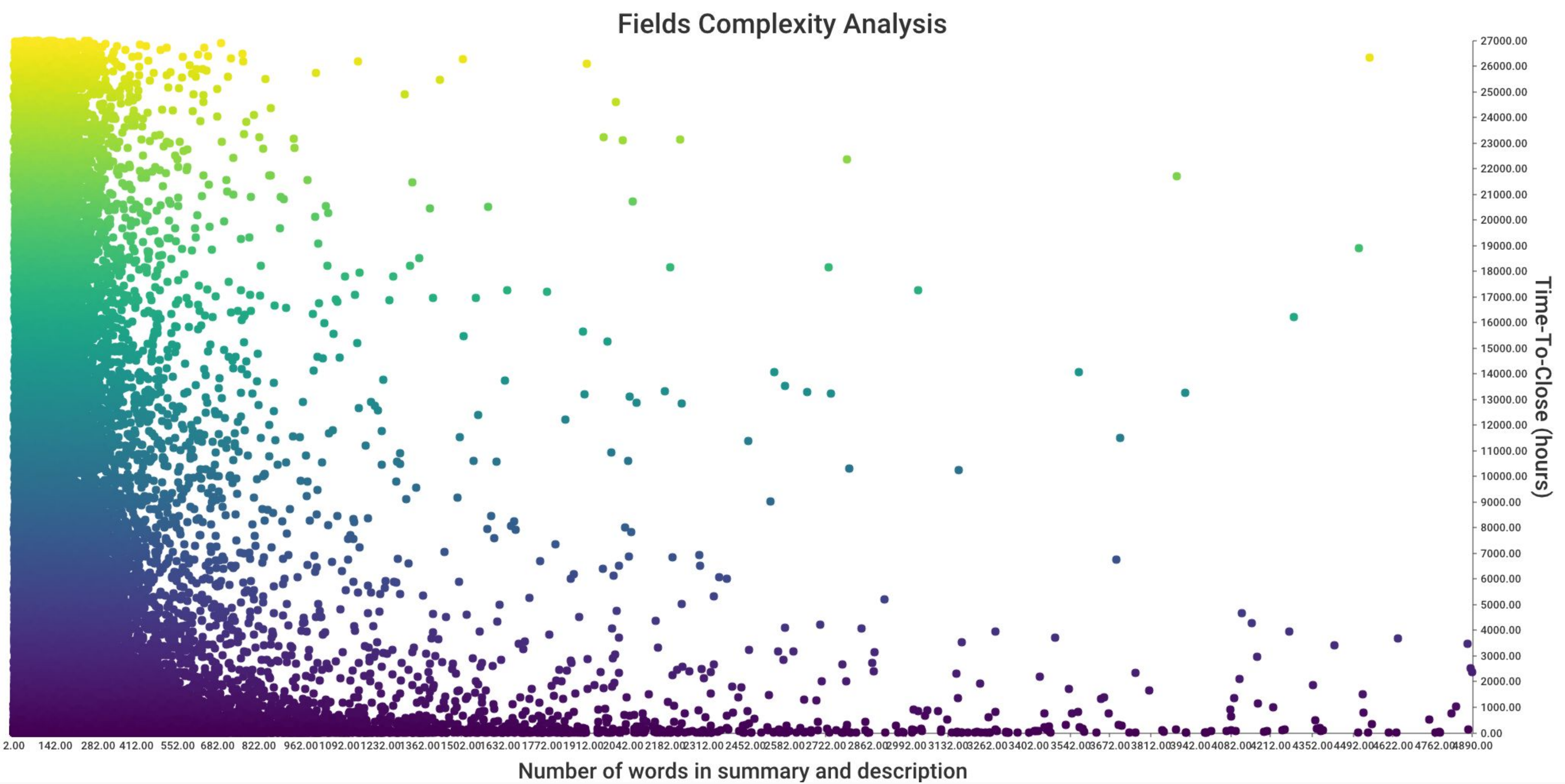
Correlations



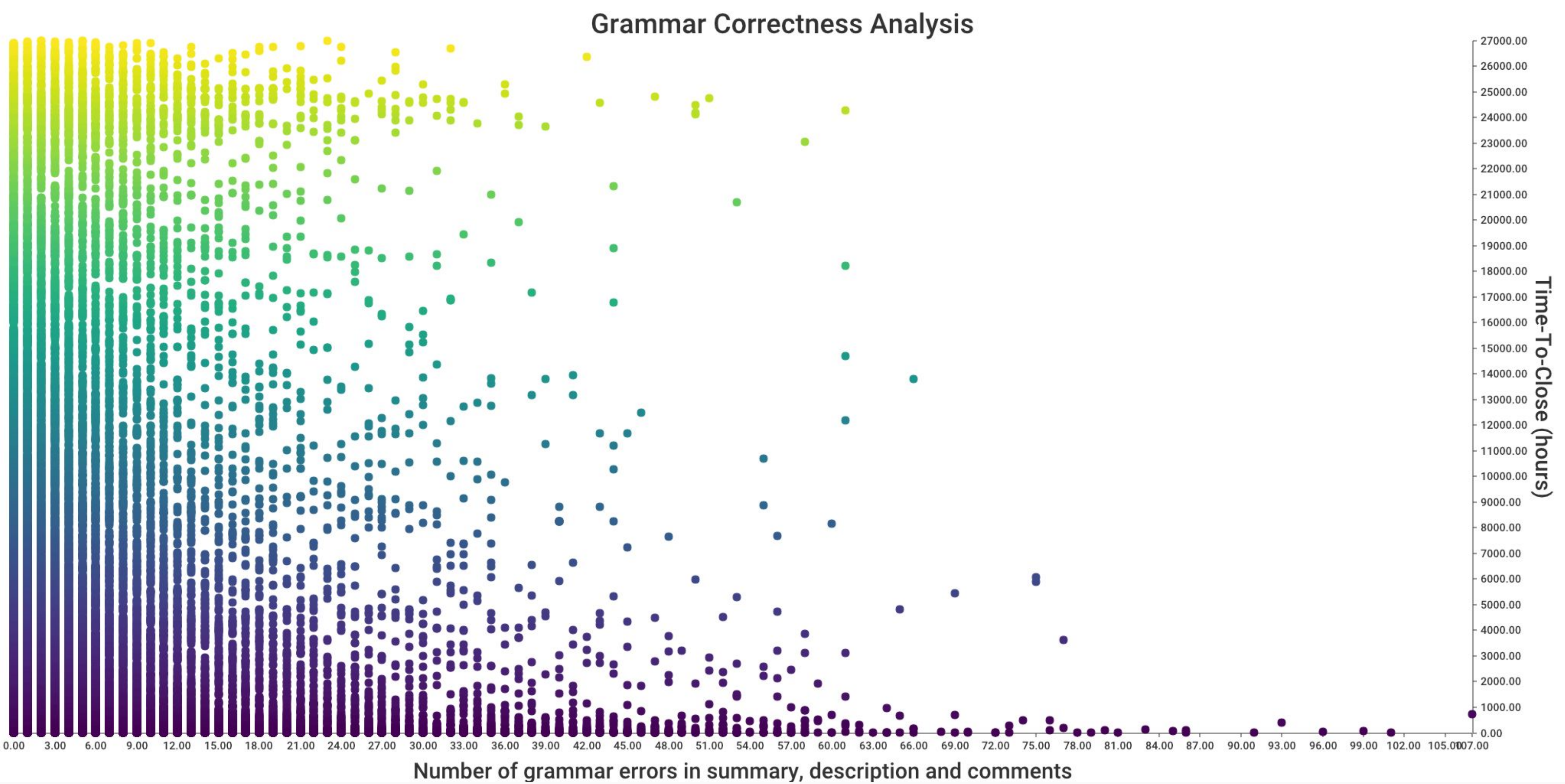
Attachment Analysis - Statistically Significant 👌



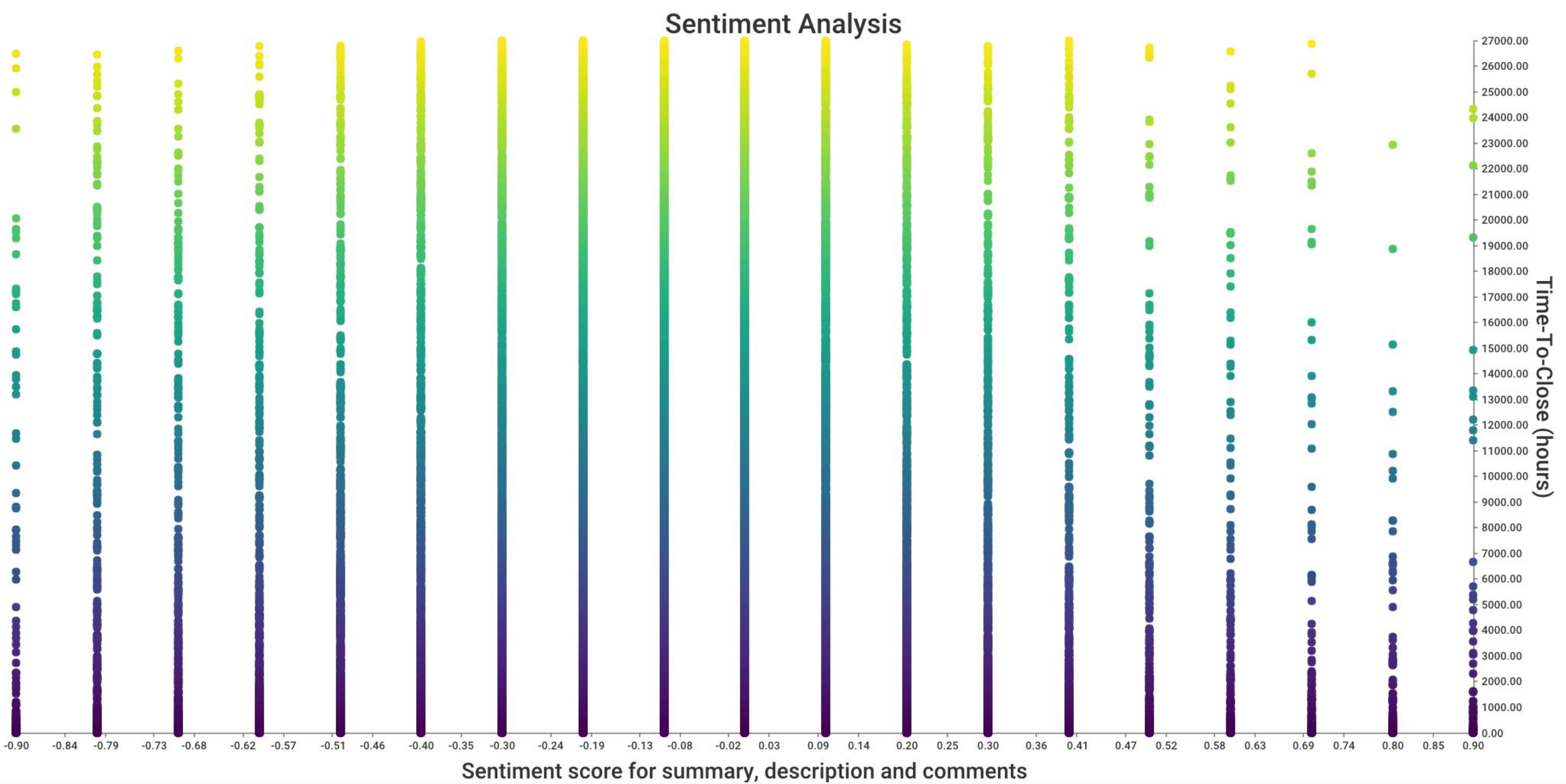
Comments Complexity Analysis - Statistically Significant 👌



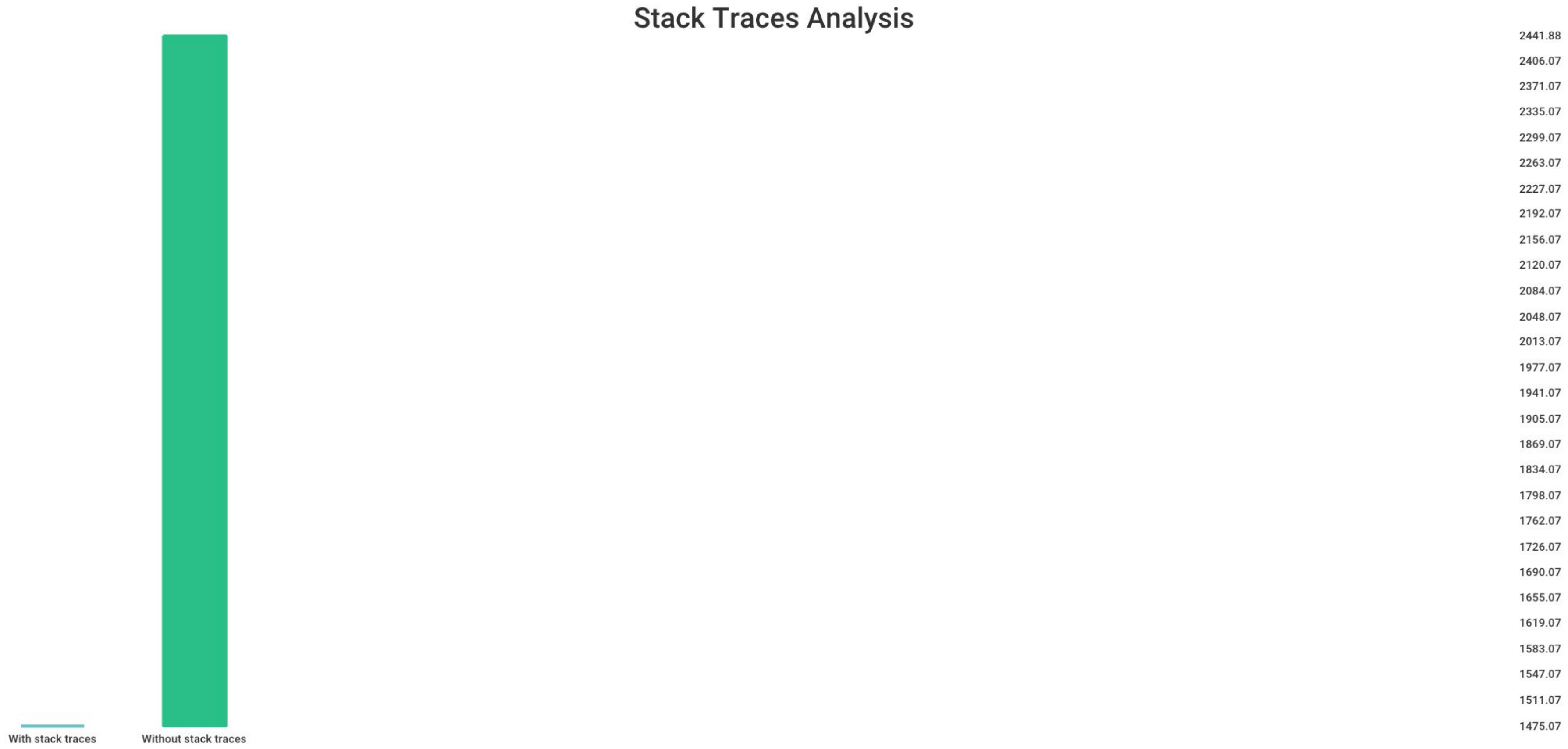
Fields (Summary & Description) Complexity Analysis - Statistically Significant 👌



Grammar Correctness Analysis - Statistically Significant 👌

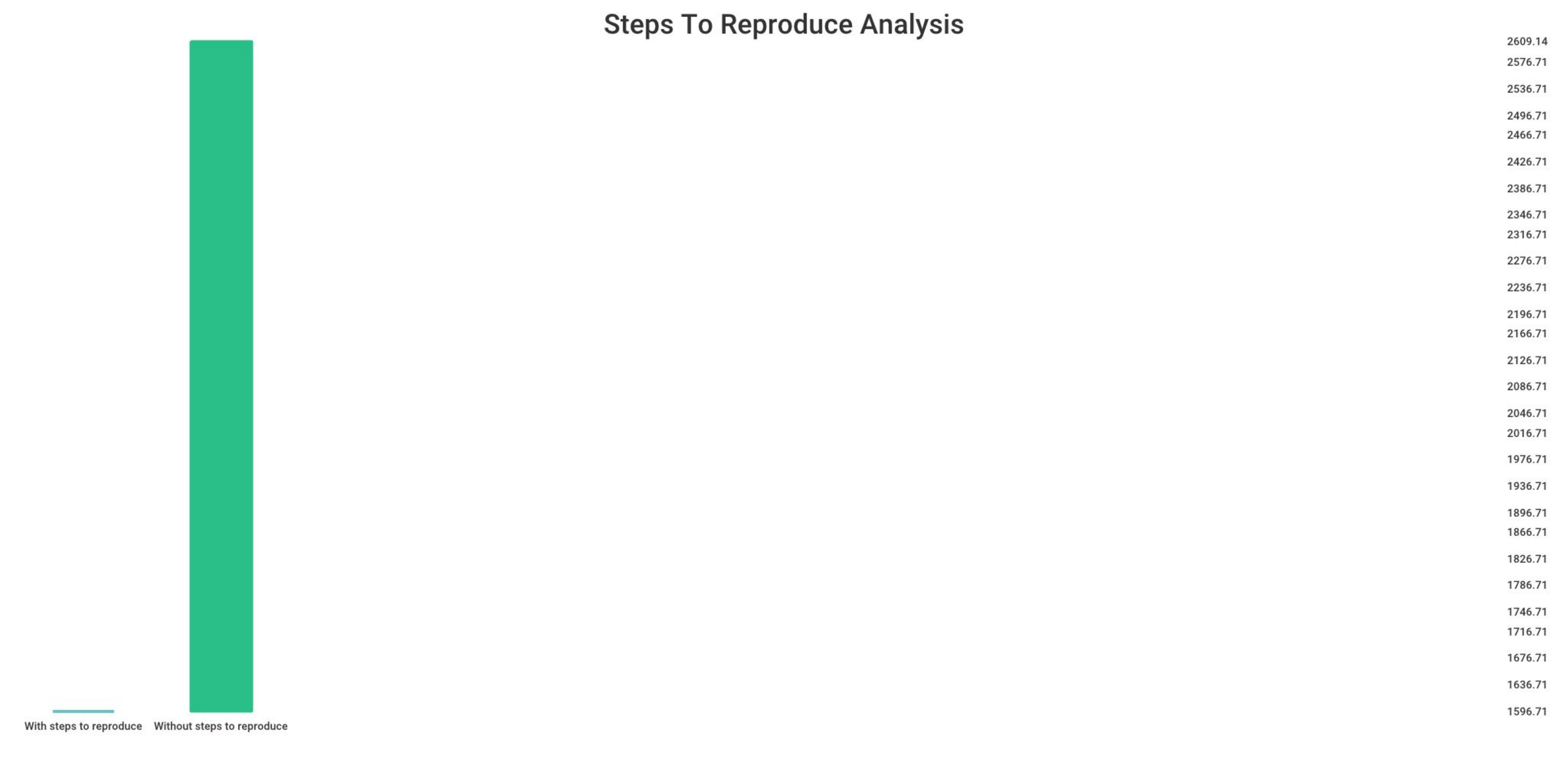


Sentiment Analysis - Statistically Significant 👌



Stack Traces Analysis

Stack Traces Analysis - Statistically Significant 👌



Steps to Reproduce Analysis - Statistically Significant 👌

Future Work

- Obtain Google Cloud Platform credits for more comprehensive analysis
- Determine ticket difficulty from information inside it - currently not known how to do it
- Create a *goodness* metric or a *recommender tool* that can automatically create high quality tickets
- Add support for Bugzilla
- Increase the current database of tickets
- Include closed source projects in the analysis

Conclusions

- Software Tickets are a vital part of every software project lifecycle
- Deriving quality score is not trivial
- Answered all RQs with statistically significant results
- Bring valuable contributions to future work in this area

Thank you!

1. N. Bettenburg, S. Just, A. Schroeter, C. Weiss, R. Premraj, and T. Zimmermann. What makes a good bug report? Pages 308–318, 2008
2. https://github.com/nclandrei/ticketguru
3. P. Hooimeijer and W. Weimer. Modeling bug report quality. pages 34–43, 2007
4. N. Bettenburg, S. Just, A. Schroeter, C. Weiss, R. Premraj, and T. Zimmermann. What makes a good bug report? pages 308–318, 2008
5. N. Bettenburg, R. Premraj, T. Zimmermann, and S. Kim. Duplicate bug reports considered harmful: really? pages 337–345, 2008
6. T. Prifti, S. Banerjee, and B. Cukic. Detecting bug duplicate reports through local references. In Proceedings of the 7th International Conference on Predictive Models in Software Engineering, page 8. ACM, 2011
7. N. Bettenburg, R. Premraj, T. Zimmermann, and S. Kim. Extracting structural information from bug reports. pages 27–30, 2008
8. J. D. Herbsleb. Global software engineering: The future of socio-technical coordination. In 2007 Future of Software Engineering, pages 188–198. IEEE Computer Society, 2007