SoMEF: A Framework for Capturing Software Metadata from its Documentation

Allen Mao, Daniel Garijo, Shobeir Fakhraei

University of Southern California, Information Sciences Institute

2019 IEEE BigData REU Symposium

Contact: allenmao AT berkeley DOT edu
Computational Sciences have increasingly become a fundamental scientific approach

- But the continuous development of new software makes it hard to keep track of or evaluate different software (or even versions)
- As a result, scientists spend much time poring through software documentation and code to understand how to use it or cite it.
- This process is time consuming and unappealing to scientists.
Easier Understanding, Reuse, and Attribution of Scientific Software

- We present SoMEF, a *Software Metadata Extraction Framework* that automatically extracts relevant software metadata from its documentation.

  ![README](https://github.com/whimian/pyGeoPressure)

  \[ \text{README} \xrightarrow{\text{SoMEF}} \{ \text{description, installation, invocation, citation} \} \]

- Examples\(^1\):
  - **Description**: A Python package for pore pressure prediction...
  - **Installation**: `pip install pygeopressure`
  - **Invocation**: `import pygeopressure as ppp`
  - **Citation**: Yu, (2018). PyGeoPressure: Geopressure Prediction in Python. *Journal of Open Source Software*, 3(30), 992, [https://doi.org/10.21105/joss.00992](https://doi.org/10.21105/joss.00992)

\(^1\)[https://github.com/whimian/pyGeoPressure](https://github.com/whimian/pyGeoPressure)
Approach

- Corpus consisted of plain text manual annotations on READMEs from 74 GitHub repositories.
- Scientific software dominates and Awesome curations of provided links to curations of scientific projects from different fields\(^2\).

**Table:** Byte fractions of languages in collected repositories

<table>
<thead>
<tr>
<th>Language</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>C++</td>
<td>32.66%</td>
</tr>
<tr>
<td>Python</td>
<td>31.16%</td>
</tr>
<tr>
<td>Jupyter Notebook</td>
<td>23.39%</td>
</tr>
<tr>
<td>JavaScript</td>
<td>4.70%</td>
</tr>
<tr>
<td>HTML</td>
<td>2.79%</td>
</tr>
<tr>
<td>Lasso</td>
<td>1.00%</td>
</tr>
<tr>
<td>Go</td>
<td>0.70%</td>
</tr>
<tr>
<td>Other</td>
<td>3.61%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>100%</strong></td>
</tr>
</tbody>
</table>

- Set of four binary classifiers with one for each category.

\(^2\)https://awesome.re/
Corpus Preparation

- **Corpus Composition**: For each binary classifier, corpus transformed where category to be predicted becomes **True** and all others become **False**.

- Random sentences from Treebank also serve as control sentences to ensure classifiers do not devolve into code vs text.

- **Balancing the Corpus**: Per corpus of classifier, all negative categories contribute equally to the 50% negative class.

<table>
<thead>
<tr>
<th>Truth Value</th>
<th>Category</th>
<th>Apprx. Ratio</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>Description</td>
<td>0.5</td>
<td>275</td>
</tr>
<tr>
<td>False</td>
<td>Installation</td>
<td>0.125</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>Invocation</td>
<td>0.125</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>Citation</td>
<td>0.125</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>Treebank</td>
<td>0.125</td>
<td>68</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>Total</strong></td>
<td><strong>1.0</strong></td>
<td><strong>547</strong></td>
</tr>
</tbody>
</table>
Data Preparation

- Tf-idf Vectorizer with unigram features.
- Since command line inputs and computer language lexicons are precise, no stemming or stop words used.
- TF-IDF matrix with 1509 features.
- Each feature is a “word”, i.e. space-delimited string of characters.
Classifiers

- Two classifiers from the *Scikit-learn* package.
- Logistic Regression *liblinear* solver and balanced class weights because of small corpus size and imbalances that may arise from undersampling.
- Multinomial Naive Bayes (MNB) additive smoothing parameter of $\alpha = 1$ as default fail-safe probability.
Results

- Five-fold cross validation
- $\geq 0.90$ ROC AUC and $\geq 0.92$ Average Precision across all categories
- Logistic Regression and MNB performed similarly per same category
- On average, citation and installation classifiers performed best.
Extraction of Other Software Metadata

- Component that returns metadata from GitHub repositories with GitHub REST API as a JSON.

<table>
<thead>
<tr>
<th>GitHub Repository</th>
<th>SoMEF</th>
</tr>
</thead>
</table>

- description
- forks
- license
- programming languages
- license
- name
- owner
- README URL
- releases
- topics
Conclusion

Summary

- SoMEF is a novel approach that employs to extract software metadata.
- Promising initial steps: minimum average 0.92 precision and 0.90 ROC AUC.

Future Work

- Corpus Expansion
- Text Separation
- Other linguistic features
- Knowledge graphs of scientific software
Acknowledgements

A big thank you to the following:

- Dr. Daniel Garijo
- Dr. Shobeir Fakhraei
- Professor Yolanda Gil
- Professor Jelena Mirkovic
- The National Science Foundation
- The REU program
Questions?