Eliminating the Paradox of Choice in Software Development

COMP 4520 – Undergraduate Honours Project Proposal

Darshan Ajay Pandhi
Department of Computer Science
University of Manitoba
Winnipeg, Manitoba Canada
pandhid@myumanitoba.ca

1 Introduction

Analysis paralysis is a phrase for being overwhelmed due the presence of many options to choose from. Make no mistake as this term is pretty relevant in the case of Software Development although there is little to no research in this area. Developers often experience decision fatigue given the myriad number of choices at every step. From choosing programming languages and making system-wide design decisions to something as simple as naming a variable or timing the need for refactoring, future-proofing or even addressing the ever-growing technical debt for instance. Choice Overload [1]hampers productivity and breeds inefficiency.

In the current state of things, finalizing important project variables like programming languages and frameworks to be used typically involves through research and investigation by the lead developer (or equivalent) before the start of the undertaking. Manual selections are made keeping in mind the following factors:

- Familiarity
- Documentation / Ease of Use
- Community Support
- Scalability
- Long term viability / Maintainability
- Licensing / Cost
- Reliability
- Security

This costs significant time and money, not to mention the fact, that this can be erroneous or subject to change given the human aspect attached.

Hence, the goal of this paper is to eliminate some of this conflict and simplify the lives of fellow developers by means of a revolutionary data-driven recommendation system. A system that intelligently suggests the best choice to take given a problem statement.

All in all, it will be capable enough to suggest the following:

- Programming languages
- Libraries / Frameworks / Packages

Such a system will help hone workflows and improve development quality and quantity since automating underrated choices like these means one less thing for human developers to worry about.

2 Dataset

Google BigQuery hosts a public dataset [2]comprising the largest released source of GitHub activity to date. This 3TB+ dataset contains a full snapshot of the content of more than 2.8 million open-source GitHub repositories. To start things off, I will focus on specific domains like mobile and web applications so as to constrain the project complexity.

Usually, projects on GitHub have a readme file that introduces the project itself. In addition, they may contain additional documentation through wiki pages covering details like third party frameworks / libraries used. In terms of the source code itself, configuration files can be a good place to start at in order to programmatically understand the project composition effectively.

3 Methodology

Following is a very high-level overview of the potential steps involved.

[1] The term "choice overload" was coined by Alvin Toffler in 1970

[2] Link: https://console.cloud.google.com/marketplace/product/github/github-repos?filter=solution-type:dataset&project=local-catalyst-300819

3.1 Filtering

Firstly, I will filter out all projects that don't meet the above criteria as they won't be helpful in improving the recommendation system.

Assumption: Post filtering, we have at least 10k repositories / projects available to properly train the recommendation system.

Fortunately, BigQuery uses familiar SQL to then query the database and assemble the relevant information.

3.2 Data Visualization

This step will involve understanding the dataset and inferring trends /patterns to formulate valuable insights. For instance: Assess popularity of frameworks with respect to time answering the most pertinent question (in the minds of developers) as to which framework is getting outdated while which one is still cool to use.

3.3 Data Analytics

Finally, given a problem statement, I determine whether it is possible to build a recommendation system accurate enough to outsource vital development choices.

Note that I will focus on text mining (natural language processing) i.e. project description, limiting actual source code mining (given the time and complexity constraints associated with it).

In essence, the recommendation system will match and leverage similar projects to the input problem to accurately output the best choices.

4 Deliverables

As part of the Undergraduate Honours Project Course, I will be aiming to deliver the final paper and a presentation summarizing the research. My target is to also publish the paper.

In addition, I will be attending various software development conferences to further my understanding and gain exposure.

5 Future Work

The trends and patterns identified with programming languages and libraries / frameworks / packages shall prove to be extremely valuable to spawn off independent research studies.

Besides, this idea of the recommendation system is pretty extensible. It can be programmed to support additional choices like

- IDEs / Code Editors and Extensions
- Architecture
- Development Tools and Software …,

making it the ultimate tool for software development.

A future iteration could employ Machine Learning techniques to better rank the different options and increase accuracy. Likewise, source code mining could be an avenue to explore in addition to the existing text mining.