Pogo: A Programming Project Advisor

Progress Report

Maneet G, Mansi M, Neetha R, Aditi M[1], Vijaysri L, Harsh K[1]

I. PROBLEM DEFINITION

In today's world of constantly evolving technology, choosing the right programming tools to build a project can be daunting. Users need to Google, search Stack OverFlow (SO), GitHub etc. separately to implement programming projects. With Pogo, users have a one-stop application which speeds up their learning.

To build Pogo, we fetch SO, Libraries.io and GitHub data from Google BigQuery and integrate them using topic modeling and graph analytics. On a user's programming task query, Pogo offers insights into relevant technologies. Each technology is accompanied with information depicting its popularity, degree of assistance and any learning dependency it might have. With these insights users can make informed decisions on technology best suited for them. Pogo thereby reduces time and effort spent in the planning and learning phase of a project. Hence, people from both academia and industry would find the tool beneficial.

II. LITERATURE SURVEY

A. StackOverflow

A recent research established that trends in SO data are strongly correlated to trends in Google searches [1]. Based on this, we can rely on SO data to calculate popularity and assistance metrics of various technologies.

Latent Dirichlet Allocation (LDA) helps in figuring out themes describing the scope of a document [2], [17]. Integral to Pogo, LDA shall map users' query to relevant SO entries. Linares-Vasquez et al. reported its use for figuring out hot topics related to mobile development [3]. One of our research questions (RQ) is greatly aligned with this study. However, we plan to generate more accurate results over a larger dataset by figuring out a near-optimal model input configuration rather than adopting the baseline one. One such method, LDA-GA takes into account documents' cohesion and separation for describing a configuration's fitness, i.e., the Silhoutte Index [4].

Research done on SO tags also suggests that a discriminative model via Supporting Vector Machines (SVM) can be used to suggest, or in our case predict tags based on existing ones [5]. Using SO for predicting most successful answers to a user query, we can use sentiment analysis and logistic regression to get best results [6].

B. GitHub

Another RQ is aligned with determining popularity of repositories. Hudson Borges [9] discusses factors that contribute to higher star count on a GitHub project and identifies a pattern of popularity growth by clustering time series data. In a similar study, regression analysis was performed to model how folder usage relates to project popularity [10]. Current practices discretize data using k-means and measure success rate by cardinality of downloads [11]. This gives an interesting pattern associating open source projects and owner features with the success rate. As proposed by Cai [12], we plan to employ a graph based approach (GRETA) to assigning tags for repositories using domain knowledge from SO. This would allow GitHub repositories to be efficiently accessed and understood.

Detecting repositories that are similar to each other is integral to Pogo. Research suggests user's starring repositories and repository read-me files are good indicators of similarities [16]. Combining multiple regression modeling with visualization and text analytics helps suggest which program-

College of Computing, [1]College of Engineering, Georgia Institute of Technology, Atlanta, GA 30318, USA

ming language can achieve the best software quality [13]. Sometimes data can be easily misunderstood. Recommendations exist on how to best use data available from GitHub and avoid analysis risks [14].

C. Neo4j

Representation and efficient querying of interconnected data are key challenges to address, while building an effective, interactive tool. The paper representing time-varying social network data as a property graph in Neo4j database produces good performance results in querying, exploratory data analysis, and research-oriented data mining [7]. Results show Neo4j to be a high-performance replacement for relational databases, especially when handling highly interconnected data. Studies have also reported empirical comparison between two graph query languages to access data in Neo4j, Cypher and Gremlin [8], [15] among which Cypher is shown to be more user friendly. Hence, we are using Cypher with Neo4j to build the dependency graph.

III. PROPOSED METHODS

Our input form takes in a detailed description of the project the user wants to carry out as shown in Figure 1. All input fields are drop-down menus having appropriately classified (see III-A) SO tags as their options. Scope of input may vary with programmer experience.

Topic-Modeling: The input will then be assigned a topic probability-distribution using our LDA model trained over the entire SO dataset (only **tags** column). At the back end, we have different buckets of questions (same number as our topics). Our model's output in response to user input will help us map our query to one or more of those buckets. We will then assess parameters like average response time for those questions, proportion of questions with accepted answers, etc. to generate an assistance score which will give the user an estimate of the degree of support he can receive from the community if he chooses to proceed with his project.

Graph-Analytics: Another set of Pogo's features are powered by large graphs of SO tags and of Github repositories (see III-D). Pogo will recommend packages based on package description from the Library.io dataset and the input programming language. Implementation strategy for recommendations is under discussion.

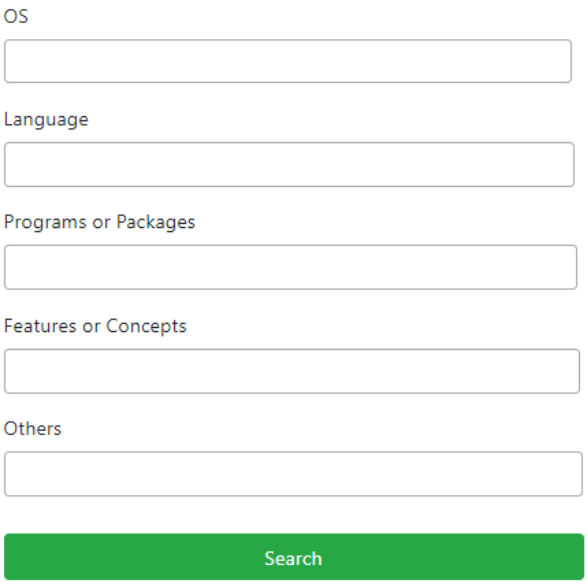


Fig. 1. A Component from the Input Form GUI

The following activities divide the work above into sub-tasks which when integrated together help build Pogo:

A. Tag Classification

A list of tags that appear more than 100 times on SO were collected using the SO dataset on Google BigQuery. To understand dependencies and fetch cleaner results, tags are classified into one of the following categories: OS, Language, Package, Feature. Using the libraries_io dataset, 199 SO tags were classified under 'Language', 7619 under 'Package', 99 under 'OS' and 1175 under 'Feature'. Additional benefit of classifying tags is in figuring out relationships between them, which will be exploited to generate input suggestions in our input form and enhance UX.

B. Cleaning SO Data

Stack Overflow Data is being queried from the Google BigQuery Data Set and being run through a parsing script to extract text elements from the titles and bodies of various questions. Text elements extracted are then run through a preprocessing script to split them into tokens and tag them as

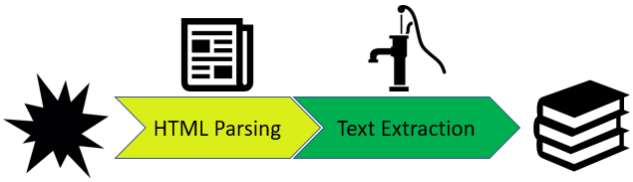


Fig. 2. Incomplete HTML to Raw Text

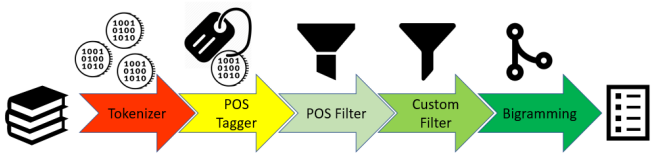


Fig. 3. Raw text to Doc

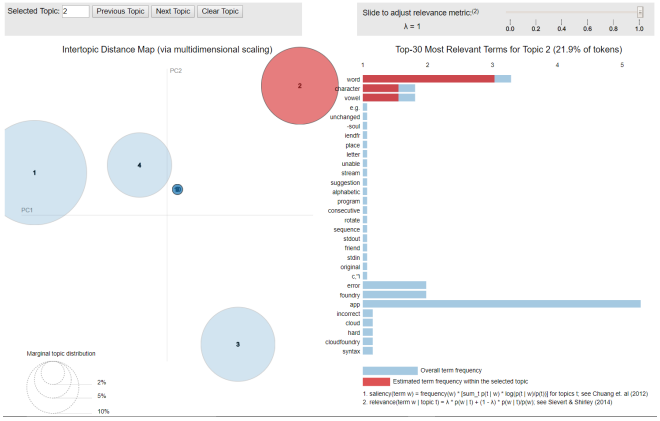


Fig. 4. gensim LDA model results and topics formed on subset

appropriate parts of speech using Python's spacy library. Once preprocessing is done, the input texts turn into a list of lists which is then used to form a dictionary of words and a corpus (bag-of-words format) all using the **gensim**, a topic modelling library for Python.

C. Running and Evaluating LDA results

We have also begun performing test runs on gensim's LDA model over small subsets of the SO data set to verify our implementation (see Figure 4). We will have 2 LDA models, one trained on just question tags and the other on title and body of questions.The topics generated by the LDA model will form the basis for most of our predictive capabilities such as Assistance Score and Hot Topics.

D. Dependency Graph

For a better user experience, pogo displays to users a graph showing the learning dependencies that their query might have. To achieve this, package names and their dependencies (collected from libraries_io dataset) are loaded as nodes and relationships into Neo4j's graph database. An example of what the graph looks like can be seen in Fig.3. If a user is interested in learning about 'dynaTree', for example, a query is made on database to display all dependencies that 'dynaTree' might have Fig.4.

E. Distributed Computing

Distributed computing accelerates computations by splitting a given task into smaller subtasks, and computing them in parallel. The Pyro library in Python will be used. To speed up the process, the worker script will be run on each cluster node prior to performing LDA. This tells gensim that it may use the node as a slave to delegate work to it. Pyros name server is run on exactly one of the machines, using command: $ python -m Pyro4.naming -n 0.0.0.0 & Subsequently gensim lda_worker.py script is run on the worker node: $ python -m gensim.models.lda_worker &. This tells gensim that it can run two jobs on each of the computers in parallel, reducing computation time, while also taking up twice the memory on each machine. This would set up the cluster running and ready to accept jobs.

IV. INNOVATION

A. Integrating data from SO, GitHub and Library.io together in the form of a learning aid.

Such a tool is not yet available based on our survey. If successful, Pogo can bring down learning-curves for new programmers, and enhance productivity for experienced ones. The performance of this tool can be measured via user feedback and number of on-line hits.

B. Using Distributed Computing to develop a scalable and deployable predictive system.

Due to the large volume of data and the strain an LDA model puts on a single system (400 MB file can take up to 2 hours for modeling), our proposed solution is to solve this by distributed computing to ensure speed and scalability.

Fig. 5. Visualizing dependencies

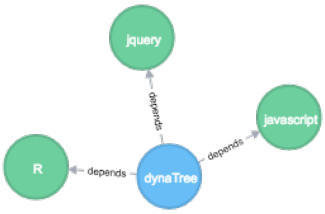


Fig. 6. Querying for dependencies (Ex: Querying all the dependencies of *'dynaTree'*)

C. Using graph databases such as Neo4j to help facilitate better decision making.

The implementation of Neo4j in tandem with our dataset will open many avenues of opportunity in terms of ease of use and better decision making with regards to the packages and concepts that are interrelated and the ones our user will be best served learning.

V. HEILMEIER QUESTIONS

1) Q1-Q4: Section I.
2) Q5: Section III.
3) Q6-Q7: Section VI.
4) Q8-Q9: Section V.

VI. PLAN OF ACTIVITIES

The work involved in building Pogo is evenly distributed among all six members in the team.

VII. EXPERIMENTS

Pogo is aimed at providing intuitive information related to a user's query. Primary plan of evaluation includes querying the system with known inputs and assessing the intuitiveness of the output. The results obtained will be assessed based on two criteria: relevance of the results with respect to user's input and how informative the visual representation of the output is to a user. The second step of evaluation is to conduct a controlled survey by collecting feedback from users in academia as well as industry.

VIII. CONCLUSION

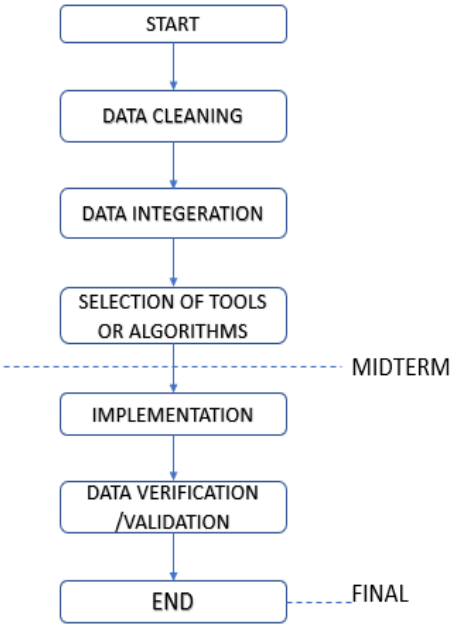
Under Construction.

IX. APPENDIX

Group Member Responsibilities:

1) Maneet: SO Data Cleaning/Text Preprocessing, LDA Training, Pogo GUI
2) Harsh: SO Data Cleaning/Text Preprocessing, LDA Training and Evaluation
3) Mansi and Neetha: Tags Cleaning and Classification, Using Neo4j to visualize dependency graph, Use Cypher to query graph database.

Old Plan of Activities:



REFERENCES

[1] Chen, C., & Xing, Z., 2016, Towards Correlating Search on Google and Asking on Stack Overflow. In proceedings of COMPSAC, the IEEE Computer Society's International Computer Software & Applications Conference (pp 83-92).

[2] Blei, D.M., 2012. Probabilistic topic models. Communications of the ACM, 55(4), pp.77-84.

[3] Linares-Vsquez, M., Dit, B. and Poshyvanyk, D., 2013, May. An exploratory analysis of mobile development issues using stack overflow. In Proceedings of the 10th Working Conference on Mining Software Repositories (pp. 93-96). IEEE Press.

[4] Panichella, A., Dit, B., Oliveto, R., Di Penta, M., Poshyvanyk, D. and De Lucia, A., 2013, May. How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms. In Proceedings of the 2013 International Conference on Software Engineering (pp. 522-531). IEEE Press.

[5] Saha, Avigit K., Ripon K. Saha, and Kevin A. Schneider. "A discriminative model approach for suggesting tags automatically for stack overflow questions." Proceedings of the 10th Working Conference on Mining Software Repositories. IEEE Press, 2013.

[6] Calefato, Fabio, et al. "Mining successful answers in stack overflow." Mining Software Repositories (MSR), 2015 IEEE/ACM 12th Working Conference on. IEEE, 2015.

[7] Ciro Cattuto, Marco Quaggiotto, Andre Panisson, Alex Averbuch. "Time-varying Social Networks in a Graph Database: a Neo4j use case." GRADES '13 First International Workshop on Graph Data Management Experiences and Systems. ACM, 2013.

[8] Florian Holzschuher, Rene Peinl. "Performance of graph query languages:comparison of cypher, gremlin and native access in Neo4j". Extending Database Technology (EDBT) '13 Proceedings of the Joint EDBT/International Conference on Database Theory (ICDT) 2013 Workshops. ACM, 2013.

[9] Hudson Borges, Andre Hora, Marco Tulio Valente. "Understanding the Factors That Impact the Popularity of GitHub Repositories". Software Maintenance and Evolution (ICSME), 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME),2016.

[10] Jiaxin Hu, Minghui Zhou, Audris Mockus. "Patterns of folder use and project popularity: a case study of github repositories".ESEM '14 Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement,2014.

[11] Fragkiskos Chatziasimidis, Ioannis Stamelos. "Data collection and analysis of GitHub repositories and users".Information, Intelligence, Systems and Applications (IISA), 2015 6th International Conference on,2015.

[12] Xuyang Cai, Jiangang Zhu,Beijun Shen. "GRETA: Graph-Based Tag Assignment for GitHub Repositories".Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual,2016.

[13] Ray, B., Posnett, D., Filkov, V. and Devanbu, P., 2014, November. A large scale study of programming languages and code quality in github. In Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (pp. 155-165). ACM.

[14] Kalliamvakou, E., Gousios, G., Blincoe, K., Singer, L., German, D.M. and Damian, D., 2014, May. The promises and perils of mining github. In Proceedings of the 11th working conference on mining software repositories (pp. 92-101). ACM.

[15] Huang, H. and Dong, Z., 2013, November. Research on architecture and query performance based on distributed graph database Neo4j. In Consumer Electronics, Communications and Networks (CECNet), 2013 3rd International Conference on (pp. 533-536). IEEE.

[16] Zhang, Y., Lo, D., Kochhar, P.S., Xia, X., Li, Q. and Sun, J., 2017, February. Detecting similar repositories on GitHub. In Software Analysis, Evolution and Reengineering (SANER), 2017 IEEE 24th International Conference on (pp. 13-23). IEEE.

[17] Jie Zou, Ling Xu, Weikang Guo, Meng Yan, Dan Yang, Xiaohong Zhang. "Which Non-functional Requirements Do Developers Focus On? An Empirical Study on Stack Overflow Using Topic Analysis." Mining Software Repositories (MSR), 2015 IEEE/ACM 12th Working Conference. IEEE, 2015.