Three Interesting Papers Related to Software Requirements

HW1 Task3 of Software Requirements Engineering

1st Yunfeng Shen
College of Computer Science and Technology
Zhejiang University
Hangzhou, China
3200104392@zju.edu.cn

The Google scholar cite of all three articles is at the end of the whole file.

I. Deep Learning Based Program Generation From Requirements Text: Are We There Yet?[3]

A. Summary

With the development of deep learning and Natural Language Processing (NLP) technology, to generate source code from natural language by applying deep learning-based approaches has been claimed to have high accuracy. Whereas, the author doubted the result for it is evaluated on datasets which are rather small, monotonous and significantly different from real-life software requirements.

Thus, the author gathered submissions from two programming contest platforms, i.e., Codeforces and HackerEarth. By removing duplications, filtering up-to-date submissions and removing low-quality ones, they successfully built an large scale dataset with high quality. Besides, to facilitate research on code generation, the authors also develop an assisting tool which area able to automatically computes the list of quality metrics for reference programs.

They chose Seq2Seq, SNM, Tree2Tree, TRANX and Coasr-to-Fine as approaches to be evaluated. After training, testing, and evaluating with the tool kit, they gained the following results:

- The length of programs has a significant negative impact on the performance of automated code generation and all the evaluated approaches show significant reduction in performance when applied on the new dataset.
- Compared with other approaches, AST-based code generation approaches have a great chance to generate syntactically correct Python programs. However, only a few programs are executable because of various runtime exceptions.
- To generate an excutable program, manual interference (especially code revision and validation) is indispensable.
- Modifying the generated programs is not significantly easier than creating programs from scratch.
- A simple *popular-based approach* might outperform the state-of-the-art deep learning-based approaches concerning the common performance metrics bilingual evaluation understudy.

B. Why it is interesting

1) In contrast to the optimism that others have consistently expressed about machine learning, the authors offer a unique challenge: *Are we there yet*? Instead, he looked for simpler solutions and got the best results. This made me think, is the application of AI technology necessarily better? Would there be a simpler and more efficient way to solve it?
2) The author find that modification of given program is not easier than writing it from scratch.

II. Recommending Software Features for Mobile Applications Based on User Interface Comparison[1]

A. Summary

Requirement elicitation is always a significant part of software development, and the emergence of app stores provide a new platform for developers to gather requirements and perform market-wide analysis. Instead of doing interviews, observation or workshops, nowadays, software developers are able to dig potential user requirements from user review or API information in source code. However, those data do not include direct description of the implemented features, and the efficiency s constrained by multiple external conditions. Therefore, a new and efficient demand mining technology is urgently needed.

A data-driven approach for recommending software features of mobile applications based on user interface comparison has been brought about. Using text similarity to measure the similarity of the appointed UI and app UIs, this approach can efficiently recommends related features which can be considered for inclusion in forthcoming release by following the steps below:

1) *UI Similarity Calculation:* Do words splitting by using Ansj for Chinese sentences, and a regular expression for English ones, and Tongyici Cilin, WordNet::Similarity to

compute text similarity respectively. Apply the genetic algorithm (GA) to calculate the similarity of UIs.
2) *Feature Identification:* Using visible texture of components as features.
3) *Sort and Recommend:* Classify features and rank the cluster in descending order of feature number it has.

Leave-on-cut validation shows that: the proposed method is meaningful and effective for recommending popular features.

B. Why it is interesting

1) Focused on new demand gathering platforms - app stores and new requirement sources - trends in competing products, which ensures the necessity and feasibility of requirements and reduces sunk costs.
2) A new way of comparison is proposed - UI-based comparison.
3) Use text-based analysis instead of image-based analysis.

III. Does Sentiment Help Requirement Engineering: Exploring Sentiments in User Comments to Discover Informative Comments[2]

A. Summary

User feedback is a typical way of expressing the opinions of the user, and is a valuable source for analysis. As the communication channel between developers and users, app stores process a large amount of information on user needs to be discovered. And the rise of NLP has significantly reduced the cost of analysis, but the limitation of its own algorithm to avoid a large number of meaningless comments has led to inefficiencies.

It have been reported that, *negative* comments are more useful compared to others while neutral sentiments have not yet been analyzed in more detail. To address these problems, a representative opinion-mining technique is needed. After excluding useless data, they are able to execute sentiment analysis via SentiStrength and dividing them into 4 types: *Weak Positive*, *Strong Positive*, *Weak Negative*, *Strong Negative*. Then, the y use the Latent Dirichlet Allocation to measure the informativeness of the comments. At last, they measured sentiment and informative score of topics and do the evaluation.

The results are as follows:

Negative and *StrongEmotional* could be expected to have high informativeness. Since comments that simply express *negative* or *Strong emotional* sentiments are quite uncommon, informative comments are densely collected in very few datasets, which means filtering out informative comments using sentiments has the potential for effective results.

B. Why it is interesting

1) The research significantly reduces the pressure on operations and maintenance teams to scroll through meaningless user comments to uncover user requirements.
2) It enables more effective filtering and analysis of user comments by classifying the positivity or negativity of user sentiments and the intensity of emotions.

References

[1] Xiangping Chen et al. “Recommending software features for mobile applications based on user interface comparison”. In: *Requirements Engineering* 24.4 (2019), pp. 545–559.

[2] Jongwook Jeong and Neunghoe Kim. “Does sentiment help requirement engineering: Exploring sentiments in user comments to discover informative comments”. In: *Automated Software Engineering* 28.2 (2021), pp. 1–26.

[3] Hui Liu et al. “Deep learning based program generation from requirements text: Are we there yet?” In: *IEEE Transactions on Software Engineering* 48.4 (2020), pp. 1268–1289.