Resumo do artigo "Neural Software Analysis"

André Baião, Gonçalo Barradas, Guilherme Grilo

Fevereiro 2022

Resumo

Devido à evolução dos softwares e ao aumento da quantidade de dados tratados nos dias de hoje é necessário tornar as ferramentas e as técnicas para os programadores num fator importante para a economia, pois estas ferramentas visam tornar os programadores mais produtivos através da automatização em vários projetos em desenvolvimento.

Apesar do sucesso da maneira convencional de construir ferramentas para programadores que é a análise de programas baseada em raciocínio lógico e preciso, existem muitos problemas que apenas podem ser parcialmente resolvidos, ou seja, classificar uma resposta como totalmente precisa e correta para programas não triviais não está correto. Assim, a análise do programa deve-se aproximar do comportamento do programa analisado, muitas das vezes com ajuda de heurísticas cuidadosamente elaboradas.

Embora as análises convencionais que têm por base a lógica, sejam geralmente elaboradas por especialistas em análise de programas, as análises de software neural baseia-se numa aprendizagem a partir de dados fornecidos.

1 Descrição do artigo analisado

Neste artigo faz-se uma apresentação do "Neural Software Analysis" que, é uma forma alternativa de ajudar os programadores de software a resolverem problemas nos seus respetivos projetos, utilizada principalmente em códigos mais elaborados onde o pensamento lógico não alcança os resultados mais precisos ou corretos.
O artigo está dividido em 10 secções:

Secção 1 Título do artigo;

Secção 2 O Resumo contém, de forma sucinta, os temas abordados ao longo do artigo como por exemplo a necessidade da análise de softwares neuronais;

Secção 3 Conhecimentos-chave,
destaca três definições/ideias importantes a reter para melhor entendimento dos assuntos tratados neste artigo.

Secção 4 Quando usar a análise de software neuronal,
de forma geral, todos os problemas de análise de programa podem ser formulados com base no raciocínio lógico ou com base no aprendizado orientado a dados, como a análise neuronal de software. Existem três dimensões para determinar se deve-se usar a análise de software neuronal.

1. Os modelos neuronais têm a capacidade de identificar padrões, por isso devem ser utilizados sempre que a informação for imprecisa;
2. Os modelos neuronais têm a capacidade de encontrar heurísticas adequadas, sendo essencial quando os dados fornecidos não têm critérios de correção definidos;

3. Os modelos neuronais necessitam de modelos de exemplo para aprender, por isso sempre que o número de dados seja suficiente deve ser utilizado;

Secção 5 Uma estrutura conceptual para analise de software neural,
inicialmente os exemplos de código adequados ao problema são extraídos. Os exemplos de código são transformados em vetores, como tokens ou árvores de sintaxe abstrata. De seguida, o modelo é treinado com recurso aos exemplos fornecidos. Após o treino, é iniciada a previsão, onde o modelo gera previsões que podem ou não ser sujeitas a validação.
Representação de software em vetores.
Os modelos neuronais trabalham com vetores de números, sendo de extrema importância a transformação dos exemplos de código extraído em vetores.
Representação de tokens de código: A melhor abordagem é o mapeamento de cada token num vetor de incorporação de tamanho fixo, com o objetivo de representar tokens semanticamente semelhantes.
Representação de trechos de código: A técnica mais simples e eficaz consiste em mapear cada token num vetor tendo por base os tokens de código. Sendo ainda possível a utilização de técnicas que representam trechos de código como gráficos de vetores.
Modelos neuronais de software.
A entrada dos vetores no modelo é efetuada em duas etapas. O resumo/codificação do código-fonte, e a previsão sobre o código.
Resumo da fonte do código. Normalmente o código é resumido através de uma rede feedforward que faz a concatenação dos vetores e mapeia-os num vetor mais curto.
Fazer uma previsão. Geralmente os modelos são modelos de classificação, que podem ser binários ou N-ário que prevê as várias classes existentes no código.
Treino: O treino é efetuado através de pesos e viés, utilizando várias funções que otimizam o modelo;

Secção 6 Exemplos de análises de software neuronais,
como análises necessárias na deteção de bugs, adaptação consoante linguagens e a finalização de códigos parcialmente completos.

Secção 7 Perspetivas e desafios,
a introdução da análise de software neuronal é algo bastante recente, por este motivo, investigadores e programadores de software tentam explorar e evoluir esta mesma forma de análise para que, no futuro possa vir a ser a principal substituta da análise com base na lógica.

Secção 8 Conclusão,
a análise de software neuronal é uma ideia bastante ambiciosa e já mostrou resultados bastantes promissores, pois facilita a resolução de problemas, reduzindo assim o esforço da análise desses mesmos problemas.

Secção 9 Agradecimentos;

Secção 10 Referências;

Lacomis, J., Yin, P., Schwartz, E., Allamanis, M., Goues, C., Neubig, G., and Vasilescu, B. DIRE: A Neural Approach to Decompiled Identifier Naming. ASE, 2019.

Li, Y., Wang, S., and Nguyen, T. DLFix: context-based code transformation learning for automated program repair. ICSE, 2020.

Referências

Pradel, M., Chandra, S., Neural Software Analysis of the Communications of the acms, Vol. 65 No. 1, 86-96 (January 2022) `https://cacm.acm.org/magazines/2022/1/257449-neural-software-analysis/fulltext#`.
Last accessed 21 January 2022