TabNine - IA

Fábia Alves, Felippe Luna, Rafael Cajé
FATEC – São José dos Campos – Prof. Jessen Vidal
fabia.alves@fatec.sp.gov.br
giuliano.bertoti@fatec.sp.gov.br

Figura 1 - Logo Tabnine

1. Introdução

O Tabnine foi criado com base em um modelo de inteligência artificial que usa Machine Learning para sugerir o que provavelmente será digitado, em tempo real focada no uso de programadores. Ou seja, medida que você começa a digitar, ele usa da inteligência para sugerir o fim da linha de código, evitando que você digite manualmente. Nesse artigo veremos mais a respeito dessa ferramenta e o quanto ela pode influenciar positivamente no ambiente profissional de muitos programadores.

2.Início

A Tabnine, foi fundada por Eran Yahav e Weiss em 2017. O nome “Tabnine” veio de uma startup sediada em Waterloo com o mesmo nome que a Codata adquiriu em 2019. Também foi anunciada o levantamento de US$ 15,5 milhões em financiamento da Qualcomm Ventures, OurCrowd e Samsung NEXT Ventures, com participação de investidores existentes que incluem Khosla Ventures, Headline Ventures, Hetz Ventures e TPY Capital. Com esta última injeção de capital, a Tabnine levantou um overall de US$ 32 milhões até o momento.

O Tabnine afirma ser a única ferramenta de desenvolvimento de device com Inteligência Synthetic desse tipo no mercado

Com base no trabalho em análise e simulação de código, percebe-se que, com a grande quantidade de semelhanças e padrões padrão no código, era inevitável que a IA fosse uma parte crítica no processo de desenvolvimento. Sendo assim foi, o Tabnine, estabelecidos como pioneiros na categoria de assistente de código de IA.”

3. Suporte

O Tabnine suporta cerca de 11 linguagens de programação tais como Python, JavaScript, Java etc. Disponibilizado gratuitamente para muitas IDEs como IntelliJ PyCharm, VS Code, Sublime, VIM, Atom e muitas outras.

O objetivo é democratizar o treinamento do modelo de IA e disponibilizá-lo para todos. Os desenvolvedores não estão limitados aos modelos de IA fornecidos pelo Tabnine, porque qualquer pessoa pode treinar seus próprios modelos conectando-se ao GitLab, GitHub ou Bitbucket e, em seguida, treinando o modelo de IA que reflete as melhores práticas para um projeto específico.

Isso permite que os líderes de desenvolvimento dimensionem as boas práticas de codificação interna e tenham mais controle sobre a origem do código-fonte.

O aumento da velocidade de desenvolvimento melhora a experiencia de trabalho do desenvolvedor, permitindo que as equipes de trabalho respondam mais rapidamente às oportunidades de mercado. Com isso os usuários podem obter um modelo de IA personalizado com base em seu código privado que permite:

- Compartilhamento de conhecimento;
- Dívida técnica reduzida;
- Revisões de código mais rápidas;
- Integração mais rápida com oTime to Value.

O valor de um modelo personalizado ajuda uma equipe especifica com uma missão especifica a ser mais produtiva. Já que o que todas as equipes têm em comum é o interesse compartilhado em uma base de código comum.

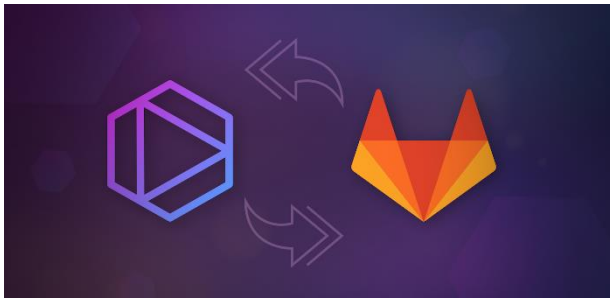

Figura 2 - Logo Tabnine e GitLab

4. Segurança

As ferramentas de preenchimento automático de código com inteligência artificial melhoraram nos últimos anos, mas ainda falta um problema importante. O Tabnine, por exemplo, não "observam o estado de segurança do código que examinam, nem o código que produzem", diz Greg Young, vice-presidente de segurança cibernética da Trend Micro.

A tarefa de verificar o código está nas mãos dos desenvolvedores. "O código fornecido não deve ser considerado totalmente seguro, a menos que realizemos uma análise de segurança exaustiva em cada função e seus usos", diz Nitescu. Os desenvolvedores devem

realizar revisões de código de segurança automatizadas e manuais e seguir as melhores práticas de domínio definidas em qualquer documentação de linguagem específica antes de usar o código gerado em ambientes de produção.

Eventualmente, os geradores de código com inteligência artificial podem evoluir para filtrar bugs e até sugerir implementações seguras, mas ainda há um longo caminho pela frente. Por enquanto, desenvolvedores e mantenedores de código são responsáveis pela segurança do código que enviam.

Outra preocupação é a segurança e privacidade. Embora todos eles garantam que não vazarão seu código-fonte, ainda precisamos fazer concessões entre segurança e produtividade antes de decidir usar ferramentas completas de código inteligente. Alternativamente, há um expediente que você pode escolher produtos completos de código inteligente de “Grandes” empresas.

Figura 3 - Inteligência Artificial e a Segurança de dados

5. Usabilidade

Outra abordagem para geração de código é a síntese de programa mais tradicional baseada em pesquisa. À medida que a tecnologia de síntese de programas amadurece, torna-se cada vez mais comum avaliar a usabilidade de sintetizadores em seres humanos. Muitos desses estudos de usabilidade são para sintetizadores de domínio específico direcionados à navegação de API, expressões regulares, raspagem da web ou consulta de dados, discussão e visualização. Esses estudos geralmente se concentra em medir o efeito da ferramenta nas taxas e tempos de conclusão da tarefa, o que é menos relevantes para as nossas perguntas.

O trabalho sobre RESL e Snippy incluem estudos de usuários de sintetizadores de programação por exemplo de propósito geral para JavaScript e Python. Embora ambos também se concentrem principalmente nos tempos de conclusão das tarefas, eles fazem algumas observações qualitativas interessantes. Por exemplo, Ferdowsifard et al. [2020] observe que uma das principais barreiras à utilidade do sintetizador é a chamada lacuna usuário-sintetizador, ou seja, a superestimação do programador das capacidades do sintetizador.

6. Programadores exploram quando confiam no modelo

Um estudo feito por uma empresa de software concorrente do Tabnine (Copilot) identificou algumas características dos usuários de softwares sintetizadores.

[4.2.2] O nível de confiança de um usuário e a empolgação com os modelos de geração de código é altamente correlacionada com qual medida em que se envolveriam na exploração. Durante a tarefa de treinamento, o Copilot produziu um grande, sugestão correta para P18 (P18 é um dos usuários estudados); eles exclamaram: "Não vou ser um desenvolvedor, vou ser um cara que faz comentários!" Esse nível de empolgação foi compartilhado por muitos de nossos participantes no início da tarefa, como P7 dizendo, “é tão emocionante vê-lo escrever [o código] para você!”. Os participantes que ficaram entusiasmados com o Copilot costumavam deixar a ferramenta guiar a solução antes mesmo de tentar resolver a tarefa por conta própria.

Às vezes, esse entusiasmo excessivo atrapalhava a conclusão de uma tarefa. Por exemplo, P10 foi o que menos avançou em relação aos outros na mesma tarefa; em nosso pós-estudo entrevista, eles admitiram que eram “um pouco dependentes demais do Copilot”:

“Eu estava tentando deixar o Copilot fazer isso por mim, talvez eu devesse ter dado tarefas menores ao Copilot e fazer o resto sozinho, em vez de depender inteiramente do Copilot.”

Esse superotimismo é característico do mal-entendido que os usuários costumam ter com os programas sintetizadores. P9 e P10 estavam atingindo a lacuna do sintetizador do usuário, que separa o que o usuário espera que um sintetizador de programa seja capaz e o que o sintetizador pode realmente fazer.

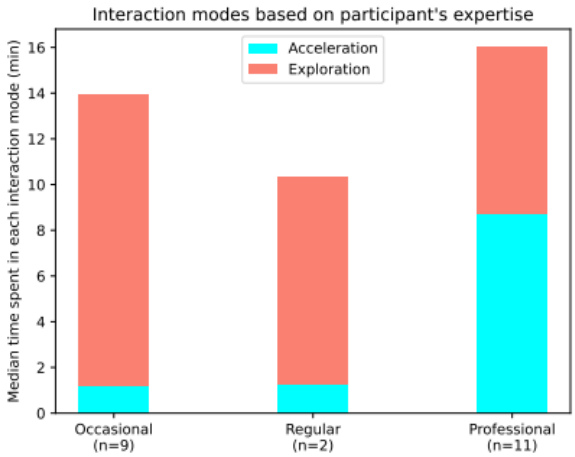

Figura 4 - Agrupados por conhecimento em linguagens

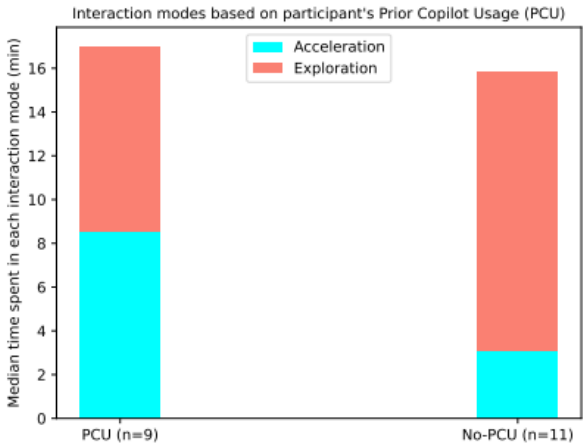

Figura 5 - Agrupados por uso anterior do Copilot

Conhecimento de linguagens: A Fig. 4 mostra o tempo médio gasto em dois modos divididos pelos participantes da perícia linguística. Podemos ver claramente que os usuários profissionais com mais domínio em linguagens gastam mais tempo acelerando do que os outros dois grupos. Isso não é surpreendente, pois é mais provável que eles já saibam como resolver a tarefa na linguagem de programação fornecida.

Uso prévio do Copilot: Podemos ver na Fig. 5 que o tempo total de interação é aproximadamente o mesmo para os participantes com e sem uso anterior do Copilot. Dado aproximadamente o mesmo tempo geral, os usuários anteriores gastam menos tempo explorando (e mais tempo acelerando) do que os usuários novatos. Atribuímos essa diferença ao efeito que observamos na Sec. 4.2.2 do estudo, onde os usuários iniciantes têm expectativas mais altas da capacidade do Copilot de resolver tarefas exploratórias de alto nível.

7. Impacto no futuro

A programação como conhecemos hoje tende a se modificar com o tempo, com o surgimento de novas tendências de mercado, como ferramentas de low-code e uso de IA auxiliando no desenvolvimento de softwares. Diferentemente do que alguns pensavam, a maioria dos softwares continuarão sendo desenvolvidos por pessoas em futuro não muito distante. Entretanto, terão o auxílio de ferramentas que garantem mais rapidez e eficiência aos projetos. Essas ferramentas de IA estão ajudando e aprimorando os humanos, não os substituindo. Além disso, estão ajudando a democratizar a codificação e o desenvolvimento de software, permitindo que indivíduos não necessariamente treinados em codificação preencham as lacunas de talentos e aprendam novas habilidades.

Também existe a revisão de código baseada em IA fornecendo garantia de qualidade antes mesmo de executar o código. Como acontece com a maioria da automação, a IA aqui acelera o trabalho. Em alguns casos, elimina certas tarefas, mas as pessoas ainda são necessárias. Muitas empresas estão adotando ferramentas de desenvolvimento em low-code como foi citado acima, que permitem que pessoas não treinadas como programadores desenvolvam aplicativos sem nenhuma linha de código. Boa parte dessas empresas observaram uma melhoria na qualidade dos produtos, além de reduzir o custo e o tempo no desenvolvimento. Em tese, podemos concluir que essas melhorias foram observadas porque a qualidade é maior. Com detecção de bugs mais rápida e precisa e mais capacidade para testar os produtos ao longo do processo de desenvolvimento, é mais provável que o software funcione melhor e seja mais fácil de usar. Com isso, o grande desafio dos líderes de tecnologia, a partir de agora, será preparar suas organizações e seus liderados para que tenham condições de encarar o futuro, que já está bem perto, com conhecimento e confiança.

Poder contar com um parceiro experiente e com expertise em soluções de TI para auxiliá-lo nessa jornada de transformação digital, pode ser uma boa opção para superar os desafios e manter-se competitivo no mercado.

Figura 6 - Inteligência Artificial é o futuro

8. Conclusão

Sob qualquer perspectiva e métrica, é inegável que a IA alcançou um tremendo sucesso, as maiores empresas tecnológicas do mundo já fazem uso efetivo de inteligências como essa (TabNine). No entanto, tal sucesso se deu pelo barateamento dos custos de processamento e de memória, surgimento de novos paradigmas, como as redes neurais profundas e a enorme quantidade de dados disponível nas redes e mídias sociais.

O desenvolvimento e o uso da IA levantam questões éticas fundamentais para a sociedade, que são de vital importância para o nosso futuro. Já existe muitos debates sobre o impacto da IA no trabalho, interações sociais (incluindo cuidados de saúde), privacidade, justiça e segurança (incluindo iniciativas de paz e guerra). O impacto social e ético da IA abrange muitos domínios, por exemplo: os sistemas de classificação de máquinas levantam questões sobre privacidade e preconceitos e veículos autônomos levantam questões sobre segurança e responsabilidade. Pesquisadores, decisores políticos, indústria e sociedade reconhecem a necessidade de abordagens que garantam as tecnologias de IA de uso seguro, benéfico e justo, para considerar as implicações da tomada de decisão ética e legalmente relevante pelas máquinas e o status ético e legal da IA.

Essas abordagens incluem o desenvolvimento de métodos e ferramentas, atividades de consulta e treinamento e esforços de governança e regulamentação.

Para encerrar, cabe relembrar uma frase do fundador da Cibernética, Norbert Wiener, que faz parte do artigo “Some Moral and Technical Consequences of Automation”, publicado na revista *Science*, em 1960:

“*Se usarmos, para atingir nossos objetivos, um órgão mecânico em cujo funcionamento não podemos interferir de forma eficaz ... é melhor estarmos bem certos de que o propósito colocado na máquina é aquele que realmente desejamos*”.

9. Referências

MARTINS, Claylson; MARTINS, Claylson. **SempreUpdate**. SempreUpdate. Disponível em: <https://sempreupdate.com.br/tabnine-usa-ia-para-ajudar-desenvolvedores/>. Acesso em: 21 set. 2022.

HEXTEC. **Tabnine atualiza o assistente de escrita de código baseado em IA para desenvolvedores**. HexTec News. Disponível em: <https://www.hextecnews.com.br/tabnine-atualiza-o-assistente-de-escrita-de-codigo-baseado-em-ia-para-desenvolvedores/>. Acesso em: 21 set. 2022.

GLEN, Stephanie. **Tabnine code completion platform adds more powerful AI**. SearchSoftwareQuality. Disponível em: <https://www.techtarget.com/searchsoftwarequality/news/252521540/Tabnine-code-completion-platform-adds-more-powerful-AI>. Acesso em: 02 out. 2022.

GitLab and Tabnine: AI-powered code completion for GitLab repositories. GitLab. Disponível em: <https://about.gitlab.com/blog/2022/03/02/bringing-ai-gitlab-repository/>. Acesso em: 19 out. 2022.

ANDRADA FISCUTEAN. **Why you can’t trust AI-generated autocomplete code to be secure**. CSO Online. Disponível em: <https://www.csoonline.com/article/3653309/why-you-cant-trust-ai-generated-autocomplete-code-to-be-secure.html>. Acesso em: 19 out. 2022.

BARKE, S.; JAMES, M. B.; POLIKARPOVA, N. **Grounded Copilot: How Programmers Interact with Code-Generating Models**. **arXiv.org**, 2022.