TP Noté 4: Linting et LLMs

I - Travail Attendu

Dans ce TP nous attendons la production d'un rapport au format PDF envoyé à <jbbesnard@paratools.fr> le Mardi suivant le TP avant 23h59. Soyez concis et efficaces mais ne renvoyez pas non plus un document en texte brut – trouvez le juste milieu. Le travail peut être fait seul ou en binôme.

Récupération des exercices:
git clone https://github.com/besnardjb/tp_linter.git

II - Notion de Linter

Le linting est un processus automatisé de vérification et d'analyse de qualité du code source, qui a pour but de détecter des erreurs syntaxiques et semblables, ainsi que des pratiques de codage inadéquates ou des violations de conventions de style.

Le principe de base du linting est d'appliquer une règle définie (ou plusieurs règles) sur le code source en analysant sa syntaxe et sa structure. Ces règles sont généralement basées sur des recommandations communément acceptées dans la communauté de développement logiciel, telles que les conventions de style de coding, la bonne pratique de codage et les meilleures pratiques.

Lorsque le linting est appliqué à un projet ou à un fichier source particulier, il examine le code en recherchant des violations des règles définies. Il peut produire des messages d'erreur ou de warning pour chaque violation identifiée. Ces messages peuvent être affichés directement dans l'éditeur de code, ou bien être écrits dans unfichier de sortie pour être consulté plus tard.

Les principaux avantages du linting sont :

- Il permet de détecter des erreurs et des incohérences dans le code dès leur apparition.
- Il facilite la conformité au style et aux bonnes pratiques de codage, ce qui améliore la lisibilité et la maintenabilité du code.
- Il aide à prévenir des bugs et des erreurs de compilation grâce à une analyse plus fine et régulière du code.

III - Notion de Large Language Model

A - Principe Général

Un Large Language Model (LLM) est un type de modèle d'intelligence artificielle conçu pour comprendre et générer du texte semblable à celui produit par les humains en fonction des entrées qu'il reçoit. Le modèle est entraîné sur de vastes quantités de données textuelles, ce qui lui permet d'apprendre des modèles et des relations dans le langage. Lorsqu'il reçoit une nouvelle entrée textuelle, le LLM génère une réponse qui correspond le

mieux au contexte et au sens de l'entrée en fonction de ce qu'il a appris des données d'entraînement.

En termes plus simples, un LLM peut être considéré comme un assistant virtuel pour les tâches basées sur le texte. Il lit et comprend le texte, puis génère une réponse ou une suggestion en fonction des informations qui lui sont fournies. Les LLM sont particulièrement utiles dans le développement logiciel pour générer des explications, suggérer des approches alternatives et fournir des extraits de code contextuellement pertinents, ce qui en fait un outil complémentaire puissant aux côtés de Stack Overflow et des linters.

B - LLM et Développement Logiciel

Les Modèles de Langage Géants (LLMs) et Stack Overflow, ainsi que les linters, remplissent des rôles distincts mais complémentaires dans le processus de développement logiciel. Discutons pourquoi les LLMs sont un bon complément entre Stack Overflow et un linter :

Types différents de problèmes :

- Stack Overflow est une excellente ressource pour les développeurs afin de trouver des réponses à des questions de codage spécifiques ou des problèmes rencontrés. Il se concentre davantage sur la fourniture de solutions à des problèmes ou des erreurs de code existants.
- Un linter, en revanche, vérifie votre code par rapport à un ensemble de règles et de normes et propose des suggestions pour corriger les erreurs de syntaxe, les problèmes de mise en forme ou les erreurs logiques potentielles. Il aide à prévenir les erreurs avant qu'elles ne surviennent.
- Les LLMs excellent dans la compréhension et la génération de réponses contextuellement pertinentes basées sur de grandes quantités de données textuelles. Ils peuvent aider à fournir des explications ou des suggestions liées à l'intention et au sens derrière le code, ce qui va au-delà de ce qu'un linter peut offrir. Par exemple, les LLMs peuvent suggérer des approches alternatives, expliquer des concepts plus en détail, voire générer des extraits de code.

Compréhension dynamique :

- Stack Overflow est une ressource statique avec des réponses indexées sur la base de mots-clés et de tags. Il ne fournit pas toujours la réponse la plus pertinente pour une situation ou un contexte spécifique.
- Un linter fournit un ensemble de règles à vérifier, qui peuvent être assez rigides. Il n'a pas la capacité de comprendre les subtilités ou les implications de votre code.
- Les LLMs peuvent comprendre dynamiquement le contexte et le sens derrière votre code en fonction des données textuelles qui leur sont fournies. Ils peuvent générer des réponses qui tiennent compte de la situation ou du contexte spécifique dans lequel le code est utilisé, rendant leurs suggestions plus applicables et précieuses pour les développeurs.

Apprentissage et amélioration continus :

- Stack Overflow et les linters sont des ressources statiques qui doivent être mises à jour manuellement pour rester à jour avec les derniers développements dans les langages de programmation et les meilleures pratiques de développement logiciel. Cependant Stack Overflow tire avantage du Crowdsourcing.
- Les LLM eux ont une vision figée du monde du fait de leur dataset d'entraînement (on exclut ici le recours à des données externes lors des requêtes). Il est donc possible qu'ils ignorent des développements logiciels récents

En résumé, Stack Overflow et les linters remplissent des rôles essentiels dans le développement logiciel, mais les LLMs offrent des avantages uniques qui complètent ces outils. En comprenant le contexte, en fournissant des solutions dynamiques et en apprenant en continu à partir de données textuelles, les LLMs peuvent aider les développeurs à écrire un meilleur code, à apprendre de nouveaux concepts et à améliorer leurs compétences générales en développement logiciel.

C - LLM et Hallucination

Les LLM sont une connaissance moyenne d'un ensemble de données massif. Ainsi, ils peuvent sembler extrêmement intelligents. Cependant, il faut connaître les limites des LLM car il ne pensent pas ce sont des imitateurs capables d'appliquer des "formes" à leurs opérations en répétant ce qu'ils ont vu en moyenne dans leur corpus d'entraînement.

Le bloc de construction de base d'un LLM est la **complétion** d'un texte c'est-à-dire trouver le mot le plus probable après une suite de mots donnée. Ainsi, dans leur construction les LLM ne sont pas voués à répondre initialement à des questions. Cependant, il a été observé une forme d'émergence alors que les modèles ont grandi (https://arxiv.org/pdf/2206.07682.pdf) et que leur architecture est devenue plus performante (Transformer Models: https://arxiv.org/pdf/1706.03762.pdf). Ainsi, ces moteurs statistiques de complétion sont devenus des assistants.

Le LLM souffre cependant de nombreux biais:

- Connaissance moyenne uniquement:
  - difficulté à énumérer des ensemble exhaustifs;
  - Capacité à inventer une réponse (ne pas admettre ne pas savoir) effet aussi appelé une hallucination (mais également une capacité pour la génération)
  - Pas de capacités analytiques par exemple 8.8*4= ?
- Absence de raisonnement sur le temps long (uniquement capable de faire de l'imitation de forme avec remplacement)
- Biais d'alignement (parfois pour raison de sécurité) et application de normes sociales issues du dataset d'entrainement et également définies par la phase de Reinforcement Learning from Human Feedback (RLHF) (https://huggingface.co/blog/rlhf). Ainsi le modèle peut suivre une "pensée dominante" par imitation. Il n'a pas d'avis personnel.

- Taille limitée de contexte. Le contexte des LLM est l'entrée qui est passée en entrée du modèle pour définir l'embedding qui génèrera la réponse. Il limite donc la fenêtre "d'attention" du modèle à quelques milliers de caractères.
- Difficulté à parler précisément les langues autres que l'anglais (mais capable de le faire) le corpus d'entrainement étant souvent en Anglais.

IV - Exercices

Exercice1 : Linter shell

Installez l'outil "shellcheck" et corrigez le contenu du script ./ex1/tofix.sh, expliquez les différentes erreurs.

1) Corrigez les erreurs dans ./ex1/tofix.sh

Exercice2 : Linter pour le code C

Installez *bear* et *clangd*. Si vous utlilisez vscode, vous pouvez alors installer l'exension clangd pour avoir les sorties linteur (et erreurs de compilation) dans votre IDE.

Tutorial officiel:
https://releases.llvm.org/9.0.1/tools/clang/tools/extra/docs/clangd/Installation.html

1) Est-ce que le programme compile ?
2) Voyez-vous les warning dans votre IDE et dans la sortie du compilateur ?
3) Compilez avec **bear -- make** , cela crée un fichier *compiles_commands.json*
    a) Configurez votre IDE pour voir les erreurs
        i) VSCODE
            (1) Installation du plugin clang
            (2) Installation du plugin errorlens
        ii) Autre:
            (1) https://releases.llvm.org/9.0.1/tools/clang/tools/extra/docs/clangd/Installation.html
    b) Corrigez et expliquez les erreurs
4) Expliquez l'avantage de *clangd* couplé à *bear* dans le cas du développement logiciel

Exercice3 : Usage et Limite des LLMs

Vous pouvez installez un LLM local avec ollama: https://ollama.com/

Puis:
1) 8GB de ram (7B):
    a) **ollama run mistral**

2) Moins de ram (3B):
    a) ollama run dolphin-phi

Note : Vous devez disposer d'au moins 8 Go de RAM disponible pour exécuter les modèles de 7 milliards de paramètres, de 16 Go pour exécuter les modèles de 13 milliards de paramètres, et de 32 Go pour exécuter les modèles de 33 milliards de paramètres.

1) Expliquez et commentez les réponses aux questions suivantes passées à un LLM.
    a) Prompt: **Liste exhaustive des villes de l'Essonne ?**
        i) Quel est le problème de ce prompt ?
        ii) La réponse peut-elle être exacte ?
        iii) Testez avec des modèles de tailles différentes (si ollama)
    b) Prompt: **Décode U2FsdXQgY29tbWVudCB2YXMgdHUgPwo=**
        i) Quel est le problème de ce prompt ?
        ii) La réponse peut-elle être exacte ?
        iii) Testez avec des modèles de tailles différentes (si ollama)
2) Utilisez le LLM pour faire une courbe avec mathplotlib
    a) Dessinez deux fonctions sinus en opposition de phase avec python mathplotlib
        i) Générez un code
        ii) Sauvegardez la sortie au format png
    b) Créez un plot 2D avec mathplotlib et représentez sin(x)*sin(y)
        i) Générez un code
        ii) Sauvegardez la sortie au format png
    c) Générez un plot des données dans ./ex3/city.dat
3) Si vous avez ollama (pour vscode):
    a) Installez l’extension “continue” dans vscode