

1. Введение

В последние годы нейросетевые инструменты стремительно развиваются и становятся достаточно мощными, чтобы генерировать полноценные решения задач, в том числе и алгоритмических. Так, в среде спортивного программирования у людей появляется помощник, использование которого ставит под вопрос честность соревнований, если они проводятся в условиях, где невозможно явно проконтролировать самостоятельность решения задач.

Из-за этого актуальной проблемой становится нужда в проверке предоставленных решений на то, были ли они сгенерированы нейросетями. На данный момент существует много способов решения данного вопроса, и их можно разделить на такие группы: анализ процесса, поведения (работа с временными данными – время между правками и посылками, отслеживание обращений к внешним сайтам, проверка буфера обмена), организационные меры (прокторинг, блокировка доступа в Интернет на некоторых соревнованиях), а также анализ самого кода. В последнем пункте неочевидно, как именно происходит проверка решения участника без каких-либо дополнительных данных, поэтому далее я разберу базовые алгоритмы этой группы, которые являются основой для более сложных принципов.

1. Внешняя оценка кода – помогает ли?

Код, сгенерированный нейросетью, имеет несколько особенностей с точки зрения внешнего вида: чаще всего это очень подробные комментарии без надобности, идеально выдержанное форматирование, общепринятые названия переменных, шаблонная реализация функций и т. д.

Из-за того, что данные особенности видно невооружённым глазом, человеку не составляет труда внести мелкие «косметические» изменения в код, что делает этот способ непоказательным.

1. Семантический анализ

Стоит начать с того, что такое семантический анализ. В контексте проверки кода это способ оценки и выявления схожих алгоритмов и структур данных исходя из их логики, а не синтаксиса и внешнего вида вне зависимости от различий в записи решения.

Нейросети генерируют код по уже известным им шаблонам, поэтому решения и их куски у разных участников соревнований, использующих искусственный интеллект, могут совпадать, поэтому нужно проверять коды на совпадение.

Некоторые из семантических способов представляют собой процесс построения графов по логике решения и последующего их сравнения. Сейчас я хочу разобрать несколько подобных способов семантического сравнения решений.

1. Графы потока управления (Control Flow Graph, CFG)

Граф выглядит следующим образом: вершины – это блоки кода, операторы (например, условные операторы или циклы), рёбра ориентированы, представляют собой возможные переходы от одних блоков кода к другим. Чтобы было понятнее, приведу пример визуализации готового графа:

Код:

Граф:

Если в графах совпадают логические смыслы вершин и переходы между ними, это может значить, что присутствует плагиат.

1. Графы потока данных (Data Flow Graph – DFG)

Здесь вершины графа представляют собой переменные и операции над ними, а рёбра показывают передачу данных от одной переменной к другой или от оператора к переменной. Приведу пример визуализации такого графа.

Код:

Граф:

Такие графы сравниваются почти так же, что и графы предыдущего типа. Сравниваются вершины, операции в них, а также рёбра – пути передачи данных из вершин между переменными и операциями.

1. Стоит понимать, что здесь описано меньшинство алгоритмов проверки честности написания кода, но именно они являются основой для более сложных и точных способов определения кода, сгенерированного нейросетью.

2. Источники информации:

<https://pharmacelera.com/blog/science/artificial-intelligence-tree-search-algorithms/>

<https://habr.com/ru/articles/668684/>

<https://medium.com/@kanhasoftUSA/how-to-detect-an-ai-written-code-from-github-50d4a01e3114>

<https://www.geeksforgeeks.org/software-engineering/software-engineering-control-flow-graph-cfg/>

<https://bears.ece.ucsb.edu/research-info/DP/dfg.html>

<https://www.geeksforgeeks.org/compiler-design/abstract-syntax-tree-vs-parse-tree/>

<https://thecode.media/kak-otlichit-kod-napisanny-ii-ot-napisannogo-chelovekom/>