УДК 004.413

ОСНОВНЫЕ ЗАДАЧИ ПРИ ВНЕДРЕНИИ БОЛЬШИХ ЯЗЫКОВЫХ МОДЕЛЕЙ НА ОСНОВЕ ТРАНСФОРМЕРОВ В IT-ПРОЕКТЫ

Горшков П. В.[1]
Научный руководитель – доктор технических наук, профессор Бессмертный И. А.[1]
[1]Университет ИТМО
narpetri@itmo.ru

Введение

В последнее время, в связи со стремительным ростом возможностей искусственного интеллекта, всё больше руководителей команд разработки стараются внедрить большие языковые модели на базе трансформеров в рабочий процесс. Всеобщая цифровизация и высокая конкуренция среди разработчиков программного обеспечения накладывают ряд ограничений и требуют повышения скорости разработки с сохранением качества и бюджетных ограничений. Но зачастую использование искусственного интеллекта приводит совершенно не к тем результатам, которые хочется получить [1]. Это происходит в том числе вследствие завышенных ожиданий от использования больших языковых моделей и отсутствия адаптированности рабочего процесса к использованию новых технологий.

Основная часть

При внедрении больших языковых моделей на базе трансформеров в IT-разработку необходимо понимать, что, как и зачем будет использоваться. Одна и та же модель может выдавать как приемлемые результаты, так и требующие существенной доработки, в зависимости от того, какие задачи ей будут поставлены, как они будут поставлены, и какой будет уровень декомпозиции. При этом одни и те же задачи, с одним уровнем декомпозиции, одинаково поставленные, но разным моделям так же могут выдавать совершенно разные результаты. Более того, даже если одной модели несколько раз поставить одну и ту же задачу, нет гарантий, что она будет выполнена одинаково.

Соответственно, если мы не можем гарантировать, что результат генерации будет в полной мере соответствовать тому, что мы ожидаем, при внедрении искусственного интеллекта ставятся две основные задачи. Это, с одной стороны, повышение эффективности (увеличение скорости разработки, повышения качества, экономия бюджета), а, с другой стороны, минимизации возможных негативных последствий (иначе очень высокий риск рассчитывать на повышение эффективности, но в итоге нарушить сроки, выйти за бюджетные ограничения и предоставить некачественный продукт только потому, что были завышенные ожидания без должных условий для их обеспечения).

Мы можем гарантировать, что код будет написан очень быстро, как никогда раньше, но при этом он может или вовсе быть нерабочим, или в нем будет много ошибок. Таким образом дообучение большой языковой модели (чтобы сгенерированный код максимально соответствовал ожиданиям), грамотная декомпозиция (чтобы результат генерации можно было легко проверить и исправить, не затрачивая больше времени, нежели на ручное написание кода), и наличие компетенций по работе с большими языковыми моделями позволяют частично минимизировать ограничения.

Существующие методологии управления IT-проектами, являющиеся результатом десятилетий исследований в программной инженерии, содержат в себе методы и практики, актуальные и при внедрении больших языковых моделей, но без должной адаптации к их использованию, они могут быть избыточны, или, наоборот, упускать критически важные моменты. При этом грамотная комбинация методов и практик, учитывающая специфику работы с искусственным интеллектом, позволит повысить эффективность разработки.

С каждым годом большие языковые модели становятся совершеннее, результаты

генерации качественнее, но это не отменяет того, что у проекта будет жизненный цикл, будет создаваться проектная документация, сгенерированный код необходимо будет валидировать и поддерживать.

Выводы

Наибольшую эффективность в IT-проектах большие языковые модели показывают при их использовании в качестве ассистентов для генерации программного кода, по примеру парного программирования из XP [2]. При этом необходимо адаптировать методы управления IT-проектами к AI-ассистированной разработке и включить промт-ориентированные шаблоны проектной документации и элементов планирования, фиксирующих контекст, ограничения и критерии качества для взаимодействия с большими языковыми моделями в инструментарий управления IT-проектами.

Литература

1. 1. Bui T.-D., Vu T. T., Nguyen T.-T., Nguyen S., Vo H. D. Correctness assessment of code generated by Large Language Models using internal representations // Journal of Systems and Software. 2025. Vol. 230. https://doi.org/10.1016/j.jss.2025.112570.
2. Haque Md. A. LLMs: A game-changer for software engineers? // BenchCouncil Transactions on Benchmarks, Standards and Evaluations. 2025. Vol. 5, no. 1. https://doi.org/10.1016/j.tbench.2025.100204.