УДК 004.8

Исследование эффективности алгоритмов генерации случайных чисел для обучения и оценки моделей векторных представлений графов знаний

Плюхин Д.А.
(Национальный исследовательский университет ИТМО, г. Санкт-Петербург)
Научный руководитель - к.т.н. Муромцев Д.И.
(Национальный исследовательский университет ИТМО, г. Санкт-Петербург)

Введение. На сегодняшний день существует множество подходов к автоматической обработке графов знаний, среди которых - использование заранее заданного набора правил, применение статистических моделей и выполнение требуемых преобразований вручную. Ввиду наиболее высокой гибкости и масштабируемости самые перспективные методы решения данной задачи основаны на применении моделей машинного обучения, при построении которых требуется выполнять процедуру оптимизации параметров, а при выборе оптимальной архитектуры - процедуру оценки. Как правило, подобные модели оперируют представлениями графов, выраженными в форме совокупности триплетов, состоящих из двух сущностей и типа отношения между ними. Таким образом, в процессе использования таких моделей во избежание необходимости полного перебора всех триплетов в графе, а также с целью обеспечения гибкости полученной модели, требуется генерировать подвыборки исходного датасета. Основным компонентом алгоритмов формирования подвыборок является генератор случайных чисел, который может быть реализован различными способами, каждый из которых характеризуется различным уровнем эффективности. Таким образом, высокая зависимость эффективности процедур использования моделей векторных представлений графов знаний от эффективности применяемых алгоритмов генерации случайных чисел обуславливает высокую актуальность данного исследования.

Основная часть. В процессе обучения и оценки моделей векторных представлений графов знаний задача формирования подвыборок формулируется в виде задачи генерации наборов триплетов, обладающих заданными свойствами. В свою очередь, задача формирования множества триплетов декомпозируется на множество задач формирования одного триплета, для решения которых применяется синхронная или асинхронная модель вычислений. При этом требуемый набор свойств триплета формулируется в виде совокупности интервалов, в одном из которых должен находиться один или несколько компонентов триплета. В частности, при обучении моделей векторных представлений графов знаний необходимо генерировать триплеты, отсутствующие в обучающей выборке.

Обеспечение выполнения условий, которым должен удовлетворять сгенерированный триплет, может быть выполнено при помощи по крайней мере одного из двух алгоритмов:

1. Итерационный алгоритм - наиболее простой метод генерации триплета, основанный на предположении о том, что чаще всего генератор случайных чисел формирует корректный результат. Если же генератор случайных чисел сформировал некорректный триплет, то алгоритм выполняется повторно. Количество повторов зависит от количества случаев, в которых генератор формирует неверный триплет;
2. Алгоритм, основанный на подсчете смещений - более сложный алгоритм формирования списков триплетов, который, в отличие от итерационного алгоритма, направлен на генерацию корректного результата с первого раза. При использовании данного подхода помимо шагов, предусмотренных итерационным алгоритмом, реализуется 3 дополнительных проверки:
    a. Идентификатор сгенерированного компонента триплета меньше минимального индекса идентификатора “запрещенного” компонента - если данное условие

выполняется, то сформированный компонент применяется для генерации триплета без изменений;

b. Идентификатор сгенерированного компонента триплета больше разности максимального индекса идентификатора “запрещенного” компонента и количества “запрещенных” элементов - если данное условие выполняется, то к сгенерированному индексу прибавляется количество “запрещенных” компонентов, после чего полученный результат используется для генерации триплета;

Если обеспечивается условие непрерывности списка идентификаторов запрещенных компонентов, то при реализации данного алгоритма достаточно проверки двух указанных условий, поскольку операция вычитания на втором шаге эквивалентна смещению индексов всех компонентов триплетов, расположенных после совокупности “запрещенных” элементов. Результатом данной операции смещения является непрерывное множество “разрешенных” значений индексов компонентов триплета, из которых произвольное значение может быть сгенерировано без дополнительных проверок. Если же условие непрерывности не обеспечивается, то требуется выполнение третьего шага:

c. поиск “свободного” промежуточного значения индекса компонента триплета среди индексов “запрещенных” компонентов триплетов и расчет требуемого значения смещения - данный шаг реализуется при помощи двоичного поиска.

Выводы. Описанные алгоритмы генерации случайных чисел были реализованы с использованием языков программирования Python и C++. По результатам реализации были осуществлены эксперименты и сформулированы следующие выводы:

1. Итерационный алгоритм обладает большей эффективностью в случае малой доли “запрещенных” компонентов триплетов в общем списке индексов при использовании языка программирования Python - данное условие, как правило, выполняется при обучении моделей векторных представлений графов знаний. По результатам экспериментов в ряде случаев при доле “запрещенных” компонентов не более 0.4 (p-value = 0.023) обеспечивается указанное свойство итерационного алгоритма, однако в большинстве случаев итерационный алгоритм демонстрирует менее высокую эффективность;
2. Использование объектов для представления промежуточных результатов генерации не приводит к заметному снижению эффективности алгоритма;
3. Реализация алгоритма, основанного на подсчете смещений, в среднем в 29.96 раз эффективнее на языке программирования C++ по критерию времени выполнения в сравнении с реализацией на языке программирования Python ($\sigma = 2.26$);
4. Что касается реализации на языке программирования C++, итерационный алгоритм во всех рассмотренных случаях характеризуется менее высокой эффективностью;
5. Был разработан модуль оценки эффективности алгоритмов генерации случайных чисел при решении задач обучения и оценки качества работы моделей векторных представлений графов знаний, который планируется использовать в дальнейшем для исследования альтернативных алгоритмов генерации случайных чисел, среди которых - алгоритм, основанный на использовании вероятностных распределений;
6. В среднем реализация алгоритма, основанного на подсчете смещений, без осуществления операций сортировки списка “запрещенных” элементов и удаления дубликатов выполняется в 1.59 раз быстрее ($\sigma = 0.37$) по сравнению с более полной реализацией, предполагающей выполнение указанных вычислений.

Список использованных источников:

1. Han X. et al. Openke: An open toolkit for knowledge embedding // Proceedings of the 2018 conference on empirical methods in natural language processing: system demonstrations. – 2018. – С. 139-144.
2. GeeksforGeeks “How to Find a P-Value from a t-Score in Python?” // Режим доступа: https://www.geeksforgeeks.org/how-to-find-a-p-value-from-a-t-score-in-python/ своб. Дата обращения: 07.02.2023
3. Khan Academy “Two-sample t-test for difference of means” // Режим доступа: https://www.khanacademy.org/math/ap-statistics/xfb5d8e68:inference-quantitative-means/two-sample-t-test-means/v/two-sample-t-test-for-difference-of-means своб. Дата обращения: 07.02.2023