Инженерный вестник Дона, №5 (2022)
ivdon.ru/ru/magazine/archive/n5y2022/7682

Формирование и анализ эффективности выборки для обучения языковых моделей распознаванию и анализу исходного кода программ

Д.Ю. Какутин, А.С. Дмитриев

Волгоградский государственный технический университет

Аннотация: В данной статье описывается формирование обучающей выборки для обучения языковых нейронных сетей для использования их в задачах, связанных с анализом и поиском совпадений и/или соответствий по смыслу/значению, а конкретно с функциями и методами в исходном коде языка программирования. Определяются нужные в выборке ключевые параметры для корректного обучения нейронной сети.
Ключевые слова: исходный код, машинное обучение, обработка естественного языка, нейронная сеть, анализ данных.

При разработке любого вида программного обеспечения разработчику приходится сталкиваться с анализом своего, а зачастую и чужого исходного кода программ [1]. Кроме прочего, описания и названия использующихся методов языка программирования дают возможность находить в открытых источниках более рациональные и эффективные реализации существующих в коде алгоритмов, избегать ошибок и улучшать общую читаемость кода.

Исходя из этих соображений, в настоящее время создаются различные виды программного обеспечения, призванного облегчить работу разработчика, уменьшить воздействие человеческого фактора на конечный результат и, соответственно, уменьшить время, затрачиваемое на разработку [2].

Одним из подобного рода случаев является использование обученных на наборах данных исходного кода, находящихся в открытом доступе (а также в собственности каких-либо частных компаний при собственных разработках ПО), нейронных сетях. Данные нейронные сети лежат в основе некоторых программ, вот пара вариантов использования:

1. AI-ассистенты для написания исходного кода – сейчас существует множество решений данного рода, в основном в своей основе имеющих обученную на исходном коде программ модель обработки естественного

языка GPT-x (где x – номер версии, от 1 до 3, от самой первой до более современной модели) [3 - 5]. В свою очередь исходная модель обучена на огромных объемах данных с английским языком, включающих в себя данные из Википедии, различных публикаций, находящихся в открытом доступе и литературы. Данные ассистенты, по сути, реализуют функционал предсказательной модели [6], обрабатывая контекст уже написанного кода, они предлагают варианты для завершения написания его участков. Весьма действенные в некоторых монотонных и громоздких задачах, тем не менее, имеющие далекую от 100% точность и безошибочность предсказаний.

2. Функционал, позволяющий искать исходный код по описанию или части кода. По пользовательскому вводу из исходных параметров (обычно строки с текстом) позволяет находить совпадения и на выходе отдать пользователю готовый исходный код. Иногда является вспомогательным дополнительным функционалом для AI-ассистента, имеющим отдельный интерфейс и не так часто использующимся. Отдельные же реализации подобного функционала в открытом доступе достаточно сложно найти, в целом их существует не очень много.

Как раз формирование выборки для обучения нейронных сетей для работы в подобных ситуациях мы и поговорим. Для начала рассмотрим существующие популярные языковые модели, их доступность и сложность обучения:

1. Семейство GPT – уже упомянутые выше модели, имеют закрытый исходный код, огромное количество параметров, доступны ограниченном количеству компаний-партнеров

2. Stanford NLP – языковая модель Стэнфордского университета с открытым исходным кодом, доступная любому разработчику. Содержит множество вариаций для различных языков помимо английского [7], достаточно большие объемы исходных тренировочных данных, неплохую

точность модели, при этом требуя достаточно больших ресурсов для обучения.

3. Google BERT – языковая модель компании Google LLC (USA, California) с открытым исходным кодом, имеющая обширный набор языков, гораздо больший, чем Стэнфордская модель, а также огромный выбор натренированных моделей с различными параметрами [8, 9]. Google использует BERT в том числе в своей поисковой системе, пока что не для всех поисковых запросов и не для всех регионов, но это создает тенденцию на повсеместное использование нейронных сетей.

4. ALBERT – A lite BERT, облегченный вариант Google BERT, содержащий меньший набор параметров и тренировочных данных, в целом более упрощенный набор моделей, при этом требующий наименьших мощностей для последующего обучения [10].

Модель ALBERT мы и будем использовать ввиду низких время- и ресурсо- затрат на обучение, при этом самой модели будет вполне достаточно для оценки корректности и репрезентативности наших данных, благодаря выходным оценочным параметрам самой модели.

Возьмем за основу уже натренированную модель ALBERT со словарем, состоящим из 30000 слов. Ее в целом будет достаточно для обработки наших данных.

Для этой модели мы будем создавать тренировочные данные посредством уже готовых скриптов, предоставленных компанией Google в репозитории ALBERT.

В качестве данных для дополнительного обучения нами взят набор данных 150k JavaScript Dataset, являющийся набором файлов с исходным кодом на языке JavaScript и использующийся в работе по построению предсказательной модели для обнаружения уязвимостей в исходном коде на языках JavaScript и Python.

Для выделения нужных данных из данного набора используем парсер исходного кода файла в AST – асбстрактное синтаксическое дерево, представляющее собой древовидную структуру исходного кода, описанного в файле. Из AST мы по ряду условий можем получить код каждой функции в файле, ее название и параметры, а также относящиеся к ней комментарии.

На выходе получается набор данных, содержащий объекты с названием функции, ее параметрами, всеми относящимися к ней комментариями в одной строчке, а также ссылкой на файл, откуда функция была взята. Этот набор мы дальше будем преобразовывать для изучения влияния различных параметров на репрезентативность (полезность) получившегося набора данных.

Главным критерием репрезентативности итоговой выборки будем считать параметр модели, называемый lm accuracy или точность языковой модели. Параметр определяется методом замены части слов в предложении масками и предсказанием моделью возможных вариантов подстановок с последующей сверкой с оригиналом. Этот параметр покажет, насколько хорошо понятна для нашей модели выборка данных, в нашем случае – насколько хорошо языковая модель для классификации предложений на английском языке после дообучения классифицирует названия и описания методов исходного кода.

Рассмотрим этапы с разными параметрами для тренировки:

1. В первом случае мы исключаем из нашей выборки функции и код, названия и описания которого превышают 1000 символов, предполагаем, что это слишком много и, вероятно, только усложнит обучение модели. Также исключаем функции с именами длиной меньше, чем три буквы. Кроме того, исключим строки с комментариями вида “copyright” и “deprecated”, что значит, что в данном контексте обозначаются авторские права и сообщения

об устаревшем функционале, они нам помешают. Текущая выборка содержит 273000 строк с описаниями кода. Пример выборки на рис. 1

```
* Returns the slide element matching the specified index.

------------------

* Returns the background element for the given slide.
* All slides, even the ones with no background properties
* defined, have a background element so as long as the
* index is valid an element will be returned.
```

Рис. 1. – Пример описаний в первой выборке

Задав в качестве начальных параметров 10000 шагов и 250 обрабатываемых строк за шаг (train_batch_size), получаем результаты на рис. 2.

```
masked_lm_accuracy = 0.047407314
masked_lm_loss = 9.948375
sentence_order_accuracy = 0.523125
sentence_order_loss = 0.69289017
```

Рис. 2. – Результаты тренировки с первой выборкой

Как мы можем видеть, параметр lm accuracy равен 0.047 или же 4.7%, что можно трактовать как непонимание моделью нашего набора.

2. Во втором случае изменим наш набор данных, включив в него в основном небольшие строки, минимально, но понятно описывающие какой-либо функционал. Исходя из первого случая предполагаем, что, вероятно 1000 символов тоже много, и оставляем лишь строки со 150 символами и меньше, уменьшив также параметры тренировки, но увеличив число шагов, чтобы улучшить наши результаты. Пример выборки на рис. 3.

```
return total number of rows in grid number

return total number of columns in grid number

returns the computed width of the native browser scroll bar number

returns true if undo point is available boolean
```

Рис. 3. – Пример описаний во второй выборке

На выходе получаем рис. 4.

```
masked_lm_accuracy = 0.09329596
masked_lm_loss = 5.7214518
sentence_order_accuracy = 0.5076
sentence_order_loss = 0.693128
```

Рис. 4. – Результаты тренировки со второй выборкой

Результаты тренировки и понимания набора данных улучшились до 9.3%, но это все еще в несколько раз меньше минимального желаемого результата.

3. В третьем случае мы усредняем наш набор, оставляя строки средней длины, в пределах от 50 до 250. Также снизим до средних значений параметр train_batch_size, определяющий количество используемых строк данных для обучения за одну итерацию. Сделаем его равным 100. Получим результат на рис. 5.

```
masked_lm_accuracy = 0.32336667
masked_lm_loss = 10.015323
sentence_order_accuracy = 0.5364
sentence_order_loss = 0.6904889
```

Рис. 5. – Результаты тренировки со третьей выборкой

Как видим, значение точности нашей модели приблизилось к минимально достаточному для использования (30–35%), для относительно небольшого набора данных это удовлетворительный результат.

В целом мы можем сказать, что не всегда максимизация параметров даст наилучший результат. В данном конкретном случае мы видим, что лучше всего обучение языковой модели с небольшим набором данных, представляющих собой описание исходного кода проходит при среднем размере предложений (строк с комментариями) и среднем размере итерационного «пакета» данных, используемого для тренировки каждый шаг. В итоге наша модель вполне может использоваться для предсказания исходного кода, то есть на основе пользовательского ввода генерировать различные варианты последующего «развития» текста, находить похожий исходный код по совпадениям описаний. Подобная нашей настройка параметров может коррелировать с тренировкой этой и подобной языковых моделей на небольших/средних наборах данных для различных задач в работе с исходным кодом программ.

Литература

1. Кучин И.Ю. Обзор существующих методов анализа программного кода // Актуальные проблемы гуманитарных и естественных наук. 2012, № 2, с 44–46.

2. Абрамова Н.А. О проблеме рисков из-за человеческого фактора в экспертных методах и информационных технологиях // Проблемы управления. 2007, № 2, с 11–21.

3. B. Brown Tom, Mann Benjamin, Ryder Nick, Subbiah Melanie, Kaplan Jared, Dhariwal Prafulla, Neelakantan Arvind, Shyam Pranav, Sastry Girish, Askell Amanda, Agarwal Sandhini, Herbert-Voss Ariel, Krueger Gretchen, Henighan Tom, Child Rewon, Ramesh Aditya, M. Ziegler Daniel, Wu Jeffrey,

Winter Clemens, Hesse Christopher, Chen Mark, Sigler Eric, Litwin Mateusz, Gray Scott, Chess Benjamin, Clark Jack, Berner Christopher, McCandlish Sam, Radford Alec, Sutskever Ilya, Amodei Dario. Language Models are Few-Shot Learners // Advances in Neural Information Processing Systems 33 (NeurIPS 2020), 2020, DOI: 10.48550/arXiv.2204.14259. URL: arxiv.org/abs/2005.14165.

4. Budzianowski Pawel, Vulic Ivan. Hello, It's GPT-2 -- How Can I Help You? Towards the Use of Pretrained Language Models for Task-Oriented Dialogue Systems // Proceedings of the 3rd Workshop on Neural Generation and Translation, 2019, pp 15 – 22, DOI: 10.18653/v1/D19-5602

5. Whitfield Dewayne. Using GPT-2 to Create Synthetic Data to Improve the Prediction Performance of NLP Machine Learning Classification Models // 2021, DOI: 10.48550/arXiv.2104.10658. URL: arxiv.org/abs/2104.10658.

6. Ladune Theo, Philippe Pierrick, Hamidouche Wassim, Zhang Lu, Deforges Olivier. Binary Probability Model for Learning Based Image Compression // International Conference on Acoustics, Speech, and Signal, 2020, pp. 2168-2172. DOI: 10.1109/ICASSP40776.2020.9053997

7. Qi Peng, Zhang Yuhao, Zhang Yuhui, Bolton Jason, D. Manning Christopher. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, 2020, pp. 101-108. DOI: 10.18653/v1/2020.acl-demos.14.

8. Wang Rui, Chen Dongdong, Wu Zuxuan, Chen Yinpeng, Dai Xiyang, Liu Mengchen, Jiang Yu-Gang, Zhou Luowei, Yuan Lu. BEVT: BERT Pretraining of Video Transformers // 2022, DOI: 10.48550/arXiv.2112.01529

9. Devlin Jacob, Ming-Wei Chang, Lee Kenton, Toutanova Kristina. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // ArXiv, 2019, DOI: 10.48550/arXiv.1810.04805. URL: arxiv.org/abs/2112.01529.

10. Lan Zhenzhong, Chen Mingda, Goodman Sebastian, Gimpel Kevin, Sharma Piyush, Soricut Radu. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations // ICLR 2020 Conference Blind Submission, 2019, DOI: 10.48550/arXiv.1909.11942. URL: arxiv.org/abs/1909.11942.

References

1. Kuchin I.Y. Aktualnye problemy gumanitarnyh i estestvennyh nauk. 2012, Vol. 2, pp. 44-46.

2. Abramova N.A. Problemy upravleniya. 2007, Vol. 2, pp 11-21.

3. B. Brown Tom, Mann Benjamin, Ryder Nick, Subbiah Melanie, Kaplan Jared, Dhariwal Prafulla, Neelakantan Arvind, Shyam Pranav, Sastry Girish, Askell Amanda, Agarwal Sandhini, Herbert-Voss Ariel, Krueger Gretchen, Henighan Tom, Child Rewon, Ramesh Aditya, M. Ziegler Daniel, Wu Jeffrey, Winter Clemens, Hesse Christopher, Chen Mark, Sigler Eric, Litwin Mateusz, Gray Scott, Chess Benjamin, Clark Jack, Berner Christopher, McCandlish Sam, Radford Alec, Sutskever Ilya, Amodei Dario. Advances in Neural Information Processing Systems 33 (NeurIPS 2020), 2020, DOI: 10.48550/arXiv.2204.14259. URL: arxiv.org/abs/2005.14165.

4. Budzianowski Pawel, Vulic Ivan. Hello, It's GPT-2 -- How Can I Help You? Proceedings of the 3rd Workshop on Neural Generation and Translation, 2019, pp 15 – 22, DOI: 10.18653/v1/D19-5602

5. Whitfield Dewayne. Using GPT-2 to Create Synthetic Data to Improve the Prediction Performance of NLP Machine Learning Classification Models // 2021, DOI: 10.48550/arXiv.2104.10658. URL: arxiv.org/abs/2104.10658.

6. Ladune Theo, Philippe Pierrick, Hamidouche Wassim, Zhang Lu, Deforges Olivier. International Conference on Acoustics, Speech, and Signal, 2020, pp. 2168-2172. DOI: 10.1109/ICASSP40776.2020.9053997

7. Qi Peng, Zhang Yuhao, Zhang Yuhui, Bolton Jason, D. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, 2020, pp. 101-108. DOI: 10.18653/v1/2020.acl-demos.14.

8. Wang Rui, Chen Dongdong, Wu Zuxuan, Chen Yinpeng, Dai Xiyang, Liu Mengchen, Jiang Yu-Gang, Zhou Luowei, Yuan Lu. 2022, DOI: 10.48550/arXiv.2112.01529

9. Devlin Jacob, Ming-Wei Chang, Lee Kenton, Toutanova Kristina. ArXiv, 2019, DOI: 10.48550/arXiv.1810.04805. URL: arxiv.org/abs/2112.01529.

10. Lan Zhenzhong, Chen Mingda, Goodman Sebastian, Gimpel Kevin, Sharma Piyush, Soricut Radu. ICLR 2020 Conference Blind Submission, 2019, DOI: 10.48550/arXiv.1909.11942. URL: arxiv.org/abs/1909.11942.