Отчет по реализации метода из статьи «Universal Adversarial Triggers for Attacking and Analyzing NLP»

Махин Артем Александрович

9 апреля 2021

1 Постановка задачи

В статье описан метод поиска универсальных триггеров для языковых моделей. Под универсальным триггером подразумевается независимые от входа последовательности токенов, которые заставляют модель выдавать определенный ответ при объединении с любыми входными данными. Рассмотрим задачу Natural language inference. На вход дается два предложения: предположение и гипотеза. Необходимо определить, является ли гипотеза следствием, противоречием или же она нейтральна по отношению к предположению.

2 Описание метода поиска

На каждом батче обновляются эмбеддинги подставляемых токенов так, чтобы новые эмбеддинги обеспечивали минимум аппроксимации Тейлора первого порядка функции потерь, то есть

$$\arg\min_{e_i \in V} = [e_i - e_{adv_j}]^T \nabla_{e_{adv_j}} \mathcal{L}$$

где V - множество эмбеддингов всех токенов в словаре модели, e_{adv_j} - эмбеддинг нынешнего j-ого триггер-токена, $\mathcal{L}$ - функция потерь модели. Для лучшего результата так же применяется beam-search.

3 Описание реализации из статьи

Авторами был использован датасет SNLI, в качестве модели была использована Enhanced Sequential Inference Model с ELMo эмбеддингами. Запустив код, можно убедиться, что метод действительно очень хорошо и быстро работает. К примеру, с токеном “nobody” 99.43% пар с логическим следствием модель классифицирует как противоречие.

4 Эксперименты

Стало интересно, сможем ли данным методом обманывать достаточно крупные модели? Для экспериментов использовался датасет MNLI, в качестве модели взята предобученная модель Roberta-large-mnli (в которой 355M параметров). Код написан, основываясь на структуре кода, предложенного в статье. Пытаемся для пар с логическим следствием добиться противоречия от модели.

Так как модель достаточно большая, работать с большими батчами на google colab не получается (не хватает памяти на cuda). Поэтому во многих функциях пришлось разбивать батч на еще более маленькие батчи, а результаты объединять (к примеру, для оценивания качества считалось количество верно определенных пар, и эти значения суммировались; для подсчета градиента так же считался градиент по небольшим батчам, результирующим градиентом по исходному батчу была их сумма). Это достаточно долго работает, даже не смотря на то, что выбиралось всего примерно 5% случайных пар из исходного датасета.

Весь код написан в experiments.ipynb. Выбрано количество токенов равное 4. В качестве начальных токенов были использованы случайные слова (просто что пришло в голову): 'Not', ' means', ' opposite', ' the'. (так же проводились эксперименты с различным количеством слов от 1 до 10, но на результат это не сильно влияло; выбирались и начальные токены, предложенные в статье ('a' и 'the'), но так же результат не сильно изменялся). Можно видеть, что в точность модели несколько падает (около 1%), но это все же далеко не то, что нужно.

5 Возможные причины

Возможные причины того, что предложенный метод не сработал:

- Слишком большая модель. Так как она обучена на большом количестве данных, в которых нет явного так называемого data biases. Вероятно, если взять очень много триггер-токенов (20, 30, 40 ...), то удастся заставить модель ошибаться, однако такие большие префиксы для строк будут больше самих строк (да и поиск займет очень много времени), поэтому потеряется смысл поиска триггер-токенов (мы фактически будем полностью заменять предложение, а не незначительно дополнять)

- Мало данных. Использовался батч размера примерно 540, в качестве всего датасета используется всего примерно 5% данных (то есть примерно 6500 пар). Вероятно, если использовать более большие батчи, то результат будет несколько лучше, однако время работы значительно увеличивается.

- И, конечно же, ошибка в коде (но все таки количество ошибок модели

вначале несколько увеличивается, что свидетельствует о том, что код ищет нужные нам вещи)

Так же авторы статьи написали, что универсальные триггеры часто передаются между моделями: триггеры, найденные для одной модели, могут так же вызывать большое число ошибок для другой модели для такой же задачи. Хотя датасеты несколько различаются, в качестве интереса были опробованы найденные триггеры для предложенной в статье модели (по одиночке, несколько сразу в различном порядке). Код представлен в test.ipynb. Однако, это не привело ни к чему: максимум отклонения по точности меньше 1 процента, что хуже, чем найденные нами токены.

6 Выводы

Метод, предложенный в статье, работает (что следует из кода, который авторы выложили), однако не для всех моделей. Для более сложных языковых моделей данным методом найти триггер-токены не получается.