**Трапер Максим**

**GPN Intelligence Cup 2024. «DS в нефтяном ритейле».**

**Задача:** из имеющегося корпуса цитат сформировать корпус тегов, которые в дальнейшем будут применятся для тегирования новых цитат, подающихся в систему.

**Методика решения:**

1. **Предобработка данных:**

Цитаты в исходном корпусе – шумные. Шумными они будут и в случае с новыми цитатами. Классические решения на основе SpellChecker, проверок по словарю и т.д. показали себя не очень эффективно. Поэтому был выбран вариант использования нейросетевого фреймоворка SAGE, а именно ai-forever/RuM2M100-418M. Это решение в плане производительности применимо как на предварительном этапе формирования корпуса тегов, так и на этапе инференса. Среднее время исправления одной цитаты – 1-3 секунды. Для онлайна не лучшее время, но приемлимое.

В тексте также присутствует специфическая лексика, не характерная для «обычного» русского языка, т.е. модель не обучалась на такой лексике. Поэтому модель наровит всегда исправить такую лексику на «правильные» варианты. Была попытка применить LogitProcessor, но оно не заработало. Поэтому был применём вариант в лоб – составление словаря специфической лексики, содержимое которого сравнивается с изменениями в цитатах на этапе постпроцессинга. Словарь составлялся в ручном режиме при просмотре данных о том, какие токены заменила модель и на какие.

1. **Следующий этап – основной, формирование корпуса тегов.**

Для этого изначально я выбрал BERTopic и легковесную LLM, которую можно запустить локально - qwen2.5\_3B, дообученный на русских текстах. Я не стал применять классические методы тематического моделирования, такие как LDA и LSA, т.к. они показали себя очень плохо на таком трудном корпусе . А вот на BERTopic я надеялся, как на более мощную модель. И на первый взгляд она показала себя неплохо (поэтому я её оставил в ноутбуке).

LLM же была применена не ради «желания впихнуть её куда угодно», а скорее в попытке добиться выделения более абстрактных тематик, на что не способны ранее названные модели, так как они ориентируются лишь на существующие в корпусе слова.

По итогу, попробовав BERTopic и LLM и объединив результаты на этапе 3, проведя кластеризацию, я остался не очень доволен качеством. А вот убрав сгенерированные теги BERTopic’ом качество, эмперически оценив, стало гораздо кратно лучше. Шумов в тегах стало гораздо меньше.

1. **Кластеризация:**

Ранее уже упомянутый процесс кластеризации сгенерированных тематик совершается по той причине, что модель тоже выдаёт шумные ответы. Иногда тематики могут семантически совпадать, но писаться по разному. Именно поэтому было необходимо объединить все сгенерированные теги в более укрупненные группы. Делал я это при помощи KMeans алгоритма.

Выбор количества кластеров проводил перебором по сетке и эмперически оценивал качество. Пытался было оценивать с помощью Silhouette score, но он оказался не очень объективен, т.к. показал наибольшее значение при большом количестве кластеров.

Проведя кластеризацию, в качестве «главного» тега в каждом кластере были выбраны самые часто встречающиеся и именно на такие теги были заменены все остальные теги кластеров.

1. **Загрузка данных в векторную БД:**

Последний подготовительный этап – загрузка цитат и сформированных для них тегов в векторную БД FAISS. Далее, для каждой новой цитаты будут искаться top\_k соседей из существующих цитат, браться их теги. Далее, высчитываются две схожести (косинусное расстояние и скалярное произведение) цитаты и каждого из тегов, берутся топ-10 по каждой из схожестей и пересечение этих множеств из топ-10 будет итоговым набором тегов.

На каждом шаге я постарался сохранять промежуточные данные, чтобы если возникает желание поработать с данными на каком-то отдельном этапе, можно было бы просто взять нужный датасет из папки data, загрузить его и работать.

**Примечание:** несмотря на использование нескольких нейросетевых моделей и даже LLM, я всё же считаю, что требование по легковесности выполнено, ибо неважно то, сколько по времени мы формировали первичный корпус тегов, так как мы делаем это в оффлайне. Зато затем, уже на инференсе для любой новой цитаты, за счёт использования векторной БД, мы формируем теги за считанные секунды. И именно поэтому предложенное решение можно считать легковесным.

**Проблемы:**

1. Нужно лучше проводить кластеризацию, более скрупулёзно, потому что иногда алгоритмы смешивают ту информацию, которую бы хотелось разделить. Возможно стоит попробовать другие методики (hdbscan, DBSCAN – я их пробовал, но они на мой личный взгляд сработали хуже)
2. Необходимо как-нибудь выделить имена собственные, чтобы они тоже могли быть тегами, иначе информация теряется. С таким малым корпусом текстов вряд ли получиться обучить модель на задачу NER, поэтому легче составить словарь.
3. Улучшить качество обработки опечаток (составить словарь терминов, возможно попробовать дообучить модель)
4. Выборка тегов зависит от нескольких параметров: количество соседей и количество берущихся из обоих метрик близости самых близких тегов к цитате. Перебрав некоторые числа, я пришёл к некоторой золотой середине в виде количества соседей = 5 и взятие от косинусного расстояния и скалярного произведения топ-10 тегов по близости к цитате, но всё же иногда результаты будут лучше при других цифрах.