Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика, искусственный интеллект и системы управления

КАФЕДРА Системы обработки информации и управления

Домашнее задание
по дисциплине «Методы машинного обучения»

Выполнил: Королев С.В.

Группа: ИУ5-23М

Москва, 2022 г.

ОГЛАВЛЕНИЕ

1. ЗАДАНИЕ

Домашнее задание по дисциплине направлено на анализ современных методов машинного обучения и их применение для решения практических задач. Домашнее задание включает три основных этапа:

1. выбор задачи;
2. теоретический этап;
3. практический этап.

Этап выбора задачи предполагает анализ ресурса paperswithcode [1]. Данный ресурс включает описание нескольких тысяч современных задач в области машинного обучения. Каждое описание задачи содержит ссылки на наиболее современные и актуальные научные статьи, предназначенные для решения задачи (список статей регулярно обновляется авторами ресурса). Каждое описание статьи содержит ссылку на репозиторий с открытым исходным кодом, реализующим представленные в статье эксперименты. На этапе выбора задачи обучающийся выбирает одну из задач машинного обучения, описание которой содержит ссылки на статьи и репозитории с исходным кодом.

Теоретический этап включает проработку как минимум двух статей, относящихся к выбранной задаче. Результаты проработки обучающийся излагает в теоретической части отчета по домашнему заданию, которая может включать:

- описание общих подходов к решению задачи;
- конкретные топологии нейронных сетей, нейросетевых ансамблей или других моделей машинного обучения, предназначенных для решения задачи;
- математическое описание, алгоритмы функционирования, особенности обучения используемых для решения задачи нейронных сетей, нейросетевых ансамблей или других моделей машинного обучения;
- описание наборов данных, используемых для обучения моделей;
- оценка качества решения задачи, описание метрик качества и их значений;
- предложения обучающегося по улучшению качества решения задачи.

Практический этап включает повторение экспериментов авторов статей на основе представленных авторами репозиториев с исходным кодом и возможное улучшение обучающимися полученных результатов. Результаты проработки обучающийся излагает в практической части отчета по домашнему заданию, которая может включать:

- исходные коды программ, представленные авторами статей, результаты документирования программ обучающимися с использованием диаграмм UML, путем визуализации топологий нейронных сетей и другими способами;
- результаты выполнения программ, вычисление значений для описанных в статьях метрик качества, выводы обучающегося о воспроизводимости экспериментов авторов статей и соответствии практических экспериментов теоретическим материалам статей;
- предложения обучающегося по возможным улучшениям решения задачи, результаты практических экспериментов (исходные коды, документация) по возможному улучшению решения задачи.

Отчет по домашнему заданию должен содержать:

1. Титульный лист.
2. Постановку выбранной задачи машинного обучения, соответствующую этапу выбора задачи.
3. Теоретическую часть отчета.
4. Практическую часть отчета.
5. Выводы обучающегося по результатам выполнения теоретической и практической частей.
6. Список использованных источников.

2. ПОСТАНОВКА ЗАДАЧИ

В результате анализа содержимого ресурса «Papers with code» была найдена статья, решающая задачи из области создания эмбеддингов документов (Document Embedding) и тематического моделирования (Topic Modeling). Статья называется «BERTopic: Нейронное тематическое моделирование с помощью метода TF-IDF на основе классов» (BERTopic: Neural topic modeling with a class-based TF-IDF procedure) [2].

Рисунок 1 демонстрирует страницу, посвященную данной статье. На данной странице можно:

- прочитать Аннотацию (Abstract) к статье;
- получить текст статьи в формате PDF;
- открыть код, реализующий методы, описанные в данной статье;
- посмотреть задачи, решение которых предлагают авторы статьи;
- посмотреть наборы данных, на которых обучалась или тестировалась модель;
- посмотреть результаты решения задачи (необязательно указываются на данной сайте авторами статей).

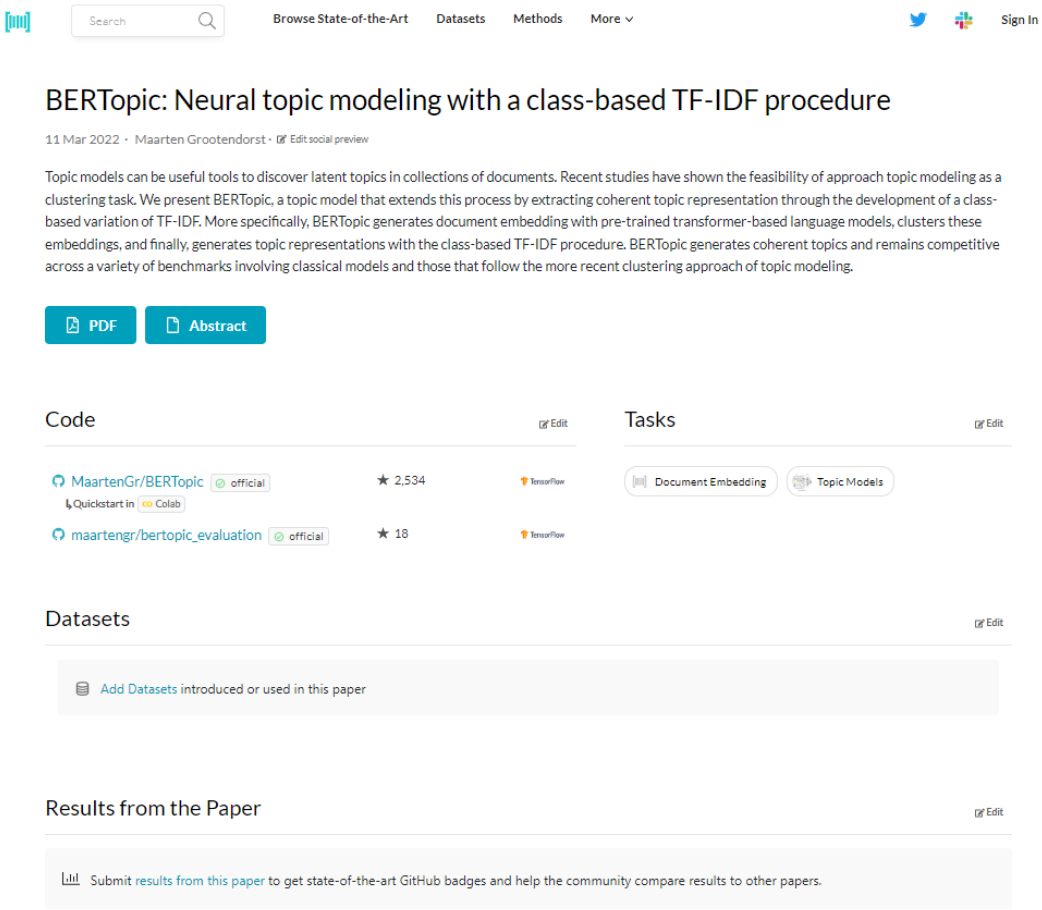

Рисунок 1 - Страница, посвященная статье

Специфика создания эмбеддингов документов состоит прежде всего в том, что под документами подразумеваются объемные тексты, обработка которых требует иного подхода по сравнению с традиционной длиной текстов, когда создавались только эмбеддинги слов или предложений. Также зачастую документами считаются тексты, обладающие специфичной лексикой, соответствующей определенной предметной области.

3. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

В рамках домашнего здания рассматриваются материалы статьи «BERTopic: Нейронное тематическое моделирование с помощью метода TF-IDF на основе классов» (BERTopic: Neural topic modeling with a class-based TF-IDF procedure).

Тематические модели могут быть полезными инструментами для обнаружения скрытых тем в коллекциях документов. Недавние исследования показали возможность применения тематического моделирования в качестве задачи кластеризации. Авторы представляют тематическую модель BERTopic, которая расширяет этот процесс, извлекая связное представление темы посредством разработки варианта метода TF-IDF на основе классов. В частности, BERTopic генерирует эмбеддинг документов с помощью предварительно обученных языковых моделей на основе трансформеров, кластеризует эти эмбеддинги и, наконец, генерирует представления тем с помощью процедуры TF-IDF на основе классов. BERTopic создает согласованные темы и остается конкурентоспособным в различных тестах, включающих классические модели и те, которые следуют более современному кластерному подходу к тематическому моделированию.

3.1. Трансформеры

Архитектура модели «трансформер» основана на механизме внимания - чрезвычайно распространенном методе в современных моделях глубокого обучения, позволяющий улучшить показатели эффективности приложений нейронного машинного перевода. В данном разделе будет рассмотрена модель Трансформер, которая использует механизм внимания для повышения скорости обучения. Более того, для ряда задач Трансформеры превосходят модель нейронного машинного перевода от Google. Однако самое большое преимущество Трансформеров заключается в их высокой эффективности в условиях параллелизации (parallelization) [3].

3.1.1. Высокоуровневое представление

В высокоуровневом представлении трансформер состоит из кодирующего компонента, декодирующего компонента и связи между ними. Кодирующий компонент – это стек энкодеров; Рисунок 2 демонстрирует 6 энкодеров, расположенных друг над другом (можно экспериментировать и с любым другим числом кодировщиков). Декодирующий компонент – это стек декодеров, представленных в том же количестве.

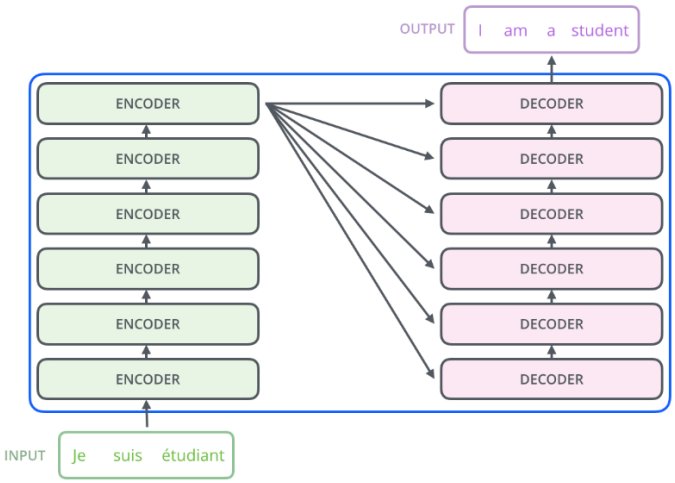

Рисунок 2 - Стек кодировщиков и декодировщиков

Все энкодеры идентичны по структуре, хотя и имеют разные веса. Каждый можно разделить на два подслоя, см. Рисунок. Входная последовательность, поступающая в энкодер, сначала проходит через слой внутреннего внимания (self-attention), помогающий энкодеру посмотреть на другие слова во входящем предложении во время кодирования конкретного слова. Мы рассмотрим этот механизм далее в статье.

Выход слоя внутреннего внимания отправляется в нейронную сеть прямого распространения (feed-forward neural network). Точно такая же сеть независимо применяется для каждого слова в предложении.

Декодер также содержит два этих слоя, но между ними есть слой внимания, который помогает декодеру фокусироваться на релевантных частях входящего предложения (это схоже с тем, как механизм внимания организован в моделях seq2seq).

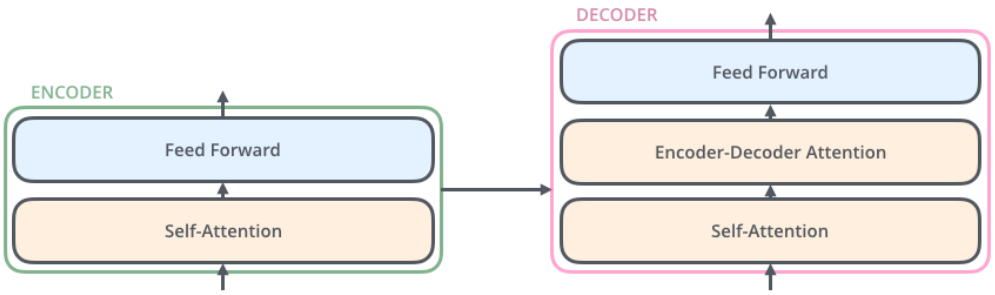

Рисунок 3 - Внутреннее устройство энкодеров и декодеров

3.1.2. Преобразование входной последовательности

Посмотрим на различные векторы/тензоры, и как они передаются от компонента к компоненту, преобразуя входную последовательность обученной модели в выходную.

Как и в случае любого NLP-приложения, мы начинаем с того, что преобразуем слово в вектор, используя алгоритм эмбеддингов слов (word embeddings). Каждое слово

преобразовывается в вектор размерностью 512. На рисунках вектора будут изображены в виде последовательности квадратов.

Эмбеддинги применяются только в самом нижнем энкодере. На уровне абстракции, общей для всех энкодеров, происходит следующее: энкодеры получают набор векторов размерностью 512 (для самого нижнего энкодера это будут эмбеддинги слов, для других – выходные вектора нижестоящих энкодеров). Размер этого набора векторов является гиперпараметром, который мы можем устанавливать, и, по сути, равен длине самого длинного предложения в обучающем корпусе.

После того как слова входящего предложения преобразовались в эмбеддинги, каждый из них в отдельности проходит через два слоя энкодера.

Энкодер получает на вход и обрабатывает набор векторов, проводя их через слой внутреннего внимания и далее – через нейронную сеть прямого распространения, пока, наконец, не передает свой выход следующему энкодеру (см. Рисунок 4).

Слова в каждой из позиций проходят через слой внутреннего внимания. Далее каждое из них переходит в отдельную, но абсолютно идентичную нейронную сеть прямого распространения.

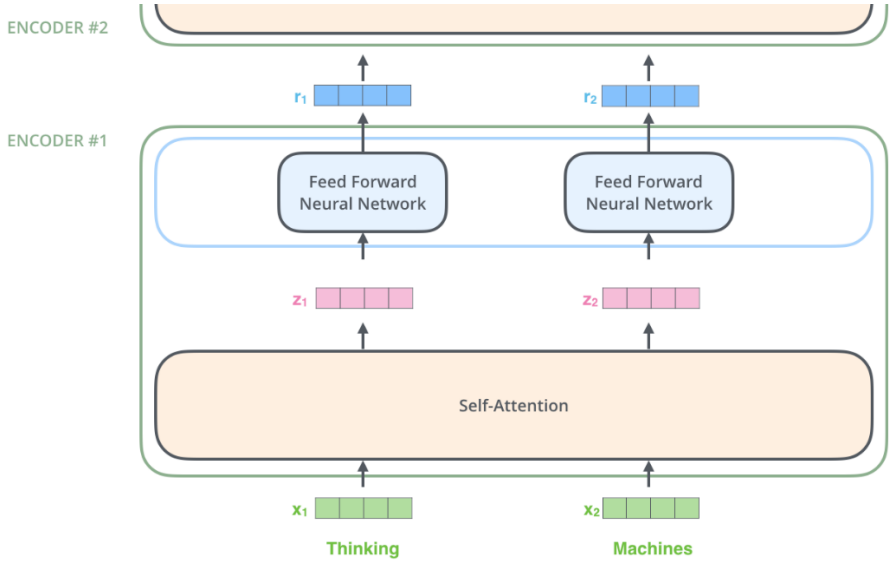

Рисунок 4 - Работа со словами в энкодере

3.1.3. Механизм внутреннего внимания

Механизм внутреннего внимания является современным методом модерлирования «понимания» других релевантных слов при обработке конкретного слова. Прежде всего рассмотрим суть данного механизма высокоуровнево.

Пусть следующее предложение – это входящее предложение, которое мы хотим перевести: «The animal didn't cross the street because it was too tired». К чему относится

«it» в этом предложении? К улице (street) или к животному (animal)? Простой вопрос для человека становится целой проблемой для алгоритма.

Когда модель обрабатывает слово «it», слой внутреннего внимания помогает понять, что «it» относится к «animal».

По мере того, как модель обрабатывает каждое слово (каждую позицию во входной последовательности), внутреннее внимание позволяет модели взглянуть на другие позиции входной последовательности и найти подсказку, помогающую лучше закодировать данное слово.

Механизм внутреннего внимания – это метод, который Трансформер использует для того, чтобы смоделировать «понимание» других релевантных слов при обработке конкретного слова.

Первый этап в вычислении внутреннего внимания – это создать три вектора из каждого входящего вектора (в нашем случае – эмбеддинга каждого слова): вектор запроса (Query vector), вектор ключа (Key vector) и вектор значения (Value vector). (см. Рисунок 5). Эти векторы создаются с помощью перемножения эмбеддинга на три матрицы, которые мы обучили во время процесса обучения.

Эти новые векторы меньше в размере, чем векторы эмбеддингов. Их размерность составляет 64, в то время как эмбеддинги и входящие/выходные векторы энкодера имеют размерность 512. Они не обязаны быть меньше, но в данной случае выбор данной архитектуры модели обусловлен желанием сделать вычисления в слое множественного внимания (multi-head attention) более стабильными.

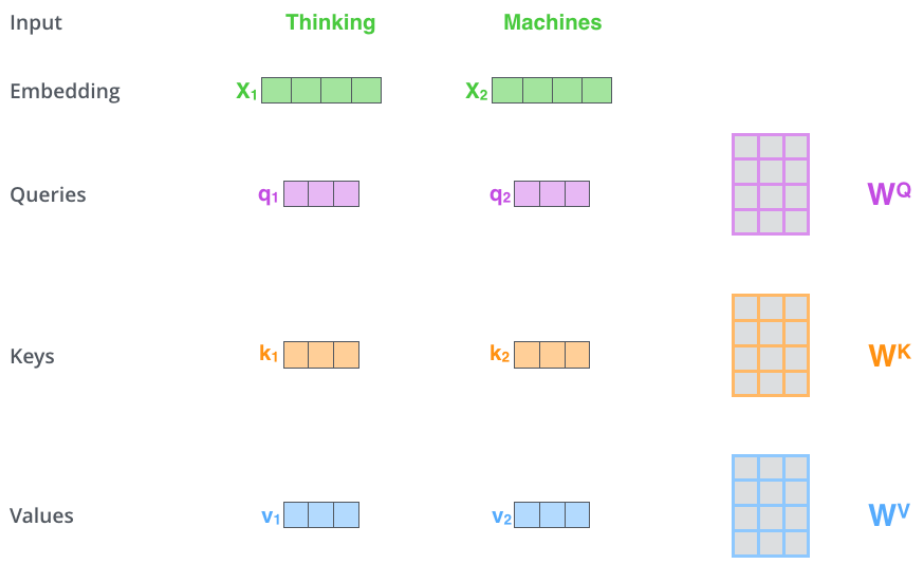

Рисунок 5 - Векторы механизма внимания

Умножение x1 на матрицу весов WQ производит q1, вектор «запроса», относящийся к этому слову. В итоге мы создаем проекции «запроса», «ключа» и «значения» для каждого слова во входящем предложении.

Второй этап вычисления внутреннего внимания – получение коэффициента (score). Допустим, мы подсчитываем внутреннее внимание для первого слова в нашем примере – «Thinking». Нам нужно оценить каждое слово во входящем предложении по отношению к данному слову. Коэффициент определяет, насколько нужно сфокусироваться на других частях входящего предложения во время кодирования слова в конкретной позиции.

Коэффициент подсчитывается с помощью скалярного произведения вектора запроса и вектора ключа соответствующего слова. Таким образом, если мы вычисляем внутреннее внимание для слова в позиции #1, первый коэффициент будет скалярным произведением q1 и k1, второй — скалярным произведением q1 и k2.

Третий и четвертый этапы – разделить эти коэффициенты на 8 (квадратный корень размерности векторов ключа, используемой в статье – 64; данное значение обеспечивает более стабильные градиенты и используется по умолчанию, но возможны также и другие значения), а затем пропустить результат через функцию софтмакс (softmax). Данная функция нормализует коэффициенты так, чтобы они были положительными и в сумме давали 1.

Полученный софтмакс-коэффициент (softmax score) определяет, в какой мере каждое из слов предложения будет выражено в определенной позиции. Очевидно, что слово в своей позиции получит наибольший софтмакс-коэффициент, но иногда полезно учитывать и другое слово, релевантное к рассматриваемому.

Пятый этап – умножить каждый вектор значения на софтмакс-коэффициент (перед их сложением). Интуиция здесь следующая: нужно держать без изменений значения слов, на которых мы фокусируемся, и отвести на второй план нерелевантные слова (умножив их на небольшие значения, например, 0.001).

Шестой этап – сложить взвешенные векторы значения. Это и будет представлять собой выход слоя внутреннего внимания в данной позиции (для первого слова).

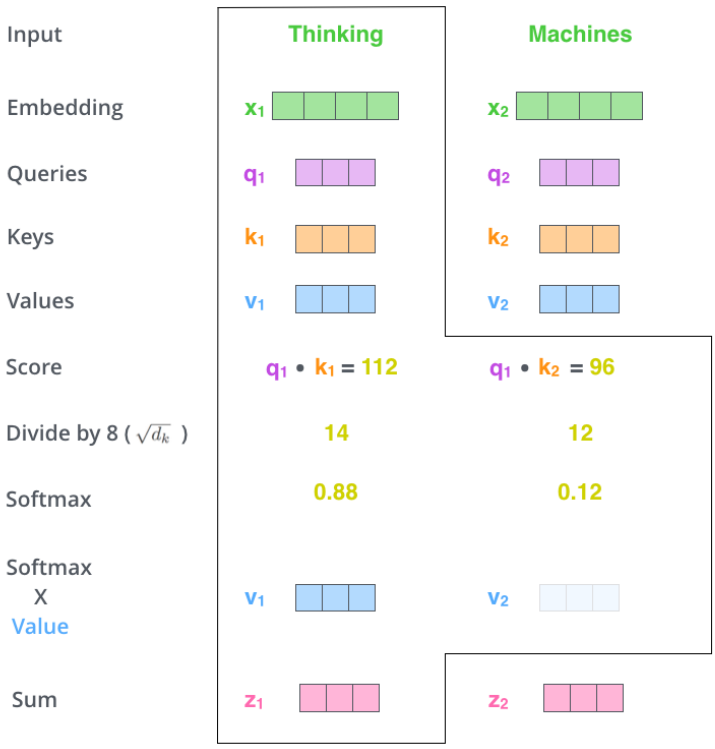

Рисунок 6 - Схема вычисления внутреннего внимания

На этом завершается вычисление внутреннего внимания. В результате мы получаем вектор, который можем передавать дальше в нейронную сеть прямого распространения. В настоящих реализациях, однако, эти вычисления делаются в матричной форме для более быстрой обработки.

3.1.4. Множественное внимание

Внутреннее внимание совершенствуется с помощью добавления механизма, называющегося множественным вниманием (multi-head attention). Данная техника улучшает производительность слоя внутреннего внимания за счет следующих аспектов:

Повышается способность модели фокусироваться на разных позициях. Да, в примере выше, z1 содержит немного от всех других кодировок, но он не может доминировать над самим словом. В случае с переводом предложения вроде «The animal didn’t cross the street because it was too tired», мы хотим знать, к какому слову относится «it».

Слой внимания снабжается множеством «подпространств представлений» (representation subspaces). Как мы увидим далее, с помощью множественного внимания у нас есть не один, а множество наборов матриц запроса/ключа/значения (Трансформер использует 8 «голов» внимания, так что в итоге у нас получается 8 наборов для каждого энкодера/декодера). Каждый из этих наборов создается случайным образом. Далее после обучения каждый набор используется для отображения входящих эмбеддингов (или

векторов с нижестоящих энкодеров/декодеров) в разных подпространствах представлений.

В случае множественного внимания, мы располагаем отдельными WQ/WK/WV матрицами весов для каждой «головы» (см. Рисунок), что в результате дает разные Q/K/V матрицы. Как мы делали ранее, умножаем X на WQ/WK/WV матрицы для получения Q/K/V матриц.

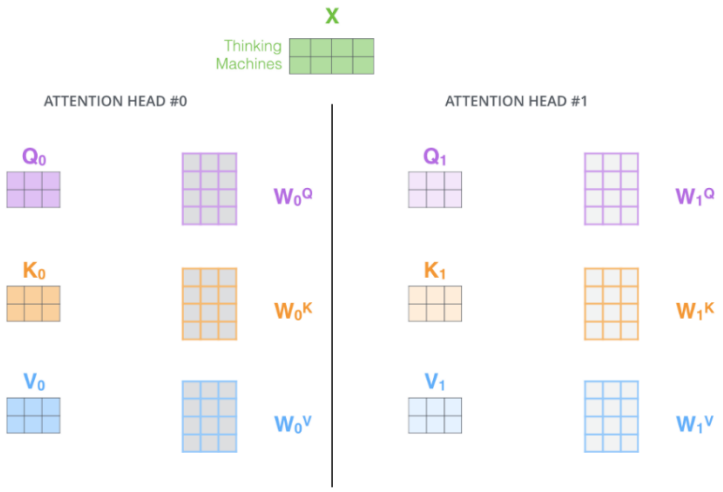

Рисунок 7 - Множественное внимание

После завершения фазы кодирования начинается фаза декодирования. Каждый этап фазы декодирования возвращает элемент выходной последовательности (в данном случае – переводное предложение на английском).

3.1.5. Полный алгоритм обработки

Рассмотрим весь процесс обработки предложения целиком (см. Рисунок). Энкодер начинает обрабатывать входящее предложение. Выход верхнего энкодера затем преобразуется в набор векторов внимания K и V. Они используются всеми декодерами в их «энкодер-декодер» слое внимания, что помогает им фокусироваться на подходящих местах во входящем предложении.

Следующие шаги повторяются до появления специального символа, сообщающего, что декодер Трансформера завершил генерацию выходной последовательности. Выход каждого этапа отправляется на нижний декодер в следующем временном промежутке, и декодеры генерируют свой результат так же, как это делают энкодеры. И точно так же, как мы делали с входами энкодеров, мы добавляем позиционное кодирование на те входы декодеров, которые указывают на позицию каждого слова.

В декодере слой внутреннего внимания может фокусироваться только на предыдущих позициях в выходном предложении. Это делается с помощью маскировки

всех позиций после текущей (устанавливая их в –inf) перед этапом софтмакс в вычислении внутреннего внимания.

Стек декодеров на выходе возвращает вектор чисел с плавающей точкой. Как можно получить из этого вектора слово? За это отвечает линейный слой и следующий за ним слой софтмакс.

Линейный слой – это простая полносвязная нейронная сеть, которая переводит вектор, созданный стеком декодеров, в значительно больший вектор, называемый логит вектором (logits vector).

Слой софтмакс переводит этот показатель в вероятности (положительные числа, сумма которых равна 1). Выбирается ячейка с наиболее высокой вероятностью и на выход данного временного отрезка подается соответствующее слово.

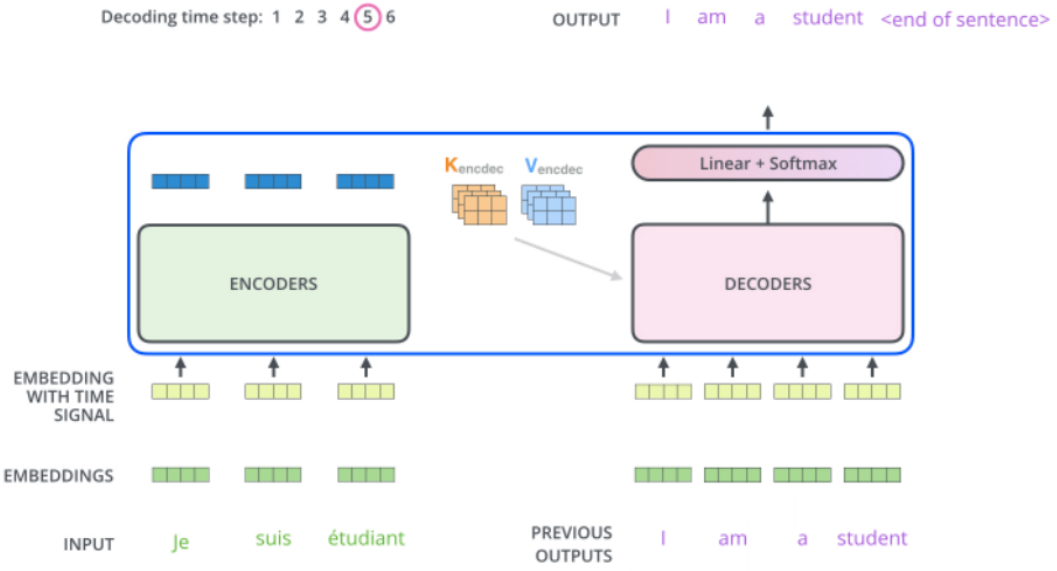

Рисунок 8 - Порядок обработки предложения

3.2. Тематическое моделирование с помощью BERTopic

Тематические модели оказались мощным инструментом для выявления общих тем и основного повествования в тексте без обучения. Обычные модели, такие как скрытое распределение Дирихле (LDA) и неотрицательная матричная факторизация (NMF) описывают документ как набор слов и моделируют каждый документ. как смесь скрытых тем.

Одним из недостатков этих моделей является то, что при представлении набора слов они игнорируют семантические отношения между словами. Поскольку эти представления не учитывают контекст слов в предложении, входной пакет слов может неточно представлять документы.

В качестве ответа на этот вопрос методы эмбеддинга текста быстро стали популярными в области обработки естественного языка. В частности, представление двунаправленного кодировщика из трансформеров (BERT) и его варианты показали отличные результаты в создании контекстных векторных представлений слов и предложений. Семантические свойства этих векторных представлений позволяют кодировать смысл текстов таким образом, чтобы подобные тексты были близки в векторном пространстве.

Хотя методы эмбеддинга использовались для решения самых разных задач, от классификации до нейронных поисковых систем, исследователи начали применять эти мощные контекстные представления для тематического моделирования. Предыдущие авторы продемонстрировали жизнеспособность кластеризации эмбеддингов с помощью методов на основе центроидов по сравнению с традиционными методами, такими как LDA, как способ представления тем. Из этих сгруппированных вложений были извлечены тематические представления путем встраивания слов и поиска тех, которые находятся в непосредственной близости от центроида кластера. Точно так же Top2Vec использует представления слов и документов модели Doc2Vec для находящихся рядом векторов тем, документов и слов. В данном случае документы группируются, а тематические представления создаются путем поиска слов, близких к центроиду кластера. Интересно, что хотя тематические представления создаются с точки зрения центроида, кластеры создаются с точки зрения плотности, а именно с использованием HDBSCAN.

Вышеупомянутые методы тематического моделирования предполагают, что слова, находящиеся в непосредственной близости от центра кластера, наиболее репрезентативны для этого кластера и, следовательно, для темы. На практике, однако, кластер не всегда будет лежать внутри сферы вокруг центра тяжести кластера. Таким образом, это предположение не может быть верным для каждого кластера документов и представления этих кластеров, и поэтому тема может ввести в заблуждение. Хотя предыдущие авторы пытались решить эту проблему путем переранжирования тематических слов на основе их частоты в кластере, первоначальные кандидаты по-прежнему генерируются с точки зрения центроида.

В рассматриваемой статье авторы представляют BERTopic, тематическую модель, которая использует методы кластеризации и вариант TF-IDF на основе классов (c-TF-IDF) для создания согласованных представлений темы. В частности, сначала создается эмбеддинг документов, используя предварительно обученную языковую модель для получения информации на уровне документа. Во-вторых, сначала уменьается

размерность эмбеддинга документов, прежде чем создавать семантически похожие кластеры документов, каждый из которых представляет отдельную тему. В-третьих, чтобы преодолеть точку зрения, основанную на центроидах, была разработана основанная на классах версия TF-IDF для извлечения представления темы из каждой темы. Эти три независимых шага позволяют создать гибкую модель темы, которую можно использовать в различных вариантах использования, таких как динамическое моделирование темы.

3.2.1. Описание BERTopic

BERTopic генерирует представление тем в три этапа. Во-первых, каждый документ преобразуется в векторное представление с использованием предварительно обученной языковой модели. Затем перед кластеризацией этих эмбеддингов размерность полученных векторов уменьшается для оптимизации процесса кластеризации. Наконец, из кластеров документов извлекаются тематические представления с использованием варианта TF-IDF на основе классов.

Эмбеддинги документов

В BERTopic загружаются документы для создания представлений в векторном пространстве, которые можно семантически сравнивать. Предполагается, что документы, содержащие одну и ту же тему, семантически схожи. Для выполнения этапа встраивания BERTopic использует структуру Sentence-BERT (SBERT). Эта структура позволяет пользователям преобразовывать предложения и абзацы в плотные векторные представления с использованием предварительно обученных языковых моделей. Она обеспечивает высочайшую производительность при выполнении различных задач по эмбеддингу предложений.

Однако эти вложения в основном используются для кластеризации семантически схожих документов и не используются напрямую при создании тем. Для этой цели можно использовать любой другой метод эмбеддинга, если языковая модель, генерирующая вложения документов, была точно настроена на семантическое сходство. В результате качество кластеризации в BERTopic будет повышаться по мере разработки новых и улучшенных языковых моделей. Это позволяет BERTopic постоянно развиваться в соответствии с современными технологиями встраивания.

Кластеризация документов

Было показано, что по мере увеличения размерности данных расстояние до ближайшей точки данных приближается к расстоянию до самой дальней точки данных. В результате в многомерном пространстве понятие пространственной локальности становится нечетким, а меры расстояния мало различаются.

Хотя для преодоления этого проклятия размерности существуют кластерные подходы, более простым подходом является уменьшение размерности эмбеддингов. Хотя PCA и t-SNE являются хорошо известными методами уменьшения размерности, UMAP продемонстрировал сохранение большего количества локальных и глобальных особенностей многомерных данных в более низких проекционных измерениях. Более того, поскольку он не имеет вычислительных ограничений на встраивание измерений, UMAP может использоваться в языковых моделях с различным размерным пространством.

Уменьшенные вложения представляют собой кластеризацию с использованием HDBSCAN. Это расширение DBSCAN, которое находит кластеры различной плотности путем преобразования DBSCAN в алгоритм иерархической кластеризации. HDBSCAN моделирует кластеры, используя подход мягкой кластеризации, позволяющий моделировать шум как выбросы. Это предотвращает отнесение несвязанных документов к какому-либо кластеру и, как ожидается, улучшит представление тем.

Представление тем

Представления темы моделируются на основе документов в каждом кластере, где каждому кластеру будет назначена одна тема. Для каждой темы мы хотим знать, что делает одну тему, основываясь на распределении кластерных слов, отличной от другой? Для этой цели мы можем изменить TF-IDF, меру представления важности слова в документе, чтобы вместо этого он позволял представлять важность термина для темы.

Классическая процедура TF-IDF объединяет две статистики: частоту терминов и обратную частоту документа:

$$W_{t,d} = tf_{t,d} \cdot \log(\frac{N}{df_t})$$

Где частота термина моделирует частоту термина t в документе d. Обратная частота документа измеряет, сколько информации термин предоставляет документу, и рассчитывается путем логарифмирования количества документов в корпусе N, деленного на общее количество документов, содержащих t.

Авторы обобщают эту процедуру на кластеры документов. Во-первых, мы рассматриваются все документы в кластере как один документ, просто объединяя документы. Затем TF-IDF настраивается для учета этого представления путем преобразования документов в кластеры:

$$W_{t,c} = tf_{t,c} \cdot \log(1 + \frac{A}{tf_t})$$

Где частота термина моделирует частоту термина t в классе c или в этом экземпляре. Здесь класс c представляет собой набор документов, объединенных в один документ для каждого кластера. Затем обратная частота документа заменяется обратной частотой класса, чтобы измерить, сколько информации термин предоставляет классу. Он рассчитывается путем логарифмирования среднего количества слов в классе A, деленного на частоту термина t во всех классах. Чтобы вывести только положительные значения, мы добавляем единицу к делению внутри логарифма.

Таким образом, эта основанная на классах процедура TF-IDF моделирует важность слов в кластерах, а не в отдельных документах. Это позволяет генерировать распределения тематических слов для каждого кластера документов.

Наконец, итеративно объединяя представления c-TF-IDF наименее распространенной темы с наиболее похожей, можно уменьшить количество тем до заданного пользователем значения.

3.2.2. Подготовка к экспериментам

Набор данных

Для проверки BERTopic использовались три набора данных, а именно 20 NewsGroups, BBC News и твиты президента Трампа.

Набор данных 20 NewsGroups содержит 16309 новостных статей по 20 категориям. Набор данных BBC News4 содержит 2225 документов с веб-сайта BBC News за период с 2004 по 2005 год. Оба набора данных были получены с помощью пакета OCTIS и предварительно обработаны путем удаления пунктуации, лемматизации, удаления стоп-слов и удаления документов, содержащих менее 5 слов.

Чтобы представить более свежие данные в краткой текстовой форме, собраны все твиты президента Трампа до и во время его президентства. Данные содержат 44 253 твита, не считая ретвитов, в период с 2009 по 2021 год. В обоих наборах данных использованы строчные буквы для всех токенов.

Модели

BERTopic будет сравниваться с LDA, NMF, CTM и Top2Vec. LDA и NMF запускались через OCTIS с параметрами по умолчанию. Модель SBERT all-mpnet-base-v2 использовалась в качестве модели эмбеддинга для BERTopic и CTM. Были смоделированы два варианта Top2Vec, один с Doc2Vec и один с моделью SBERT all-mpnet-base-v27.

Для объективного сравнения между BERTopic и Top2Vec параметры HDBSCAN и UMAP были зафиксированы между тематическими моделями.

Чтобы измерить обобщаемость BERTopic для языковых моделей, в экспериментах с BERTopic использовались четыре разные языковые модели, а именно универсальный кодировщик предложений, Doc2Vec и модели SBERT all-MiniLM-L6-v2 (MiniLM) и all-mpnet-base-v2 (MPNET).

Метрика оценивания

Производительность тематических моделей в этой статье отражается двумя широко используемыми показателями, а именно согласованностью тем (topic coherence - TC) и разнообразием тем (topic diversity - TD). Для каждой тематической модели ее согласованность тем оценивалась с использованием нормализованной точечной взаимной информации (NPMI). Было показано, что эта мера согласованности имитирует человеческое суждение с разумной эффективностью. Мера находится в диапазоне от [-1, 1], где 1 указывает на идеальную связь (ассоицацию). Разнообразие тем — процент уникальных слов по всем темам. Показатель находится в диапазоне от [0, 1], где 0 указывает на повторяющиеся темы, а 1 указывает на более разнообразные темы. В диапазоне от 10 до 50 тем с шагом 10 оценка NPMI рассчитывалась на каждом этапе для каждой тематической модели. Все результаты усреднялись по 3 запускам на каждом шаге. Чтобы оценить модели динамических тем, оценка NPMI была рассчитана для 50 тем для каждого временного шага, а затем усреднена. Все результаты были усреднены по 3 запускам.

3.2.3. Оценка результатов

Рисунок 9 демонстрирует, что BERTopic обычно имеет высокие показатели согласованности тем во всех наборах данных. Он имеет самые высокие оценки в слегка предварительно обработанном наборе данных, твитах Трампа, оставаясь при этом конкурентоспособным в тщательно предварительно обработанных наборах данных, 20 NewsGroups и BBC News. Хотя BERTopic демонстрирует конкурентоспособные оценки разнообразия тем, он постоянно уступает CTM. Это согласуется с результатами, указывающими на большое разнообразие тем, хотя и с использованием другого показателя разнообразия тем.

	20 NewsGroups		BBC News		Trump	
	TC	TD	TC	TD	TC	TD
LDA	.058	.749	.014	.577	-.011	.502
NMF	.089	.663	.012	.549	.009	.379
T2V-*MPNET*	.068	.718	-.027	.540	-.213	.698
T2V-*Doc2Vec*	.192	.823	.171	.792	-.169	.658
CTM	.096	.886	.094	.819	.009	.855
BERTopic-*MPNET*	.166	.851	.167	.794	.066	.663

Рисунок 9 – Сравнение работы разных методов моделирования тем

Рисунок 10 демонстрирует стабильность BERTopic с точки зрения согласованности тем и разнообразия тем для языковых моделей SBERT. В результате меньшая и более быстрая модель all-MiniLM-L6-v2 может быть предпочтительнее, когда доступна ограниченная мощность графического процессора.

	20 NewsGroups		BBC News		Trump	
	TC	TD	TC	TD	TC	TD
BERTopic-*USE*	.149	.858	.158	.764	.051	.684
BERTopic-*Doc2Vec*	.173	.871	.168	.819	-.088	.536
BERTopic-*MiniLM*	.159	.833	.170	.802	.060	.660
BERTopic-*MPNET*	.166	.851	.167	.792	.066	.663

Рисунок 10 - Сравнение разных языковых моделей

Хотя USE и Doc2Vec в BERTopic в целом имеют одинаковую производительность, Doc2Vec имеет низкие оценки в наборе данных твитов. Это отражено в результатах, где Top2Vec с Doc2Vec имеет низкую производительность. Эти результаты свидетельствуют о том, что Doc2Vec испытывает трудности с созданием точных представлений набора данных Трампа.

Что касается согласованности тем, Top2Vec с эмбеддингами Doc2Vec демонстрирует конкурентоспособную производительность. Однако, когда используются эмбеддинги MPNET, как согласованность тем, так и разнообразие падают во всех наборах данных, что позволяет предположить, что Top2Vec может не лучше всего подходить для эмбеддингов, отличных от тех, которые созданы с помощью Doc2Vec.

В свою очередь, это также говорит о том, почему BERTopic остается конкурентоспособным независимо от модели эмбеддинга. Разделив процесс эмбеддинга документов и построение распределения слов по темам, BERTopic обеспечивает гибкость процедуры эмбеддинга.

4. ПРАКТИЧЕСКАЯ ЧАСТЬ

Авторы статьи выложили в общий доступ все материалы, относящиеся к опубликованной статье, включая коды программ, наборы данных, обученные модели и результаты [4].

4.1. Обзор страницы в GitHub

Рисунок 11 демонстрирует страницу на сайте GitHub, посвященную данной статье.

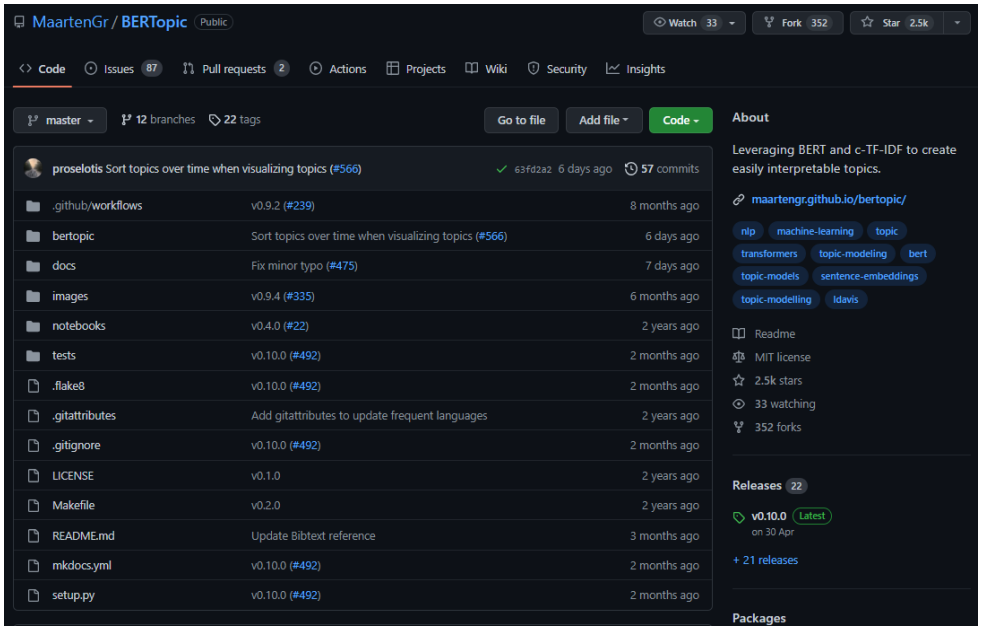

Рисунок 11 - Страница статьи в GitHub

На данной странице авторы дают краткое описание содержимого репозитория, а также описывают порядок работы с ним в нескольких разделах страницы.

4.1.1. Описание статьи

В первом разделе помимо описания авторы также указывают ссылки на сторонние источники, связанные со статьей (см. Рисунок 12).

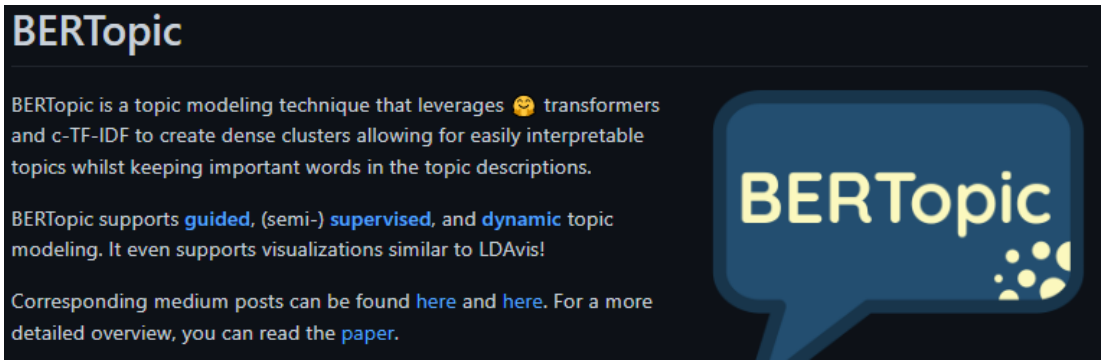

Рисунок 12 - Описание материалов

BERTopic поддерживает управляемое, (полу-) контролируемое и динамическое тематическое моделирование.

Текст статьи можно найти на сайте arxiv.org [5].

4.1.2. Установка

Далее разработчики описывают порядок загрузки и установки репозитория локально для подробного изучения и тестирования работы.

Необходимо установить соответствующий пакет:

```
pip install bertopic
```

Также возможно потребуется установить дополнительные зависимости для использования ряда инструментов из сферы NLP:

```
pip install bertopic[flair]
pip install bertopic[gensim]
pip install bertopic[spacy]
pip install bertopic[use]
```

4.1.3. Демо-пример

В следующем разделе авторы оставили ссылку на демострационные примеры работы модели на платформе Google Colaboratory. (см. Рисунок 13).

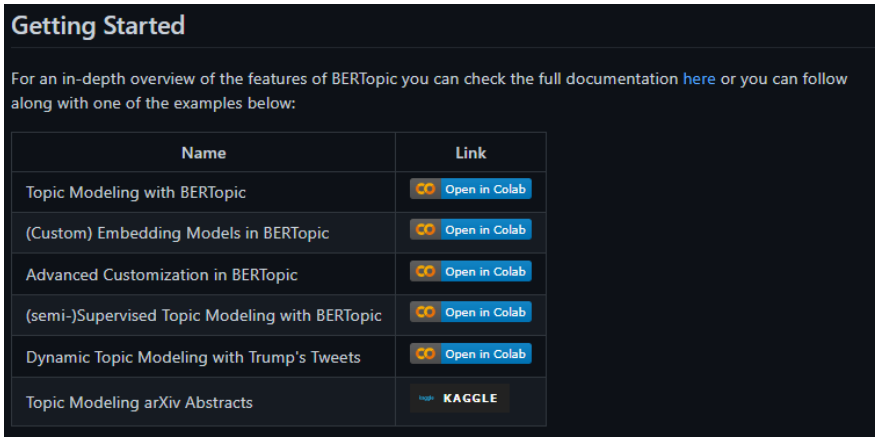

Рисунок 13 - Описание демонстрационного примера

4.1.4. Быстрый запуск

Далее следует раздел, посвященный самостоятельному проведению экспериментов с предоставленными материалами.

Начнем с извлечения тем из известного набора данных 20 групп новостей, содержащего документы на английском языке:

```
from bertopic import BERTopic
```

```
from sklearn.datasets import fetch_20newsgroups

docs = fetch_20newsgroups(subset='all',  remove=('headers', 'footers',
'quotes'))['data']

topic_model = BERTopic()
topics, probs = topic_model.fit_transform(docs)
```

После создания тем и их вероятностей мы можем получить доступ к часто сгенерированным темам:

```
>>> topic_model.get_topic_info()

Topic  Count  Name
-1     4630   -1_can_your_will_any
0      693    49_windows_drive_dos_file
1      466    32_jesus_bible_christian_faith
2      441    2_space_launch_orbit_lunar
3      381    22_key_encryption_keys_encrypted
```

-1 относится ко всем выбросам и обычно должен игнорироваться. Далее, давайте взглянем на наиболее часто сгенерированную тему, тему 0:

```
>>> topic_model.get_topic(0)

[('windows', 0.006152228076250982),
 ('drive', 0.004982897610645755),
 ('dos', 0.004845038866360651),
 ('file', 0.004140142872194834),
 ('disk', 0.004131678774810884),
 ('mac', 0.003624848635985097),
 ('memory', 0.0034840976976789903),
 ('software', 0.0034415334250699077),
 ('email', 0.0034239554442333257),
 ('pc', 0.003047105930670237)]
```

Для генерации тем на других языках предлагается инициализировать модель следующим образом:

```
topic_model = BERTopic(language="multilingual")
```

4.2. Демонстрационный пример работы модели

Авторами статьи был предоставлен .ipynb-файл, в котором можно проверить работу обученных моделей на практике [6]. Далее будет выполнен обзор и тестирование предложенных практических материалов.

4.2.1. Подготовка данных

В начале файла следуют служебные строки кода, необходимые для установки и импорта всех библиотек. Далее следует фрагмент с загрузкой исходного набора данных и его сохранения в виде списка. В качестве данных использовалась выгрузка материалов с сайта EMVCo, заранее очищенная от стоп-слов и сохраненная в виде txt-файлов, см. Рисунок 14.

```
from sklearn.datasets import fetch_20newsgroups
docs = fetch_20newsgroups(subset='all',  remove=('headers', 'footers', 'quotes'))['data']
```

```
docs[:2]
```

```
["\n\nI am sure some bashers of Pens fans are pretty confused about the lack\nof any kind
recent Pens massacre of the Devils. Actually,\nI am  bit puzzled too and a bit relieved. H
to put an end\nto non-PIttsburghers' relief with a bit of praise for the Pens. Man, they\n
Devils worse than I thought. Jagr just showed you why\nhe is much better than his regular
is also a lot\nfo fun to watch in the playoffs. Bowman should let JAgr have a lot of\nfun
of games since the Pens are going to beat the pulp out of Jersey anyway. I was very disapp
the Islanders lose the final\nregular season game.          PENS RULE!!!\n\n",
 'My brother is in the market for a high-performance video card that supports\nVESA local
RAM.  Does anyone have suggestions/ideas on:\n\n  - Diamond Stealth Pro Local Bus\n\n  - O
1280\n\n  - ATI Graphics Ultra Pro\n\n  - Any other high-performance VLB card\n\n\nPlease
Thank you!\n\n  - Matt\n']
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import os
```

```
directory = '/content/drive/MyDrive/datasets/emvco_cleaned'

docs=[]

for filename in os.listdir(directory):
    f = os.path.join(directory, filename)
    # checking if it is a file
    if os.path.isfile(f):
      with open(f, encoding = 'utf-8') as f:
         doc_text = str(f.read())

      docs.append(doc_text)

print(len(docs))
```

```
965
```

Рисунок 14 – Загрузка текстов

4.2.2. Загрузка модели и обработка текстов

Далее выполняется загрузка модели BERTopic с настройками по умолчанию, см. Рисунок 15. Также данный код выполняет преобразование текстов в эмбеддинги, снижает размерность и выполняет кластеризацию полученных ветекторов.

```
from bertopic import BERTopic

topic_model = BERTopic(language="english", calculate_probabilities=True, verbose=True)
topics, probs = topic_model.fit_transform(docs)
```

```
Downloading: 100% 1.18k/1.18k [00:00<00:00, 31.7kB/s]
Downloading: 100% 190/190 [00:00<00:00, 5.71kB/s]
Downloading: 100% 10.2k/10.2k [00:00<00:00, 322kB/s]
Downloading: 100% 612/612 [00:00<00:00, 18.7kB/s]
Downloading: 100% 116/116 [00:00<00:00, 3.76kB/s]
Downloading: 100% 39.3k/39.3k [00:00<00:00, 1.03MB/s]
Downloading: 100% 349/349 [00:00<00:00, 9.98kB/s]
Downloading: 100% 90.9M/90.9M [00:01<00:00, 56.9MB/s]
Downloading: 100% 53.0/53.0 [00:00<00:00, 1.33kB/s]
Downloading: 100% 112/112 [00:00<00:00, 3.45kB/s]
Downloading: 100% 466k/466k [00:00<00:00, 1.50MB/s]
Downloading: 100% 350/350 [00:00<00:00, 10.5kB/s]
Downloading: 100% 13.2k/13.2k [00:00<00:00, 293kB/s]
Downloading: 100% 232k/232k [00:00<00:00, 1.34MB/s]
Batches: 100% 31/31 [00:29<00:00, 10.21it/s]
2022-06-20 13:53:42,759 - BERTopic - Transformed documents to Embeddings
/usr/local/lib/python3.7/dist-packages/numba/np/ufunc/parallel.py:363: NumbaWarning: The TBB threading layer
  warnings.warn(problem)
2022-06-20 13:53:53,009 - BERTopic - Reduced dimensionality
2022-06-20 13:53:53,110 - BERTopic - Clustered reduced embeddings
```

Рисунок 15 - Загрузка модели и обработка текстов

4.2.3. Анализ полученных данных

Из текстов был извлечен ряд тем, которые можно вывести по мере убывания количества текстов, отнесенных к данной теме, см. Рисунок 16. Первая тема с индексом -1 обозначает выбросы, которые не входят в кластеры тематик.

```
freq = topic_model.get_topic_info(); freq.head(10)
```

	Topic	Count	Name
0	-1	195	-1_application_payment_data_card
1	0	121	0_emvco_product_terminal_document
2	1	71	1_put_test_00_lt
3	2	41	2_bb_3ds_message_acs
4	3	40	3_cvm_specification_reader_emvco
5	4	37	4_specification_pcd_picc_bulletin
6	5	33	5_change_application_accumulator_editorial
7	6	32	6_aid_value_entry_terminal
8	7	30	7_yes_ics_support_registration
9	8	28	8_atr_terminal_basic_character

Рисунок 16 - Топ-10 выделенных тем

Можно вывести топ слов, отнесенных, например, к теме 0 и теме 1. См. Рисунок.

```
topic_model.get_topic(0)  # Select the most frequent topic
```

```
[('emvco', 0.033296448680755465),
 ('product', 0.03237638693911596),
 ('terminal', 0.029911284579989032),
 ('document', 0.0283740648874216),
 ('party', 0.025847865264736303),
 ('approval', 0.025650053072237902),
 ('bulletin', 0.02563007451159864),
 ('service', 0.021292192661204978),
 ('case', 0.0203393781846066),
 ('set', 0.01841943207055006)]
```

```
topic_model.get_topic(1)  # Select the most frequent topic
```

```
[('put', 0.04462891371266428),
 ('test', 0.04335003529049949),
 ('00', 0.042242457183546266),
 ('lt', 0.041576280743855526),
 ('picc', 0.022471094693103554),
 ('pcd', 0.02110108608198897),
 ('type', 0.02051378061557065),
 ('send', 0.020253274282481023),
 ('emvco', 0.020139469044251953),
 ('frame', 0.01843981092715872)]
```

Рисунок 17 - Топ слов в темах 0 и 1

Полученные темы можно визуализировать с помощью библиотеки-сервиса Plotly. Графики являются интерактивными и позволяют изменять масштаб и взаимодействовать с элементами.

Визуализируем все тематики на карте с учетом расстояния тем друг от друга, см. Рисунок 18.

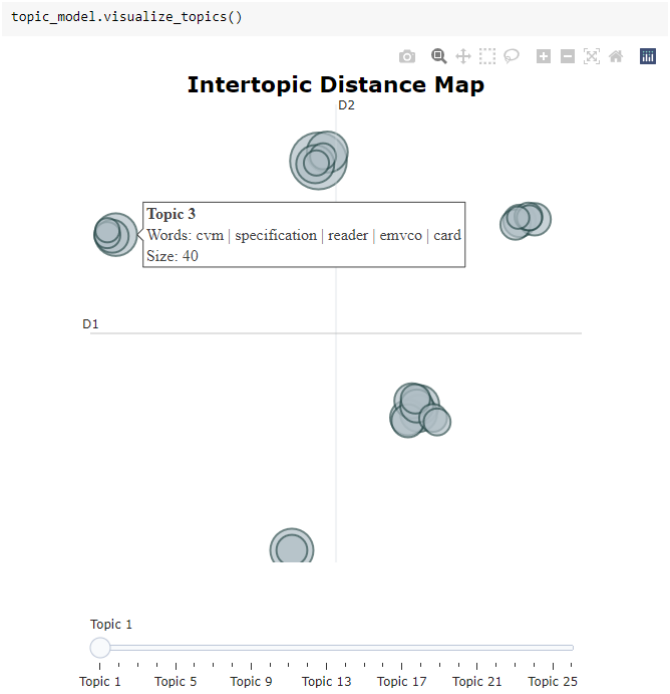

Рисунок 18 - Карта полученных тем

Также можно визуализировать вероятности (уверенность) отнесения каждого документа к определенным темам, см. Рисунок 19.

```
topic_model.visualize_distribution(probs[1], min_probability=0.015)
```

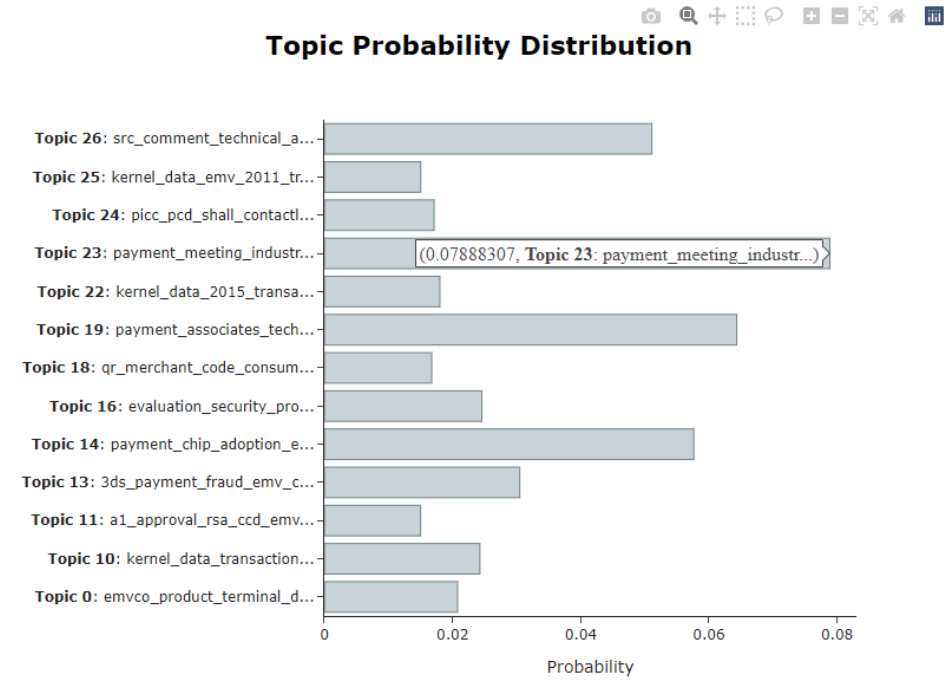

Рисунок 19 - Вероятность отнесения к темам

Также можно провести иерархическую кластеризацию полученных тем и посмотреть, какие темы можно друг с другом объединить, см. Рисунок.

```
topic_model.visualize_hierarchy(top_n_topics=50)
```

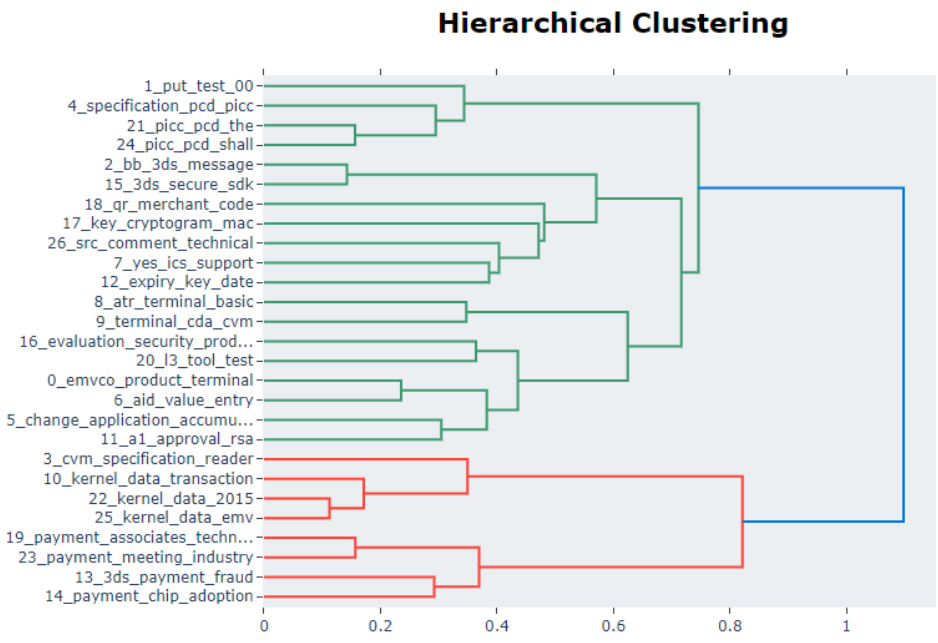

Рисунок 20 - Иерархическая кластеризация тем

Также для каждой темы можно вывести топ слов, которые отличают одну тему от другой с помощью метода c-TF-IDF.

```
topic_model.visualize_barchart(top_n_topics=25)
```

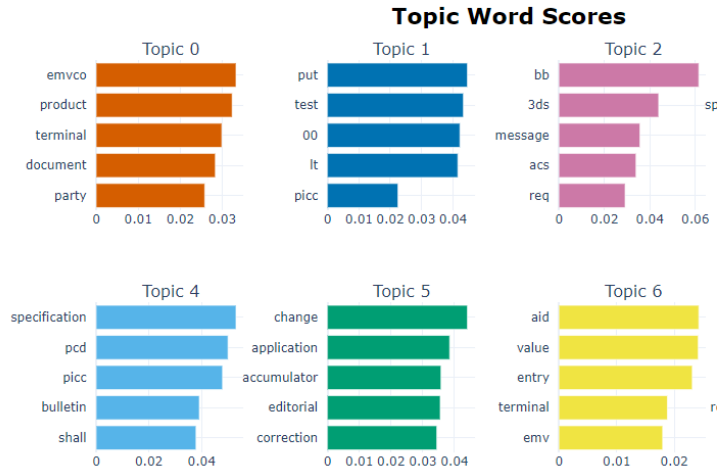

Рисунок 21 - Визуализация топ-5 слов в темах

Также можно вычислить и визуализировать сходство тем между собой в виде матрицы, см. Рисунок.

```
topic_model.visualize_heatmap(n_clusters=20, width=1000, height=1000)
```

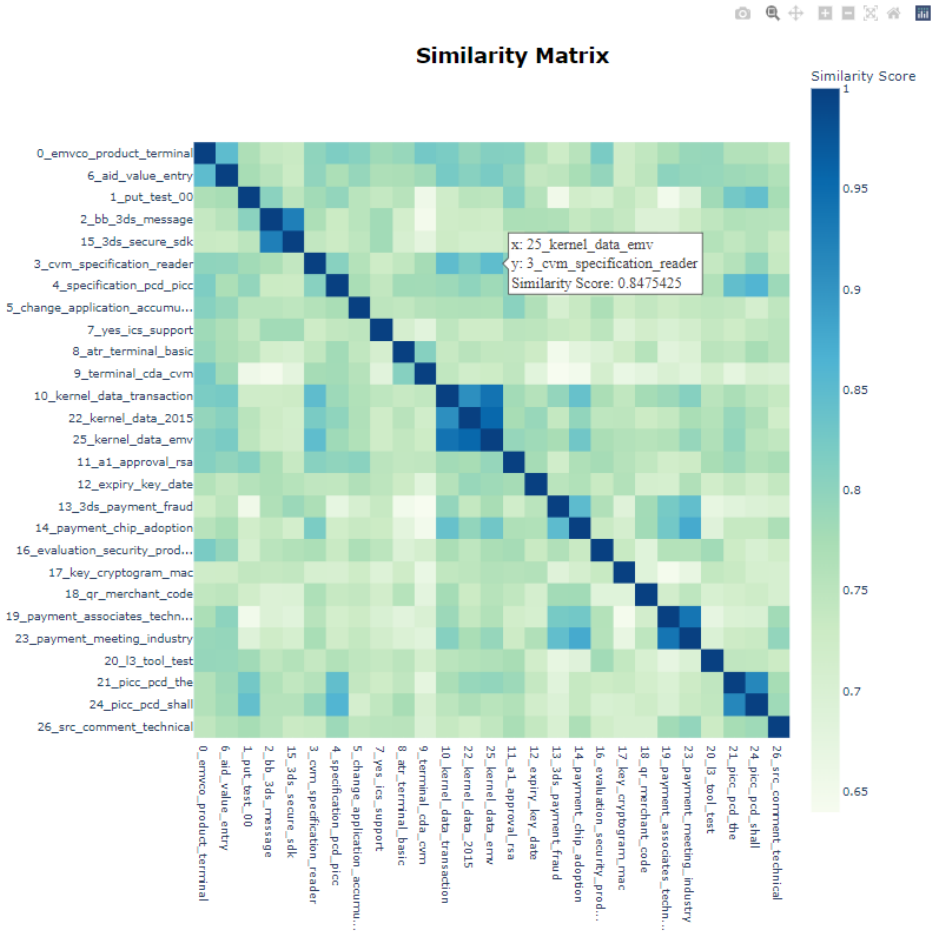

Рисунок 22 - Матрица сходства тем между собой

5. ВЫВОДЫ

В рамках домашнего задания был выполнен обзор теоретических и практических материалов, связанных со статьей «BERTopic: Neural topic modeling with a class-based TF-IDF procedure».

Была разработана BERTopic, тематическая модель, которая расширяет подход к кластеризации эмбеддингов, используя самые современные языковые модели и применяя процедуру TF-IDF на основе классов для создания представлений темы. Благодаря разделению процесса кластеризации документов и создания тематических представлений в модель вводится значительная гибкость, что упрощает ее использование. В статье представлен углубленный анализ BERTopic, начиная от оценочных исследований с классическими показателями согласованности тем и заканчивая, в том числе, анализом времени выполнения. Эксперименты показывают, что BERTopic обучается ассоциативным связям языка и демонстрирует конкурентоспособную и стабильную производительность в различных задачах.

В практической части был выполнен обзор содержимого репозитория авторов статьи на GitHub, а также было выполнено тестирование моделей в демо-примере. В результате тестирования можно подтвердить, что модель действительно способна выделять темы из набора документов и проводить кластеризацию.

6. СПИСОК ИСТОЧНИКОВ

1. Browse State-of-the-Art. – Текст. Изображение: электронные // Papers With Code : [сайт]. – URL: https://paperswithcode.com/sota (дата обращения: 15.06.2022).
2. BERTopic: Neural topic modeling with a class-based TF-IDF procedure. – Текст. Изображение: электронные // Papers With Code : [сайт]. – URL: https://paperswithcode.com/paper/bertopic-neural-topic-modeling-with-a-class (дата обращения: 15.06.2022).
3. Transformer в картинках. – Текст. Изображение: электронные // Хабр : [сайт]. – URL: https://habr.com/ru/post/486358/ (дата обращения: 15.06.2022).
4. BERTopic. – Текст. Изображение: электронные // GitHub : [сайт]. – URL: https://github.com/MaartenGr/BERTopic (дата обращения: 15.06.2022).
5. BERTopic: Neural topic modeling with a class-based TF-IDF procedure. – Текст. Изображение: электронные // arXiv : [сайт]. – URL: https://arxiv.org/pdf/2203.05794v1.pdf (дата обращения: 15.06.2022).
6. BERTopic.ipynb. – Текст. Изображение: электронные // Colaboratory : [сайт]. – URL: https://colab.research.google.com/drive/1FieRA9fLdkQEGDIMYl0I3MCjSUKVF8C-?usp=sharing (дата обращения: 15.06.2022).