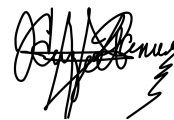


Санкт-Петербургский государственный университет

КИЗЕЕВ Данил Владимирович



Выпускная квалификационная работа
«Применение BERT к задаче автоматической категоризации
постов в социальной сети»

Уровень образования: **бакалавриат**

Направление: **01.03.02 «Прикладная математика и информатика»**

Основная образовательная программа: **СВ.5004.2017 «Прикладная**
математика и информатика»

Профиль: исследование операций и принятие решений в задачах
оптимизации, управления и экономики

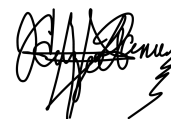
Научный руководитель:
доцент кафедры теоретической кибернетики,
к.ф.-м.н., Липкович Михаил

Рецензент:
аналитик ООО НКО «ЮМани»,
Горлова Марина Владимировна

Санкт-Петербург
2021

Saint Petersburg State University

KIZEEV Danil Vladimirovich



Bachelor Thesis

«BERT applying for automatic categorization posts in the social network»

Qualification level: **bachelor**

Specialty: **01.03.02 «Applied mathematics and computer science»**

Main educational program: **CB.5004.2017 «Applied mathematics and computer science»**

Specialization: Operational Research and Decision Making in Optimization,
Control and Economics Problems

Scientific Supervisor:

Associate Prof. of Theoretical Cybernetics Dep.,

Candidate of Physico-Mathematical Sciences,

Lipkovich Mikhail

Reviewer:

“YooMoney”, NBCO LLC analyst,

Gorlova Marina Vladimirovna

Saint Petersburg

2021

Содержание

Введение	5
Постановка задачи	7
Пример исходных данных для изображения и ожидаемых результатов	8
Обзор литературы	9
Глава 1. Модель BERT	12
Алгоритм решения задачи	12
Считывание данных	13
Обработка данных	13
Реализация в языке Python	14
Преобразование распределений меток в предложения	14
Получение представлений объектов с помощью BERT	14
Реализация в языке Python	15
Загрузка данных на видеокарту	15
Понижение размерности векторов с помощью UMAP	16
Реализация в языке Python	17
Кластеризация с помощью метода k -средних	17
Обоснованность использования нормализации векторов	18
Реализация в языке Python	20
Оценка качества кластеризации с помощью метрики «Силуэт»	20
Метод локтя и метод среднего силуэта	20
Реализация в языке Python	22
Сопоставление категориям объектов	22
Разметка данных	23
Оценка качества модели	23
Глава 2. Модель BERTopic	25
Мера TF-IDF	25
Мера c -TF-IDF	25
Количество тематик	25

Удаление выбросов	26
Визуализация кластеров тематик	26
Порог на значения близости	26
Оценка качества модели	27
Глава 3. Модель word2vec	30
Основной алгоритм	30
Реализация в языке Python	31
Разметка для векторов категорий	31
Порог на значения косинусной близости	31
Оценка качества модели	32
Подбор границы для косинусного меры близости	33
Выводы	34
Заключение	36
Список литературы	37

Введение

В наше время происходит бурное развитие социальных сетей. Пользователи потребляют, производят и делятся большим количеством информации (контентом). Это могут быть изображения, аудио- и видеофайлы, комментарии в текстовом виде. Анализ различных типов информации приведён в [1, 2].

Контента, создаваемого пользователем, очень много и зачастую пользователю самому сложно найти то, что его интересует. Далее будем называть публикации, создаваемые пользователями в соц.сетях, *постами*. Для того, чтобы помочь пользователю и предложить контент, были придуманы рекомендательные системы [3, 4].

Так формируется общая задача рекомендации: дана социальная сеть и её пользователи, опубликованные ими посты. Каждый пост состоит из изображения или видео. Требуется создать модель, которая будет предлагать интересный пользователю контент. В данной работе мы не создаём рекомендации, а решаем промежуточную задачу.

В процессе регистрации пользователь выбирает некоторое количество интересных ему разделов. Чтобы была возможность рекомендовать пользователю контент из области его интересов, нужно узнать тематику выложенного поста. Это производится с помощью определения принадлежности поста к той или иной категории. В нашей ситуации все категории предзаданы и каждой из них нужно сопоставить пост. Данная задача имеет название *автоматической категоризации* [5, 6] постов.

Важно заметить, что задача категоризации отличается от задачи классификации. Это происходит потому, что категории имеют особый семантический смысл, когда в задаче классификации мы трактуем классы как абстрактные группы.

В настоящее время часть задач, связанных с анализом данных [7] и рекомендательными системами решается, используя инструменты *машинного обучения* [8] и *нейронных сетей* [9]. В частности, для задач обработки естественного языка (NLP — англ. *Natural Language Processing*) было создано большое количество моделей представлений слов в векторном виде, некоторые из которых — **BERT** [10] и **word2vec** [11] — будут использованы в данной работе.

Сформировав из распределений объектов по меткам предложения, трактуя их как документы, становится возможным обратиться к области тематического моделирования для категоризации полученных документов. Для этого используется один из методов определения тематик в документе — **BERTopic** [12].

В работе предлагаются три модели для автоматической категоризации постов. Все расположены в порядке улучшения качества предсказаний: модель, использующая для представления слов в векторном пространстве нейросеть **BERT**; модель, использующая готовую библиотеку **BERTopic**; и, наконец, модель, использующая для представления слов нейросеть **word2vec**.

Постановка задачи

Дано:

- Посты пользователей, которые могут содержать один из двух объектов — либо изображение, либо видео.
- Предзаданное множество категорий
 $C = [\text{мода, фитнес, гармония, видеоигры, красота, животные, юмор, искусство, путешествия, музыка, танцы, спорт, окружающая среда, техника, еда}]$, обозначающие общие и наиболее известные темы. Некоторые из данных категорий пользователь выбирает при регистрации на ресурсе. Таким образом мы явно узнаём то, что может быть интересно человеку.
- С помощью сервиса Image Annotation Google API¹ произведено аннотирование изображений и видео. Таким образом по каждому объекту мы имеем словари, ключами которого являются уникальные идентификаторы объекта (ID), а значениями — пары меток и «уверенностей» модели. Обозначим это множество как L .
К примеру, для изображения, на котором показана играющая с мячом собака, метки могут быть: «собака», «мяч», «свежий воздух», «трава».
- Количество уникальных меток составляет **4500**; количество объектов или постов — **36214**. Категорий, на которые нужно разбить объекты, **15** штук.

Требуется каждой категории из множества C поставить в соответствие наиболее подходящие объекты из множества L .

¹<https://cloud.google.com/vision/docs/reference/rest/v1/images/annotate>

Пример исходных данных для изображения и ожидаемых результатов



Рис. 1: Пример объекта.

```
('images/c_f_51bcd1a20cf23bf9736ecd58db607b105982e2.jpeg',  
[{'description': 'Pool player', 'score': 0.987},  
 {'description': 'Indoor games and sports', 'score': 0.9811},  
 {'description': 'Cue stick', 'score': 0.9795},  
 {'description': 'Pool', 'score': 0.9768},  
 {'description': 'Recreation room', 'score': 0.9736},  
 {'description': 'Recreation', 'score': 0.972},  
 {'description': 'Billiard room', 'score': 0.9657},  
 {'description': 'Shoulder', 'score': 0.9656},  
 {'description': 'Textile', 'score': 0.9583},  
 {'description': 'Joint', 'score': 0.954}])
```

Рис. 2: Список меток и уверенностей модели аннотации.

Ожидаемый результат для данного примера: «мода», «фитнесс».

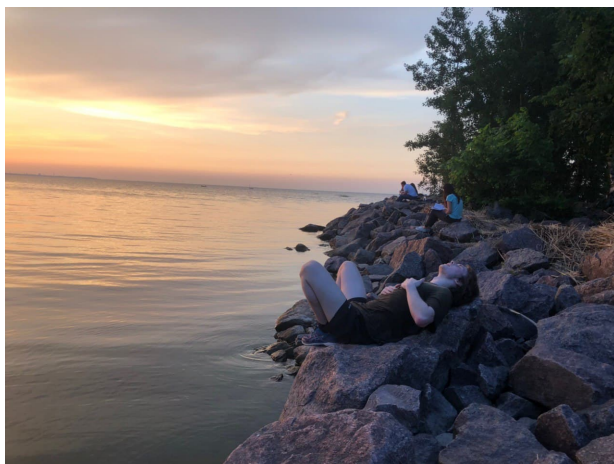


Рис. 3: Пример объекта, которому нужно сопоставить категорию.

```
('images/d_a_fsbs31af20cf2sdfdfgd58dfsb1yt12sz.jpeg',  
[{'description': 'Lying man', 'score': 0.993},  
 {'description': 'Rocks', 'score': 0.9913},  
 {'description': 'Coastline', 'score': 0.9783},  
 {'description': 'Sky', 'score': 0.9754},  
 {'description': 'Sunset', 'score': 0.9752},  
 {'description': 'Sea', 'score': 0.996},  
 {'description': 'Shirt', 'score': 0.9512},  
 {'description': 'Textile', 'score': 0.9501},  
 {'description': 'Pair of men', 'score': 0.9499},  
 {'description': 'Trees', 'score': 0.949}])
```

Рис. 4: Распределение по меткам.

Ожидаемый результат для данного примера: «гармония», «путешествия».

Обзор литературы

Задача рекомендации пользовательских публикаций очень важна во время развития социальных сетей, т.к. социальные сети охватывают всё больше пользователей, а значит, для нужд бизнеса (рекламы) необходимо разработать метод, которым мы сможем заинтересовать пользователя, на большее время задержав его на портале [13]. Это приводит к извлечению большей выгоды, из чего следует важность решения данной задачи.

В данной работе мы используем облегченную в вычислительном плане версию BERT — DistilBERT, предложенную в [14]. Сделанная для обработки последовательностей слов, модель BERT была произведена для осуществления машинного перевода[10]. В будущем показав себя отличным образом на задачах ответов на вопросы [15], классификации [16], создания чат-ботов[17], анализ поисковых запросов[18] и др. [19]. Хорошее качество результатов основано на том, что модель способна лучше понимать контекст слов по сравнению с существующими моделями за счёт обучения контекстно-зависимых представлений [20]. Это является продолжением развитием модели word2vec. Модель word2vec строит лишь единственное векторное представление для объекта, даже если на самом деле их несколько.

В данной работе BERT используется для получения представлений слов в векторном виде для последующей работы с ними.

В результате работы алгоритма BERT получаются векторы размерности 768. Работа в высоких размерностях трудна [21], поэтому размерность стоит понизить. Для этого был выбран метод UMAP, предложенный в [22]. Метод производит построение взвешенного графа ближайших соседей на основании метрики Римана и использования теоремы о Нерве покрытия. Далее алгоритм создаёт граф в пространстве требуемой размерности, которая меньше исходной, и, минимизируя сумму дивергенций Кульбака-Лейблера с помощью стохастического градиентного спуска, приближает исходный граф. Данный метод хорошо показал себя, например, в анализе данных из биологии [23].

Для того, чтобы отобрать похожие объекты, в данной работе используется кластеризация. Для кластеризации был выбран широко известный метод k -средних, описанный в [24]. Оптимальное количество кластеров выбирается по методу локтя [25] и методу среднего силуэта [26, 27]. Сравнение объектов в векторном пространстве производится на основании косинусной близости [44].

Можно объединить распределения по меткам в предложения простой конкатенацией строк. Получив таким образом примитивные документы из одного предложения, можно обратиться к области тематического моделирования [28, 29]. Вторая модель использует технику BERTopic, которая находит тематики в документах. В основе подхода BERTopic используются: нейронная сеть BERT для представления слов в векторном пространстве, алгоритм кластеризации HDBSCAN [30], а также модификацию TF-IDF [31] для классов: с TF-IDF [32]. TF-IDF используется для нахождения важности слов в тексте для последующего сравнения документов и классификации [33]. Сравнение объектов в векторном пространстве производится на основании косинусной близости.

Можно видеть, что автором BERTopic используется структура, схожая с той, которую мы используем в первом методе, но с некоторыми отличиями: различаются алгоритмы кластеризации, а также мы не взвешиваем слова мерой TF-IDF. На практике это приводит к тому, что алгоритм BERTopic показывает лучшую работу, хоть количество предсказанных постов и остаётся малым.

В третьей модели используется вышеупомянутый метод word2vec. Мы используем предобученные векторы lexvec, которые показали хорошее качество работы во многих задачах NLP [34, 35]. Метод word2vec основан на обучении «мешка слов» [36]. Данная модель не учитывает всех связей слов и контекста. Поскольку предложения в нашем случае создаются искусственно — конкатенацией строк, то имеет смысл воспользоваться именно этим методом для

улучшения качества. Также каждое слово для нахождения наиболее весомых взвешивается с помощью меры TF-IDF.

Для оценки качеств моделей будут использоваться метрики полноты, точности и F-мера [37, 38, 39]. Для визуализации и лучшего понимания различий в предсказаниях и эталонных значениях в работе используется матрица ошибок [40].

Задача автоматической категоризации важна и довольно часто возникает [5, 6, 41, 42]. Авторы статьи [5] решают задачу категоризации текстов посредством обучения специальной модели на уже размеченных данных. Справляются они со своей задачей подобно задаче классификации, показывая качество работы категоризацией медицинских статей из журналов. В статье [6] производится категоризация медицинских снимков и изображений посредством обучения собственной модели. Данный подход также похож на решение задачи классификации, поскольку используются размеченные данные для обучения модели. В реальной жизни производить разметку вручную затруднительно, поэтому подходы использующие разметку нуждаются в доработке. В нашей работе мы пытаемся уйти от этой проблемы и не использовать предварительно размеченные данные. В работах [41, 42] используется модель Латентного размещения Дирихле (LDA), которая способна выявить схожие объекты с помощью «неявных» групп. В первой работе авторы используют модель LDA для сообщений об ошибках, т.к. часто неправильная категоризация сообщений об ошибках в работе программ задерживает исправление самих ошибок. Во второй работе авторы используют свою модель на основе LDA для категоризации исходного кода программ. Тем не менее, модель LDA неустойчива к шуму в исходных данных, что отрицательно сказывается на полученных результатах.

Глава 1. Модель BERT

Алгоритм решения задачи

Распределения и категории представлены в *строковом* формате — для выполнения преобразований над словами нужно представление в виде вектора — набора действительных чисел. Далее необходимо позить размерность и кластеризовать объекты. Сопоставив векторам категорий векторы объектов, мы сможем сделать оценку качества.

Алгоритмы кластеризации плохо работают с векторами большой размерности в связи с т.н. «проклятием размерности» [21]. Таким образом, есть смысл понизить размерность, выделяя главные компоненты. Сделать это можно либо с помощью техники PCA [43], выполняющей линейное снижение размерности, либо UMAP[22], выполняющей нелинейное снижение размерности. В нашем случае был выбран метод UMAP, поскольку он сохраняет структуру графа в пространстве меньшей размерности.

Далее нужно кластеризовать полученные векторы для постов в группы до дальнейшего сопоставления множеству заданных категорий.

Здесь используется широко известный алгоритм k -средних, основанный на идее ЕМ-алгоритма и минимизирующий критерий т.н. *инерции* [24]. В качестве меры близости двух векторов осмысленно использовать косинус углов между векторами — основную метрику, используемую в NLP и для кластеризации, и для оценки близости [44]. Поскольку алгоритм k -средних работает в *евклидовой метрике*, нужно немного доработать полученные векторы. Доработка производится нормализацией — приведением длины векторов к единичной.

Таким образом, применяется следующий подход:

1. Представление слов в векторном виде (**BERT**).

2. Понижение размерности для векторов (**UMAP**).
3. Кластеризация слов в векторном пространстве (**KMeans**).
4. Сопоставление каждой категории изображений (**Cosine Similarity**).

Считывание данных

Данные были предоставлены в формате **.csv** — англ. *comma separated value*. Для работы с данными их нужно прочитать и преобразовать в удобные для программирования и обработки форматы — списки, словари.

В нашей работе данные считываются с помощью модуля `json`.

Обработка данных

Первым шагом необходимо произвести предобработку данных, чтобы избежать ошибок при считывании. Данные были **.csv**-таблиц были считаны, очищены от пустых значений, приведены к удобному формату в нижнем регистре. Мы не учитываем имён собственных для улучшения качества.

На рисунках 5,6 показана обработка данных.

```
{
  'description': 'Human',
  'score': 0.9823172092437744,
  'topicality': 0.9823172092437744},
{
  'mid': '/m/01bsxb',
  'description': 'Collage',
  'score': 0.9496289491653442,
  'topicality': 0.9496289491653442},
{
  'mid': '/m/02l215',
  'description': 'Reflection',
  'score': 0.9309725761413574,
  'topicality': 0.9309725761413574},
{
  'mid': '/g/1tr17zw8',
  'description': 'Flash photography',
  'score': 0.8121163845062256,
  'topicality': 0.8121163845062256},
{
  'mid': '/m/0b0fq',
  'description': 'Photomontage'
```

Рис. 5: Начальный формат данных.

```
'vehicle door',
'automotive mirror',
'automotive tire',
'landscape',
'white',
'fender'],
['monochrome',
'monochrome photography',
'atmospheric phenomenon',
'terrestrial animal',
'black',
'snout',
'black-and-white',
'grey',
'wildlife',
'working animal'],
['leg',
'human leg',
```

Рис. 6: Список меток.

Реализация в языке Python

Для обработки данных использовались встроенные функции языка Python, предназначенные для чтения и записи файлов. Для хранения строковых переменных используется структура данных, именуемая списком. Для удобного и быстрого считывания файлов в нужном формате используется библиотека `pickle`.

Преобразование распределений меток в предложения

Поскольку модель **BERT** была разработана для работы с последовательностями, то для решения нашей задачи нужно получить предложения. Мы не будем делать их осмысленными, а склеим для каждого поста его метки в одну строку. Без использования предложений модель будет работать плохо и показывать низкое качество предсказаний.

В итоге были получены **36214** предложений.

Получение представлений объектов с помощью BERT

Модель **BERT** основана на архитектуре Трансформер, которая сама по себе является логическим развитием рекуррентных нейронных сетей [45].

Архитектура Трансформер использует модель *внимания* [45], схожую по механизму на работу одноимённого когнитивного процесса. При переводе слова с одного языка на другой приходится несколько раз рекурсивно возвращаться к изначальному предложению для лучшего понимания *контекста*. Схожим образом и работает модель *внимания* в нейронных сетях — она «подсказывает» наиболее связанное слово с тем, которое просматривается в данный момент.

Дообучение модели не будет производиться. Возьмём только представле-

ния слов в векторном пространстве для дальнейшего использования.

Реализация в языке Python

В первой части используется облегчённую модель **BERT — DistilBERT** [14]. Она предобучена на англоязычной Википедии и датасете **BookCorpus**. Загружается она из библиотеки **transformers** от компании **HugginsFace** по названию **distilbert-base-uncased**.

Для *токенизации* наших слов/словосочетаний будем использовать функцию **tokenizer** из **BertTokenizer**.

В нейронной сети 12 слоёв. Каждый из этих слоёв содержит матрицу веса размером $768 \times n$, где n — количество предложений. Матрица весов и является наборами представлений слов — остаётся выбрать какой-то один из этих слоёв. Поскольку исходный датасет (данные англоязычного сайта Wikipedia² и датасет **BookCorpus**) не связаны напрямую с нашими данными, то есть смысл использовать один из средних слоёв. Модели имеют свойство «переобучаться» под текущий датасет. Из этих соображений возьмём набор весов с 8 слоя.

Загрузка данных на видеокарту

Большие размерности данных (768×36214) занимают много памяти. Для ускорения работы нейросети и использования дополнительной памяти, данные загружались на видеокарту.

Это стало возможным благодаря сайту Kaggle³. На данном ресурсе существует возможность использования ОЗУ объёмом 13 Гб и видеокарты **Tesla P100** с памятью объёмом 16 Гб.

²<https://wikipedia.com>

³<https://kaggle.com>

Данные при загрузке были разделены на маленькие подгруппы, называемые *батчами* и перемещены для вычислений на видеокарту. Это было осуществлено с помощью специализированного ПО от компании **NVIDIA** — **CUDA**⁴.

Понижение размерности векторов с помощью UMAP

Чтобы добиться сравнения векторов категорий с векторами меток, нужно, чтобы векторы имели одинаковую размерность. При снижении числа компонент для малого количества векторов (в нашем случае категорий 15 штук) пропадают исходные зависимости, а именно: косинусы углов изменяются и хорошего качества добиться трудно.

Это проиллюстрировано на рисунке 7:

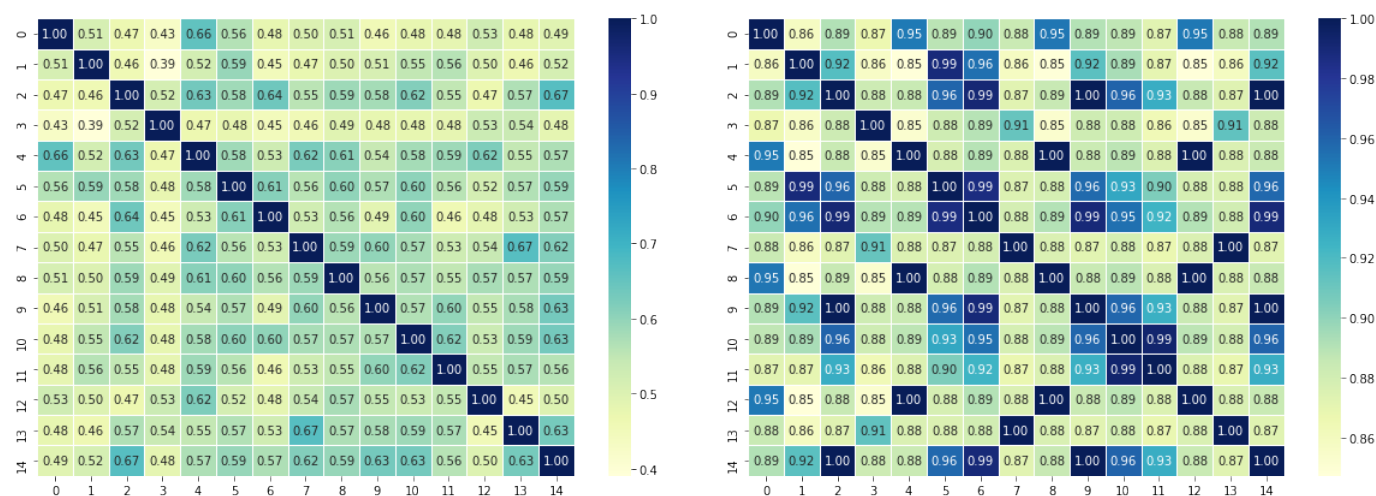


Рис. 7: Матрицы расстояний между векторами категорий для исходной пониженной размерности. Многие зависимости не сохранились.

Таким образом, имеет смысл понижать размерность только для меток объектов, кластеризовать их, и далее вернуться в исходное пространство для нахождения центроидов.

⁴<https://developer.nvidia.com/>

Реализация в языке Python

Для реализации используем библиотеку **umap**.

В данной методике есть несколько параметров, которые нужно подбирать под каждый набор данных индивидуально. Оптимизация по ним влечёт лучшее представление данных в пониженном пространстве. Будем подбирать следующие параметры:

- **min_dist** — минимальное расстояние для выбора соседней вершины.
- **n_neighbors** — максимальное число соседей, которое войдёт в компоненту сильной связности.

Понижать размерность будем до числа **n_components** = 150.

Для подбора параметров будем использовать понижение размерности до двух, выводить на график распределение наших данных и смотреть, в каком случае лучше выделяются кластеры из данных. На рисунках 8 и 9 предоставлены графики подбора оптимального числа ближайших соседей в графе и минимального расстояния для формирования графа. Осмысленно взять число ближайших соседей равное 50 и минимальное расстояние равным 0.1ю

Кластеризация с помощью метода k -средних

Для того, чтобы кластеризация была осмысленна, нам требуется использовать метрику *косинусной близости* [44]. Метод k -средних соседей использует евклидову метрику, но мы можем изменить наши данные так, чтобы расстояния по евклидовой метрике имела характер, одинаковый с косинусной близостью.

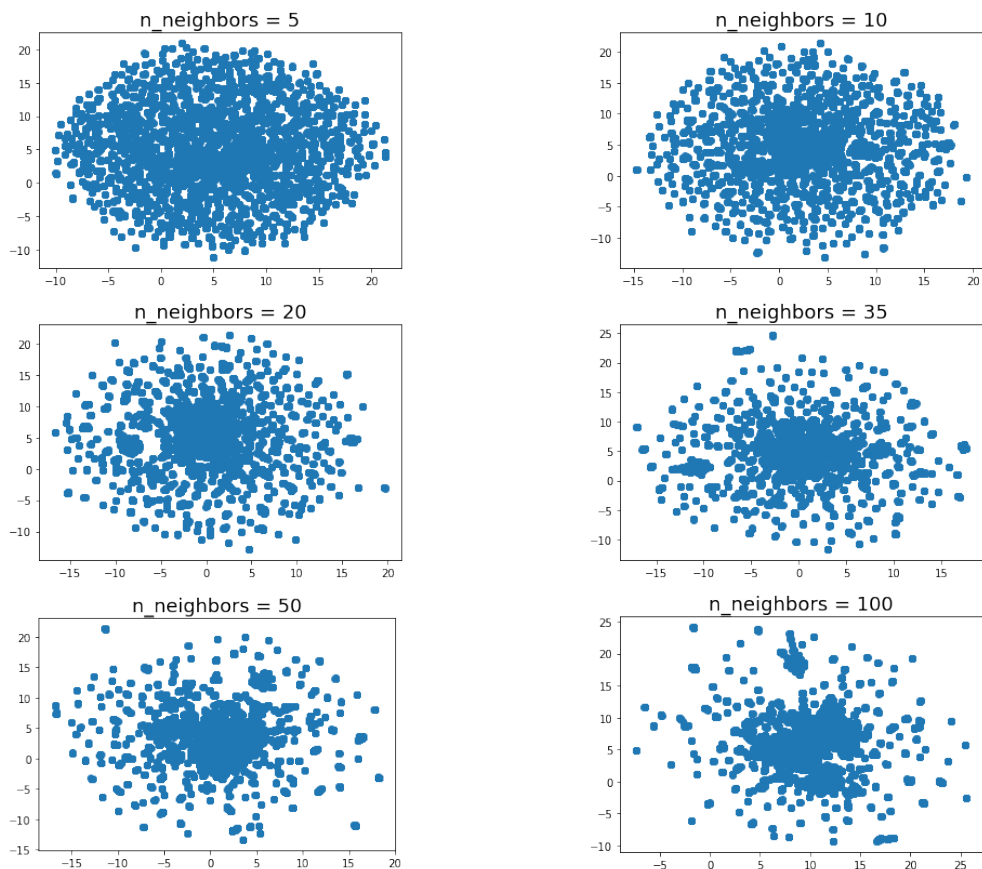


Рис. 8: Подбор оптимального количества ближайших соседей, $n_neighbors = 5, 10, 20, 35, 50, 100$.

Обоснованность использования нормализации векторов

По теореме косинусов для двух векторов x, y в евклидовом пространстве косинусная близость — это

$$\cos(x, y) = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2 \sum_i y_i^2}}. \quad (1)$$

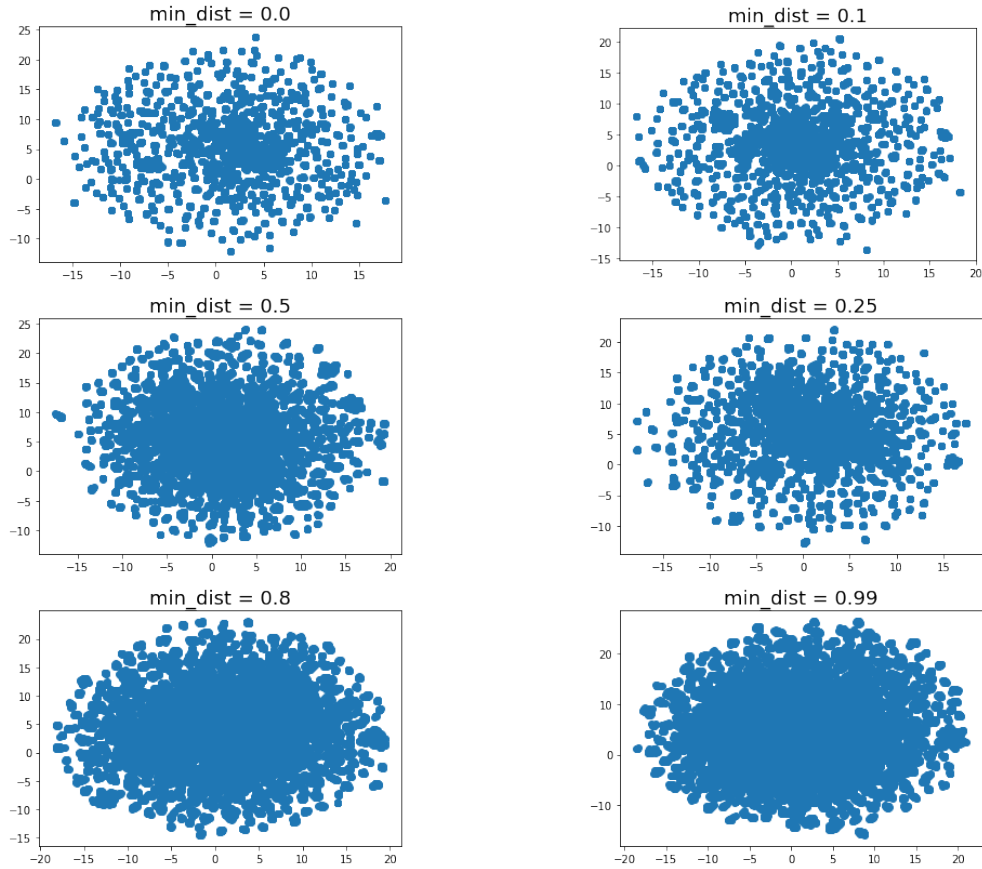


Рис. 9: Подбор минимального расстояния, $\text{min_dist} = 0.0, 0.1, 0.25, 0.5, 0.8, 0.99$

Для нормированных векторов x, y получим $\cos(x, y) = \frac{\sum_i x_i y_i}{1} = \sum_i x_i y_i$.

Для квадрата евклидова расстояния:

$$\begin{aligned}
 \|x - y\|^2 &= \sum_i (x_i - y_i)^2 \\
 &= \sum_i x_i^2 + \sum_i y_i^2 - 2 \sum_i x_i y_i \\
 &= 2(1 - \cos(x, y))
 \end{aligned}$$

Таким образом, меньшие расстояния в евклидовой метрике будут означать большие косинусы углов между векторами — то, что нам нужно.

Реализация в языке Python

Для нормировки векторов будем пользоваться функцией `Normalizer()` из библиотеки `sklearn.preprocessing`. Данная функция производит стандартную нормировку делением вектора покомпонентно на его длину.

Для кластеризации мы будем использовать алгоритм k -средних. Техника работает в 5 шагов, основываясь на идее ЕМ-алгоритма.

Оценка качества кластеризации с помощью метрики «Силуэт»

Для того, чтобы оценивать качество кластеризации используется метрика «среднего Силуэта» [26].

Значения метрики находятся в интервале $[-1, 1]$ и интерпретировать её можно следующим образом: чем ближе число к -1 , тем хуже получилось разбиение по кластерам. Чем ближе значение силуэта к 0 , тем больше кластеры накладываются друг на друга и тем труднее различимы. Наконец, чем это значение ближе к 1 , тем кластеры более разделимы и сгруппированы.

Метрика показывает, насколько ближе находится точка к своему кластеру по отношению к другим.

Метод локтя и метод среднего силуэта

1. **Метод локтя** — выбираем оптимальное число кластеров по резкому изгибу кривой на графике.
2. **Метод среднего силуэта** — усреднённое по всем точкам исходных данных значение метрики «Силуэт».

В методе k -средних количество кластеров, на которое нужно разделить множество объектов, является параметром. Поскольку изначально мы не знаем оптимальное число кластеров, требуется способ для его определения.

На рисунках 10 и 11 представлены сравнительные графики для двух вышеописанных методов. Графики разделены на два для лучшей читабельности.

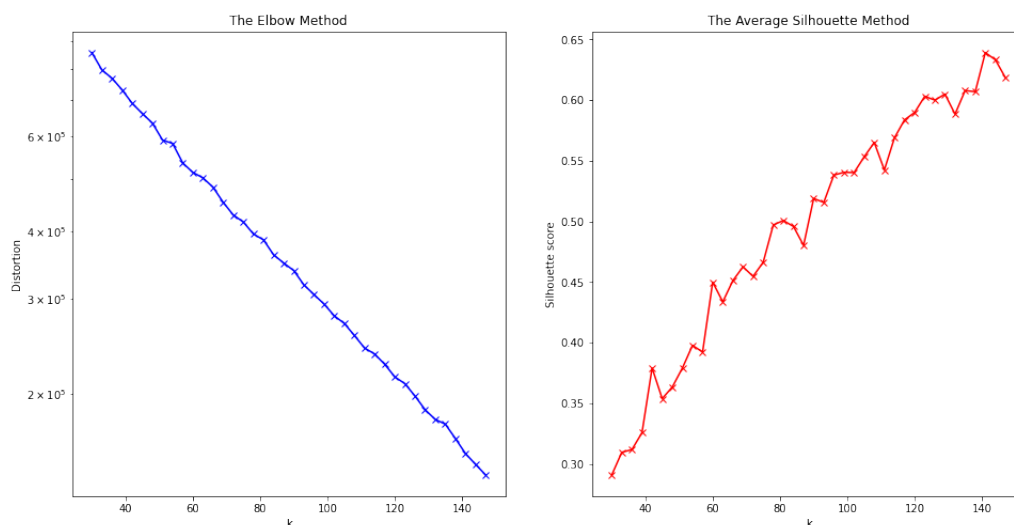


Рис. 10: Сравнительные графики для кластеров $k = 30, 33, \dots, 150$. По методу локтя неясно, какое количество кластеров лучше взять. Метод Силуэта возрастающую ломаную.

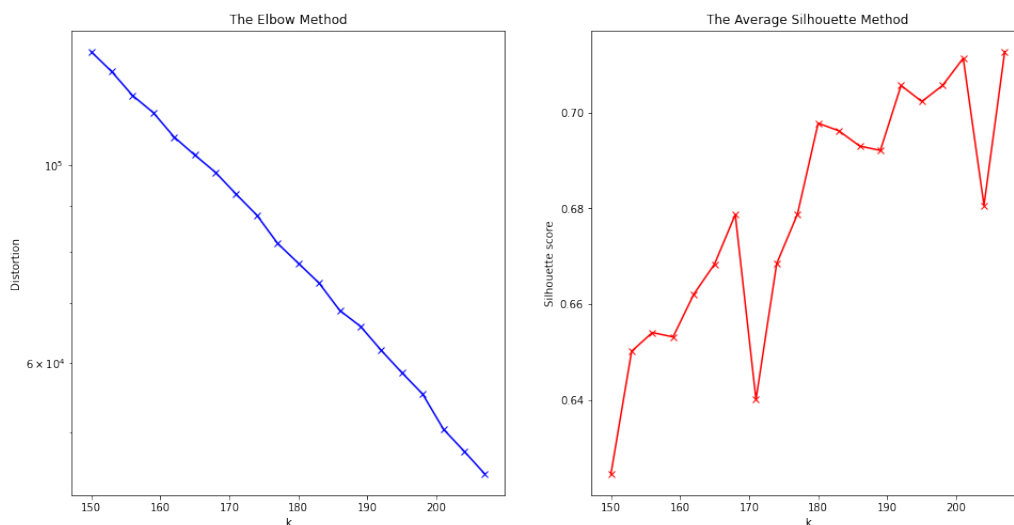


Рис. 11: Сравнительные графики для кластеров $k = 150, 153, \dots, 210$. По методу локтя снова неясно, какое количество кластеров лучше взять. Метод Силуэта показывает возрастающую ломаную.

По данным графикам можем видеть, что метод локтя несостоятелен — нет резкого «изгиба» на графике. По методу среднего силуэта нельзя однозначно

определить оптимальное число кластеров — ломаная на графике возрастает. Дальнейшие вычисления затруднительны с вычислительной точки зрения и занимают много времени. Остановимся и возьмём для оптимального значения кластеров число, равное **200**. Значение метрики «силуэт» на нём равно 0.714.

Реализация в языке Python

Поскольку в методе KMeans используется большое количество вычислений, связанное с расчётом расстоянием между каждой из точек исходного множества, следует производить расчёты на графическом ядре.

Для использования метода k -средних используем функцию **KMeansTF()** из библиотеки **KMeansTF**. Эта функция работает на интерфейсе другой библиотеки — **TensorFlow**, которая, в свою очередь, использует большой набор инструментов для параллелизации процессов на графических ядрах.

Для сравнения качества кластеризации используем функцию с помощью метрики среднего силуэта **silhouette_score** из библиотеки **sklearn.metrics**. У функции два аргумента: исходные кластеризуемые объекты и полученное распределение множества объектов по кластерам.

Сопоставление категориям объектов

Для сопоставления мы будем использовать **косинусную близость** (1). В нашем случае нормировки конечная формула принимает вид $\cos(x, y) = \sum_i^n x_i y_i$.

Сопоставление производится нахождением косинусов углов между векторами предзаданных категорий и векторами полученных при кластеризации центроидов.

Разметка данных

Для того, чтобы иметь эталонные значения на небольшом наборе данных, было вручную размечено **5000** постов.

Оценка качества модели

Для оценки качества будем использовать три метрики — F-меру, точность и полноту [37].

Также будем смотреть на матрицу ошибок [40]— это матрица различий между эталонными (размеченными) и полученными моделью значениями.

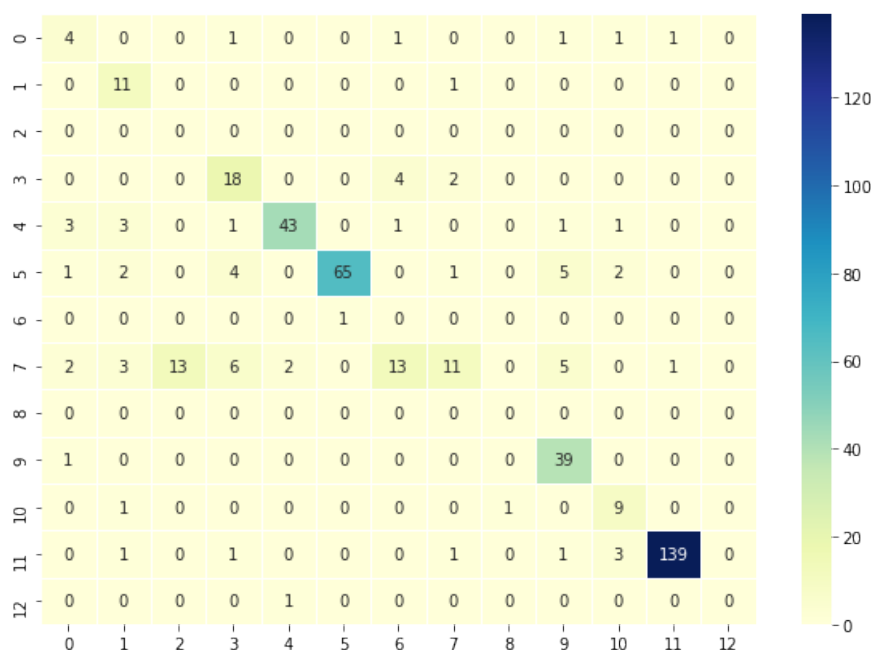


Рис. 12: Матрица ошибок для вектора предсказаний. У модели довольно хорошие показатели F-меры, но количество предсказаний скудно.

На рис. 12 показано отличие предсказанных категорий от размеченных. Можно видеть, что общее количество объектов из размеченного вручную датасета от размеченных моделью постов, приблизительно равно **300**. Для этого достаточно сложить все числа матрицы. Данное явление произошло из-за

того, что некоторые посты не попали в 15 данных категорий, данные разбивались на 200 кластеров. И, таким образом, посты были ошибочно отнесены моделью к другим категориям.

Значение F -меры равно 0.72 на наших данных — довольно высокий результат. Но из-за малого количества рекомендаций (всего 3000 из 36000) признать эту модель годной нельзя.

Глава 2. Модель BERTopic

В данном разделе мы используем модель, использующая нейросеть для представления слов в векторном виде — BERT, алгоритм понижения размерности UMAP, алгоритм кластеризации HDBSCAN и меру TF-IDF для классов.

В этой части, как и в первой, формируются предложения из распределений по меткам. Далее будем трактовать предложения как документы и оценим качество модели, используя F-меру, полноту и точность.

Мера TF-IDF

Основной принцип данной меры [31] — дать наибольший вес часто употребляющемуся уникальному слову. Тогда некоторые часто употребляющиеся слова как, например, союзы или местоимения, будут иметь низкие веса. Далее можно удобно и эффективно отличать документы друг от друга с помощью полученных весов.

Мера c-TF-IDF

Для того, чтобы отделить наборы документов друг от друга, используется мера TF-IDF для классов [32]. Отличие от обычной меры TF-IDF в том, что теперь мы объединяем документы, принадлежащие одному кластеру, в один. А далее к полученным документам применяем меру TF-IDF.

Количество тематик

Для данной модели количество тематик является гиперпараметром. Обратимся к первой части и возьмём количество тем для обучения, полученным с помощью метода кластеризации меток объектов (200).

Удаление выбросов

Алгоритм кластеризации HDBSCAN удаляет часть данных, которые не может кластеризовать, и помечает их выбросами. В нашем случае это есть примерно 1/3 от всего количества:

	Topic	Count
0	-1	11582
1	121	659
2	139	406
3	202	405
4	50	363

Рис. 13: Топ-5 тематик по количеству документов в них.
Меткой «-1» обозначается тематика для выбросов.

Визуализация кластеров тематик

С помощью встроенной функции `visualize_topics()` мы можем получить удобную визуализацию распределения тематик. Для представления графика используется библиотека **plotly**. Сам график интерактивен — можно сдвигать ползунок для нахождения определённой темы. Кластер, изображённый в виде диска, для каждой отдельной темы выделяется красным цветом. Также возможно навести курсор на один из дисков и получить слова, наиболее встречающиеся в данной теме.

Далее предоставлен на рис. 14 представлен скриншот распределения по тематикам, которое является интерактивным. Для использования интерактивного графика нужно перейти по ссылке⁵

Порог на значения близости

Для подсчёта тематик, близких к исходным категориям используется внутренняя функция `find_topics`. В качестве аргументов она принимает два значения: слово или словосочетания, похожую тему к которому нужно найти, и

⁵<https://www.kaggle.com/taciturno/working-bertopic?scriptVersionId=63943338&cellId=10>

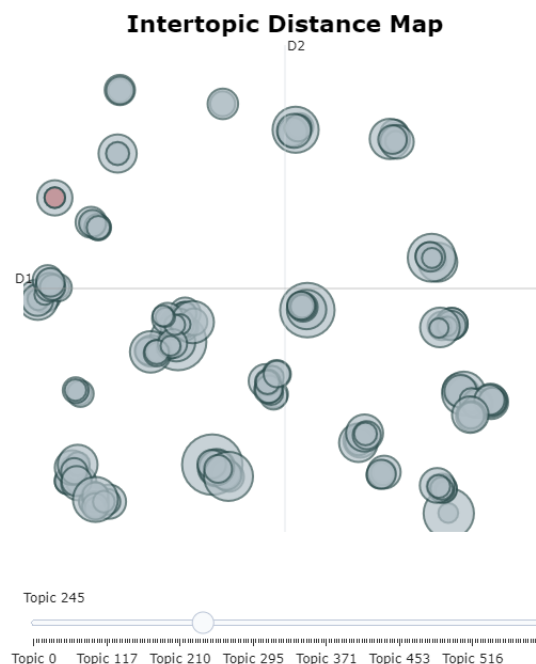


Рис. 14: Статическая версия распределения по тематикам.

число похожих тематик.

Функция возвращает два объекта. Первый — список похожих тематик. Второй — значения «похожести». Выберем число похожих тематик равным **15**.

Для того, чтобы выяснить оптимальный порог «доверия» нашей модели, мы можем поставить порог на значения похожести и оптимизировать его по значениям метрик. Рассмотрим пороги от 0 до 1 с шагом 0.1.

Оценка качества модели

Качество будет оцениваться с помощью матрицы ошибок — это матрица различий между эталонными (размеченными) и полученными моделью значениями.

На рис. 15 предоставлена матрица ошибок для вектора эталонных значений и вектора предсказанных моделью категорий.

Значения метрик полноты, точности для разных порогов указаны на ри-

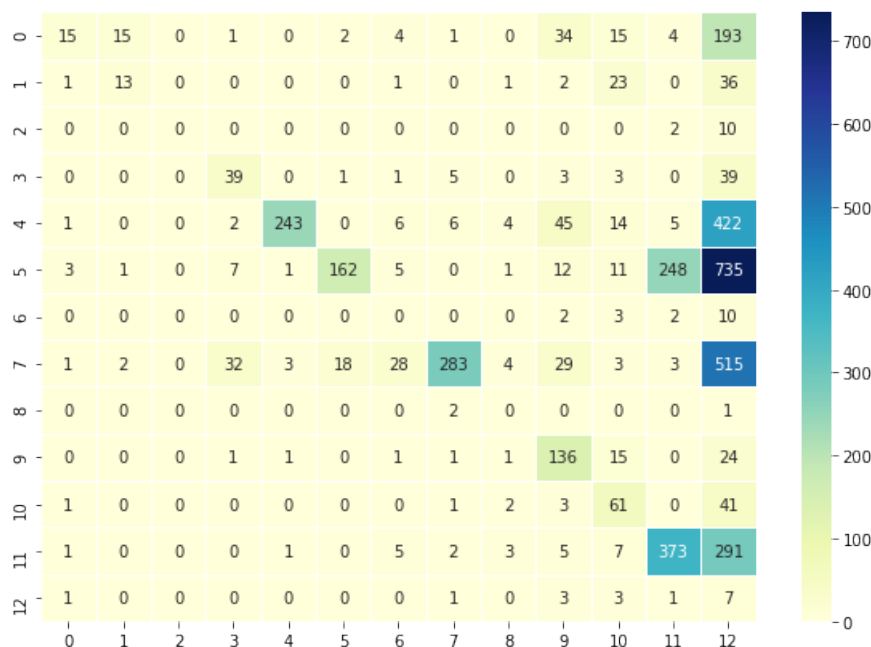


Рис. 15: Матрица ошибок для вектора предсказаний. Уже визуально можно сказать, что модель точна, но скудна на количество предсказаний.

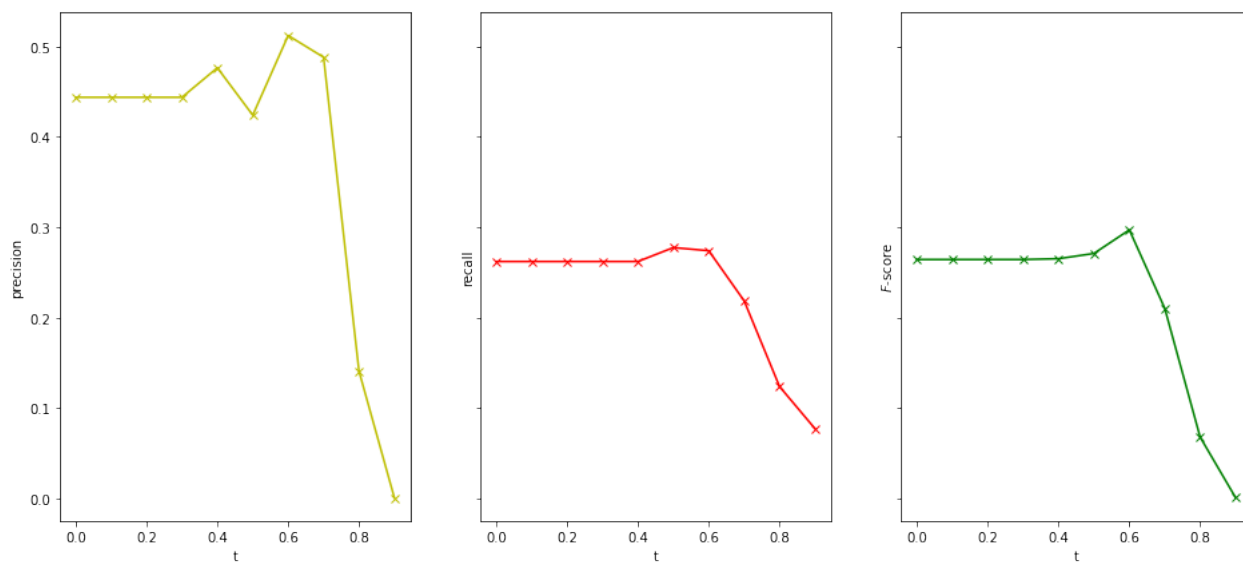


Рис. 16: Значения метрик точности, полноты и F-меры для разных значений порога $t = 0, 0.1, \dots, 0.9$.

сунке 16.

По рисунку видно, что оптимальное значения «доверия» нашей модели равно 0.5. Также видно, что у модели точность выше, чем полнота — это

объясняется тем, что во время кластеризации треть исходных данных были помечены выбросами и не категоризованы.

Качество данной модели превосходит качество первой. Тем не менее, для обучения модели мы также использовали искусственные предложения. Поскольку они не имеют смысла, предлагается использовать аннотирование для улучшения качества. Постараемся улучшить качество предсказаний, применив другой подход.

Глава 3. Модель word2vec

Для третьей модели мы используем нейросеть word2vec и статистическую меру важности слов TF-IDF, описанную ранее.

Основной алгоритм

В данном методе существуют два принципиальных момента: 1). Обучение модели TF-IDF для выяснения весов каждого слова. 2). Нахождение представления объектов в векторном виде.

Как и прежде, будем создавать предложения из распределений меток, но уже лишь для обучения модели TF-IDF. Трактуя каждое предложение как единый документ, найдём таким образом вес каждого слова на этом объединении документов. В распределениях объектов на метки довольно часто в качестве одной метки встречаются *словосочетания*, состоящие из нескольких слов (максимально до 3). Тогда осмысленно будет обучать модель TF-IDF на **n-граммах** длиной от 1 до 3.

При нахождении мер слов осмысленно не учитывать т.н. *«стоп-слова»* — часто встречающиеся лексемы, которые не дадут конструктивной информации об документе. Например: союз «и», частица «не» и т.п.

С другой стороны, для составления векторов объектов и передаче их модели word2vec мы не будем использовать полученные предложения, а сами векторы объектов будем получать агрегацией векторов слов, входящих в распределение объекта по меткам. Каждое слово при этом будем домножать на его вес в обученной модели TF-IDF.

Реализация в языке Python

Воспользуемся функцией `TfidfVectorizer()` из библиотеки `sklearn.feature_extraction.text` с параметром `n_gram = (1, 3)` для обучения на n-граммах.

Для загрузки стоп-слов используется библиотека `nltk` и функция `stopwords.words()` с параметром `'english'` для обработки слов на латинице.

Для загрузки модели `word2vec` используется функция `KeyedVectors.load_word2vec_format` из библиотеки `gensim.models` с двумя параметрами: первый — путь файла, по которому находятся предобученные векторы; второй — параметр `binary = False` для чтения векторов не в бинарном формате, а в текстовом. В качестве загружаемых векторов используются предобученные векторы `lexvec`.

Разметка для векторов категорий

В данном подходе мы не будем пользоваться методом дополнения каждой категории синонимами или близкими по значению словами, как делали это ранее.

Поскольку изначально для каждой категории нам дано всего лишь одно слово, мы разметим вручную на каждую категорию по 30 объектов, а потом обратимся к меткам объекта и агрегируем векторы меток, используя веса из уже обученной модели TF-IDF.

Порог на значения косинусной близости

До сих пор за ближайший вектор к вектору объекта считается значение с наибольшим значениям косинуса угла между двумя векторами. Однако, значения косинусов не всегда приближаются к единице. Можно поставить границу или *порог* для значения косинусной близости, при достижении или не

достижении которого вектор будет соотнесён к какой-то осмысленной категории или же наоборот, соотнесён к пустой категории.

Найдём **порог** для значений косинусов, по которому предсказывается наибольшее число постов. Таким образом станет известным значение, при котором обобщающая способность нашей модели будет максимальна. В будущем это значение порога и будем использовать для предсказаний модели.

Оценка качества модели

Как и в предыдущих случаях, качество оценивать будем с помощью матрицы ошибок, метрик полноты, точности и F-меры.

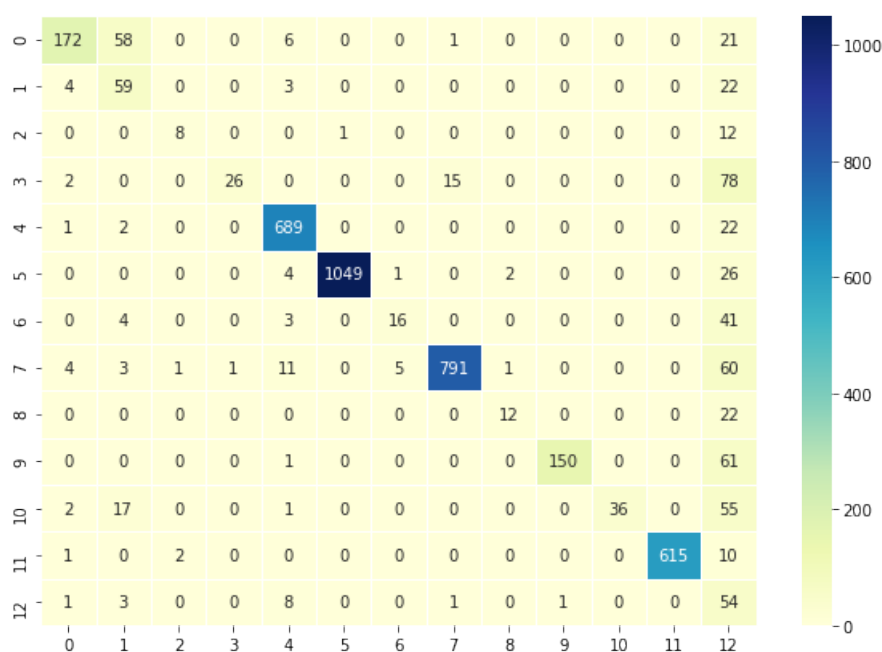


Рис. 17: Матрица ошибок для вектора предсказаний третьей модели. Видно, что количество предсказаний увеличилось при сохранении качества.

На рисунке 17 показана матрица ошибок для предсказаний модели и истинными значениями категорий объектов при пороге на косинусную близость $t = 0.5$. Значение F-меры в данном случае равно 0.644.

Подбор границы для косинусного меры близости

Как и в предыдущем случае будем смотреть на значения метрик точности, полноты и F-меры для порогов $t = 0, 0.1, \dots, 1$:

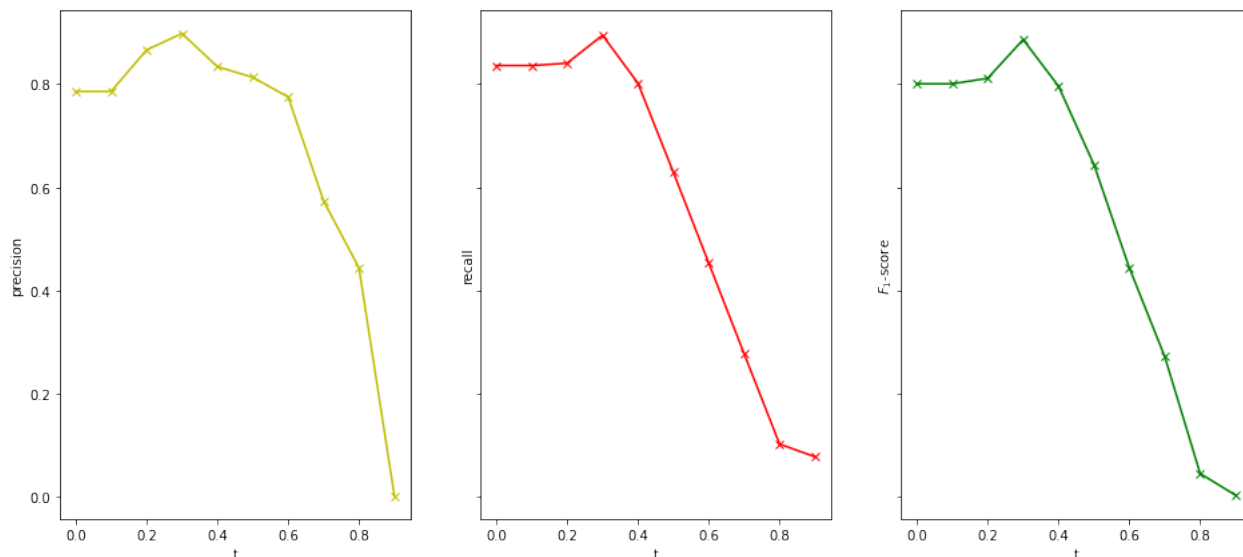


Рис. 18: Подбор оптимального порога. Значения границы равны $t = 0, 0.1, \dots, 1$. Видно, что оптимальное значение достигается при $t = 0.3$

По рисунку 18 видно, что максимальное значение F-меры достигается при значении порога $t = 0.3$ и равно 0.89.

Как и прежде, значение точности всегда больше, чем значение полноты. Это значит то, что модель делает точные предсказания, но теряет некоторые посты, которые принадлежат искомым категориям.

В данной модели достигается оптимальное значение между количеством и качеством категоризации постов. В сравнении со всеми предыдущими подходами можно уверенно заявить, что эта модель работает лучше и именно её и стоит использовать в дальнейшем.

Выводы

Перед нами стояла сложная задача категоризации постов. Для её решения было предложено три модели.

В **главе 1** рассматривалась модель с использованием BERT. Мы применили на практике подходы UMAP для снижения размерности и Kmeans для кластеризации. После нахождения расстояний между векторами категорий и векторами центроидов кластеров мы сделали сопоставление.

Первая модель показала низкое качество. Это получилось из-за специфики построения предложений, которые не были осмысленны. Поэтому модель BERT, построенная для работы с осмысленными предложениями, плохо справилась с задачей. В качестве улучшения качества предлагается следующий подход: обратиться к сфере *автоматической маркировки изображений* [46] и для каждого объекта получить аннотации — осмысленные предложения. К полученным таким способом предложениям и применить уже описанную модель BERT.

В **главе 2** рассматривалась модель с использованием подхода BERTopic, сделанного для нахождения тематик в документах. Сформировав документы искусственным образом — объединением меток — мы получили документы и применили к ним модель.

Внутри себя данная библиотека использует алгоритмы понижения размерности UMAP и кластеризации HDBSCAN. Для сравнения мер близости векторов используется косинус угла между ними.

Вторая модель показала лучшее качество: увеличилось количество постов при хороших показателях рассматриваемых метрик. Тем не менее, предлагается улучшить качество обращением к модели аннотаций для составления осмысленных предложений.

В **главе 3** рассматривалась модель с использованием представления слов `word2vec` и меры TF-IDF. Нейронная сеть, как и в первой главе была взята предобученной. В качестве предобученных векторов были взяты векторы **lexvec**. Модель TF-IDF была обучена на искусственно объединённых меток в предложения.

В третьей модели векторы объектов были получены с помощью агрегации меток самих объектов: бралось векторное представление из модели `word2vec` и домножалось на значение меры TF-IDF. Для получения эффективных векторных представлений категорий, на каждую из них было размечено по 30 картинок. Конечное сопоставление векторов категорий и векторов объектов было осуществлено с помощью косинусной меры близости.

Третья модель имеет сравнительно лучшее качество и количество предсказаний по сравнению с двумя предыдущими вариантами. Было найдено значения порога для значений косинусной близости. Это увеличит обобщающую способность модели.

Заключение

Для решения задачи категоризации при построении моделей были успешно использованы методы машинного обучения и нейронных сетей.

Были предложены три модели, в порядке увеличения качества предсказаний, также были предложены варианты улучшения данных моделей.

Для решения исходной задачи были применены методы естественной обработки языка в нестандартной для данной области: автоматической категоризации постов в социальной сети.

Список литературы

- [1] **Kim, Dae-Hee; Spiller, Lisa; Hettche, Matt.** . *Analyzing media types and content orientations in Facebook for global brands.* //Journal of Research in Interactive Marketing. – 2015 – Т. 9. – С. 4-30.
- [2] **Martino, Francesco ; Spoto, Andrea.** *Social Network Analysis: A brief theoretical review and further perspectives in the study of Information Technology.* //PsychNology Journal. – 2006 – Т. 4. – С. 53-86.
- [3] **Resnick P., Varian H. R.** *Recommender systems.* //Communications of the ACM. – 1997. – Т. 40. – №. 3. – С. 56-58.
- [4] **Burke R.** *Hybrid recommender systems:Survey and experiments.* //User modeling and user-adapted interaction. – 2002. – Т. 12. – №. 4. – С. 331-370.
- [5] **Lam W., Ruiz M., Srinivasan P.** *Automatic text categorization and its application to text retrieval.* //IEEE Transactions on Knowledge and Data engineering. – 1999. – Т. 11. – №. 6. – С. 865-879.
- [6] **Lehmann T. M. et al.** *Automatic categorization of medical images for content-based retrieval and data mining.* //Computerized Medical Imaging and Graphics. – 2005. – Т. 29. – №. 2-3. – С. 143-155.
- [7] **Van Der Aalst W.** *Data science in action.*//Process mining. – Springer, Berlin, Heidelberg, 2016. – С. 3-23.
- [8] **Hastie T., Tibshirani R., Friedman J.** *The elements of statistical learning: data mining, inference, and prediction.* – Springer Science & Business Media, 2009.
- [9] **Bishop C. M. et al.** *Neural networks for pattern recognition.* – Oxford university press, 1995.
- [10] **Devlin J. et al.** *Bert: Pre-training of deep bidirectional transformers for language understanding.* //arXiv preprint arXiv:1810.04805. – 2018.

- [11] **Mikolov T. et al.** *Efficient estimation of word representations in vector space.* //arXiv preprint arXiv:1301.3781. – 2013.
- [12] **Maarten Grootendorst** *BERTopic: Leveraging BERT and c-TF-IDF to create easily interpretable topics.* //Zenodo (2020) <https://doi.org/10.5281/zenodo.4381785>
- [13] **Ricci F., Rokach L., Shapira B.** *Introduction to recommender systems handbook.* //Recommender systems handbook. – Springer, Boston, MA, 2011. – C. 1-35.
- [14] Sanh V. et al. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter //arXiv preprint arXiv:1910.01108. – 2019. – C.5
- [15] **Wang Z. et al.** *Multi-passage bert: A globally normalized bert model for open-domain question answering* //arXiv preprint arXiv:1908.08167. – 2019.
- [16] **Santiago González-Carvajal, Eduardo C. Garrido-Merchán** *Comparing BERT against traditional machine learning text classification.* arXiv, 2020. — 12 cтp.
- [17] **Yoo S. Y., Jeong O. R.** *An Intelligent Chatbot Utilizing BERT Model and Knowledge Graph* //Journal of Society for e-Business Studies. – 2020. – T. 24. – №. 3.
- [18] **Montti, Roger.** *Google’s BERT Rolls Out Worldwide.* //Search Engine Journal. — 2019
- [19] **Annamoradnejad, Issa.** *ColBERT: Using BERT Sentence Embedding for Humor Detection.* arXiv:2004.12765 — 2020.
- [20] **Giulianelli M. et al.** *Under the hood: Using diagnostic classifiers to investigate and improve how language models track agreement information* //arXiv preprint arXiv:1808.08079. – 2018.

- [21] **Donoho D. L. et al.** *High-dimensional data analysis: The curses and blessings of dimensionality* //AMS math challenges lecture. – 2000. – T. 1. – №. 2000. – C. 32.
- [22] **Leland McInnes, John Healy, James Melville** *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction.* //arXiv, 2018. – 64 cтp.
- [23] **Becht, E., McInnes, L., Healy, J. et al.** *Dimensionality reduction for visualizing single-cell data using UMAP.* //Nat Biotechnol 37, 38–44 (2019). <https://doi.org/10.1038/nbt.4314>
- [24] **Lloyd, Stuart P.** *Least squares quantization in PCM.* //IEEE Transactions on Information Theory. 28 (2): 129–137 (1982)
- [25] **Bholowalia P., Kumar A.** *EBK-means: A clustering technique based on elbow method and k-means in WSN* //International Journal of Computer Applications. – 2014. – T. 105. – №. 9.
- [26] **Rousseeuw P. J.** *Silhouettes: a graphical aid to the interpretation and validation of cluster analysis* //Journal of computational and applied mathematics. – 1987. – T. 20. – C. 53-65.
- [27] **Kodinariya T. M., Makwana P. R.** *Review on determining number of Cluster in K-Means Clustering* //International Journal. – 2013. – T. 1. – №. 6. – C. 90-95.
- [28] **Jacobi C., Van Atteveldt W., Welbers K.** *Quantitative analysis of large amounts of journalistic texts using topic modelling* //Digital Journalism. – 2016. – T. 4. – №. 1. – C. 89-106.
- [29] **Rehurek R., Sojka P.** *Software framework for topic modelling with large corpora* //In Proceedings of the LREC 2010 workshop on new challenges for NLP frameworks. – 2010.

- [30] **Birant D., Kut A.** *ST-DBSCAN: An algorithm for clustering spatial-temporal data* //Data and knowledge engineering. – 2007. – T. 60. – №. 1. – C. 208-221.
- [31] **Juan Ramos J. et al.** *Using tf-idf to determine word relevance in document queries* //Proceedings of the first instructional conference on machine learning. – 2003. – T. 242. – №. 1. – C. 29-48.
- [32] **Özgür A., Özgür L., Güngör T.** *Text categorization with class-based and corpus-based keyword selection* //International Symposium on Computer and Information Sciences. – Springer, Berlin, Heidelberg, 2005. – C. 606-615.
- [33] **Aizawa A.** *An information-theoretic perspective of tf-idf measures* //Information Processing and Management. – 2003. – T. 39. – №. 1. – C. 45-65.
- [34] **Salle A., Idiart M., Villavicencio A.** *Matrix factorization using window sampling and negative sampling for improved word representations* //arXiv preprint arXiv:1606.00819. – 2016.
- [35] **Alexandre Salle and Marco Idiart and Aline Villavicencio.** *Enhancing the LexVec Distributed Word Representation Model Using Positional Contexts and External Memory.* //arXiv:1606.01283. – 2016
- [36] **Wallach H. M.** *Topic modeling: beyond bag-of-words* //Proceedings of the 23rd international conference on Machine learning. – C. 977-984. – 2006.
- [37] **Powers D. M. W.** *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation* //arXiv preprint arXiv:2010.16061. – 2020.
- [38] **Van Rijsbergen, C. J.** *Information Retrieval.* (2nd ed.). //Butterworth-Heinemann, 1979.
- [39] **Goutte C., Gaussier E.** *A probabilistic interpretation of precision, recall and F-score, with implication for evaluation.* //European conference on information retrieval. – Springer, Berlin, Heidelberg, 2005. – C. 345-359.

- [40] **Stehman S. V.** *Selecting and interpreting measures of thematic classification accuracy.* //Remote sensing of Environment. – 1997. – T. 62. – №. 1. – C. 77-89.
- [41] **Somasundaram K., Murphy G. C.** *Automatic categorization of bug reports using latent dirichlet allocation.* //Proceedings of the 5th India software engineering conference. – 2012. – C. 125-130.
- [42] **Tian K., Revelle M., Poshyvanyk D.** *Using latent dirichlet allocation for automatic categorization of software.* //2009 6th IEEE International Working Conference on Mining Software Repositories. – IEEE, 2009. – C. 163-166.
- [43] **Yang J. et al.** *Two-dimensional PCA: a new approach to appearance-based face representation and recognition.* //IEEE transactions on pattern analysis and machine intelligence. – 2004. – T. 26. – №. 1. – C. 131-137.
- [44] **Huang A. et al.** *Similarity measures for text document clustering.* //Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand. – 2008. – T. 4. – C. 9-56.
- [45] **Clark K. et al.** *What does bert look at? an analysis of bert's attention.* //arXiv preprint arXiv:1906.04341. – 2019.
- [46] **Venugopalan S. et al.** *Captioning images with diverse objects.* //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2017. – C. 5753-5761.