ОТЧЕТ ПО КЛАСТЕРИЗАЦИИ ТОВАРОВ И ФОРМИРОВАНИЮ СИСТЕМЫ КАТЕГОРИЙ

Данный отчет представляет собой обзор и описание метода, примененного для категоризации объектов и формирования универсальной системы категорий. В качестве данных, для которых проводилась кластеризация, были использованы описания товаров из интернет-магазинов (маркетплейсов).

ОПИСАНИЕ МЕТОДА КЛАСТЕРИЗАЦИИ

1. **BERTopic**

BERTopic – это техника тематического моделирования на основе использования технологии Transformer, которая формирует интерпретируемые кластеры, включая наиболее важные слова из каждого кластера в его название.

Данный инструмент поддерживает работу не только с английским языком, но и с другими, включая русский, что делает этот алгоритм привлекательным для нашей разработки.

Кроме того, данный инструмент позволяет выводить вероятностное распределения для каждого объекта по существующим категориям, однако этот функционал замедляет работу алгоритма при работе на больших данных.

Алгоритм BERTopic структурно состоит из 4 компонент:

- Модель получения эмбеддингов на основе трансформеров
- Понижение размерности через алгоритм UMAP
- Кластеризация HDBSCAN
- Тегирование кластеров с помощью c-TF-IDF
- MaximalMarginalRelevance (опционально)

1. *Модель получения эмбеддингов на основе трансформеров*

Может быть получена на основе использования таких библиотек, как Sentence Transformers, Flair, Gensim, SpaCy, передать модели этих библиотек можно передать в BERTopic.

Также в BERTopic можно передать уже построенные векторные представления, так для тестирования различных алгоритмов для текущего проекта были реализованы следующие алгоритмы получения векторов:

- Bag of Words
- Tf-Idf
- pretrained Word2Vec
- finetuned Word2Vec
- Bert tokenizer

Сравнительная характеристика алгоритмов:

1. **Bag of Words**

В результате работы данного алгоритма получается вектор, длиной которого является количество различных слов из словаря всех документов, по которым строится индекс. Компонентами вектора будут являться количества раз, которое повторилось слово в тексте.

+простой способ

-длина векторов не фиксирована

-вектора ортогональны

-вектор большой размерности

2. **Tf-Idf**

Tf-Idf - следующая ступень по построению векторного представления после Bag of Words. Этот алгоритм для каждого слова будет вычислять его индекс обратной частоты, то есть насколько редкий этот терм в корпусе статей, на которых проихводится обучение. Длина вектора будет совпадать с длиной вектора в алгоритме Bag of Words. Составлять вектор статьи будут коэффициенты TF*IDF, где TF - это частота слова в тексте, IDF - это обратная частота слова в корпусе текстов.

+учитывает вес слова в контексте всех документов

-вектор все еще большой размерности (размерность равна количеству слов в словаре)

3. **Word2Vec**

Гипотеза этой модели состоит в том, что “Близкие слова по смыслу встречаются в одном контекстею.”.

Word2Vec состоит из семейства моделей: Continuous Bag-Of-Words, Skip-gram, обе модели обучают нейронную сеть со скрытым слоем, из которого получаются эмбеддинги, только в первом случае предсказывается слово по контексту, а во втором контекст по слову.

Векторное представление в word2vec основывается на контекстной близости: слова встречающиеся рядом с одинаковыми словами, в векторном представлении будут иметь близкие координаты векторов-слов. Близость измеряется косинусным расстоянием.

Проблема данной модели состоит в том, что невозможно установить тип семантических отношений между словами: синонимы и антонимы будут одинаково близки, т.к. употребляются в схожих контекстах, также проблема word2vec состоит в том, что она не работает с словами вне словаря (OOV problem).

Качество построенной модели можно проверить с помощью оценки семантической близости между ними, для этого часто используются WordSim353 и SimLex-999

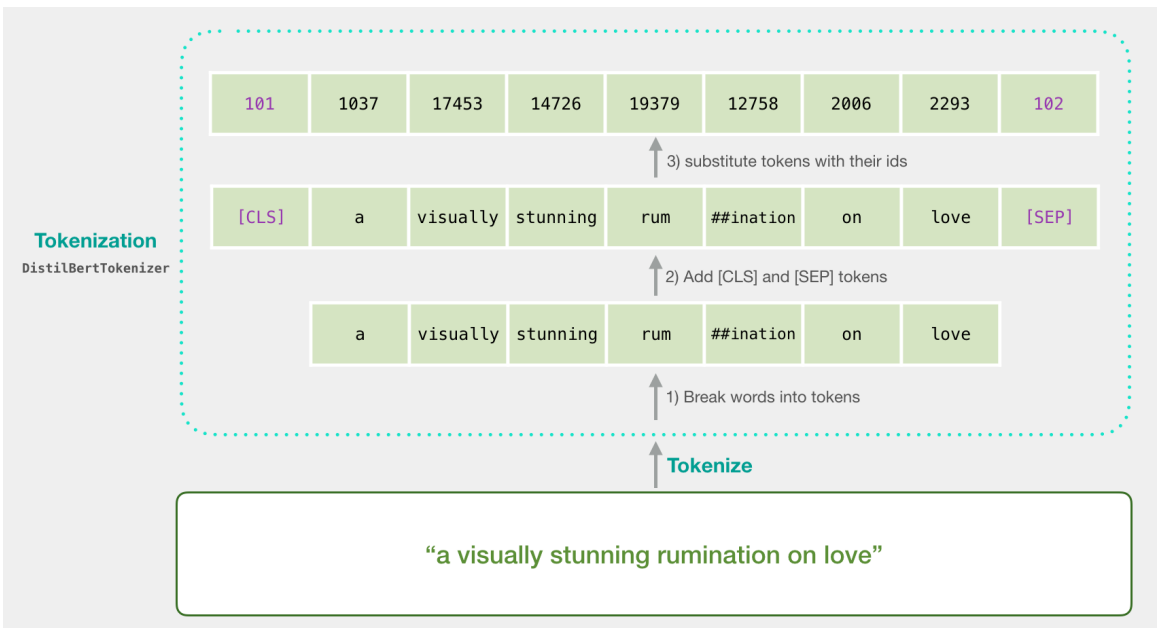
4. **BERT**

Алгоритм построения векторного представления BERT состоит в том, что:

1) происходит разбиение на токены (причем некоторые слова могут разбиться на подслова, в случае, если это можно сделать, отделив корень от суффикса с окончанием, такие слова помечаются ##, чтобы пониматься, что это часть предыдущего слова)

2) добавляются служебные токены: [CLS] -начало предложения, хранит в себе всю информацию о тексте, [SEP] - разделитель между предложениями, [PAD] - токен для выравнивания длин последовательностей (все последовательности одинаковой длины - специфика данного алгоритма)

3) Преобразование исходных слов и служебных токенов в численное представление индексами - для каждого слова свой индекс.



реализация данных алгоритмов предоставлена на гихаб→vectorizing→ vectorizing_funcs

В итоге была использована библиотека Sentence Transformers со словарем paraphrase-multilingual-MiniLM-L12-v2, которая в свою очередь работает с множеством языков. С помощью данной библиотеки были получены наиболее репрезентативные результаты.

Итоговая размерность эмбеддингов составляет (384,).

```
from sentence_transformers import SentenceTransformer
from bertopic import BERTopic

embedding_model = SentenceTransformer("paraphrase-multilingual-MiniLM-L12-v2")
topic_model = BERTopic(embedding_model = embedding_model )
```

2. *Понижение размерности с помощью алгоритма UMAP*

Этап понижения размерности необходим т.к. модели кластеризации дают лучшие результаты на данных с низкой размерностью, а также засчет понижения размерности ускоряется обучение.

Алгоритм UMAP является алгоритмом по умолчанию в BERTopic, размерность понижается с 384 до 5. При остановке уменьшения размерности на значении 5 удалось сохранить информацию, а также добиться понятных результатов кластеризации.

UMAP является наиболее эффективным, поскольку он сохраняет значительную часть многомерной локальной структуры в более низкой размерности.

Основным гиперпараметром UMAP является n_neighbors, число соседей. Для каждого вектора выбираются n_neighbors его соседей и анализируется расстояние до каждого из них. Алгоритм UMAP работает таким образом, что пытается сохранить соответствующее расстояние между корневым объектом и n_neighbors-ым соседом. (n_neighbors = 15)

3. *Кластеризация HDBSCAN*

По умолчанию BERTopic как алгоритм кластеризации использует HDBSCAN

HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) –алгоритм, основанный на плотности иерархической пространственной кластеризации
приложений с шумом. Алгоритмы, основанные на плотности кластеризации объединяют объекты в кластеры путем поиска таких областей в пространстве данных, которые имеют высокую плотностью и при этом окружены областями с меньшей плотностью. Особенность HDBSCAN, как алгоритма иерархической кластеризации, состоит в том, что он способен идентифицировать кластеры с переменной плотностью благодаря созданию иерархии вложенных кластеров.

Результатом работы алгоритма является иерархия HDBSCAN, где
различные уровни плотности соответствуют различным значениям радиуса. Метки, присвоенные компонентам иерархии, являются номерами кластеров.

Одним из основных плюсов алгоритма является то, что первоначально можно не указывать количество кластеров, а затем путем объединения похожих урезать количество полученных топиков, получается добиться лучших результатов

4. *Тегирование кластеров с помощью c-TF-IDF*

С помощью алгоритма c-TF-IDF выявляются ключевые слова для каждого из полученных кластеров. Таким образом появляется интерпретируемость кластеризации и возможность ее визуальной и ручной оценки. Алгоритм c-TF-IDF отвечает на вопрос: “что отличает один кластер, исходя из их содержания, от другого?”.

Алгоритм c-TF-IDF, построен на базе TF-IDF, целью которого является сравнить важность слов в различных документов, в то время как в алгоритме c-TF-IDF все тексты, принадлежащие одному топику объединяются в один единый текст, затем итоговая оценка TF-IDF дает на этом наборе наиболее важные слова внутри топика.

$$c-TF-IDF_i = \frac{t_i}{w_i} \times \log \frac{m}{\sum_j^n t_j}$$

Частота каждого слова t извлекается для каждого класса i и делится на общее количество слов w. Можно сказать, что в результате этого действия получается упорядочивание частых слов внутри топика. Затем общее количество несвязанных документов m делится на общую частоту встречаемости слова t во всех классах n.

5. Topic Representation with MMR

MaximalMarginalRelevance

MMR пытается уменьшить избыточность результатов, в то же время сохраняя релевантность запроса результатов для уже ранжированных документов/фраз.

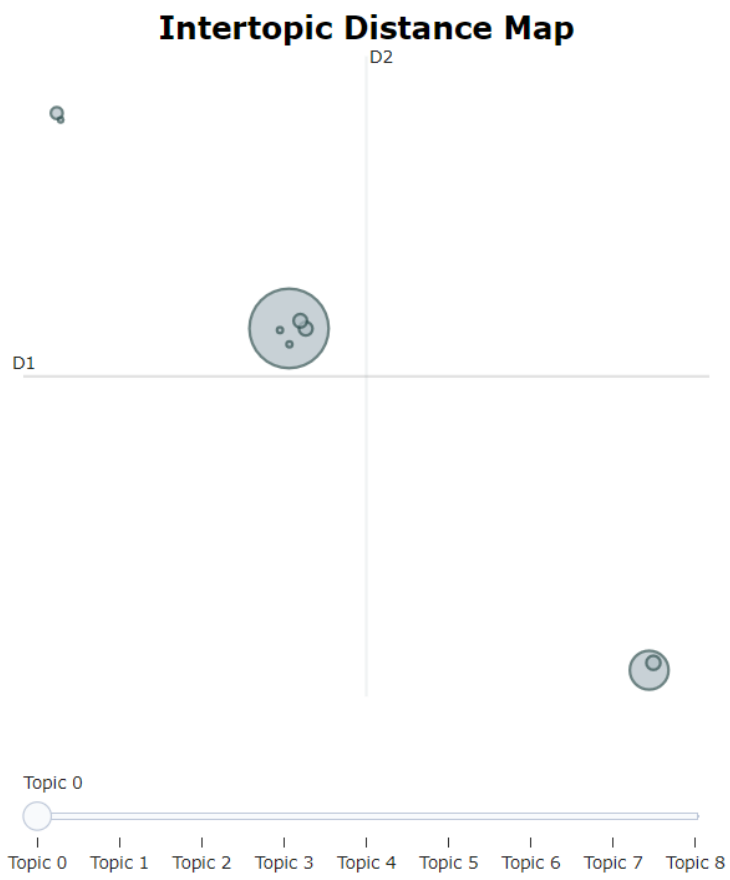
С помощью данного алгоритма можно улучшить полученные результаты засчет того, что повышается согласованность терминов в пределах одного кластера, а также за счет удаления нерелевантных терминов. Также данный алгоритм позволяет удалить однокоренные и синонимичные слова из топика.

Реализация всех вышеперечисленных функций : гитхаб→topics→ extract_topics.py

Также в ноутбуке (указать путь) предоставлена реализация на тестовых данных

Также BERTopic предоставляет возможность визуализировать полученные разбиения.

Примеры после нашего обучения на тестовых данных (4500 товаров различных категорий):



Первоначальные категории: ['Детское творчество и досуг', 'Аксессуары', 'Карповая рыбалка',
'Ножи и аксессуары', 'Автоэлектроника и навигация']

Из представленных визуализаций можно увидеть, что модель довольно неплохо распределила данные, нашла все категории, но основная проблема полученных результатов - невозможность использовать их для дальнейших задач, так как у топиков нет четкого названия, а товары с похожим написанием, но различным значением могут (но редко) попасть в другую категорию.

Также была обучена модель на всем наборе данных, обучение такой модели занимает около 60 минут, а вес модели составляет около 2GB

Модель обученная на тестовых данных (4500 товаров, 5 категорий) - https://drive.google.com/file/d/1s7XxBddePVNjLg79NAclFJrAQImcp7vf/view?usp=share_link

Модель обученная на всех данных (264000 товаров, 1288 категорий) - https://drive.google.com/file/d/1n7NHYnMChq9rPc9GH4LZCWzNsV3fYhdx/view?usp=share_link

Проведение экспериментов:

Построение векторного пространства

В случае использования **BERTokenizer**, на 5 категорий получается 43 кластера, при этом кластеры в себе могут содержать несколько категорий как на картинке ниже. Так происходит как на кластерах, в которые попало много данных, так и в меньших кластерах. При этом если менять настройки алгоритма кластеризации (в нашем случае hdbscan), то увеличить качество топиков удается только увеличением и так не малого кол-ва кластеров.

```
[('для', 0.05900073413425729),
 ('нож', 0.05151484661427805),
 ('ножей', 0.04966008220425838),
 ('набор', 0.04331289812127765),
 ('видеорегистратор', 0.039526259014753325),
 ('автомобильный', 0.036333633401226126),
 ('вида', 0.035549456552582445),
 ('заднего', 0.035455362449180794),
 ('подставка', 0.03536209013883487),
 ('см', 0.03480239367757083)]
```

При использовании **tf-idf** для построения векторного представления результаты получаются лучше, чем с BERT embeddings, но хуже чем с SentenseTransformers. Разделяются топики достаточно хорошо, но много похожих. Если установить min_cluster_size=30,min_samples=None, то топиков становится меньше, за счет их объединение и получается 26 кластеров, что при уменьшении по косинусной близости дает не лучший вариант, тк объединяются различные категории.

При тестировании **word2vec** модели и раличных параметров кластеризации получить качественные кластеры не получается, при достаточно большом количестве кластеров, внутри одного кластера все равно встречаются слова из различных категорий.

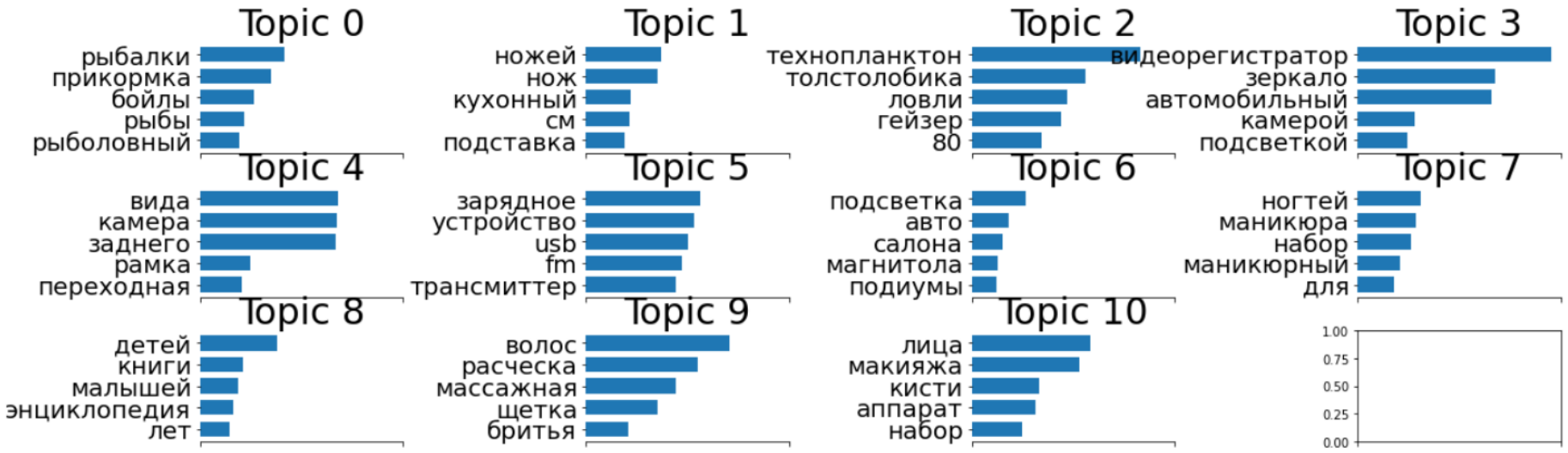
```
[('для', 0.07554845154103484),
 ('набор', 0.05469308402590035),
 ('ножей', 0.05092074133438142),
 ('книга', 0.033995975433245916),
 ('лица', 0.03325224478575474),
 ('подставка', 0.030338573327385283),
 ('детей', 0.027372042276250043),
 ('макияжа', 0.026618324864428 72),
 ('нож', 0.025759164432914875),
 ('рыбалки', 0.0255493777689258)]
```

Методы кластеризации

Все следующие примеры использовали SentenceTransformer embeddings.

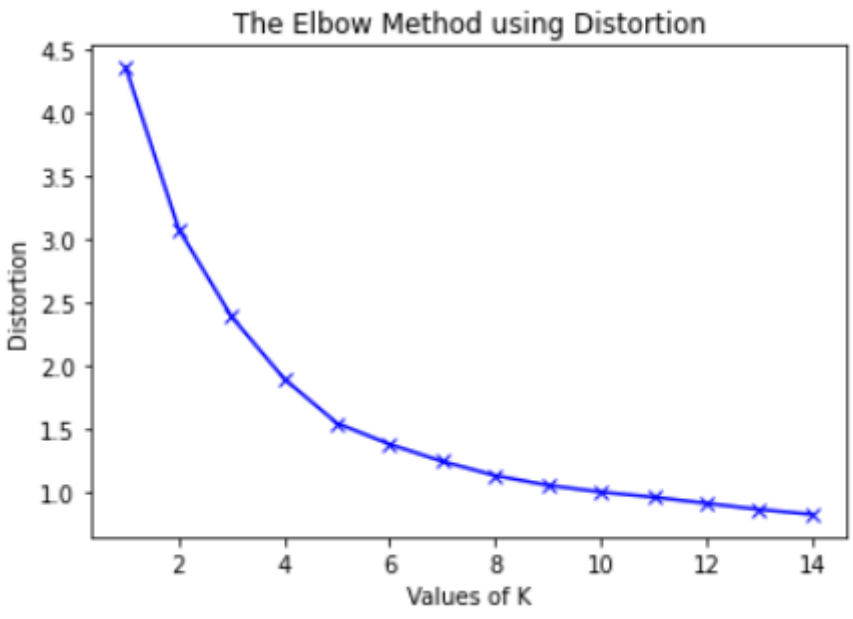
1) hdbscan

Плюс этого метода заключается в том, что для него необязательно указывать количество кластеров, он это делает автоматически в отличии от других алгоритмов

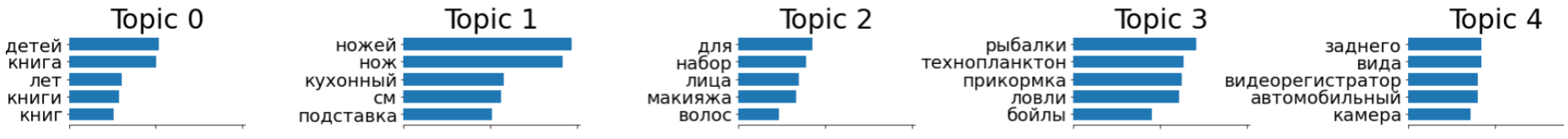


2)k-means

Согласно методу локтя точно определить кол-во необходимых кластеров сложно, но виден некий перегиб в пределах 5 кластеров (что соответствует изначальному кол-ву категорий)



При ипольлзовании алгоритма, получилось достичь довольно неплохого качества



Основная проблема данного метода кластеризации является в том, что необходимо определить кол-во категорий на большом кол-ве данных (~1100 категорий)

Кластеризация без уменьшения размерности

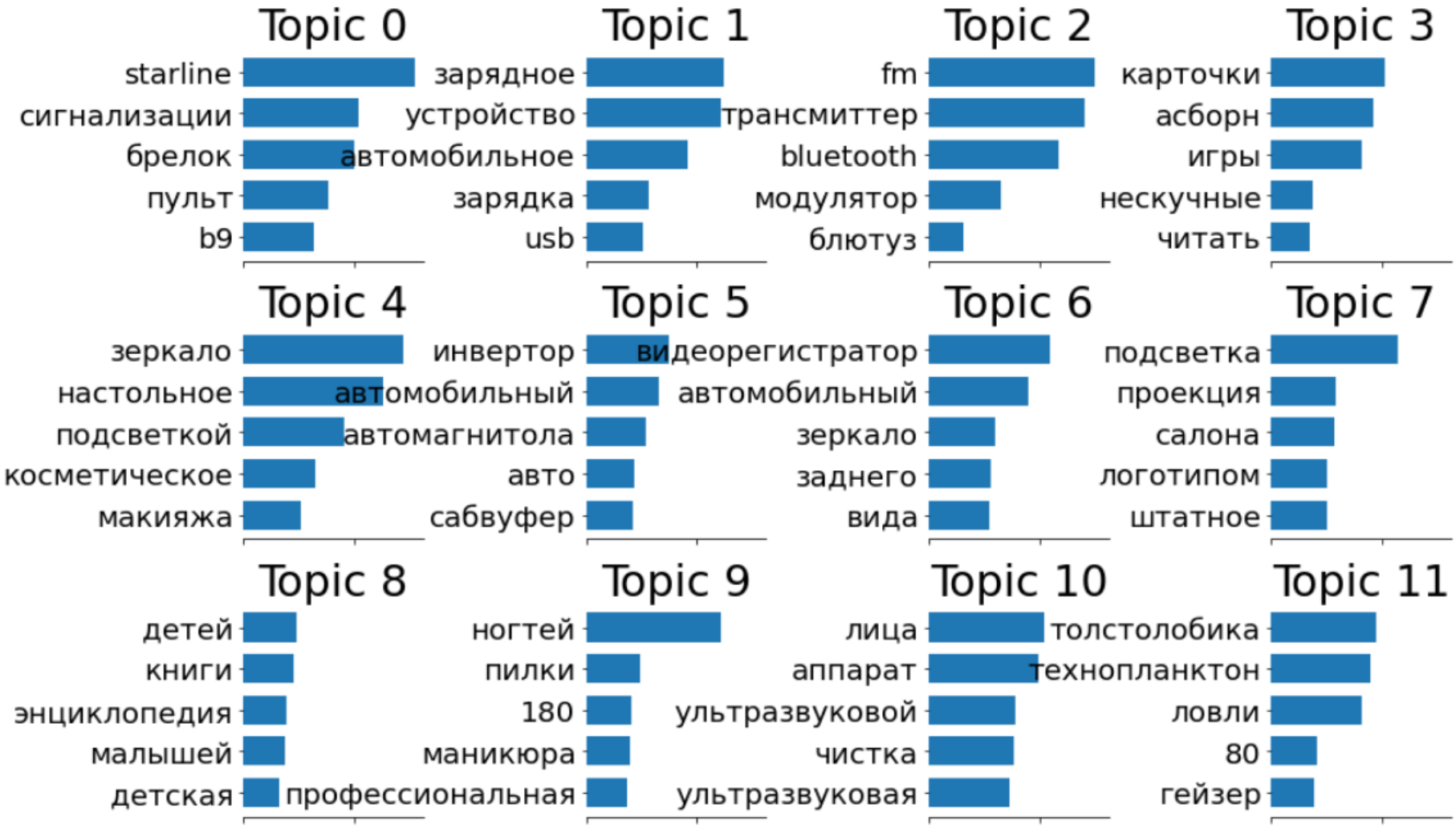
В данной работе применяется алгоритм уменьшения размерности, тк в этом случае алгоритмы кластеризации показывают лучшее качество за меньшее время для обучения.

В качестве алгоритма уменьшения размерности был выбран алгоритм UMAP, тк он сохраняет значительную часть многомерной локальной структуры в более низкой размерности.

Уменьшение размерности до использования алгоритма кластеризации помогает…

При отсутствии данного шага, качество разбиений значительно падает(ноу)

Также было замечено, что при отсутствии шага уменьшения размерности, но тьюнинга алгоритма кластеризации возможно добиться таких же результатов, что и с шагом уменьшения размерности, но все же будем использовать, тк на больших данных это дает прирост к скорости.



Ноутбук для проведения жкспериментов - TopicModelingExperiments.ipynb