# Министерство образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего учебного заведения

Факультет цифровых технологий и химического инжиниринга (ЦиТХИн)

Направление подготовки:  
Информационные системы и технологии 09.03.00

# Курсовая работа

Разработка информационной системы интеллектуального анализа отзывов пользователей контента

|  |  |
| --- | --- |
| Проверяющий: | Выполнил: студент 2-курса очной |
| Старший преподаватель кафедра ИКТ | Формы обучения группы КС-23 |
| Васецкий Алексей Михайлович | Бяков Александр Александрович |

Москва 2024

# Постановка задачи

## Стек Технологий:

1. СУБД: PostgreSQL
2. Платформа для администрирования и настройки СУБД: pgAdmin4
3. Язык программирования: python

## Описание задачи:

Суть данного проекта состоит в следующем: с помощью парсинга на языке программирования - python, происходит сохранение комментариев, отзывов, в базу данных PostgreSQL. После чего, на данном наборе различных комментариев мы обучаем нейронную модель Doc2Vec / e5-large-v2, так называемый алгоритм «Эмбеддинга» который преобразует текст (комментарий) в многомерный вектор в пространстве. На данном этапе мы получаем набор многомерных векторов, для которых мы можем применить алгоритм кластеризации, dbScan или kmeans, в зависимости от нужных требований. Скластеризованные комментарии мы проецируем на двумерную плоскость, и, таким образом, получаем множество групп, причем каждая такая группа будет иметь свой смысл, например, благодарности.

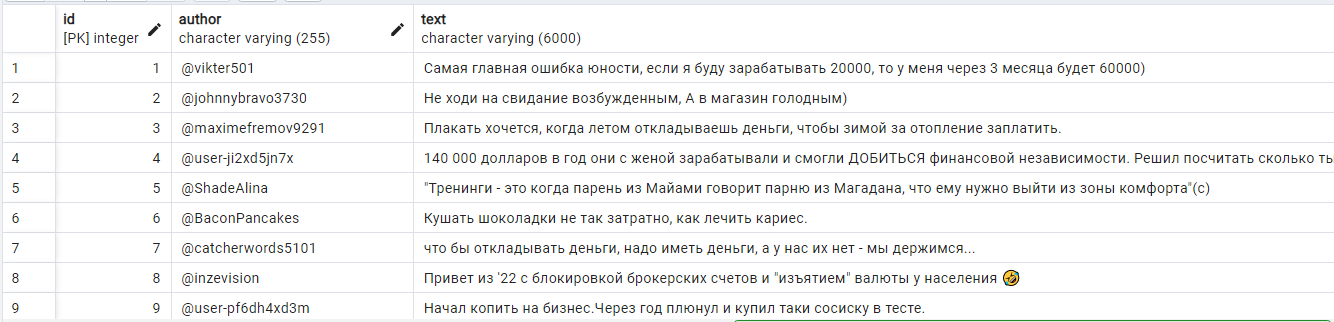
Практическая польза проекта состоит в быстром оценивании и просмотре сгруппированных комментариев для анализа.

# Обзор

## 1. Парсинг ютуб-комментариев

На данном этапе, использовался готовый парсинг на яп: python, так называемая библиотека YoutubeCommentDownloader, с ее помощью удается легко и просто вернуть набор комментариев для определенной ссылки на видео с хостинга. Однако у такого метода есть два нюанса. Первый - состоит в том, что не существует аргумента для парсинга определенного количества комментариев, метод всегда возвращает полный набор комментариев. Второй – низкая производительность. Так, например, в ходе сбора данных, было установлено, что загрузка, примерно, 20тыс. комментариев занимает 1000 секунд, что эквивалентно 17 минутам. Поэтому в будущем планируется произвести замену этой библиотеки на более эффективный парсинг, с использованием API YouTube.

Для задачи нам достаточно сохранять только текст, поэтому таблица будет выглядеть примерно следующим образом, хотя добавить колонки можно в любой момент:



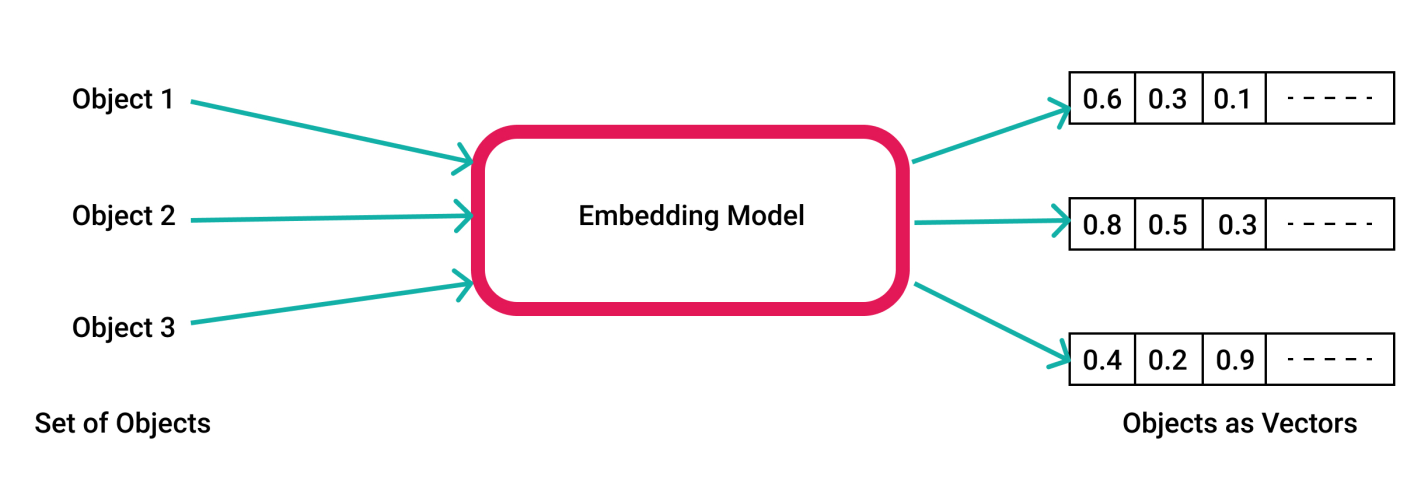
## 2. База данных

В качестве базы данных было принято решение использовать СУБД: PostgreSQL. Подключение и любая работа с ней, так же происходит программно и по большей части автоматически. С ее помощью можно создать новую базу данных, подключится уже к имеющей, производить вставку комментариев, получать набор комментариев, а так же, при надобности удалить таблицу.

## 3. «Эмбеддинг»

Прежде чем переходить к кластеризации точек в пространстве, сначала нам необходимо их представить в виде точек. Для этого и существуют метод «Эмбеддинга» - векторное представление чего-либо: слов или даже целых комментариев, что нам и нужно. По сути Эмбеддинг это некая нейронная модель, которая преобразует текст в вектор многомерного пространства, и чем этот вектор большей размерности, тем, как правило, он более точно представляет этот самый текст, но и вычисления алгоритм производит гораздо дольше, что сильно сказывается на производительности.

Для лучшего понимания представим себе автобус и попробуем выделить для него некоторые характеристики. Например: цвет, марка, корпус. Чем больше мы задаем для него параметров, тем более точно мы можем описать структуру автобуса. То же самое относится и к представлению текста в виде многомерного вектора, где каждая его размерность, по сути, и есть числовая характеристика.



Готовых моделей существует уже огромное количество. В работе использовались две модели – Doc2Vec и e5-large-v2. Чуть позже результаты показали, что Doc2Vec не так уж и хорошо работает с текстом, особенно большим, единственный плюс, который можно отметить, так это его быстрота преобразования. Модель - e5-large-v2 намного лучше справляется с этой задачей (топ #1 по эмбеддингу), но и требует огромных вычислений. Так, например, преобразование 3240 комментариев заняло у меня около получаса вычислений, поэтому для него нужны совсем другие мощности.

## 4. Оптимизация

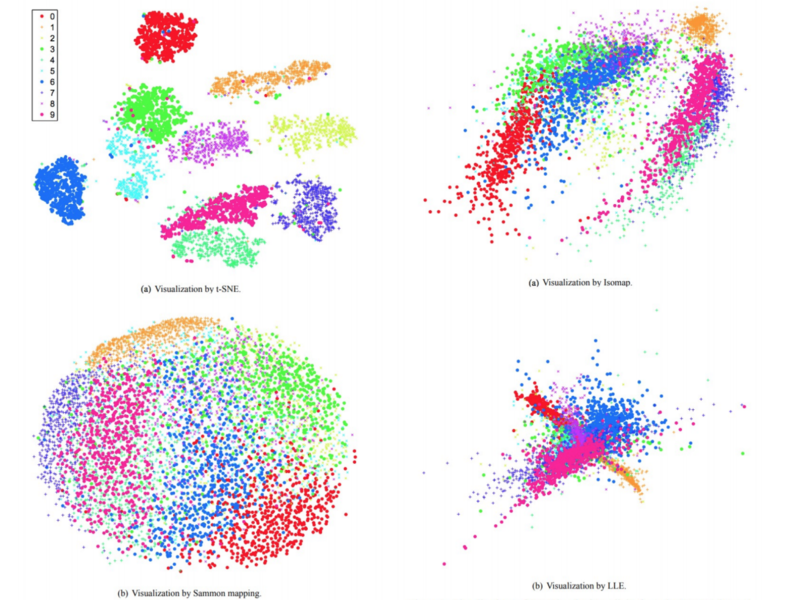
Чтобы не производить каждый раз огромные вычисления для одного набора комментариев, можно просто сохранять вектора в бинарном файле, причем лучше сразу в двух вариантах – исходные вектора 1024 мерного пространства и упрощенные – двумерные вектора. С этой целью отлично справляется библиотека «numpy» и пишется сам код буквально в две строчки.

## 5. Стохастическое вложение соседей с t-распределением (алгоритм t-SNE)

t-SNE - это алгоритм машинного обучения, для визуализации точек многомерного пространства. Он является техникой нелинейного снижения размерности, т.е. это некая проекция точки на плоскость, только в разы эффективнее и лучше. Здесь он используется для того, чтобы наши комментарии представить в виде точек на двумерной плоскости, которые изначально были многомерными. В частности, метод моделирует каждый объект высокой размерности двух- или трёхмерной точкой таким образом, что похожие объекты моделируются близко расположенными точками, а непохожие точки моделируются с большой вероятностью точками, далеко друг от друга отстоящими. Затем t-SNE определяет похожее распределение вероятностей по точкам в пространстве малой размерности и минимизирует расстояние Кульбака — Лейблера между двумя распределениями с учётом положения точек. Заметим, что исходный алгоритм использует евклидово расстояние между объектами как базу измерения сходства, это может быть изменено сообразно обстоятельствам.

Этот алгоритм может быть полезен не только в визуализации точек на плоскости, но и частичной Кластеризации. Ведь, как было сказано раннее, непохожие точки моделируются с большей вероятностью точками, которые стоят друг от друга далеко. Поэтому получается как бы, некие группы еще до группировки. И после t-SNE алгоритма очень удобно применять кластеризацию.

Хотя так может быть не всегда, поэтому стоит отметить, что результаты t-SNE могут зависеть от конкретных гиперпараметров и данных, с которыми он работает. Кроме того, при нелинейном уменьшении размерности могут происходить потери, которые могут повлиять на качество кластеризации.



Помимо алгоритма t-SNE, так же существую и другие техники для нелинейного снижения размерности. Но для данной задачи t-SNE подходит куда лучше, ведь группы в таком случае практически не пересекаются – а значит, анализировать будет куда проще и удобнее. (1) – t-SNE, (2) – Isomap, (3) – Sammon mapping, (4) - LLE

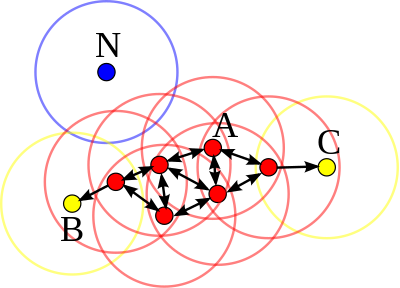
## 6. Алгоритмы кластеризации

### 6.1 DBSCAN

Алгоритм, основанный на плотности пространственная кластеризация для приложений с шумами. Если дан набор точек в некотором пространстве, алгоритм группирует вместе точки, которые тесно расположены (точки со многими близкими соседями), помечая как выбросы точки, которые находятся одиноко в областях с малой плотностью (ближайшие соседи которых лежат далеко).

У этого алгоритма есть два главных параметра: Первый – eps: это область, которая охватывает одна точка и min\_samples: минимальное количество точек, которые обязаны попасть в кластер, иначе кластер не сформируется и останется пустым. Label точки задается отрицательным: -1

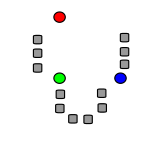
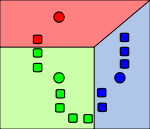
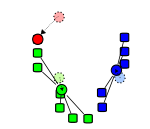
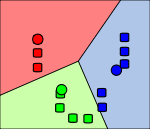
Демонстрация приведена:



### 6.2 kMeans - метод k-средних

Действие алгоритма таково, что он стремится минимизировать суммарное квадратичное отклонение точек кластеров от центров этих кластеров. Алгоритм представляет собой версию EM-алгоритма. Он разбивает множество элементов векторного пространства на заранее известное число кластеров k. Основная идея заключается в том, что на каждой итерации перевычисляется центр масс для каждого кластера, полученного на предыдущем шаге, затем векторы разбиваются на кластеры вновь в соответствии с тем, какой из новых центров оказался ближе по выбранной метрике.

Демонстрация приведена ниже:

У каждого из этих методов есть свои плюсы и минусы. Пожалуй, главной особенностью (а может и нет) алгоритма kMeans является то, что у него можно задать изначальное количество кластеров. Поэтому еще до алгоритма кластеризации мы изначально знаем количество групп комментариев. По расчетным данным получается примерно так, что для выборки из 3240 комментариев необходимо отразить не менее 100 кластеров. Иначе многие группы сплываются в слишком большой кластер, который не имеет общего смысла.

DBSCAN более медленный, по сравнению с kMeans, однако и у него есть свои плюсы. Пожалуй, главный из них заключается в том, что он может находить кластеры произвольной формы и размера, в отличие от других алгоритмов кластеризации, которые могут находить только кластеры определенной формы. Кроме того, DBSCAN не требует заранее заданного количества кластеров и может автоматически определять их количество.

# Компоновка программы

# C:\Users\byako\AppData\Local\Microsoft\Windows\INetCache\Content.Word\компоновка.png

# 

# Файловая структура программы

\_\_pycache\_\_

clustering

├── \_\_pycache\_\_

└── Clustering.py

collectData

├── \_\_pycache\_\_

└── commentLoader.py

GraphsPlotting

├── \_\_pycache\_\_

├── Graphs

└── countComments.py

Plots # примеры

├── data-test-dbscan.png

└── data-test-kmeans.png

vecFiles # примеры

├── data\_mygap1.npy

├── data\_mygap1p.npy

├── data1.npy

└── data2.npy

dialogWindows.py

Embedding.py

Embeddingv2.py

KMeans.txt

main.py

output.txt

PostgreConnect.py

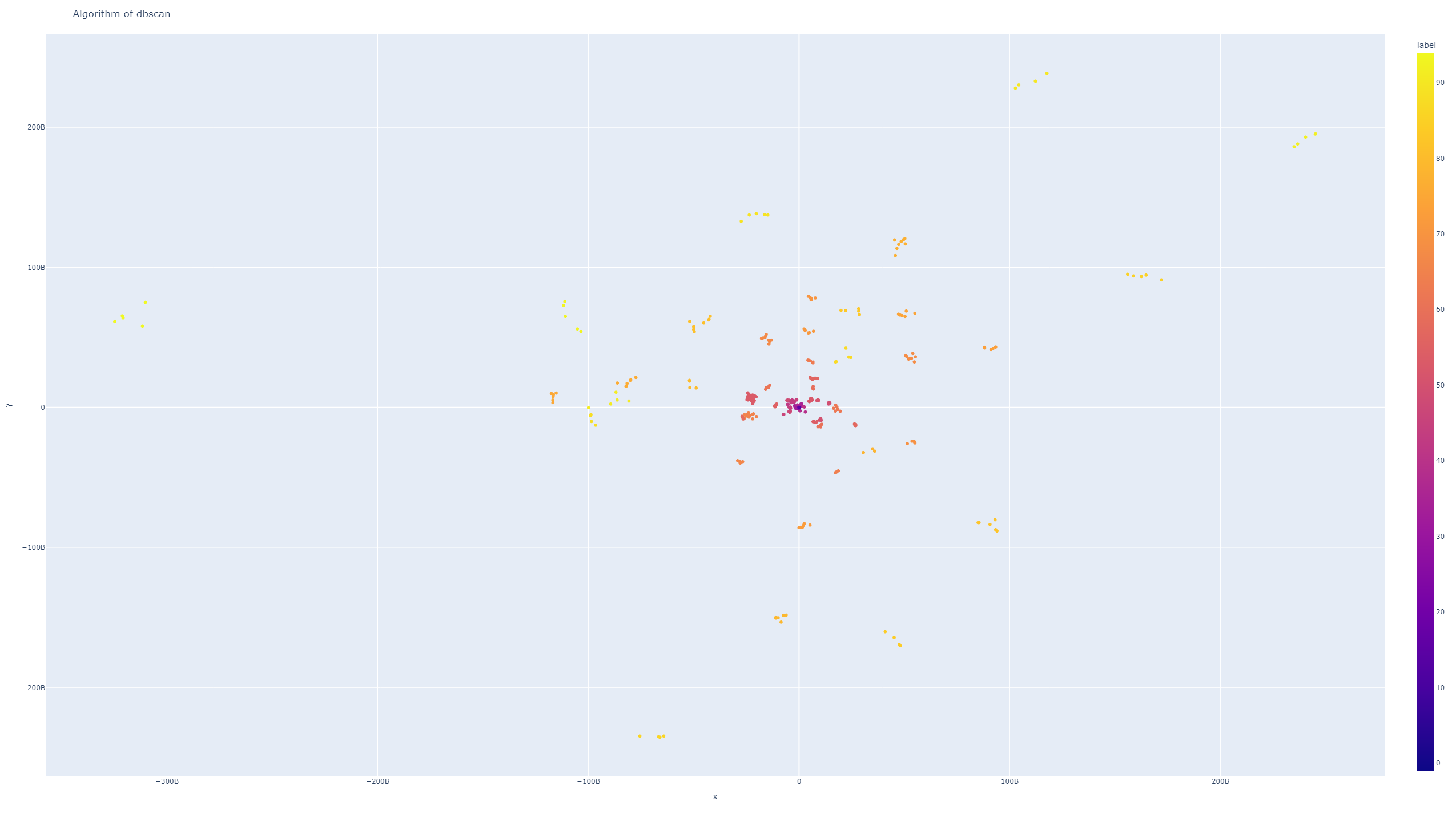
readIni.py

vecToFile.py

writeClusters.py

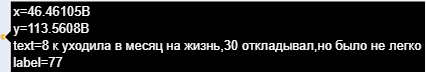
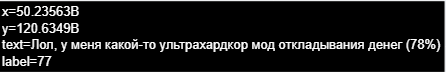
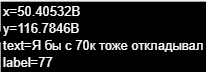
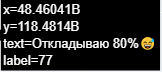
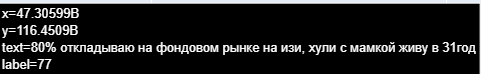
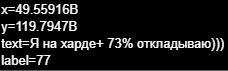
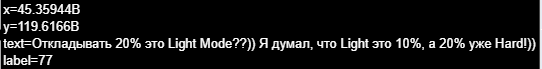
# Результаты DBSCAN

## Выборка 3240 комментариев с одного из видео у канала mygap



Таким образом, получили свыше 90 различных кластеров при eps = 2

Ниже приведены некоторые комментарии одного кластера под номером 77:

Смысл данных комментариев очень похож между собой. Все они говорят про откладывания денег. Делаем вывод, что алгоритм сработал хорошо.

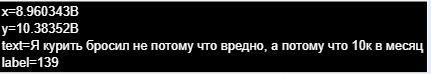
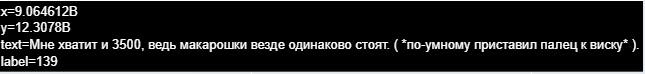
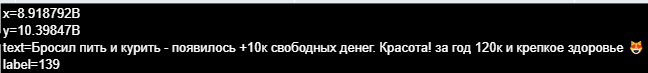
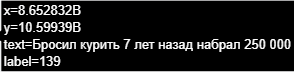
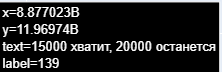
# Результаты kMeans

## Выборка 3240 комментариев с того же самого видео у канала mygap

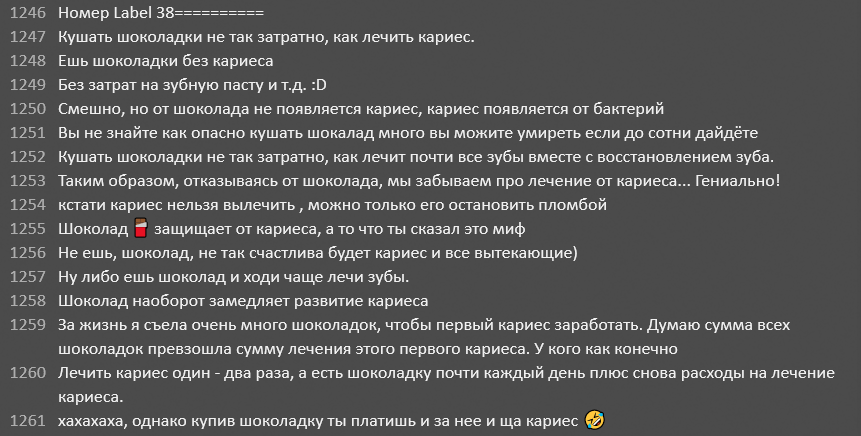
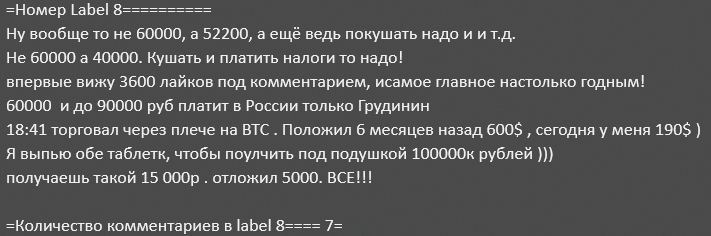


В параметрах для kMeans мы указывали 150 различных кластеров.

Ниже приведены некоторые комментарии одного кластера под номером 139:

Ниже приведен один из кластеров под номером 38 для kMeans алгоритма и 8 для DBSCAN, которые были сохранен в \*.txt файлы

# Вывод

В рамках данного проекта была успешно реализована система анализа комментариев с помощью парсинга на языке программирования Python и машинного обучения. Благодаря использованию алгоритма Doc2Vec/e5-large-v2, комментарии были преобразованы в многомерные векторы, что позволило применить алгоритм кластеризации для группировки похожих комментариев.

В результате кластеризации были получены группы комментариев, каждая из которых имеет свой смысл, например, благодарности, жалобы, вопросы, и т.д. Проекция кластеризованных комментариев на двумерную плоскость позволила визуально представить полученные группы и выявить скрытые закономерности в текстовых данных.

Эта система может быть полезна в различных областях, таких как анализ отзывов о продуктах, мониторинг социальных сетей и анализ текстовых данных в целом. Она позволяет автоматически выявлять тренды и паттерны в текстовых данных, что может быть полезно для принятия бизнес-решений, улучшения качеств обслуживания клиентов и т.д. Кроме того, она позволяет быстро оценивать весь набор комментариев, разбитых по группам, для анализа.

В целом, проект демонстрирует эффективность использования машинного обучения и алгоритмов кластеризации для анализа текстовых данных и выявления скрытых закономерностей в них. И, по большей части, проект основан на анализе, нежели практической части.