**Описание процесса сбора данных из ВКонтакте (VK API)**

Для выполнения проекта был реализован скрипт на Python, предназначенный для автоматического сбора текстовых постов из открытых сообществ социальной сети «ВКонтакте» с использованием VK API.

**1. Определение целей и тематики**

Перед началом сбора данных была сформирована структура по темам, каждая из которых включала определенные сообщества. Например:

* «Политика» — сообщество *kommersant.politics*,
* «Наука» — *nauka.tass*,
* «Технологии» — *haierrussia*, *store77*,
* и т.д.

Это позволило систематизировать данные по тематическим категориям.

**2. Настройка параметров сбора**

Были заданы следующие параметры:

* **Диапазон дат**: от 1 января 2022 года до 28 апреля 2024 года;
* **Максимальное число постов**: до 500 постов на сообщество;
* **Минимальная длина текста**: 50 символов;
* **Пауза между запросами**: 0.34 секунды для соблюдения ограничений VK API;
* **Формат хранения данных**: JSON и CSV.

**3. Авторизация через VK API**

Для работы с VK API использовался персональный токен доступа (ACCESS\_TOKEN). Он позволяет выполнять запросы от имени пользователя и получать доступ к методам API, в частности к wall.get для выгрузки постов из сообществ.

**4. Получение и фильтрация данных**

Основная функция, отвечающая за сбор данных, выполняла следующие действия:

* Подключение к API и получение ID сообщества;
* Получение постов с помощью метода wall.get, постранично;
* Фильтрация по дате публикации и минимальной длине текста;
* Очистка текста от HTML-тегов, ссылок и лишних пробелов;
* Извлечение хэштегов;
* Формирование структуры: текст, дата, тема, хэштеги, ID поста.

**5. Обработка ошибок и повторные попытки**

Каждый API-запрос оборачивался в безопасную функцию с повторными попытками в случае ошибок (например, превышения лимита запросов, временной недоступности сервиса и др.).

**6. Сохранение результатов**

По завершении сбора по каждой тематике данные сохранялись:

* в формате **JSON** — для хранения полной структуры поста,
* в **CSV** — для удобства анализа и импорта в аналитические инструменты (например, Excel, pandas, Power BI и т.д.).

Все данные сохранялись в локальную папку vk\_dataset/.

**7. Вывод**

В результате был собран тематически размеченный датасет постов из ВКонтакте, пригодный для анализа содержания, выявления трендов, построения классификаторов, анализа тональности и других исследовательских или прикладных задач в рамках проекта.

**Этап предобработки данных**

**1.Загрузка и чтение данных**

На первом этапе мы загружаем набор текстов и сохраняем его в виде таблицы (DataFrame).Мы рассматриваем каждый текст как отдельный документ — это могут быть статьи, отзывы, комментарии и т.д.

**2. Приведение текста к нижнему регистру**

Все буквы в тексте приводятся к **нижнему регистру**. Это необходимо для того, чтобы одинаковые по смыслу слова (например, *"Apple"* и *"apple"*) не считались разными. Такое приведение устраняет избыточность и помогает объединить статистику по словам.

**3. Удаление спецсимволов, цифр и лишнего**

Мы используем регулярные выражения (re.sub), чтобы:

* удалить **цифры** (например, "2023", "1000"),
* убрать **спецсимволы**, такие как !, ?, #, @, кавычки и знаки пунктуации,
* оставить только **буквы и пробелы**.

Таким образом, текст очищается от лишнего "шума", не несущего полезной смысловой нагрузки для тематического анализа.

**4. Лемматизация**

После очистки текста применяется **лемматизация** с помощью инструмента Mystem от Яндекса — это морфологический анализатор, который приводит каждое слово к его начальной форме (лемме).

**Примеры:**

* "кошки" → "кошка"
* "бежала" → "бежать"
* "машины" → "машина"

Это необходимо, чтобы объединять статистику по разным формам одного и того же слова. Например, слова "делать", "делаю", "делал", "делают" — после лемматизации все превратятся в "делать".

Также на этом этапе мы:

* удаляем стоп-слова — самые частотные и бессмысленные для анализа слова вроде *"это"*, *"в"*, *"на"*, *"и"*.
* оставляем только **осмысленные слова длиной от 3 символов**, чтобы исключить короткие слова без значения.

**5. Создание матрицы "документ-слово"**

После лемматизации создаётся **матрица частот слов** (doc\_term\_matrix) с помощью CountVectorizer. В этой матрице:

* каждая строка — это документ (текст),
* каждый столбец — это отдельное слово (после обработки),
* значения — количество раз, которое слово встречается в документе.

Также настраиваются фильтры:

* игнорируются слишком редкие слова (встречаются только в одном документе),
* исключаются слишком частые слова (встречаются почти в каждом тексте),
* убираются стандартные английские стоп-слова (если тексты на английском).

**Классические методы**

В NLP текстовые данные нельзя напрямую использовать в алгоритмах машинного обучения, так как они работают с числами. Для преобразования текста в числовые векторы применяются методы

## **Bag of Words (BOW, "Мешок слов")**

## **8 слайд**

## LSA (Latent Semantic Analysis) — это алгоритм обработки естественного языка, который помогает находить скрытые темы в текстах. Он работает на основе математического метода — сингулярного разложения матрицы

* На вход подаётся матрица "документы × слова".
* LSA раскладывает её на три компонента, выделяя **тематические паттерны**.
* Уменьшает шум и объединяет слова с близким значением (например, "автомобиль" и "машина").

## Мы протестировали несколько вариантов LSA с разным числом тем: **5, 10, 15, 20, 30**. Для оценки использовали **объяснённую дисперсию** — метрику, которая показывает, насколько хорошо модель улавливает структуру данных.

## При 30 темах объяснённая дисперсия достигла **14.6%** — это максимальное значение в нашем эксперименте.

## 9 слайд

LDA **(Latent Dirichlet Allocation)** - это алгоритм тематического моделирования, который представляет документы как смеси тем, а темы как распределения слов. В отличие от других методов, LDA использует вероятностный подход на основе распределения Дирихле.

Для оценки качества моделей мы использовали метрику перплексии. Эта метрика показывает, насколько хорошо модель предсказывает новые данные. Чем меньше значение перплексии, тем лучше качество модели.

Мы также протестировали несколько вариантов с разным количеством тем. Результаты показали, что наименьшая перплексия достигается при использовании 5 тем и составляет 1533.96. С увеличением количества тем значение перплексии последовательно растет, что говорит об ухудшении качества модели.

10 слайд

В отличие от предыдущего эксперимента с scikit-learn, здесь мы использовали когерентность как метрику и реализовали LDA через библиотеку Gensim.

Метрика когерентности оценивает, насколько семантически связаны между собой слова внутри каждой темы. Значения этой метрики находятся в диапазоне от -1 до 1, где более высокие значения указывают на лучшую интерпретируемость и качество тем.

## Наивысший показатель когерентности - 0.4895 - был достигнут при построении модели с 5 темами.

## ПОКАЗЫВАЕМ ФАЙЛ HTML С КРАСИВЫМИ КРУГЛИКАМИ

## 11 слайд

## PLSA (Probabilistic Latent Semantic Analysis) — это вероятностная модель, которая автоматически выделяет темы в текстах, анализируя распределение слов по документам. Она основана на предположении, что:

## Документы состоят из смеси скрытых тем, а темы состоят из наборов слов с определённой вероятностью.

## Каждая тема описывается распределением вероятностей слов (например, тема "спорт" может включать слова "мяч", "гол", "матч" с высокими вероятностями).

## Каждый документ представляется как смесь тем (например, документ может на 60% состоять из темы "политика" и на 40% из темы "экономика").

## Слайд 12

## Мы сравнили четыре модели тематического моделирования (LSA, pLSA, LDA из sklearn и LDA из gensim) по метрике когерентности.

Из результатов видно что LDA из sklearn показала наилучший результат (0.5472) - это значит, что она формирует самые осмысленные и четкие темы и лучше всего подходит для нашего анализа текстов

Тематическое моделирование успешно идентифицировало 5 из 6 исходных категорий датасета:

* ✓ Политика
* ✓ Искусство
* ✓ Литература
* ✓ Спорт
* ✓ Технологии
* ✗ Наука (слабо выделена, возможно, смешивается с другими темами)

## Слайд 13

## Далее мы перешли к нейросетевым методам BERT

## Сначала тексты преобразуются в векторные представления с помощью модели BERT, которая учитывает контекстное значение слов. Полученные эмбеддинги имеют высокую размерность, поэтому мы уменьшаем ее с помощью алгоритма UMAP, сохраняя при этом важные структурные взаимосвязи между текстами. Затем проводим кластеризацию полученных низкоразмерных представлений с использованием алгоритма HDBSCAN, который автоматически определяет оптимальное количество кластеров и хорошо работает с данными сложной формы. На завершающем этапе анализируем полученные кластеры, выявляя их тематическую принадлежность, и визуализируем результаты для наглядного представления структуры текстовых данных.

## Слайд 14

## Двунаправленность В отличие от предыдущих моделей BERT анализирует контекст слева и справа от каждого слова одновременно. Например Для слова "банк" в фразе *"Я положил деньги в банк"* модель учитывает и *"положил деньги"*, и *"в"*.

## Слои и механизмы

## Self-Attention (Механизм внимания):Каждое слово "видит" все остальные слова в предложении и вычисляет их важность для себя. Например, в фразе "коричневая лиса" слово "коричневая" сильно влияет на значение "лиса".Feed

## Forward Networks:

## Каждый слой содержит полносвязную нейросеть для дополнительной обработки данных.

## Слайд 15

Мы применили библиотеку BERTopic для автоматического выделения тем в русскоязычных текстах. Модель сочетает BERT-эмбеддинги для учета контекста и алгоритмы UMAP+HDBSCAN для кластеризации.

Настройки включали очистку от стоп-слов (например, "который", "весь") и фильтрацию редких/частых терминов. В результате получилось 7 интерпретируемых тем.

Самая большая тема (1491 документ) связана с историей — в топе слова типа "xix", "xviii", "telegram". Технологические бренды (Apple, Samsung, Haier) образуют отдельные кластеры. Например, тема про iPhone включает слова "pro", "max", "ios".

157 документов не попали ни в одну тему — вероятно, они содержат уникальный контент или шум.

Слайд 16

Мы провели оценку, насколько хорошо автоматически выделенные темы соответствуют реальному тематическому разделению документов. Исходные данные содержали 6 категорий: политика, наука, спорт, литература, искусство и технологии - примерно по 500 документов в каждой.

Модель BERTopic выделила 7 основных тем и группу из 157 документов без четкой тематики. Анализ показал, что лучше всего система справилась с литературой - 454 документа этой категории попали в одну тему. Тема искусства разделилась между несколькими кластерами. Категории политика и наука не были четко выделены моделью.

**Выводы о применимости методов для русскоязычного корпуса**

1. **LDA является оптимальным методом для общего тематического моделирования**:
   * Наивысшая когерентность тем (0.5472)
   * Наилучшее соответствие ожидаемым категориям
   * Хороший баланс между качеством, вычислительной сложностью и интерпретируемостью
2. **BERT-подходы превосходят в задачах тонкого тематического анализа**:
   * Обнаружение подкатегорий и скрытых семантических связей
   * Детальное разделение технологической тематики
   * Способность работать с минимально обработанными текстами
   * Автоматическое определение оптимального числа тем
3. **LSA менее эффективна для данного корпуса**:
   * Низкая объясненная дисперсия (14.6%)
   * Хуже интерпретируемые темы
   * Требует большого числа компонент для адекватного представления данных
4. **pLSA занимает промежуточное положение**:
   * Качество близко к LDA, но вычислительно более сложная
   * Часто объединяет разные категории в одну тему