Федеральное государственное бюджетное образовательное учреждение высшего образования

**«**ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ

РОССИЙСКОЙ ФЕДЕРАЦИИ**»**

**Департамент анализа данных**

**и машинного обучения**

**Пояснительная записка к курсовой работе**

по дисциплине:

**Технологии анализа данных и машинного обучения**

на тему:

**Предварительный анализ данных и построение признаков в задачах распознавания текста**

**Выполнил:**

Студент ИТиАБД

Группы ПИ19-4

Баканов Артем Иванович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Научный руководитель:**

Диденко Александр Сергеевич

Доцент, к.э.н.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Оглавление**

**Введение3**

**Постановка задачи4**

**Предварительный анализ данных4**

**Обработка данных5**

**Векторизация текста21**

**Графики24**

**Заключение24**

**Список источников25**

**Приложения25**

# **Введение**

В ходе данной работы требуется продемонстрировать владение основными навыками работы с методами обработки данных и выделения признаков из них: владение основными инструментальными средствами библиотек языка программирования Python, методами и приемами подготовительного и описательного анализа данных, средствами визуализации данных, умение делать выводы из проведенного анализа.

Для того, чтобы строить готовые модели по распознаванию текста или выполнять иные задачи по этой теме, всегда необходимо грамотно подготовить текстовые данные: избавиться от ошибок и бессмысленных элементов, привести к единому формату сам текст, выделить существенные части и слова. Лишь после вышеназванных действий можно приступать к выделению признаков. В данной работе будет применяться один из методов по векторизации текстовых данных. И на ее основе можно в последующем строить модели по распознаванию текста.

Векторизация текста – это процесс перевода текста на натуральном (естественном) в цифровую форму, которая в последующем пригодна для алгоритмической обработки. В курсовой работе будет использоваться метод векторизации на основе частоты встречаемости слов в тексте. Итогом станет матрица признаков (параметров), то есть таблица частот слов в тексте.

Анализировать тонны имеющийся информации вручную – дело очень сложное, трудоемкое и неблагодарное. С появлением компьютеров в нашей жизни мы начали задумываться как можно ускорить этот процесс. Поэтому в современном мире компьютерные вычисления и машинное обучение – неотъемлемая часть решения задач распознавания текста.

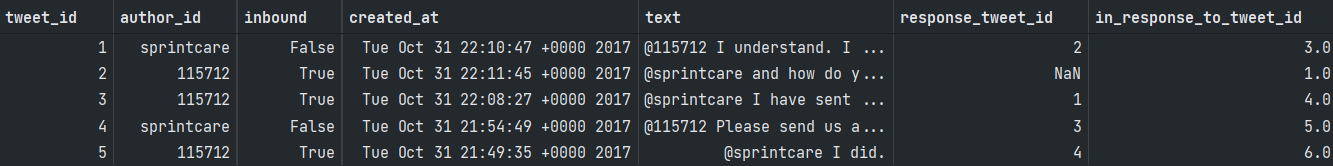
# **Постановка задачи**

В данной курсовой работе необходимо провести предварительный анализ и очистку текстовых данных для последующего выделения и построения признаков для корректного распознавания текста. Под выделением и построением признаков имеется в виду грамотная векторизация текста, то есть перевод текста на естественном языке (в нашем случае английском языке) в цифровую форму, пригодную для алгоритмической (компьютерной) обработке.

# **Предварительный анализ данных**

Изначальный датасет является случайным набором сообщений и данных из Twitter на английском языке (в основном). Структура таблицы представляет собой подобие содержания таблицы из названной социальной сети и включает в себя 7 уникальных полей данных:

* **Tweet\_id** – уникальный идентификатор сообщения
* **Author\_id** – уникальный идентификатор (никнейм) автора сообщения
* **Inbound** – сообщение отправлено или принято (булевское значение)
* **Created\_at** – дата и время создания сообщения
* **Text** – само текстовое сообщение
* **Response\_tweet\_id** – идентификатор сообщения, полученный в ответ на текущий текст
* **In\_response\_to\_tweet\_id** – идентификатор сообщения, на которое пишется ответ



Изображение 1 – изначальный набор данных

Для курсовой работы потребуется лишь одно поле – text. Именно этот столбец данных будет в последующем рассматриваться и обрабатываться. Но он также содержит множество особенностей и некорректных слов, которые буду дальше изучать и исправлять.

# **Обработка данных**

Изображение выглядит как текст

Автоматически созданное описание

Изображение 2 – количество записей

Текущий датасет содержит 196824 записей данных (текстовых сообщений), занимающие около 3 МБ дискового пространства.

Изображение выглядит как текст

Автоматически созданное описание

Изображение 3 – преобразование в нижний регистр

Преобразование в нижний регистр – это распространенный метод предварительной обработки текста. Идея состоит в том, чтобы преобразовать входной текст в тот же формат оболочки, чтобы “текст”, “Текст” и “ТЕКСТ” обрабатывались одинаково.

Это более полезно для методов векторизации текста, таких как frequency (CountVectorizer), TF-IDF, поскольку помогает объединять одни и те же слова вместе, тем самым уменьшая дублирование и получая правильные значения count/tf-idf. Это может быть бесполезно, когда мы выполняем такие задачи, как пометка части речи, где правильный регистр дает некоторую информацию о частях речи, и анализ настроений, при котором верхний регистр относится к гневу, крику, возмущению и т. д.

Изображение выглядит как текст

Автоматически созданное описание

Изображение 4 – удаление никнеймов

Далее удаляем уникальные идентификаторы (никнеймы) авторов сообщений, так как данные никнеймы не несут в себе никакой смысловой информации.

Изображение выглядит как текст

Автоматически созданное описание

Изображение 5 – удаление URL-ссылок

Следующим шагом предварительной обработки является удаление всех URL-адресов, присутствующих в данных. Например, если мы проводим анализ Twitter, то есть большая вероятность, что в твите будет какой-то URL-адрес. И для нашей задачи они не нужны, так как не являются конкретно словами.

Изображение выглядит как текст

Автоматически созданное описание

Изображение 6 – удаление HTML-тэгов

Еще один распространенный метод предварительной обработки, который пригодится во многих местах, – это удаление HTML-тегов. Это особенно полезно, если мы удаляем данные с разных веб-сайтов. В итоге мы можем получить HTML-строки как часть нашего текста.

Изображение выглядит как текст

Автоматически созданное описание

Изображение 7 – преобразование апострофов

Не менее важным шагов является удаление апострофов в тексте, тем самым раскрывая сокращения в полные слова. Часто в сообщениях пользователи для ускорения набора текста уменьшают (сокращают) слова, но нам необходимо иметь полный набор, поэтому необходимо избавиться от таких надстрочных запятых.

Изображение выглядит как текст

Автоматически созданное описание

Изображение 8 – преобразование аббревиатур

Это важный шаг предварительной обработки текста, если мы имеем дело с данными чата. Люди действительно используют много аббревиатур (сокращенных слов) в чате и поэтому может быть полезно расширить эти слова для наших целей анализа.

Изображение выглядит как текст

Автоматически созданное описание

Изображение 9 – удаление цифр

Зачастую обычные цифры и номера не несут в себе никакой смысловой нагрузки, значит нет необходимости держать их в наших данных.

Изображение выглядит как текст

Автоматически созданное описание

Изображение 10 – удаление знаков пунктуации

Еще одним распространенным методом предварительной обработки текста является удаление знаков препинания из текстовых данных. Это снова процесс стандартизации текста, который поможет одинаково относиться к "привет" и "привет!!!".

Нам также необходимо тщательно выбрать список знаков препинания для исключения в зависимости от варианта использования. Например, **string.punctuation** в *Python* содержит следующие символы препинания: !"#$%&\'()\*+,-./:;<=>?@[\\]^\_{|}~`

Изображение выглядит как текст

Автоматически созданное описание

Изображение 11 – преобразование эмоджи и эмотиконов

С ростом и расширением использования платформ социальных сетей наблюдается взрывной рост использования смайликов (эмоджи) и в нашей повседневной жизни. Также существуют эмотиконы – это смайлики, состоящие из символов клавиатуры, которые, будучи собраны определенным образом, представляют выражение лица, например, :) или :-(.

Зачастую эмоджи и эмотиконы дают некоторую ценную информацию для обработки текста и поэтому их удаление может быть не очень хорошим решением. Один из способов работы с ними – это преобразовать их словесный формат, чтобы их можно было использовать в последующих процессах моделирования.

Изображение выглядит как текст

Автоматически созданное описание

Изображение 12 – выделение «неанглийских» предложений

На данном шаге выводим все текстовые сообщения, которые не прошли проверку, содержащие особые спецсимволы и слова и буквы из других естественных языков.

Изображение выглядит как текст

Автоматически созданное описание

Изображение 13 – удаление «стоп-слов»

Стоп-слова – это часто встречающиеся слова в естественном языке, такие как «the», «a», «I» и так далее. В большинстве случаев их можно удалить из текста, поскольку они не предоставляют ценной информации для последующего анализа. В таких моментах, как выявление части речи, мы не должны избавляться от них, так как они дают очень важную информацию о самом слове, как части речи (исходя из контекста).

Эти списки стоп-слов уже составлены для разных языков, и мы можем безопасно их использовать. Например, список стоп-слов для английского языка из библиотеки **nltk** в *Python*.

Изображение выглядит как текст

Автоматически созданное описание

Изображение 14 – исправление орфографии

Опечатки часто встречаются в текстовых данных, и точно необходимо исправить, как минимум, большую часть орфографических ошибок, прежде чем проводить анализ и строить модели. В данной конкретной работе воспользуемся пакетом **pyspellchecker** из языка программирования *Python* для исправления орфографии.

Изображение выглядит как текст

Автоматически созданное описание

Изображение 15 – исправление орфографии

Лемматизация – это один из двух возможных процессов сокращения изменяемых слов до их основы (корня). Например, если в тексте есть два слова «красивый» и «красиво», то произойдет сокращение до корня слова, то есть получится в итоге «красив».

Лемматизация отличается, например, от стемминга, тем, что она гарантирует, что корневое слово, которое также называется леммой, принадлежит языку. В результате этот процесс, как правило, протекает медленнее, чем стемминг. Таким образом, в зависимости от требований к скорости, мы можем выбрать либо стемминг, либо лемматизацию. Для текущего датасета предпочтительно использовать лемматизацию ввиду большого количества сложных слов, которые требуют более тщательной обработки. В этой работе используется **WordNetLematizer** из пакета **nltk** для лемматизации наших предложений.

Изображение выглядит как текст

Автоматически созданное описание

Изображение 16 – часто встречающиеся слова

Изображение выглядит как текст

Автоматически созданное описание

Изображение 17 – удаление часто встречающихся слов

В текстовых датасетах одной спецификации или направленности очень часто встречаются повторяющиеся слова, которые либо для модели или другой цели не так важны, либо практически не несут в себе содержательной или смысловой нагрузки. Поэтому на данном шаге удаляем 10 самых часто встречающихся слов.

Изображение выглядит как текст

Автоматически созданное описание

Изображение 18 – редко встречающиеся слова

Изображение выглядит как текст

Автоматически созданное описание

Изображение 19 – удаление редко встречающихся слов

Аналогично предыдущему шагу, удаляем из датасета 20 самых редко встречающихся слов.

# **Векторизация текста**

После вышеописанных действий по обработке текстовых данных приступим к основной и главной части работы: выделение признаков из датасета. В текущем случае под этим подразумевается векторизация текста, то есть перевод текста в цифровую форму для последующего построения моделей или для иных целей, связанных с обработкой текста.



Изображение 20 – разделение датасета

Для начала нам необходимо разделить обработанный датасет на «обучающую» и «тестовую» части в соотношении 80 к 20, чтобы в последующем грамотно провести векторизацию текста: сначала обучим нашу модель векторизации, а затем закрепим результат на тестовой части. В данной ситуации без разницы в каком порядке или последовательности делить датасет. Поэтому воспользуемся случайным разделением. В этом нам поможет библиотека **sklearn** и функция из нее – **train\_test\_split**.

Изображение выглядит как текст

Автоматически созданное описание

Изображение 21 – векторизация текста

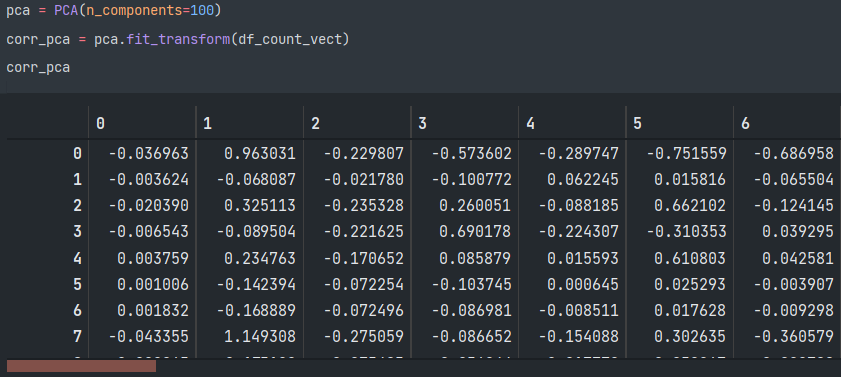
Приводим в действие алгоритм векторизации текста. Воспользуемся **CountVectorizer** из пакета **sklearn** языка программирования *Python*. Получаем матрицу, значения которой представляют собой количество вхождений данного слова в текстовую запись, то есть произошла векторизация текста на основе частоты каждого слова.

Изображение выглядит как текст, внутренний, монитор, снимок экрана

Автоматически созданное описание

Изображение 22 – наиболее часто встречающееся слово

В данном датасет самое часто встречающееся осмысленное слов – service – 2267 раз.

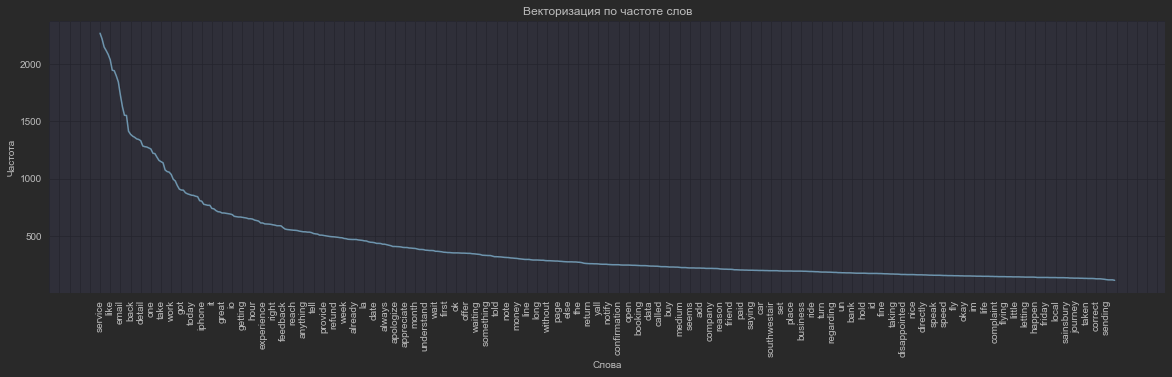


Изображение 23 – метод главных компонентов

В итоге мы получили матрицу векторизации, но появляется другая проблема – признаков очень много. Не все удастся грамотно и правильно обработать, отследить, учесть. Но некоторая доля этих признаков (параметров) не так сильно влияют на конечный результат, так как находятся в сильной связи друг с другом, следовательно и закономерно, часть из них можно отбросить. Например, если корень одного слова является частью другого, то его можно просто убрать без существенных потерь информации.

Для решения данной проблемы воспользуемся немало известным алгоритмом – методом главных компонент (principal components analysis; PCA). Для данного датасета он компонует все признаки до 100 единиц, выделяя наиболее связанные друг с другом параметры. В итоге получается матрица скомбинированной корреляции, то есть каждая ячейка этой матрицы представляет собой «усиленную» (сжатую) версию корреляции «сплюсованных» компонентов (по столбцам) и всех элементов векторизованной матрицы (по строкам). Алгоритм PCA имеется все в той же библиотеке **sklearn**.

# **Графики**



Изображение 24 – график частоты появления слов

Изображение выглядит как текст, снимок экрана, электроника, дисплей

Автоматически созданное описание

Изображение 25 – график метода главных компонентов

# **Заключение**

В ходе данной работы мы узнали, как правильно обработать данные и выделить признаки в задачах по распознаванию текста. Мы научились правильно подходить к анализу данных, разведочному анализу данных, научились предобрабатывать данные, выявлять излишние признаки методом главных компонент, векторизировать текстовые данные, получая матрицы частоты слов. Теперь данные полностью готовы для последующих построений моделей по распознаванию текста или иных текстовых задач.

# **Список источников**

* М.В. Коротеев. Об основных задачах дескриптивного анализа данных.
* М.В. Коротеев. Учебное пособие по дисциплине “Анализ данных и машинное обучение” - 2018.
* W. McKiney. Pandas: powerful Python data analysis toolkit - 2016 (1971p)
* <https://ru.wikipedia.org/wiki/Регулярные_выражения> - информация по построениям регулярных выражений
* <https://habr.com/ru/company/Voximplant/blog/446738/> - основы обработки текстовых данных
* <https://python-school.ru/blog/nlp-vectorization-methods/> - методы векторизации текста
* <https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html> - векторизация текста на основе частоты слов
* <https://www.nltk.org> – библиотека *Python* для обработки естественных языков
* <https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html> - метод главных компонент
* <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html> - функция деления датасета на «обучающую» и «тестовую» части
* <https://www.nltk.org/api/nltk.stem.wordnet.html> - алгоритм лемматизации текста

# **Приложения**

<https://github.com/Attilene/TextRecognitionWork> - рабочий программный код курсовой работы