**Сентимент анализ текста**

**Цель работы**: Изучение работы с текстовыми данными и их анализ.

Будем проводить анализ на выборке отзывов на фильмы «Large Movie Review Dataset»

Сентимент анализ или анализ тональности текстов – определение полярности эмоциональных оценок в исследуемом тексте, который содержит мнения, суждения, эмоции, отношение автора к сущностям, личностям, вопросам, событиям, темам и их атрибутам.

Передо мной стояла задача бинарной классификации отзывов о фильме – разделить данные на положительные и негативные отзывы. Для начала я выделила основные этапы для решения этой задачи:

1. Изучение структуры исходных данных
2. Предварительная обработка датасета
3. Визуальная оценка проделанной работы
4. Преобразование текста в цифры
5. Применение различных классификаторов
6. Оценка качества классификации, если результаты не устраивают, то доработать/изменить пункты 2,4

**Исходные данные**

Базовый набор данных содержит 50 000 отзывов, равномерно разделенных на 25 000 обучающих и 25 000 тестовых наборов. Общее распределение меток сбалансировано (25 тыс. положительных и 25 тыс. отрицательных). Также включены дополнительные 50 000 немаркированных документов для обучения без учителя.

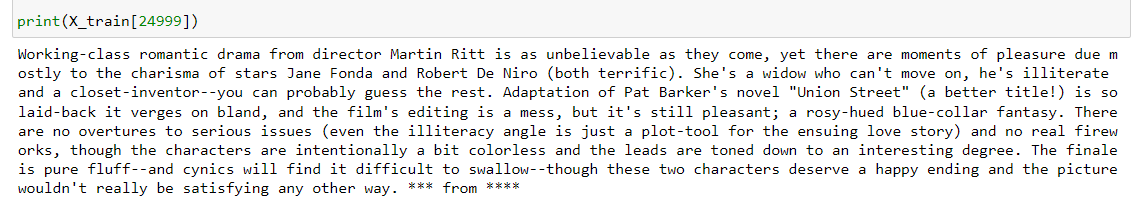
Файлы

Есть два каталога верхнего уровня [train/, test/], соответствующие тренировочному и тестовому наборам. Каждый содержит каталоги [pos/, neg/] для отзывов с бинарными метками положительных и отрицательных.

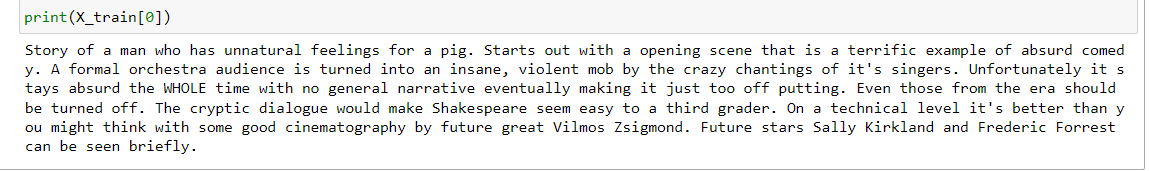
Посмотрим на несколько отзывов, которые у нас есть:

Обучающая выборка:

1. Положительный отзыв



1. Отрицательный отзыв



Тестовая выборка:

1. Положительный отзыв

Изображение выглядит как текст, Шрифт, число, снимок экрана

Автоматически созданное описание

1. Отрицательный отзыв

Изображение выглядит как текст

Автоматически созданное описание

В наших отзывах есть знаки препинания, html теги, которые не несут смысловой нагрузки при разделении отзывов на классы. Так же встроенные библиотеки в питоне чувствительны на регистр, поэтому нам требуется обработка текста.

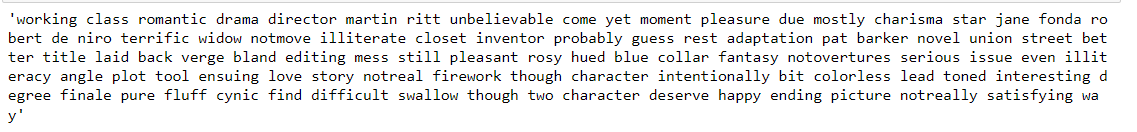
**Предварительная обработка текста**

При обработке текста будем использовать общеизвестные методы: преобразуем текст в нижний регистр, используя регулярные выражения, оставляем только буквы и удаляем ненужные пробелы, чтобы между словами было по одному пробелу, так же удаляем стоп слова. Но при работе со стоп словами было принято решение добавить в список слова film, movie, br – так как они никак не влияют на окрас отзыва, а по частоте занимают лидирующие позиции. Так же было выдвинуто предположение, что в отзывах о фильмах будем часто употребляться частица не, но в данном случае важен контекст, поэтому я объединила частицы не со следующим стоящим словом. Дальше стоял вопрос, что использовать лемматизацию или стемминг. Поэтому я провела исследования с 2-мя этими методами и в конце произведу сравнение этих двух методов.

**Лемматизация:**

Посмотрим на те же отзывы, что мы смотрели выше, но уже обработанные

Обучающая выборка: положительный отзыв



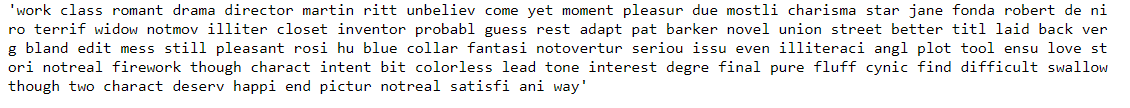
Тестовая выборка: отрицательный отзыв

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Стемминг**

Обучающая выборка: положительный отзыв



Тестовая выборка: отрицательный отзыв

Изображение выглядит как текст, снимок экрана, Шрифт, линия

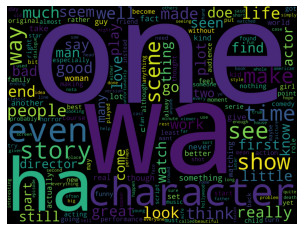
Автоматически созданное описание

**Вывод:** Стемминг – это грубый эвристический процесс, который отрезает «лишнее» от корня слов, часто это приводит к потере словообразовательных суффиксов. Лемматизация – это более тонкий процесс, который использует словарь и морфологический анализ, чтобы в итоге привести слово к его канонической форме – лемме. Как мы и видим, при лемматизации сохраняются части речи, а при стемминге все части речи одного слова приводятся к основе слова и тем самым теряется смысловой контекст. Так же при лемматизации в тестовой выборке мы получаем словарь из 71432 уникальных слов, а при стемминге на 20000 слов меньше – 54066.

**Проведенные исследования**

Для оценки проделанной работы построим **облако слов** и проанализируем полученные результаты. (Используем лемматизацию)

Облако слов для обучающей выборки: Облако слов для тестовой выборки:

 Изображение выглядит как текст, Шрифт, снимок экрана, Графика

Автоматически созданное описание

Как видим слова на тестовой и обучающей выборке пересекаются, следовательно можно предположить, что при хорошем обучении будет и хорошие результаты на тестовой выборке.

Построим облака слов для классов для того, чтобы визуально оценить различимость классов либо их идентичность

Положительные отзывы:

Изображение выглядит как текст, Шрифт, снимок экрана, Графика

Автоматически созданное описание

Отрицательные отзывы:

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

**Вывод:** получили удивительные результаты: как в положительных, так и в отрицательных отзывах встречаются такие слова как “like” и “good”. Но при этом в отрицательных отзывах встречаются такие слова “bad”, “nothing”, а в положительных “love”, “best”, которые не встречаются в противоположных классах.

Строить облако слов для текста, обработанного с помощью стемминга, не имеет смысла, так как они визуально очень похожи, и на глаз сложно найти различия.

Проделанный анализ помог нам изучить нашу выборку, увидеть разделимость классов. Теперь можно приступать к преобразованию слов с цифры, так как методы классификации, которые мы будем применять, работают только с цифрами.

**Решение задачи классификации**

Из наиболее известных и наиболее просто реализуемых метод будем использовать **мешок слов (CountVectorizer)**. Достаточно большое количество слов встречаются реже 5 раз. Поэтому создадим словарь из 20000 слов. И пересчитаем коэффициенты с помощью метода tfidf.

Мешок слов присваивает оценку слову на основе его появления в конкретном документе. При этом не учитывается тот факт, что это слово может также иметь высокую частоту встречаемости и в других документах. TFIDF решает эту проблему путем умножения частоты термина слова на обратную частоту документа. TF расшифровывается как “Частота термина”, а IDF-как “Обратная частота документа”.

В итоге было получено 2 матрицы размером (25000;20000).

Теперь проведем классификацию: для начала возьмем 5 классификаторов (логистическая регрессия, случайный лес, k-ближайших соседей, наивный байесовский классификатор), а потом оберем парочку лучших и будем работать с ними.

**Логистическая регрессия:**

**Лемматизация**

Изображение выглядит как текст, Шрифт, снимок экрана, алгебра

Автоматически созданное описание

**Стемминг**

Изображение выглядит как текст, Шрифт, снимок экрана, алгебра

Автоматически созданное описание

**Случайный лес:**

**Лемматизация**

Изображение выглядит как текст, снимок экрана, Шрифт, График

Автоматически созданное описание

В данном классификаторе мы можем вывести наиболее информативные признаки.

Изображение выглядит как текст, снимок экрана, диаграмма, График

Автоматически созданное описание

**Стемминг**

Изображение выглядит как текст, снимок экрана, Шрифт, линия

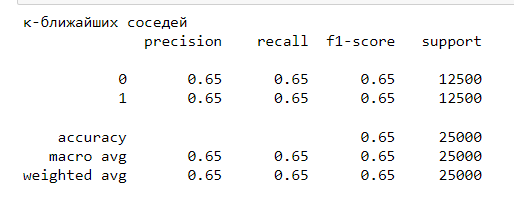
Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

**k-ближайших соседей:**

**Лемматизация**



**Стемминг** – не будем проводит исследования, так как лемматизация показала плохие результаты

**Наивный байесовский классификатор:**

**Лемматизация**

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

**Стемминг** – не будем проводит исследования, так как лемматизация показала плохие результаты

**Вывод:** как мы видим из полученных классификаторов, лучше всего отработали логистическая регрессия и случайный лес. Результаты при обработке семмингом и лемматизацией примерно одинаковы: случайный лес отработал одинаково, а на логистической регрессии наблюдается различие в 1%.

Методы k-ближайших соседей и наивный байесовский классификатор показали слишком плохие результаты, поэтому в дальнейшей работе не будем ими пользоваться.

На данный момент самое наибольшее значение классификации, которое получилось достигнуть – 89%.

**RFECV**

У нас получилась достаточно хорошая классификация, но при этом у нас достаточно много признаков с которыми сложно работать, скорее всего большинство из них не несут информации для классификации. Поэтому применим метод **RFECV** для отбора информативных признаков без потери качества классификации.

Будем удалять признаки с шагом 500 и до 500 признаков. А для выбора признаков будем использовать лучший классификатор – логистическая регрессия

**Лемматизация**

График зависимости количества признаков от качества классификации. 12500

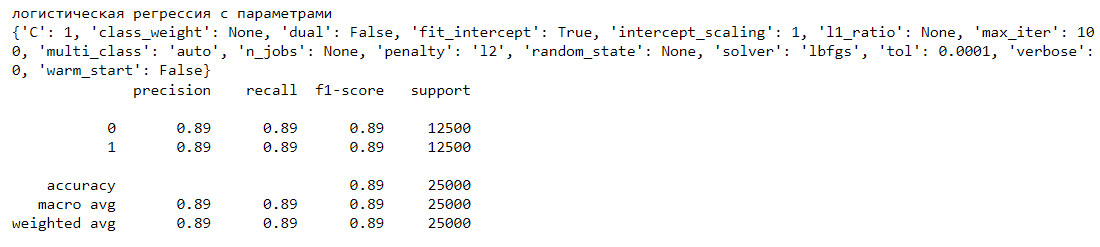
Изображение выглядит как текст, снимок экрана, линия, Шрифт

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Классификация:

**Логистическая регрессия:**

  
**Случайный лес:**

Изображение выглядит как текст, снимок экрана, Шрифт, График

Автоматически созданное описание

**Стемминг: 1500**

График зависимости количества признаков от качества классификации.

Изображение выглядит как текст, снимок экрана, График, линия

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, информация

Автоматически созданное описание

**Классификация:**

**Логистическая регрессия:**

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

**Случайный лес:**

Изображение выглядит как текст, снимок экрана, Шрифт, График

Автоматически созданное описание

**Вывод:** С помощью RFECV нам удалось сжать признаковое пространство до 1500 при использовании стемминга. А при обработке текста с помощью лемматазации признаковое пространство сжалось всего лишь до 12500, но качество классификации всего на 1% лучше. При отборе наиболее информативных признаков используется grid\_scores- среднее по кроссвалидации и более достоверный результат, чем accuracy. При обработке текста лемматизацией и стеммингом можно заметить отличия тысячную, что можно считать за одинаковый результат. Можно сказать, что по результатам в плане качества классификации – не принципиально что использовать, стемминг или лемматизацию, т.к. результат практически одинаковый. Но при использовании стемминга получилось в разы сильнее сжать пространство, чем при лемматизации.

Попробуем другой метод преобразования текста в цифры – fast text.

Теперь попробуем другой метод преобразования текста в цифры, а именно fast text. Будем проводить исследования для лемматизации. Попробуем обучить fast text на исходных данных и на предобученной модели. Смотреть на результаты данного метода будем не только по классификации, но и по наиболее близким словам по значению, которые выдает fast text к слову “good”

При преобразовании текста в вектора я использовала среднее

Обучение на исходной выборке:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Дообучение (используем предобученную модель)

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Классификация стала хуже: на логистической регрессии значение упал на 3%.

Вывод: Count\_Vectorizer вместе с TFIDF на моей выборке работают лучше, чем fasttext.

**Полный вывод по работе**:

В данной работе я познакомилась с Text Mining: пред обрабатывать текст, преобразовывать сова в цифры. Я провела бинарную классификацию отзывов о фильмах – при этом смогла добиться 89% точности. Лучше всего показал метод Count Vectorize и tfidf. Так же на качество классификации сказалась обработка текста: лемматизация и стемминг. Так же неплохо показал себя метод RFECV – получилось снизить признаковое пространство в 13.3 раза, при этом не потеряв точность. Fast text показал хуже всего.

Чтобы улучшить качество можно попробовать другие методы взвешивания, другие нейросетевые методы