Моделирование настроения новостей

Кармазин Василий ПИН-43
Уманский Александр ПИН-43

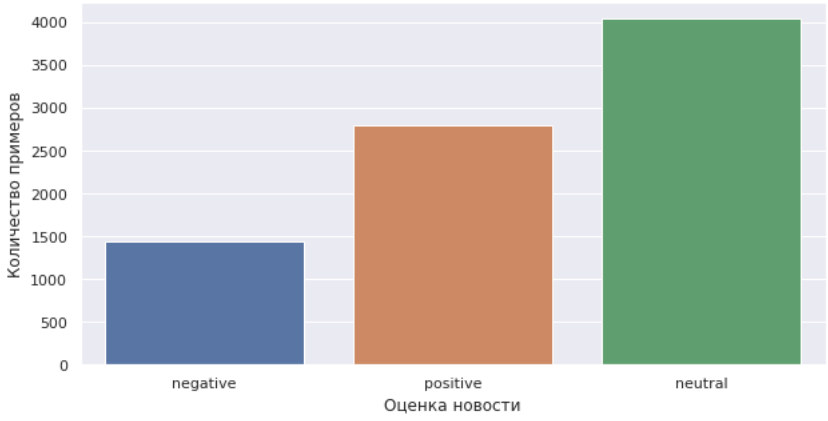
17 Декабря 2019

1 Вступление

Каждый день мы встречаемся с различными новостями. Информация идёт отовсюду: телевидение интернет, радио, **ДОПОЛНИТЬ** Защита детей от хуёвого контента

2 Обработка новостей

Мы использовали данные из открытых источников.[1] Данные включают в себя 8263 различные новостные статьи с тремя различными метками оценки настроения: *Негативные*, *Позитивные*, *Нейтральные*. (Рис. 1).



[1] https://www.kaggle.com/c/sentiment-analysis-in-russian

Рис. 1: Распределение настроения новостей в данных

Откажемся от части информации в данных, которые упростят эксперимент, а также избавимся от выбросов и аномалий.

Наши ограничения:

- Возьмём из новостей только русские слова
- Приведём все слова к нормальной форме через библиотеку pymorphy2[2].
- Не будем использовать новостные статьи, где больше 10000 символов

После данной обработки осталось 7732 статьи.

3 Описание моделей

Мы попробуем два метода:

Наивный, который основывается на построении множеств для каждого класса и взвешивания их частоты встречаемости в статьях.

Статистический, преобразуем наши данные так, чтобы можно было воспользоваться статистическими методами, а именно, логистической регрессией.

Проверять качество моделей будем метрикой F1-score[3]

3.1 Наивный метод

Сначала построим множества слов для каждого класса статей: S_{neg}, S_{pos}, S_{neu} - множества слов из статей с негативной, позитивной и нейтральной меткой, соответственно.

Построим множества уникальных слов для множеств S_{neg} и S_{pos}.

$$W_{neg} = S_{neg}/S_{pos} \text{ и } W_{pos} = S_{pos}/S_{neg}$$

[2]https://pymorphy2.readthedocs.io

[3]https://en.wikipedia.org/wiki/F1-score

Пусть $Freq_y(x)$ - функция, которая определяет частоту встречаемости слова x в множестве y. Тогда для оценки настроения новости воспользуемся простым алгоритмом. (Algorithm 1).

$sentiment = 0$
for $word$ ***in*** $words$ **do**
  if $word$ ***in*** W_{pos} **then**
    $word_{freq} = Freq_{neg}(word)$
    $polarity = 1$
  end
  if $word$ ***in*** W_{pos} **then**
    $word_{freq} = Freq_{pos}(word)$
    $polarity = -1$
  end
  if $word$ ***in*** W_{neu} **then**
    $neutral_{freq} = Freq_{neu}(word)$
    if $word_{freq} > \frac{neutral_{freq}}{2}$ **then**
      $sentiment \mathrel{+}= \frac{word_{freq} * polarity}{2}$
      continue
    end
    if $neutral_{freq} > word_{freq}$ **then**
      continue
    end
    $sentiment \mathrel{+}= word_{freq} * polarity$
  end
end

Algorithm 1: Оценка настроения новостной статьи

Пройдёмся по всем словам статьи, проверяем имеется ли слово в W_{pos} или W_{neg}, если да, то сравниваем частоты встречаемости этого слова относительно *Нейтральных*. Суммируем все слова статьи и получаем общую оценку настроения статьи.

Дальше нормируем оценку на количество слов в тексте и подбираем два порога, которые будут определять нейтральную часть.

3.2 Логистическая регрессия

Преобразуем текст в численные значения, чтобы на них обучить логистическую регрессию.

Будем использовать TF-IDF[4] - это статистическая мера оценки важности слова, основываясь на частотах встречаемости слова в тексте и в документе.

Частота слова t в документе d:

$$tf(t,d) = \frac{n_t}{\sum_k n_k}$$

Обратная частота документа:

$$idf(t,D) = \log \frac{|D|}{|\{\, d_i \in D | t \in d_i \,\}|}$$

TF-IDF является произведением двух сомножителей:

$$tf - idf(t,d,D) = tf(t,d) \times idf(t,D)$$

На самом деле, **TF-IDF** можно применять не только к одиным словам, но и к нескольким, тем самым считая меру важности комбинаций слов, или применять к нескольким символам: униграммам, биграммам, триграммам и тд.

Мы будем использовать все три статистики, но стоит учесть, что это много информации, в том числе и лишней, поэтому мы выберем $N = 30000$ самых часто встречаемых случаев.

Тогда текст статьи представляется в виде вектора $news_{vector} \in \mathcal{R}^N$, где по элементам 0, если в тексте нет слова, и tf-idf(слова), если есть.

На $news_{vector}$ векторах будем строить многоклассовую логистическую регрессию, которую называют Softmax Regression[5].

4 Эксперименты

Наивный метод показал, что он хорошо различает между собой хорошие и плохие новости, а вот нейтральные сильно путаются, что видно на распределении нормированной оценки (Рис. 2).

[4]https://ru.wikipedia.org/wiki/TF-IDF

[5]http://deeplearning.stanford.edu/tutorial/supervised/SoftmaxRegression/

Рис. 2: Распределение нормированной оценки настроения

Результаты множества слов.

- В хороших словах встречаются такие: делегация, экспедиция, упражнение, наставник, транскаспийский и тд.
- В плохих словах: терентьев, оштрафовать, взяточничество, тюремный, вирус, санкционирование и тд.

Подобрали пороговые коэффициенты: если оценка текста выше 0.1, то он позитивный, если ниже -0.05, то негативный, иначе нейтральный.

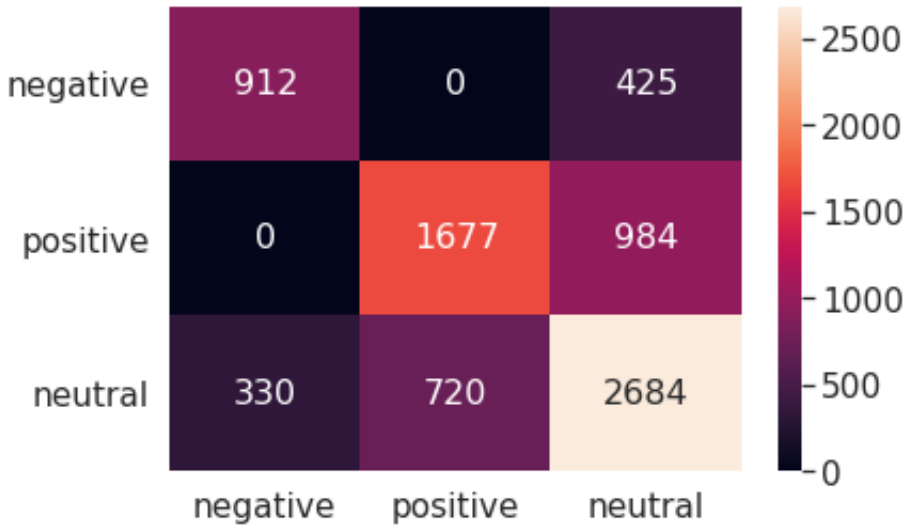


Рис. 3: Матрица ошибок наивной модели

На матрице ошибок наивного метода (Рис. 3). видно, что модель никогда не путает хорошие новости с плохими и наоборот.

Основные метрики по наивной модели

	precision	recall	f1-score	support
negative	0.73	0.68	0.71	1337
neutral	0.66	0.72	0.69	3734
positive	0.70	0.63	0.66	2661

Логистическая регрессия показала лучше результат, в целом, она распознаёт больше статей и меньше ошибается (Рис. 4). В этой модели иногда бывают ошибки между хорошими и плохими новостями, но такие ошибки очень редкие, ими можно пренебречь.

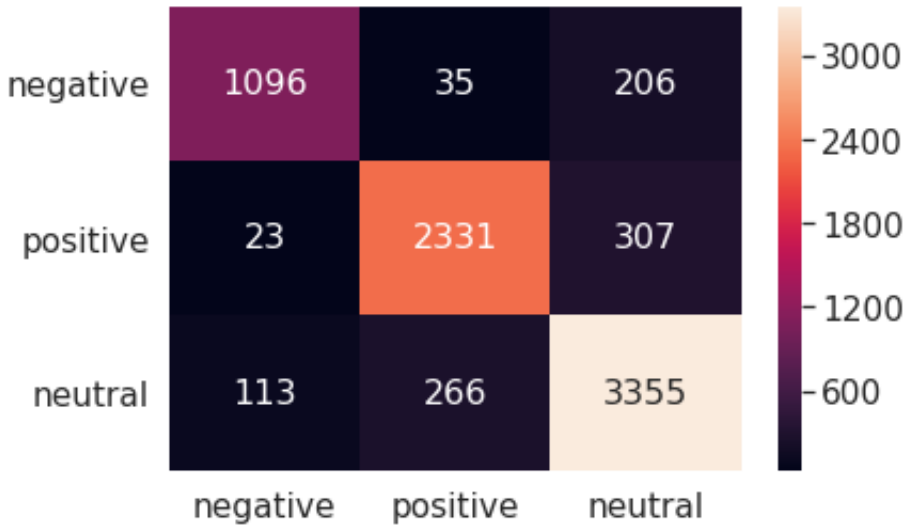


Рис. 4: Матрица ошибок многоклассовой логистической регрессии

По матрице ошибок видно, что модель чаще путает позитивные и негативные классы с нейтральными, чем между самими классами. По метрикам модель имеет сильно лучше результат, чем наивный метод.

Основные метрики по логистической регрессии

	precision	recall	f1-score	support
negative	0.89	0.82	0.85	1337
neutral	0.87	0.90	0.88	3734
positive	0.89	0.88	0.88	2661

Код экспериментов находится в открытом доступе[6]

5 Результаты

Эту хкйню можно выпилить

[6]