РЕФЕРАТ

Выпускная квалификационная работа содержит 36 страниц, 10 рисунка, 3 таблицу, XX использованных источника.

КЛЮЧЕВОЕ СЛОВО 1, КЛЮЧЕВОЕ СЛОВО 2, ...

Краткое описание содержания работы.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Классификация текстов — невероятно популярная задача. Мы пользуемся текстовыми классификаторами в почтовом клиенте: он классифицирует письма и фильтрует спам. Другие приложения включают классификацию документов, обзоров и так далее.

Обычно классификация текстов используется не как самостоятельная задача, а является частью более крупного пайплайна. Например, голосовой помощник классифицирует ваше высказывание, чтобы понять, что вы хотите (например, установить будильник, заказать такси или просто поболтать) и передать ваше сообщение другой модели в зависимости от решения классификатора. Другой пример — поисковый движок: он может использовать классификаторы для определения языка запроса, чтобы предсказать его тип (например, информационный, навигационный, транзакционный) и понять хотите ли вы увидеть картинки или видео помимо документов и прочего.

Моя работа посвящена одному из приложений классификации — автоматическому определению эмоциональной оценки русскоязычных текстов. Главная особенность заключается в том, что предсказание базируется на эмоциональных моделях, предложенных Робертом Плутчиком и Полом Экманом.

Поскольку большинство датасетов для классификации содержат только одну правильную метку. А у нас многоклассовая классификация.

Задачи интеллектуальной обработки текста делятся на три типа: синтаксические, основанные на понимании смысла и третий — не просто понимание написанного, а еще и генерация нового текста. Сентимент-анализ относится ко второму типу.

ОСНОВНАЯ ЧАСТЬ

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Обзор предметной области

основании его содержимого, к конкретному классу из определенного множества. входного текста к одному или нескольким классам, множество классов определено заранее. лексики и эмоциональной оценки авторов. установка.

Сентимент-анализ, как направление компьютерной лингвистики, стал очень популярен последние десятилетие. С появлением больших данных насыщенных эмоциональной составляющей возникла потребность в их обработке. Компании начали устраивать соревнования с внушительными призовыми фондами, а исследователи искать лучшую архитектуру для классификации этих данных. Так в открытом доступе появились большие размеченные датасететы с отзывами, данными из соцсетей и новостями.

Сам термин «сентимент» означает совокупность чувств и взглядов, как основа для действия или суждения; общая эмоциональная установка. Целью сентимент-анализа является выделении этих тональных компонент из текста. Рассмотрим его применение на разных уровнях [9].

Пусть есть целый документ, тогда, как правило, в нем можно выделить один субъект и один объект, а так же сентимент. Ярким примером такого уровня является отзыв. Здесь автор выступает в качестве субъекта, а предмет отзыва — в качестве объекта. Это уровень документа.

Если документ более сложный, то можно рассматривать его на уровне предложений. В результате можно определить эмоциональную окраску всего документа или предложений по отдельности. Зависит от поставленной задачи.

Также анализ бывает на уровне аспектов. Смысл его в том, что эмоциональная установка определяется не для конкретно объекта, а для его отдельных составляющих — аспектов. Например, для объекта «компьютер» можно выделить аспекты «производительность», «дизайн», «сборка» т.д., другими словами, к аспектам относится все то, к чему могут быть выражены эмоции. Данная задача очень востребована, потому что позволяет точнее определять отношение автора к объекту, а для некоторых задач это очень важный критерий.

Последний вид анализа самый сложный и проводится уровне именованных сущностей (Named Entities). Именованная сущность — это абстрактный или физический объект, который может быть обозначен соб-

ственным именем. Сама по себе задача извлечения именованных сущностей (Named Entity Recognition) не из простых, а вкупе с сентимент-анализом становится действительно комплексным решением.

Методы сентимент анализа можно также разделить на несколько основных направлений [9]:

- *метод основанный на правилах (rule-based).* Здесь используются наборы правил классификаци, составленные экспертом и эмоционально размеченные словари. Определенный класс присваивается в зависимости от найденных ключевых слов и их использования с другими ключевыми словами. Такой метод достаточно эффективный, но очень трудоемкий;
- *метод основанный на словарях.* Самый простой метод, основанный на подсчете сентиментных единиц содержащихся в словарях тональности. Очень сильно зависит от размера словаря и не очень точен в разрыве с правилами русского языка. Попытка получения сентимента из текста таким способом описана в части ++ этой работы;
- *методы основанные на машинном и глубоком обучении.* (machine learning, deep learning) Наиболее популярная группа методов в сентимент-анализе, работающих на способности алгоритмов обобщать выделенные из текста признаки;
- *гибридные методы.* Позволяют одновременно использовать несколько подходов. что-нибудь

Разберем подробнее машинное и глубокое обучение. Суть метода в выделении признаков из текста и последующем их обобщении с помощью разнообразных алгоритмов. Для выделения признаков используют, как простые алготитмы по типу мешок слов (Bag of Words) или TF-IDF, так и небольшие нейронные сети для генерирования эмбеддингов, например, Word2Vec ++, GloVe ++, FastText ++. Существуют и более сложные алгоритмы, которые формируют признаки на уровне предложений, к ним относятся ELMo ++, BERT (Bidirectional Encoder Representations from Transformers) ++ и др.

Чтобы обработать выделенные признаки используют разнообразные алгоритмы. К классическому обучению относятся:

- байессовский классификатор (Naive Bayes classifier);
- дерево решений (Decision Tree);
- логистическая регрессия (Logistic Regression);
- метод опорных векторов (Support Vector Machine).

Среди алгоритмов глубокого обучения можно выделить:
- реккурентные нейронные сети (RNN);
- сверточные нейронные сети (CNN);
- полносвязные нейронные сети (FCNN) и т.д.

1.2 Задача классификации и сентимент-анализа текстов

Пусть есть описание документа $d \in X$, где X — векторное пространство документов и фиксированный набор меток $C = \{c_1, c_2, \ldots, c_n\}$. Из обучающей выборки (множества документов с заранее известными метками — эмоциями) $D = \{\langle d, c\rangle | \langle d, c\rangle \in X \times C\}$ и с помощью метода обучения G необходимо получить классифицирующую функцию $G(D) = f$, которая отображает документы в пространство меток $f : X \to C$.

1.3 Данные и предобработка текстов

или

1.4 Представление слов

То, как модели видят данные отличается от того, как из видят люди. Например, мы легко можем понять предложение «Да будет свет», но модели не могут — им нужны векторы с признаками. Такие векторы являются представлениями слов, которые может обработать наша модель.

Самой простой формой представления слов является дискретное представление, т.е. one-hot представление: все слова представляются в виде вектора, размерность которого совпадает с числом слов словаре. Причем все компоненты кроме i-го равны нулю, а позиция, соответствующая i-му слову равна единице. Очевидно, что такой способ не самый лучший. Во-первых такое представление зависит от положения слов в словаре, а это нежелательно, потому что задает бессмысленные отношения между словами. Во-вторых размеры такого словаря растут прямо пропорционально количеству слов в нем, а это значит размерность его может доходить до сотен тысяч и работать с ними станет очень вычислительно накладно. В-третьих такое представление совершенно не учитывает значение слов, для решения этой проблемы обратимся к дистрибутивной семантике.

1.4.1 Дистрибутивная семантика

Чтобы зафиксировать значение слов в их векторах, нам нужно сначала определить понятие значения. Для этого возмем несколько предложений

- возтмем несколько предожений
- построим вектора
- способы из сравнить (косинусное расстояние)

1.4.2 Основная идея архитектуры нейронных сетей

Чтобы объяснить работу неронных сетей нужно опредить из чего они состоят. Для этого рассмотрим простейший линейный бинарный классификатор – перцептрон Розенблата. Пространство данных разделяется на два множества гиперплоскостью, а метка класса будет ставиться в зависимости от значения линейной функции от входных признаков.

$$\text{sign}(w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_d x_d), \tag{1}$$

где $x = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$. Мы ищем такие веса $w_0, w_1 \ldots, w_d \in \mathbb{R}^d$, чтобы sign от скалярного произведения признаков и весов $w^\top x$ совпадал с верной меткой $y(x) \in -1, 1$, но для этого добавим фиктивную переменную в вектор $x = (1, x_1, x_2, \ldots, x_d) \in \mathbb{R}^{d+1}$, чтобы размерности сохранялись.

Теперь обучим эту функцию, для этого нам нужна функция ошибки, она называется критерий Перцептрона:

$$L_P(w) = -\sum_{x \in M} y(x)(w^\top x), \tag{2}$$

где M — множество неверно классифицируемых примеров. В качестве оптимизатора выберем градиентный спуск. С помощью него мы минимизируем суммарное отклонение предсказаний классификатора от верных, но только в неправильную сторону. Верное предсказание никак не влияет на функцию ошибки. В результате получается кусочно-линейная функция, которая почти везде дифференцируема и этого достаточно для применения градиентного спуска. Процесс обучение выглядит так: если предсказание верное, то не делаем ничего, если классификатор ошибся, то делаем градиентный шаг.

Такая модель перцептрона линейная и результат ее работы не слишком содержателен. Чтобы из перцепртонов можно было составить сеть,

нужно добавить нелинейность. Такой нелинейностью будет функция активации. Они бывают разные, самая распространенная — сигмоида рис. 1:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

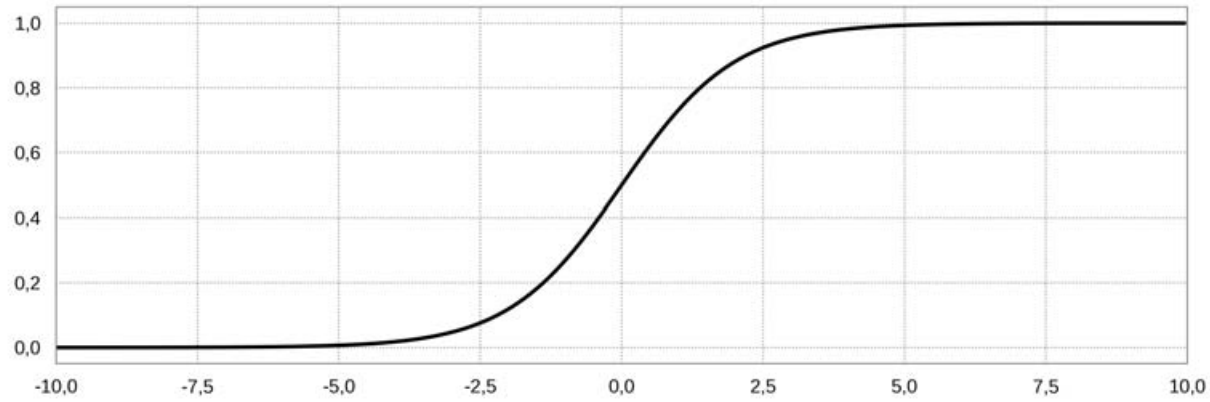

Рис. 1 — Сигмоида

Обучить этот прецептрон также не составляет труда. Просто теперь мы будем решать задачу бинарной классификации, а функция ошибки будет cross-энтропией:

$$L(w) = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log \sigma(w^\top x_i) + (1 - y_i)\log(1 - \sigma(w^\top x_i))). \tag{4}$$

Эта функция дифференцируема, значит мы можем сделать градиентный шаг. При этом прецептрон с сигмоидой реализует логистическую регрессию. Обобщение этой модели можно найти в разделе 1.5.3.

Графическое изображение структуры перцепрона представлено на рис. 2, *а*.

На рис. 2, *б* изображена несложная нейронная сеть, на ее примере покажем как можно векторизовать вычисления в слое нейронов с применением функции активации.

Пусть у нас в слое k нейронов с весами $w_1, w_2, \ldots, w_k$, $w_i = (w_{i1}, \ldots, w_{in})^\top$, на вход вектор $x = (x_1, x_2, \ldots, x_n)^\top$. В результате получим выход $y_i = f(w_i^\top x)$, где f — функция активации. Эти вычисления можно представить в векторной форме:

$$\begin{pmatrix} y_1 \\ \ldots \\ y_k \end{pmatrix} = y = f(Wx) = \begin{pmatrix} f(w_1^\top x) \\ \ldots \\ f(w_k^\top x) \end{pmatrix}, \text{ где } W = \begin{pmatrix} w_1 \\ \ldots \\ w_k \end{pmatrix} = \begin{pmatrix} w_{11} & \ldots & w_{1n} \\ \ldots & & \ldots \\ w_{kk} & \ldots & w_{kn} \end{pmatrix}.$$

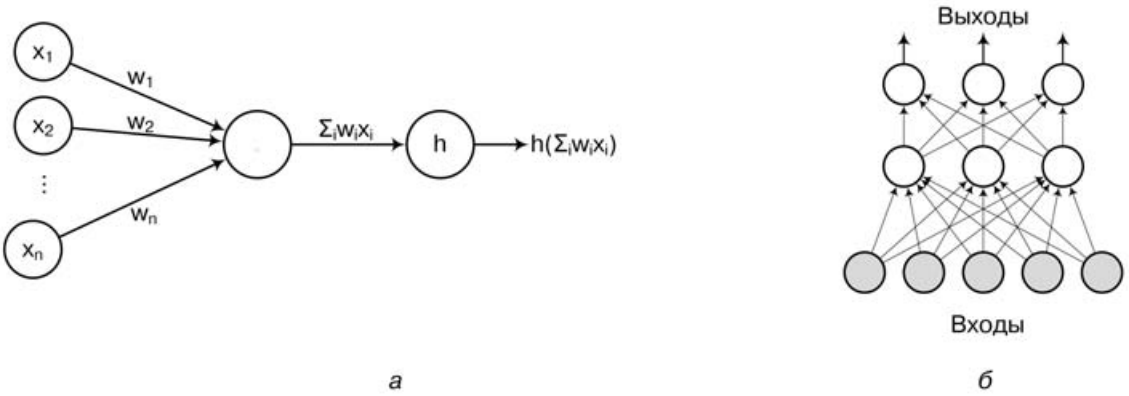

Рис. 2 — (а) граф вычислений перцептрона; (б) полносвязная нейронная сеть с одним скрытым слоем.

Обычно в нелилинейных неронах примемяется сигмоида 3, о которой писалось выше. Но существуют и другие функции активации, например, ReLU $= max(0, x)$. Чтобы понять, как она появилась рассмотрим сумму бесконечного ряда сигмоид, каждая из которых смещена на единицу относительно предыдущей:

$$f(x) = \sigma(x + \frac{1}{2} + \sigma(x - \frac{1}{2} + \sigma(x - \frac{3}{2}) + \dots$$

$f(x)$ можно представить в виде интегала:

$$\int_{\frac{1}{2}}^{\infty} \sigma(x + \frac{1}{2} - y)dy,$$

Его значение можно приюблизить единичными прямоугольниками:

$$\sum_{i=0}^{\infty} \sigma(x + \frac{1}{2} - i) \approx \int_{\frac{1}{2}}^{\infty} \sigma(x + \frac{1}{2} - y)dy = \\ = \Big[-\log(1 + e^{x+\frac{1}{2}-y})\Big]_{y=\frac{1}{2}}^{y=\infty} = \log(1 + e^x),$$

а это в точности интеграл от сигмиоды:

$$\int \sigma(x)dx = \log(1 + e^x) + C.$$

Получается бесконечный ряд сигмоид гораздо более выразительная функция активации и это почти тоже самое что и $\log(1 + e^x)$

смещением на $1/2$ и тд. Это дает возможность отличать сильно активированные нейроны от слабо активированных. Но считать производную такой функции очень накладно, поэтому ее приближение – хороший компромисс. Существуют еще разные модификации ReLU, например, PReLU или LReLU. В целом она показала лучшую эффективность по сравнению

с сигмоидой и tanh, следовательно используется повсеместно. где классов больше 2-х. Ее удобно использовать на последних слоях нейронной сети, чтобы преобразовать выход в «вероятность».

1.4.3 Распределенные представления слов word2vec

Подход к обучению моделей распределенных представлений слов был описан в работе Йошуа Бенджи с соавторами [2], которая была продолжена в [8]. Идея подхода описанного в [2] основанна на задаче построения языковой модели, процесс обучения выглядит так:

- всем токенам из словаря $i \in V$ ставят в соответствие вектор признаков w_i размерности d ($w_i \in \mathbb{R}^d$). Стандартным значением d является 300;
- теперь можно определить вероятности для каждого токена i, что он появится в контексте $c_1, \ldots, c_n$. Для этого определим функцию от векторов признаков w:

$$\hat{p}(i|c_1, \ldots, c_n) = f(w_i, w_{c_1}, \ldots, w_{c_n}; \theta), \tag{5}$$

  где $w_{c_1}, \ldots, w_{c_n}$ — векторы признаков токенов из контекста, а f — фунция с параметрами θ, которая принимает векторы признаков;
- максимизируя логарифм правдоподобия большого корпуса текстов можно обучить векторы признаков w и параметры θ

$$L(W, \theta) = \frac{1}{K} \sum_t \log f(w_k, w_{k-1}, \ldots, w_{k-n+1}; \theta) + R(W, \theta), \tag{6}$$

  где K размер окна контекста, а $R(W, \theta)$ — регуляризация.

Для получения функции f можно использовать нейронную сеть. Модель word2vec строится на описаниif нейросетевой модели, предложенной в [2]. Она была разработана Томасом Миколовым с соавторами и опубликована в работах [6, 5], причем в двух вариациях:

- CBOW (Continious Bag Of Words) — по контексту восстановить слово;
- skip-gram — восстановить контекст в зависимости от слова;

Архитектура word2vec представляет собой полносвязную нейронную сеть с одним скрытым слоем рис. 3.

Принцип работы модели CBOW рис. 3, *а* выглядит так:

- на вход сети подаются one-hot вектора размерности V, где V — это размер словаря;

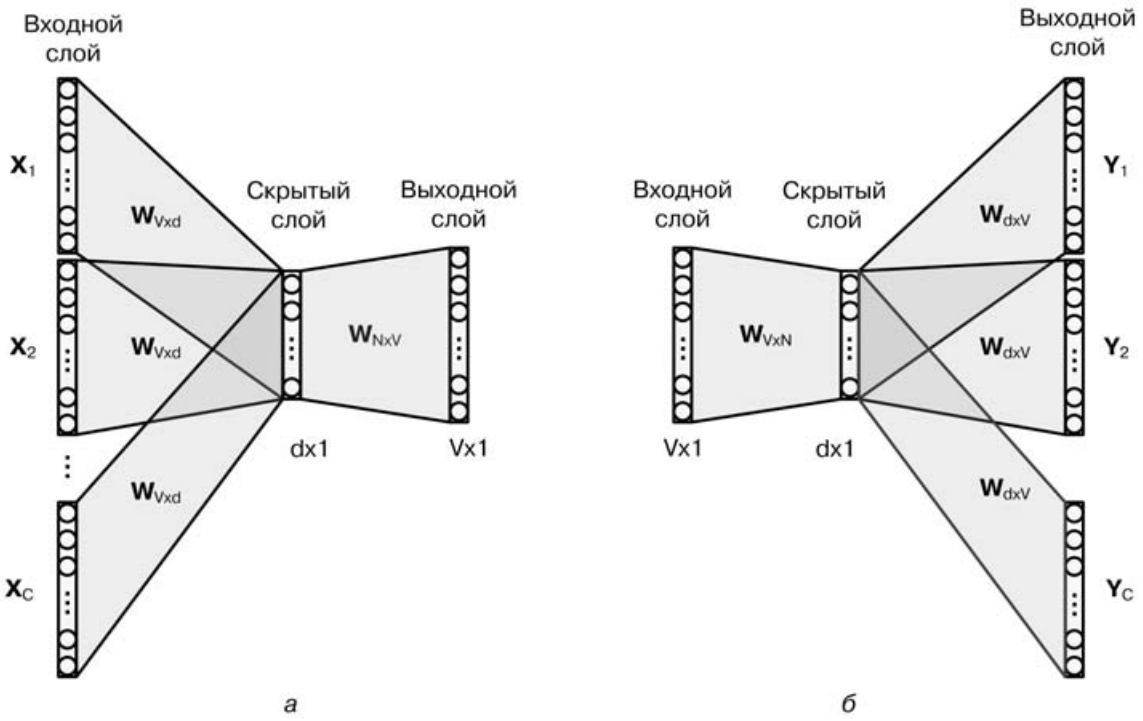

Рис. 3 — (а) CBOW; (б) skip-gram.

- скрытый слой — это матрица W размерности $V \times d$, которая переводит наши представления слов в d - мерное пространство;
- на выходе для каждого слова в словаре берем среднее всех полученных векторов и получаем оценку u_j, где $j = 1, \ldots, V$.

Чтобы найти апостериорное распределение модели, просто вычисляем softmax:

$$\hat{p}(i|c_1, \ldots, c_n) = \frac{\exp u_j}{\sum_{j'=1}^{V} \exp u_{j'}}. \tag{7}$$

Для аппроксимации апостериорным распределением распределения данных используем loss-функцию для одного окна:

$$L = -\log p(i|c_1, \ldots, c_n) = -u_j + \log \sum_{j'=1}^{V} \exp u_{j'}. \tag{8}$$

Принцип работы модели skip-gram рис. 3, *б* полностью противоположный. До этого мы усредняли контекст, чтобы получить среднее слово в окне, а теперь будем предсказывать слова контекста исходя из центрального слова. На выходе мы получаем $K-1$ мультиномиальных распределений (центральное слово не учитывается):

$$\hat{p}(c_k|i) = \frac{\exp u_{kc_k}}{\sum_{j'=1}^{V} \exp u_{j'}}, \tag{9}$$

loss-функция для окна размера K выглядит так:

$$L = -\log p(c_1, \ldots, c_n|i) = -\sum_{k=1}^{K} u_{kc_k} + n \log \sum_{j'=1}^{V} \exp u_{j'}. \tag{10}$$

Возникает вопрос, как же обучить такую модель? Этот процесс хорошо описан в докладе Голдберга и его соавторов [7].

Подробно разберем модель skip-gram для корпуса документов D. Нашей задачей стоит нахождение оптимальных параметров модели θ, чтобы максимизировать функцию правдоподобия:

$$L(\theta) = \prod_{i \in D} \left(\prod_{c \in C(i)} p(c|i;\theta) \right) = \prod_{(i,c) \in D} p(c|i;\theta), \tag{11}$$

где $C(i)$ — множество контекстных слов внутри окна вокруг центрального слова i. Вероятность $p(c|i;\theta)$ определяется, как softmax-функция, зависящая от всех возможных векторов контекста.

$$p(c|i;\theta) = \frac{\exp \tilde{w}_c^\top w_i}{\sum_{c'} \exp \tilde{w}_{c'}^\top w_i}, \tag{12}$$

где $\tilde{w}_c$ — вектор признаков слова из контекста c, который отличается от w_i. Для каждого слова i надо обучить два вектора признаков w_i и $\tilde{w}_i$, в первом случае это слова выступает в качестве центрального, во втором в качестве контекстного.

Эта особенность обучения, когда мы берем два разных вектора одного и того же слова вместо одного, описана в [7]. И мотивирована тем, что слова редко встречаются в контексте себя самих. Вот, например, слово «мотивация» вряд ли можно встретить в контексте другого слова «мотивация», под это правило попадают почти все слова. Поэтому в процессе обучения модель сведет вероятности $p(i|i,\theta)$ к нулю. А если вектора контекста и центрального слова будут равны нулю, то норма вектора $|w_i| = w_i^\top w_i$ тоже будет равняться нулю, а это очень не желательно. Поэтому для каждого слова мы обучаем два разных вектора.

Теперь выразим максимум функции правдоподобия для всего корпуса через логарифм 11 и 12:

$$\begin{aligned} \arg\max_\theta \prod_{(i,c) \in D} p(c|i;\theta) &= \arg\max_\theta \sum_{(i,c) \in D} \log p(c|i;\theta) = \\ &= \arg\max_\theta \sum_{(i,c) \in D} \left(\exp \tilde{w}_c^\top w_i - \log \sum_{c'} \exp \tilde{w}_{c'}^\top w_i \right). \end{aligned} \tag{13}$$

Оптимизируя данную функцию мы получаем хорошее распределенное представление слов. Но для этого нужно решить сложнейшую задачу: суммировать скалярные произведения всех возможных слов и их контекста $\sum_{c'} \tilde{w}_c^\top w_i$ при том, что размер словаря может достигать миллионов.

Чтобы уменьшить количество вычислений Миколов с соавторами [5] предложили элегантный метод: negative sampling. Нам не нужно считать всю сумму $\sum_{c'} \tilde{w}_c^\top w_i$, а только случайно выбрать несколько ее элементов в качестве отрицательных примеров (примеры в которых слово не находится в определенном контексте) и обновить только их. Т.е. теперь нам нужно посчитать только небольшую сумму $\sum_{c' \in D'} \tilde{w}_c^\top w_i$, где D' — случайное подмножество отрицательных примеров.

По сути negative sampling — это тоже правдоподобие, но другого события. Пусть у нас есть слово i и его контекст c, наша задача максимизировать вероятность $p((i,c) \in D; \theta)$, параметризированную вектором θ, т.е. правдоподобие появления пары (i,c):

$$\arg\max_\theta \prod_{(i,c)\in D} p((i,c) \in D; \theta) = \arg\max_\theta \sum_{(i,c)\in D} \log p((i,c) \in D; \theta). \tag{14}$$

Выразим $p((i,c) \in D; \theta)$ через softmax. Но так как это бинарное событие, то заменим softmax сигмоидой $\sigma(x) = \frac{1}{1+\exp(-x)}$:

$$p((i,c) \in D; \theta) = \frac{1}{1 + \exp(-\tilde{w}_c^\top w_i)} \tag{15}$$

Максимизируем логарифм правдоподобия:

$$\begin{aligned} \arg\max_\theta \sum_{(i,c)\in D} \log p((i,c) \in D; \theta) = \\ = \arg\max_\theta \sum_{(i,c)\in D} \log \frac{1}{1 + \exp(-\tilde{w}_c^\top w_i)}. \end{aligned} \tag{16}$$

Из 16 видно, что оптимальное значение логарифма будет получено при максимальном значении скалярного произведения $\tilde{w}_c^\top w_i$. Сделаем равные векторы с большой нормой и можно без проблем получить правдоподобие почти равное единице. Подвох заключается в том, что модель обучается на данных для бинарной классификации, но мы рассматриваем только набор состоящий из положительных примеров. Классификатор, который всегда предсказывает «да» — плохой. Поэтому имеет смысл добавить отрицательных примеров, просто случайно выбирая слова и контекст, которых нет в данных. После того, как мы получим набор отрицательных данных, максимизация правдоподобия будет выглядеть так:

$$\arg\max_{\theta} \prod_{(i,c)\in D} p((i,c) \in D;\theta) \prod_{(i,c)\in D'} p((i',c') \notin D;\theta) \tag{17}$$

Выразим в 17 пару $(i,c) \in D$:

$$\begin{aligned}
&\arg\max_{\theta} \prod_{(i,c)\in D} p((i,c) \in D;\theta) \prod_{(i,c)\in D'} 1 - p((i',c') \in D;\theta) = \\
&= \arg\max_{\theta} \left[\sum_{(i,c)\in D} \log p((i,c) \in D;\theta) + \sum_{(i,c)\in D'} \log\left(1 - p((i',c') \in D;\theta)\right) \right] = \\
&= \arg\max_{\theta} \sum_{(i,c)\in D} \left[\log \frac{1}{1+\exp\left(-\tilde{w}_c^\top w_i\right)} + \sum_{(i,c')\in D'} \log \frac{1}{1+\exp\left(\tilde{w}_{c'}^\top w_i\right)} \right] = \\
&= \arg\max_{\theta} \sum_{(i,c)\in D} \left[\log \sigma(\tilde{w}_c^\top w_i) + \sum_{(i,c')\in D'} \log \sigma(-\tilde{w}_{c'}^\top w_i) \right]
\end{aligned} \tag{18}$$

Получили формулу для negative sampling из [5]. Значит мы для каждого окна случайно берем несколько отрицательных примеров D' и делаем градиентный шаг для loss-функции:

$$L = \log \sigma(\tilde{w}_c^\top w_i) \sum_{(i,c')\in D'} \log \sigma(-\tilde{w}_{c'}^\top w_i) \tag{19}$$

Аналогичные рассуждения можно провести для модели CBOW.

1.4.4 ELMo

Создание этой модели породило новую эру распределенных представлений слов. Word2vec показал, что с помощьювектор (набор чисел), чтобы представлять слова в виде, который может передавать их семантику или смысловые отношения (т.е. способность различать схожие и противоположные по смыслу слова или находить параллели в отношениях таких словарных пар, как «Стокгольм» – «Швеция» и «Каир» – «Египет»), а также синтаксические или грамматические отношения (например, определять, что отношение между «имел» и «иметь» такое же, как между «был» и «быть»).

Сообщество быстро осознало, что лучше всего использовать эмбеддинги, предобученные на больших объемах текста, чем обучать их вместе с моделью на зачастую достаточно маленьком наборе данных. Стало возможным загружать списки слов и их эмбеддингов, созданных с помощью

предварительного обучения Word2Vec или GloVe. Так выглядит пример эмбеддинга в GloVe для слова «stick» (где размерность эмбеддинга – 200):

1.5 Классификация

1.5.1 Анализ качества классификации

Для оценки качества классификации используется Macro F1-мера — нормализованное по всем классам среднее гарманическое метрик P и R:

$$\text{Macro F1} = \frac{1}{n}\sum_{i=1}^{n} 2\frac{P_i \times R_i}{P_i + R_i} = \frac{1}{n}\sum_{i=1}^{n} \text{F1}_i,$$

где P — точность и R — полнота.

Ее преймущество в том, что она не учитывает дисбаланс классов.

Определение 1. Точностсь (*precision*) — показывает долю объектов, которые классификатор определил, как принадлежащие действительно правильному классу.

$$P = \frac{TP}{TP + FP},$$

где TP — те метки которые мы ожидали и получили, FP — те метки которые мы не ожидали, но получили на выходе.

Определение 2. Полнота (recall) — показывает какую долю объектов из всего множества конкретного класса калассификатор определил верно.

$$R = \frac{TP}{TP + FN},$$

где FN — те метки которые мы ожидали, но не получили на выходе.

1.5.2 Кросс-валидация

Методом оценки аналитической модели и её поведения на независимых данных является перекрестная проверка k-fold cross validation, при k = 10.

1.5.3 Логистическая регрессия

Логистическая регрессия — классическая дискриминативной линейная модель классификации. Дискриминативная значит, что нас интересует $P(y = k|x)$, а не совместное распределение $p(x, y)$. Свое начало она берет из расстояния Кульбака-Лейблера. Оно задается формулой:

$$KL(P||Q) = \int \log \frac{dP}{dQ} dP, \tag{20}$$

, где P — истинное распределение, а Q — приближенное. Для дискретного случая:

$$KL(P||Q) = \sum_y p(y) log \frac{p(y)}{q(y)}, \tag{21}$$

а если раскрыть получаем:

$$\begin{aligned} KL(P||Q) &= \sum_y p(y) \log \frac{p(y)}{q(y)} = \\ &= \sum_y p(y) \log p(y) - \sum_y p(y) \log q(y) = -H(p) + H(p, q), \end{aligned} \tag{22}$$

, где $H(p)$ — энтропия распределения p, а $H(p, q)$ — наша кросс энтропия. Из этой суммы видно, что нам нужно минимизировать $H(p, q)$. Для бинарной классификации loss-функция будет выглядеть так:

$$L(w) = H(p_{data}, q(w)) = -\frac{1}{N} \sum_{i=1}^{N} (y_i \log \hat{y}_i(w) + (1 - y_i) \log(1 - \hat{y}_i(w))) \tag{23}$$

где p_{data} — распределение наших данных, $q(w)$ — апостериорное распределение, $\hat{y}_i(w)$ — оценка вероятности при входных параметрах w и y_i — истинное предсказание. Два слагаемых мы получаем, т.к. события несовместные. Например, в тексте говорится о кошечках или о собачках, события появления кошечки или собачки несовместные, т.е. $p(\text{кошечки}) = 1 - p(\text{собачки})$. Если мы предсказываем кошечку (1), как абсолютный 0 или собачку (0), как 1, то ошибка будет бесконечной из первого и второго слагаемых соответственно – это не допустимо.

Перед тем, как перейти к нескольким классам, рассмотрим сначала задачу классификации с Байейсовской точки зрения: определим для каждого класса C_k плотность $p(x|C_k)$ и какие-то априорные распределения

$p(C_k)$ (пускай это будут размеры классов, т.е. мы ничего не знаем о примере, но предполагаем с какой вероятностью он относится к конкретному классу) и найдем $p(C_k|x)$. Для двух классов:

$$\begin{aligned} p(C_1|x) &= \frac{p(x|C_1)p(C_1)}{p(x|C_1)p(C_1) + p(x|C_2)p(C_2)} = \\ &= 1/\frac{(p(x|C_1)p(C_1) + p(x|C_2)p(C_2))}{p(x|C_1)p(C_1)} = \\ &= 1/(1 + \frac{p(x|C_2)p(C_2)}{p(x|C_1)p(C_1)}) = \frac{1}{1+e^{-a}} = \sigma(a), \end{aligned} \tag{24}$$

где

$$a = \ln \frac{p(x|C_1)p(C_1)}{p(x|C_2)p(C_2)}, \qquad \frac{1}{1+e^{-a}} = \sigma(a). \tag{25}$$

Используя логистическую регрессию мы делаем предположение о виде аргумента сигдмоиды a — это будет скалярное произведение вектора признаков на вектор данных: $a = w_\top x$. Сигмоида переводит результат вычисления этой линейной функции на отрезок $[0; 1]$ и как результат мы получаем апостериорную вероятность первого или второго классов:

$$p(C_1|x) = y(x) = \sigma(w_\top x), \qquad p(C_2|x) = 1 - p(C_1|x), \tag{26}$$

чтобы обучить эту модель мы можем просто оптимизировать правдоподобие по w.

Для набора x_n, t_n, где x_n – входы, а $t_n \in \{0; 1\}$ – соответствующие метки классов, получается такое правдоподобие:

$$p(t|w) = \prod_{n=1}^{N} y_n^{t_n}(1 - y_n)^{1-t_n}, \quad \text{где} \quad y_n = p(C_1|x_n). \tag{27}$$

И теперь мы, максимизируя логарифм вероятности, ищем наилучшие параметры функции правдоподобия для этого можно использовать различные оптимизаторы.

$$\text{Likelihood}(w) = -\ln p(t|w) = -\sum_{n=1}^{N}[t_n \ln y_n + (1 - t_n)\ln(1 - y_n)]. \tag{28}$$

Теперь можно легко обобщить задачу на несколько классов. Только вместо сигмоиды будем использовать softmax функцию. Для K классов получаем:

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{\sum_{j=1}^{K} p(x|C_j)p(C_j)} = \frac{e^{a_k}}{\sum_{j=1}^{K} e^{a_j}}, \tag{29}$$

где количество аргументов $a_k = \ln p(x|C_k)p(C_k)$ равняется количеству классов. Функция правдоподобия почти не изменилась. Пусть на вход метки класса подаются в формате one-hot векторов, тогда для набора векторов $T = t_n$ функция правдоподобия выглядит следующим образом:

$$p(T|w_1, \ldots, w_K) = \prod_{n=1}^{N} \prod_{k=1}^{K} p(C_k|x_n)^{t_{nk}} = \prod_{n=1}^{N} \prod_{k=1}^{K} y_{nk}^{t_{nk}}. \tag{30}$$

, где $y_{nk} = y_k(x_n)$. Опять переходим к логарифму и получаем функцию максимального правдоподобия для K классов:

$$\text{Likelihood}(w_1, \ldots, w_K) = -\ln p(T|w_1, \ldots, w_K) = \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \ln y_{nk}. \tag{31}$$

1.5.4 Метод опорных ввекторов

1.5.5 Случайный лес

2 ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Используемые инструменты

- Python 3
- NumPy
- Pandas
- Matplotlib
- Pymorphy2
- Razdel
- TensorFlow
- Scikit-learn
- JavaScript
- HTML и CSS

2.2 Сбор данных

Самым главным этапом перед созданием моделей-классификаторов является сбор данных, этот процесс включает в себя несколько основных этапов:

- обработка и подготовка текстов;
- разметка текстов;
- обработка полученных результатов.

2.2.1 Обработка и подготовка текстов

В качестве основного текста для разметки был взят роман Михаила Афанасьевича Булгакова «Мастер и Маргарита».

Обработка документа:

а) произведение было очищено от нежелательных подстрок регулярными выражениями;

б) разделено на тексты по символу перевода строки «\n»;

в) из полученных текстов восстановлена прямая речь;

г) тексты содержащие больше 52 слов разделены с использованием библиотеки «razdel».

Формирование заданий:

- для разметки выделены тексты от 5 до 52 слов;
- для каждого текста определен контекст: не менее 40 слов перед и не менее 15 после текста.

2.2.2 Разметка текстов

Чтобы приступить к разметке сначала нужно определить множество меток классов, для этого обратимся к истории создания эмоциональных моделей ведушими профессорами в области изучения эмоций. В 1980 году Роберт Плутчик в своей работе [1] определил колесо эмоций рис. 4. Данная модель была взята за основу и дополнена моделью Пола Экмана, которую он описал в работе [3] 2004 года и обновил в статье [4] 2011 года.

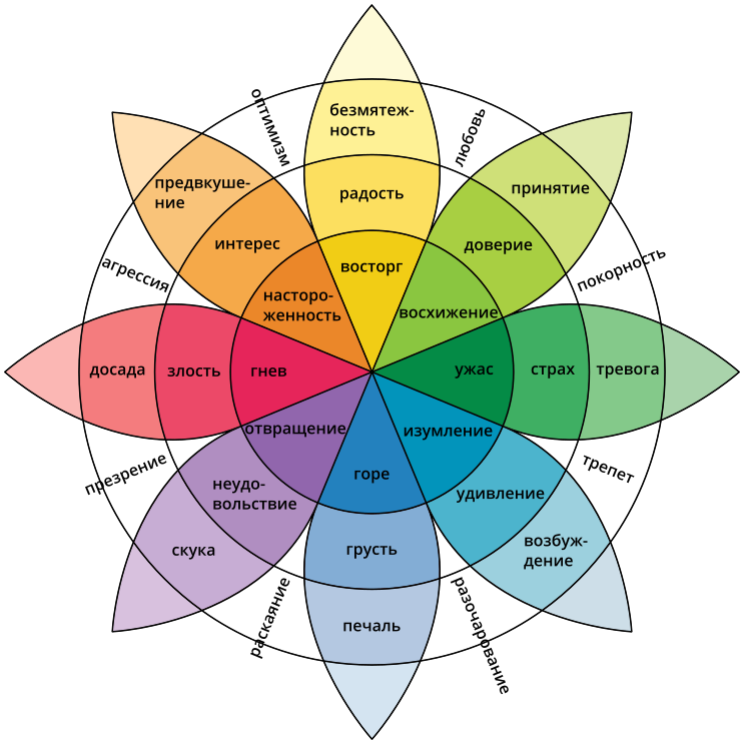

Рис. 4 — Колесо эмоции Роберта Плутчика

В результате получилось множество, состоящее из 9 основных эмоций и их производных, дополненное нейтральным классом:

- **Злость** (anger) — желание выразить агрессию или причинить зло, общая для обеих моделей.
  Может проявляться в словах, мимике, поступках. Примеры: злость

на оскорбление, на несправедливость, злость на плохое отношение.
Гнев — более интенсивная эмоция, досада — менее.

- **Интерес** (anticipation) — предчувствие важного события, только в модели Плутчика.
  Проявляется в нетерпении, волнении. Примеры: ожидание праздника, ожидание начала каникул, ожидание плохой оценки.
  Настороженность — более интенсивная, предвкушение — менее.
- **Радость** (joy) — чувство удовольствия, весёлого настроения и счастья, общая для обеих моделей.
  Проявляется в смехе, улыбке, ласковом обращении к другим. Примеры: радость по поводу подарка, общения с другом.
  Восторг — более интенсивная, безмятежность — менее.
- **Доверие** (trust) — открытое теплое отношение к чему бы то ни было (другу/животному/миру/...), только в модели Плутчика.
  Проявляется в уверенности в положительном исходе. Примеры: доверие другу при встрече с неожиданностями, доверие к собаке, что не укусит, доверие к доктору, что он делает полезные вещи.
  Восхищение — более интенсивная, принятие — менее.
- **Страх** (fear) — состояние перед реальным или предполагаемым бедствием, общая для обеих моделей.
  Проявляется в волнении, напряжении. Пример: страх наказания, страх проигрыша, страх попасть в аварию.
  Ужас — более интенсивная, тревога — менее.
- **Удивление** (surprise) — эмоциональная реакция на неожиданную ситуацию, общая для обеих моделей.
  Удивление может проявляться в хороших и плохих ситуациях. Примеры: получил плохой отзыв на работу вместо ожидаемого хорошего, директор школы привел в класс собаку, одноклассник вырос на 10 см за лето.
  Изумление — более интенсивная, возбуждение — менее.
- **Грусть** (sadness) — отсутствие радости, неудовлетворенность происходящим, отстраненность, общая для обеих моделей.
  Проявляется в нежелании веселиться с другими, желании заботы и участия. Примеры: мама уехала в командировку надолго, не покупают собаку или велосипед, никак не дается математика.
  Горе — более интенсивная, печаль — менее.
- **Неудовольствие** (disgust) — эмоциональная реакция на неприятную ситуацию или объект.
  Проявляется в неприятии человека, любых вещей, ситуаций, общая для обеих моделей. Примеры: когда сталкиваешься с неприятным запахом, грязными вещами, плохим поведением.

Отвращение — более интенсивная, скука — менее.

- **Презрение** (contempt) — пренебрежительное отношение к кому-чему-нибудь морально низкому, недостойному, подлому. Презрение связано с чувством превосходства. Также оно может перейти в безразличное отношение к кому-чему-то. Только в модели Экмана.
- **Нейтральное** (neutral) — безэмоциональное повествование.

Разметка осуществлялась с помощью краудсорсинговой платформы «Яндекс.Толока».

Определение 3. Краудсорсинг — это привлечение добровольцев и экспетртов для выполнения определенной работы, действующих на добровольной или комерческой основе.

Для получения более точной разметки данных были использованы встроенные методы и инструменты контроля качества:

- график времени выполнения страницы заданий рис. 5 нужен, чтобы видеть на сколько вдумчиво эксперт расставляет метки;
- график выполнения заданий рис. 6 показывает сколько выполнено страниц заданий, сколько просрочено и сколько пропущено. По нему можно судить о сложности выполнения задания и использовать эту информацию в процессе формирования новых пулов заданий;
- сформирована система правил, которая позволяет контролировать процесс разметки в автономном режиме:
  - Если пропущенных подряд страниц заданий $\geqslant 10$, то заблокировать на проекте на 2 дня;
  - Если отправленных страниц заданий $\geqslant 50$, то заблокировать на проекте на 2 дня;
  - Минимальное время на страницу заданий — 250 сек. Учитывать последних страниц заданий — 15. Если количество ответов $\geqslant 5$ и количество быстрых ответов $\geqslant 5$, то заблокировать на проекте на 2 дня и т.д.
- размечены контрольные задания, с их помощью можно отслеживать примерную точность (accuracy), как всего набора данных, так и набора, полученного от одного эксперта;
- каждое задание выполняло три различных эксперта (перекрытие x3);
- агрегация результатов производилась методом Дэвида-Скина. Он автоматически оценивает для каждого исполнителя $|L|^2$ параметров, где L — множество возможных различных значений для агрегации и возвращает итоговый ответ и его статистическую значимость.

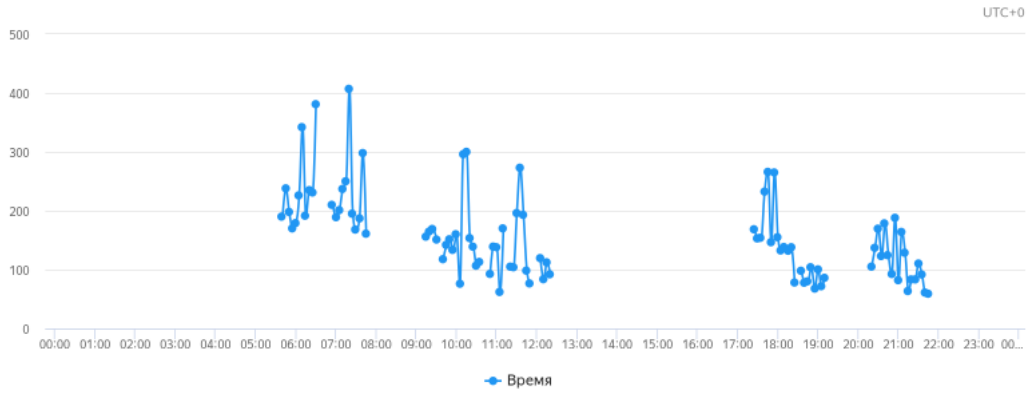

Рис. 5 — Время выполнения страницы заданий (детализация по 5 минут)

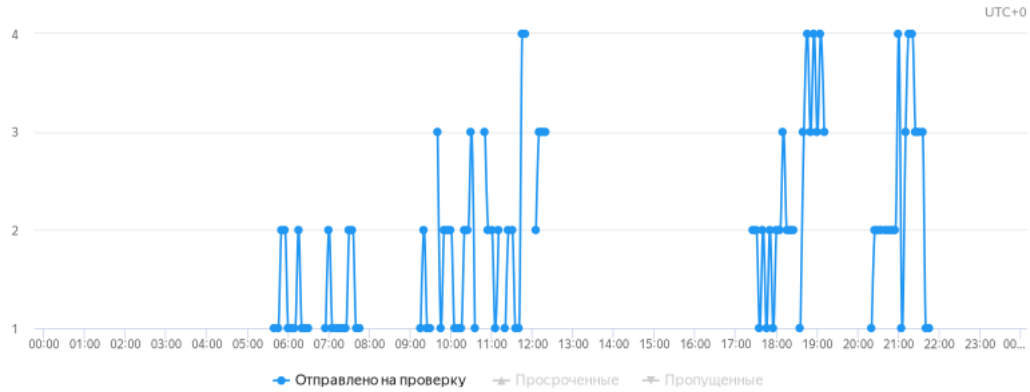

Рис. 6 — Выполнение страниц заданий (детализация по 5 минут)

Была разработана форма задания рис. 7. Каждое задание состоит из трех текстовых блоков. Страница заданий состоит из двух не размеченных заданий и одного контрольного, такое разбиение оптимально для получения качественных результатов. Эксперт должен прочитать каждый текстовый блок и отметить эмоции, которые, по его мнению, описаны в выделенном фрагменте.

В результате был сформирован набор данных с таким распределением классов рис. 8.

2.3 Модели классификации

2.3.1 Предобработка текстов

Чтобы работать с текстами, сначала их нужно нормализовать. Этот процесс включает несколько этапов.

а) приводим текст к нижнему регистру;

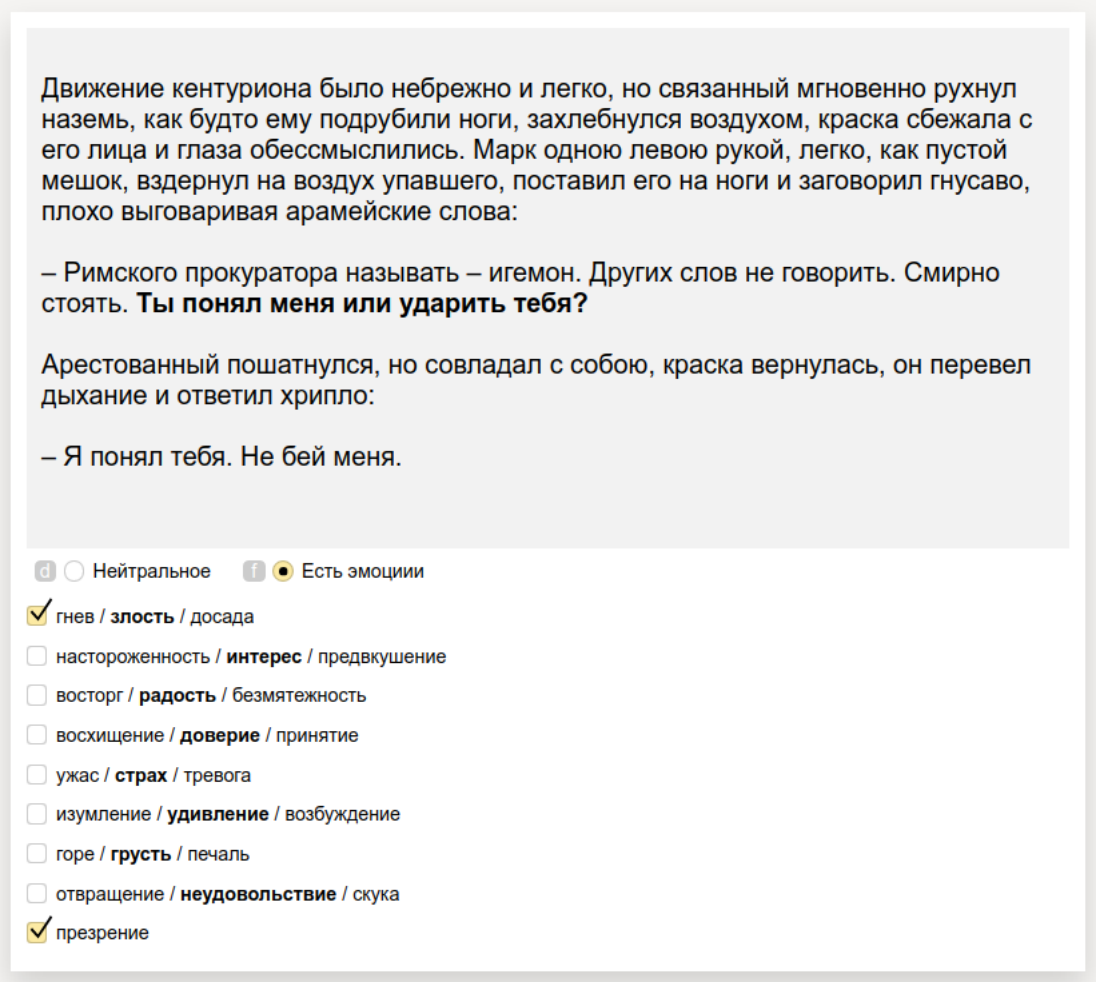

Рис. 7 — Форма задания в сервисе «Яндекс.Толока»

б) удаляем все «не слова» и «стоп-слова»;

в) лемматизируем текст.

Вот пример обработки небольшого текста:

Квартира простояла пустой и запечатанной только неделю. → квартира простаивать пустой запечатывать неделя

2.3.2 Представление предложений

Теперь текст нужно перевести в векторное пространство R^n, где n — размерность признаково пространства используемой модели. Пусть текст D состоит из слов $d \in D$, f — модель, строящая отображение пространства слов в векторное пространство действительных чисел $f(d) \to R^n$. Тогда текст для классификатора выглядит так:

$$\frac{1}{\#D} \sum_{d \in D} f(d) \in R^n$$

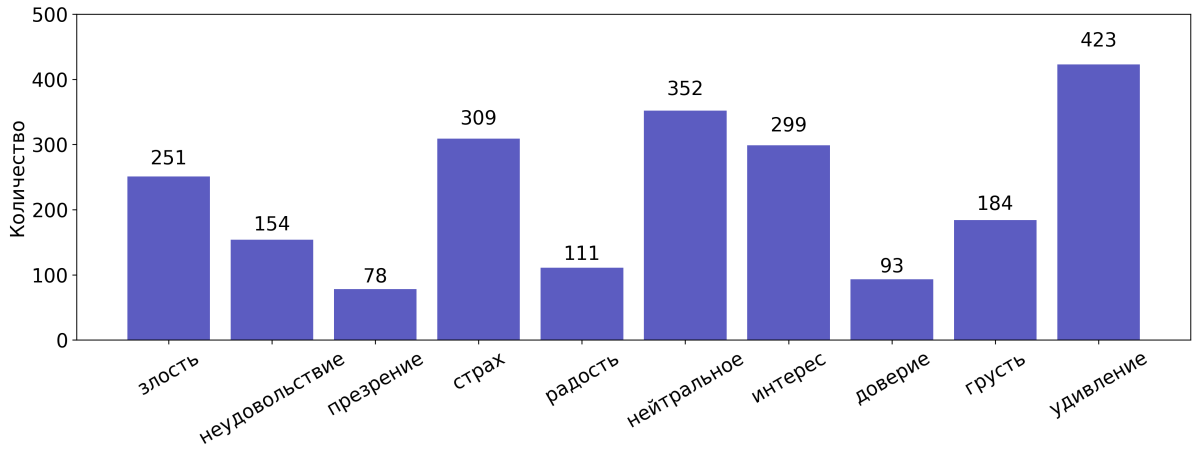

Рис. 8 — Распределение классов в итоговом наборе данных

В этой работе использованы модели предобученные на корпусе русскоязычных текстов «Тайга»:

- word2vec & skip-gram: *tayga_upos_skipgram_300_2_2019* ($n = 300$);
- ELMo: *tayga_lemmas_elmo_2048_2019* ($n = 2048$).

Особенность применения ELMo заключается в том, что берется среднее значение всех слоев для каждого слова.

2.4 Архитектура моделей классификации

Для классификации были выбраны алгоритмы классического машинного обучения:

- случайный лес (Random Forest);
- логистическая регрессия (Logistic Regression);
- метод опорных векторов (Support Vector Machine).

Схематично модели классификации представлены на рис. 9.

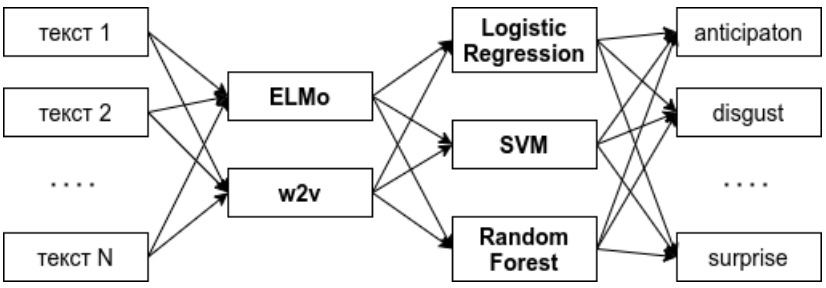

Рис. 9 — Архитектура моделей классификации

2.5 Эксперименты

Для эмоциональной модели Роберта Плутчика результаты получились следующие:

Таблица 1 — Значения Macro F1 меры

Macro F1	**log reg**	**SVM**	**random forest**	**tuned random forest**
w2v	0.21519548	0.27992996	0.23415391	0.23932876
ELMO	0.25900211	0.32828541	0.31197309	0.43861588

Для эмоциональной модели Пола Экмана:

Таблица 2 — Значения Macro F1 меры

Macro F1	**log reg**	**SVM**	**random forest**	**tuned random forest**
w2v	0.25493596	0.18245115	0.21223037	0.37710993
ELMO	0.20242784	0.3007757	0.34183484	0.42405119

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *PLUTCHIK R.* a General Psychoevolutionary Theory of Emotion // Theories of Emotion. — Elsevier, 1980. — С. 3—33. — DOI: 10.1016/b978-0-12-558701-3.50007-7.
2. *Bengio Y.*, *Ducharme R.*, *Vincent P.* A neural probabilistic language model : тех. отч. — 2001. — С. 1137—1155.
3. *Ekman P.* Emotions revealed. Т. 328. — 2004. — С. 0405184. — ISBN 0805072756. — DOI: 10.1136/sbmj.0405184.
4. *Ekman P.*, *Cordaro D.* What is meant by calling emotions basic // Emotion Review. — 2011. — Т. 3, № 4. — С. 364—370. — ISSN 17540739. — DOI: 10.1177/1754073911410740.
5. Distributed representations of words and phrases and their compositionality / T. Mikolov [и др.] // Advances in Neural Information Processing Systems. — 2013. — Окт. — ISSN 10495258. — arXiv: 1310.4546. — URL: http://arxiv.org/abs/1310.4546.
6. Efficient estimation of word representations in vector space / T. Mikolov [и др.] // 1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings. — International Conference on Learning Representations, ICLR, 01.2013. — arXiv: 1301.3781. — URL: http://ronan.collobert.com/senna/.
7. *Goldberg Y.*, *Levy O.* word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method : тех. отч. — 2014. — arXiv: 1402.3722. — URL: http://arxiv.org/abs/1402.3722.
8. A neural probabilistic structured-prediction model for transition-based dependency parsing / H. Zhou [и др.] // ACL-IJCNLP 2015 - 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, Proceedings of the Conference. Т. 1. — Association for Computational Linguistics (ACL), 2015. — С. 1213—1222. — ISBN 9781941643723. — DOI: 10.3115/v1/p15-1117. — URL: https://www.aclweb.org/anthology/P15-1117.
9. *Semina T. A.* Sentiment analysis: Modern approaches and existing problems. // Социальные и гуманитарные науки. Отечественная и зарубежная литература. Серия 6: Языкознание. Реферативный журнал. — 2020. — С. 47—64.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ 1

ПРИЛОЖЕНИЕ 2

pic example

$$F \to x \mid y \mid (S)$$
$$T \to F \mid T * F$$
$$S \to T \mid S + T$$

(а)

(б)

Рис. 10 — (а) Арифметические выражения; (б) Look R96 2016.

Таблица 3 — Расчет параметров

Параметр x_i	Параметр x_j				Первый шаг		Второй шаг	
	X_1	X_2	X_3	X_4	w_i	$K_{вi}$	w_i	$K_{вi}$
X_1	1	1	1.5	1.5	5	0.31	19	0.32
X_2	1	1	1.5	1.5	5	0.31	19	0.32
X_3	0.5	0.5	1	0.5	2.5	0.16	9.25	0.16
X_4	0.5	0.5	1.5	1	3.5	0.22	12.25	0.20
Итого:					16	1	59.5	1