Государственное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКИ И СИСТЕМ УПРАВЛЕНИЯ

КАФЕДРА ТЕОРЕТИЧЕСКОЙ ИНФОРМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

Пояснительная записка
к дипломному проекту на тему:

Автоматический анализ тональности сообщений сети Twitter на основе комбинированных методов обучения и словарей

Студент–дипломник _____ Русначенко Н. Л.

Научный руководитель _____ Лукашевич Н. В.

Москва 2016

Аннотация

В данной работе рассматривается применение методов машинного обучения к решению задачи тональной классификации русскоязычных сообщений сети *Twitter* в сфере банков и телекоммуникаций. В качестве подхода к классификации сообщений рассматривается использование метода *«опорных векторов» (SVM)*. Для повышения качества классификации было объявлено множество вспомогательных признаков, в особенности признаки на основе лексиконов оценочных слов. Сравниваются результаты в зависимости от типа обучающей коллекции (сбалансированная/не сбалансированная), от их объемов, преимущество применения признаков на основе лексиконов к каждому из типов. Была предпринята попытка участия в соревнованиях по тональной классификации сообщений (*SentiRuEval-2016*). Результаты соревнований продемонстрировали устойчивое третье место по обеим задачам. После соревнований были предприняты попытки улучшения качества путем более тонкой настройки классификатора, а также извлечением большей информации из лексиконов. Финальные настройки позволили добиться качества, сравнимого с победителем соревнования.

Пояснительная записка к работе содержит текст на 82 листах формата А4, 24 таблицы, 1 рисунок, список литературы из 20 библиографических ссылок.

Содержание

ВВЕДЕНИЕ

В настоящее время одним из наиболее популярных сервисов распространения коротких новостей является социальная сеть *Twitter*. Большинство пользователей сети часто выражают свое мнение о том, что им понравилось или не понравилось в определенной сфере услуг. Доступность данных сети извне дает возможность обработки и анализа высказанных мнений.

В этой работе рассматривается построение модели на основе *SVM* классификатора для определения тональности сообщений сети *Twitter* заданной тематики. Подразумевается построение моделей применительно к следующим тематикам: отзывы в банковской и телекоммуникационных сферах. Каждое сообщение может быть отнесено к одному из трех тональных классов: негативному, нейтральному, и положительному.

В ходе построения и настройки модели исследовались различные признаки для представления содержания сообщений. Особое внимание уделялось применению словарей оценочных слов для повышения качества классификации.

1 Обзор предметной области

На сегодняшний день, большинство пользователей крупных социальных сетей, таких как *Twitter*, предпочитают высказывать свои отзывы и мнения о товарах и услугах в формате коротких сообщений. Возможность быстрого реагирования на сообщения со стороны компаний, предоставляемых эти услуги, реализуема лишь в случае их автоматической обработки.

Как известно, одним из направлений в решении задачи классификации является использование методов машинного обучения. Ввиду существенного роста объема доступной информации социальных сетей, такие проблемы как адаптация и обучение классификационных моделей становятся все менее значительными. В связи с этим, рассмотрим наиболее популярные методы машинного обучения [1], которые находят свое применение к задачи тональной классификации сообщений.

1.1 Подходы к тональной классификации на основе методов машинного обучения

Основой работы рассматриваемых методов является представление исходных сообщений m в формате вектора нормализованных слов $\{F_1, F_2, \ldots, F_n\}$. В качестве значений каждой из размерности вектора $\vec{m}$ можно сопоставить $n_{F_i}(m)$ – число вхождений терма F_i в рассматриваемое сообщение m. Такая модель является вариацией *Bag Of Words* [1]. В результате, сообщение m представляет собой вектор:

$$\vec{m} = (n_1(m), n_2(m), \ldots, n_k(m)) \tag{1}$$

1.1.1 Метод «Наивного Байеса»

Один из подходов к классификации сообщений заключается в определении класса c^*, к которому относится рассматриваемое сообщение m на основе метрики *максимального правдоподобия*:

$$c^* = argmax_c P(c|m) \tag{2}$$

В основе вычисления условной вероятности $P(c|m)$ лежит правило Байеса:

$$P(c|m) = \frac{P(c,m)}{P(m)} = \frac{P(c) \cdot P(m|c)}{P(m)} \tag{3}$$

Классификатор, построенный на основе правила, представленного в формуле 3, называется *NB*-классификатором. Для оценки условной вероятности в формуле 3, предполагается независимость термов сообщения, и вычисляется следующим образом:

$$P_{NB}(c|m) = \frac{P(c) \cdot (\Pi_{i=1}^{n} P(F_i|c)^{n_i(m)})}{P(m)}$$

Несмотря на простоту реализации алгоритма, независимость термов F_i сообщения является ограничением для достижения реального правдоподобия. Работа [2] демонстрирует оптимальность метода Наивного Байеса для большинства задач классификации, в которых присутствуют признаки, позволяющие установить тесную связь с соответствующими классами. В тоже время, использование более сложных методов позволяет добиться лучших результатов.

1.1.2 Метод «Опорных векторов»

В отличие от метода «Наивного Байеса», подход на основе рассматриваемого метода предполагает поиск гиперплоскости, разделяющей сообщения разных классов. Построение выполняется на этапе обучения модели. На этом этапе решается задача поиска нормали $\vec{w}$ к гиперплоскости, причем разбиение классов должно производиться с максимально возможным отступом.

Для поиска нормали составляется *оптимизационная задача с граничными условиями*:

$$\vec{w} = \sum_{j=1}^{N} \alpha_i c_i \vec{m}_i,\ \alpha_i \geq 0 \tag{4}$$

В уравнении 4, коэффициент $c_i \in \{-1, 1\}$ указывает на принадлежность сообщения m_i соответствующему классу; α_i – коэффициент решения задачи двойной оптимизации. Среди всех векторов $\vec{m}$, для которых выполнено условие $\alpha_i > 0$, называются «опорными». Определения класса, к которому

относится рассматриваемый документ, осуществляется на основе стороны гиперплоскости, на которую падает проекция вектора $\vec{m}$.

В общем случае, классификатор построенный на рассматриваемом подходе, позволяет достичь лучших результатов классификации если сравнивать с аналогом на основе метода «Наивного Байеса» [3].

1.2 Признаки, используемые для классификации сообщений

1.2.1 Векторизация сообщений

Под термином *сообщение* будем понимать вектор, состоящий из нормализованных слов – *термов*. Векторизация сообщения – процесс преобразования сообщения в вектор, в котором каждому терму сообщения сопоставляется некоторое числовое значение.

В п. 1.1 рассматривался один из самых простых способов векторизации сообщений: *Bag Of Words*. Помимо факта присутствия терма и числа его вхождений в сообщение, такая векторизация не несет в себе никакой дополнительной информации терма относительно всей коллекции сообщений.

В качестве дополнительной информации, в векторизацию сообщения может быть заложена частота встречаемости термов. Для определения такой частоты используется мера *tf-idf*, которая вычисляется следующим образом:

$$tf\text{-}idf(t, d, D) = tf(t, d) \cdot idf(t, D) \tag{5}$$

В формуле 5, параметр t – терм, d – документ, являющийся элементом коллекции документов D. Функция tf определяет *частоту встречаемости* (от англ. *term frequency*) терма t в документе d (см. формулу 6).

$$tf(t, d) = \frac{n_i}{|d|} \tag{6}$$

Под *idf* мерой понимается *инвертированная частота терма* (от англ. *inverted document frequency*) в коллекции документов, и определяется формулой 7. Чем больше значение *idf(t, d)*, тем выше уровень *«уникальности»*

рассматриваемого терма t для коллекции документов D.

$$idf(t, D) = log\left[\frac{|D|}{|\{d_i \in D | t_i \in d_i|\}|}\right] \quad (7)$$

1.2.2 Вспомогательные признаки для сообщений

Для повышения качества классификации, вектор каждого сообщения можно дополнить вспомогательными признаками. Основная задача, которая ставится при введении признаков — это отделение одного множества классов от другого, при том что оба множества не являются пустыми. Примерами таких признаков могут служить:

- Учет *эмотиконов* в сообщении: «:)», «;D», «:(», «((», и т.д.;
- Учет знаков препинания: «?», «!», «...», и т.д.;
- Учет числа слов написанных в верхнем регистре.
- Признаки на основе предварительно составленных *тональных лексиконов*.

Признаки, построенные на основе первых трех пунктов, позволяют примерно разделить множество не эмоциональных сообщений от эмоциональных. Использование вспомогательных признаков на основе *тональных лексиконов* позволяет предварительно произвести оценку сообщения, и тем самым *«подсказать»* классификатору наиболее вероятный класс сообщения.

1.3 Способы построения тональных лексиконов

В любом предложении естественного языка содержатся ключевые слова (и словосочетания), на основании которых можно извлечь дополнительную информацию. Если рассматривать предложение с точки зрения тонального анализа, то некоторые слова могут придавать негативную или положительную окраску сообщению в целом. В связи с этим, умение точно извлекать тональные слова или словосочетания с целью составления дополнительной метаинформации о сообщении, может оказать положительное влияние на качество классификации.

Можно сказать, что возникает необходимость в словаре, состоящего из пар $\langle w, t \rangle$, где w — слово или словосочетание, а t — его оценка. Словарь, представленный в таком формате называется *лексиконом*. Задача, которую решает лексикон в области тонального анализа — сопоставление тональной окраски для каждого слова содержащегося в нем. Очевидно, что формализация оценочного параметра приводит к представлению параметра t в формате числового значения, возможно с учетом знака. Обычно в качестве оценки рассматривают действительные значения в области $[-1, 1]$.

Рассмотрим основные методы вычисления оценочных параметров, на основе которых возможно автоматическое создания тональных лексиконов.

1.3.1 Вычисление оценки на основе метрики log-likelihood

Метод, предложенный в [4] предполагает наличие двух корпусов, на основании которых вычисляется частотная оценка принадлежности слова к каждому из корпусов. В общем случае, под *«словом»* понимается терм предложения, однако реальное применение в [4] находит свое место в составлении списка частот для определенных частей речи.

Тем не менее, в общем случае для подсчета на уровне термов *ожидаемой степени принадлежности* $E_i(w)$, применяется следующая формула:

$$E_i(w) = \frac{N_i \sum\limits_{j=1}^{2} O_i(w)}{\sum\limits_{j=1}^{2} N_i} \tag{8}$$

Здесь, параметр i является индексом коллекции, $O_j(w)$ – обозреваемое значение, которое соответствует числу вхождений w в корпус j, а N_j – общее число слов в корпусе j.

Далее, на основании полученных степеней принадлежности терма соответствующему классу, применяют метрику log-likelihood для вычисления степени «важности» терма:

$$-\ln \lambda = \sum_i O_i \ln \left(\frac{O_i}{E_i} \right) \tag{9}$$

Если произвести сортировку словаря на основе формулы параметра формулы 9 в формате убывания значения, то вершину списка будут представлять термы w, для которых наблюдается большая разница в степени принадлежности рассматриваемых корпусов. Соответственно, термы с минимальной разницей окажутся в конце отсортированного списка.

Степень важности терма является положительной величиной. Дополнительно можно ввести знаковый параметр на основании разницы между показаниями формулы 8 для двух классов.

1.3.2 Предсказывание семантической ориентации прилагательных

В работе [5] рассматривается способ определения тональной оценки прилагательных, которые являются аргументами конъюнктов. Основой подхода является гипотеза об ограничении ориентации прилагательных в зависимости от типа объединяющего из конъюнкта и наличия префиксов отрицаний:

Этот бриллиант красивый **и** *дорогой*

Бриллиант красивый, **но** *дорогой*

Это украшение некрасивое **и** *дорогое*

Украшение некрасивое, **но** *дорогое*

Для построения автоматического извлечения семантически ориентированной информации на основе содержимого корпуса большого объема. Извлечение и оценка тональности прилагательных, а также связей между ними реализуется следующими этапами:

1. Из корпуса выбираются все конъюнкты с релевантными морфологическими связями.

   Извлечение конъюнктов выполнялось с помощью формальной грамматики, примененной к корпусу из 21 миллиона словоформ. В результате было собрано около 15 тысяч пар прилагательных;

2. Применяется регерессионная модель для объединения информации разных конъюнктов с целью определения ориентации каждой из связанных

пар. Результатом является граф с одинокого либо разно ориентированными связями между прилагательными;

3. Разделение прилагательных на две группы с разными ориентациями с помощью алгоритма кластеризации. Каждая группа включает в себя максимально возможное число прилагательных. Для этого, к построенному графу применяется функция Φ, параметром которой является $\mathcal{P}$ – разбиение множества прилагательных на группы C_1 и C_2:

$$\Phi(\mathcal{P}) = \sum_{i=1}^{2} \left(\frac{1}{|C_i|} \sum_{x,y \in C_i, x \neq y} d(x,y) \right) \tag{10}$$

Где $|C_i|$ – мощность соответствующего класса; $d(x,y)$ – степень различия прилагательных x и y. Далее, осуществляется поиск параметра $\mathcal{P}_{min}$ для построения наилучшего разбиения;

4. Для групп $C_i,\ i = \overline{1,2}$ вычисляются средние частоты упоминания входящих в них прилагательных. Та группа, для которой полученное значение наибольшее – маркируется как тонально-положительная. Ранее, в работе [6] показывается, что противопоставление качественным прилагательным (т.е. прилагательные противоположной группы) в 81% случаев оказывается семантически не размеченным. Отмечается, что этот факт демонстрирует корреляцию с ориентацией прилагательного, которая наиболее вероятно является положительной.

Применение описанного выше подхода к задаче классификации конъюнктов позволили добиться довольно высоких показаний точности (91% и выше). Отмечается потенциальная возможность применения метода к другим частям речи.

1.3.3 Вычисление оценки на основе тональности словосочетаний

В работе [7] предлагается подход который в отличие от статьи [5], рассмотренной в п. 1.3.2, применим к любым словосочетаниям. Алгоритм состоит из выполнения следующих шагов:

1. Извлечение *синтаксических конструкций*, для которых будет производиться дальнейшая оценка. В качестве таких конструкций могут быть фразы, содержащие прилагательные или наречия (п. 1.3.2). В тоже время, при извлечении одной лишь части речи может возникнуть проблема нехватки контекста для определения оценки. Так, например прилагательное «непредсказуемость» может иметь как негативную окраску в словосочетании «непредсказуемое поведение», так и положительную: «непредсказуемый сюжет».

2. Производится оценка извлеченных синтаксических конструкций на основе *меры взаимной информации* (от англ. *Pointwise Mutation Information, PMI*). Вычисление меры производится между двумя словами $word_1$ и $word_2$, и определяется следующим образом:

$$PMI(word_1, word_2) = log_2 \left[\frac{P(word_1 \ \wedge \ word_2)}{P(word_1) \cdot P(word_2)} \right] \qquad (11)$$

Где $P(word_1 \ \wedge \ word_2)$ – вероятность размещения слов $word_1$ и $word_2$ вместе. Соотношение числителя к знаменателю в формуле 11 определяет степень зависимость одного слова от другого.

Семантической ориентацией (от англ. *Semantic Orientation, SO*), для рассматриваемой конструкцией называется величина, которая вычисляется на основе тональных маркеров “*positive*” и “*poor*”, являющихся аргументами меры точечной взаимоинформации:

$$SO(word) = PMI(word, \text{“}excellent\text{”}) - PMI(word, \text{“}poor\text{”}) \qquad (12)$$

Наибольшее значение в формуле 12 достигается в случае наличия наибольшей связи рассматриваемой конструкции $phrase$ с маркером “*excellent*”; наименьшее, в случае наибольшей связи с “*poor*” маркером.

Формула 11 рассматривалась применительно к параметрам *word*. Для вычисления уравнения 12 от параметра $phrase$, используется функция на основе числа совпадений $hits(phrase)$, а также оператор $NEAR$ расположения слов в непосредственной близости. В сочетании с формулами

11 и 12, семантическая ориентация вычисляется следующим образом:

$$SO(phrase) = log\left[\frac{hits(phrase\ NEAR\ \text{“}excellent\text{”})hits(\text{“}poor\text{”})}{hits(phrase\ NEAR\ \text{“}poor\text{”})hits(\text{“}excellent\text{”})}\right] \quad (13)$$

3. Для подсчета оценки могут быть использованы формулы 12 и 13.

Эксперименты, проведенные для набора из 410 обзоров с ресурса мнений пользователей *Epinions*, показывают среднюю точность классификации порядка 74%. Как оказалось, самым сложным источником для анализа оказались обзоры на фильмы; на такой коллекции точность классификации опустилась до уровня 64%. Наилучшие показатели были достигнуты на коллекциях мнений об автомобилях и банках, на которых точность оказалась в пределах 80–84%.

1.4 Оценка качества классификационной модели

1.4.1 Точность и полнота

Чтобы произвести оценку качества работы классификатора на некотором наборе сообщений, необходимо чтобы для этого набора существовали эталонные значения. Применительно к задаче тональной классификации, под значениями понимается класс, к которому необходимо отнести соответствующее сообщение.

Таким образом, для каждого сообщения ответ может быть получен как со стороны классификатора, так и группой экспертов. Все возможные случаи ответов для фиксированного класса A удобнее всего представить в таблице 1. Такое представление носит название *таблицы контингентности.*

Таблица 1: Таблица контингентности для класса A

Принадлежность сообщений к классу A		эксперты	
		положительная	отрицательная
классификатор	положительная	TP	FP
	отрицательная	FN	TN

На основе таблицы 1 можно рассчитать следующие характеристики качества работы классификатора для соответствующего класса:

- **Полнота** — число найденных сообщений, которые действительно принадлежат соответствующему классу относительно всех сообщений соответствующего класса:

$$R_A = \frac{TP}{TP + FN} \tag{14}$$

- **Точность** — количество сообщений, которое классификатор правильно отнес к соответствующему классу по отношению ко всему объему сообщений определенных системой в этот класс:

$$P_A = \frac{TP}{TP + FP} \tag{15}$$

На практике возникает необходимость в метрике, которая бы позволяла одновременно обе характеристики: точность и полноту. Для этого предусмотрена $F-$мера, которая в общем случае вычисляется по формуле:

$$F(\beta) = \frac{(1 + \beta^2)P \cdot R}{\beta^2 \cdot P + R} \tag{16}$$

В случае, если $\beta = 1$, то формула 16 преобразуется к гармоническому среднему:

$$F_1 = \frac{2 \cdot PR}{P + R} \tag{17}$$

1.4.2 $F_1 - micro$ и $F_1 - macro$ меры качества

Рассмотрим случай, когда необходимо рассмотреть параметры качества работы классификатора относительно нескольких классов одновременно. Пусть имеется K классов, относительно которых будут вычисляться параметры полноты, точности, и $F-$меры. Относительно каждого класса можно составить таблицы контингентности, аналогичные таблице 1.

Одним из методов вычисления среднего значения параметров точности и полноты является *микроусреднение* [8]:

- *Микроусреднением полноты* — является обобщением формулы 14 на

случай нескольких классов:

$$R_{micro_{(1,\dots,N)}} = \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N} TP_i + \sum_{i=1}^{N} FP_i} \tag{18}$$

- *Микроусреднением точности* — называется обобщением формулы 15 на случай нескольких классов:

$$P_{micro_{(1,\dots,N)}} = \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N} TP_i + \sum_{i=1}^{N} FN_i} \tag{19}$$

Другой метод усреднения значений полноты и точности называется *макроусреднением* [8]. Параметры на основе такого подхода, вычисляются следующим образом:

- *Макроусреднение полноты* — вычисление среднего значения параметров полноты каждого из классов:

$$R_{macro_{(1,\dots,N)}} = \frac{\sum_{i=1}^{N} R_i}{N} \tag{20}$$

- *макроусреднение точности* — вычисление среднего значения параметров точности каждого из классов:

$$P_{macro_{(1,\dots,N)}} = \frac{\sum_{i=1}^{N} P_i}{N} \tag{21}$$

Таким образом, на основе каждого из метода усреднений может быть вычислена F_1 мера:

$$F1_{macro_{(1,\dots,N)}} = \frac{2 \cdot P_{macro_{(1,\dots,N)}} R_{macro_{(1,\dots,N)}}}{P_{macro_{(1,\dots,N)}} + R_{macro_{(1,\dots,N)}}} \tag{22}$$

$$F_{1_{micro_{(1,\ldots,N)}}} = \frac{2 \cdot P_{micro_{(1,\ldots,N)}} R_{micro_{(1,\ldots,N)}}}{P_{micro_{(1,\ldots,N)}} + R_{micro_{(1,\ldots,N)}}} \tag{23}$$

Макроусреднение придает одинаковый вес каждому из усредняемых классов, в то время как при микроусреднении вес учитывается на основе числа документов в классе. Поскольку F_{macro} мера игнорирует параметр TN, то смещение среднего значения будет производиться в сторону того класса, для которого классификатор сработал лучше (большее значение TP); в тоже время, при использовании F_{micro}, смещение будет произведено в сторону наибольшего класса. [8]

Таким образом, результаты полученные на основе микроусреднения являются мерой эффективности на коллекциях большого объема. Для получения аналогичного эффекта на коллекциях малого объема, необходимо использовать макроусреднение. [9]

1.5 Подходы к решению задачи тональной классификации с использованием лексиконов

1.5.1 Автоматическое порождение тональных лексиконов

В работе [10] описан способ построения лексикона на основе метода «удаленного контроля». В качестве исходных сообщений, авторы подхода использовали корпус сообщений сети *Twitter*, содержащий для каждого сообщения *метки мнений* (*positive* и *negative*). Такие метки легли в основу обучения контроля полярности классификатора.

Задача контроля полярности ставится следующим образом. Пусть имеется размеченные данные $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{n}$, на основе которых необходимо построить функцию принятия решения $f(\mathbf{x}) \to \mathbf{y}$, которая бы на основе входных параметров определяла бы результирующую метку сообщения. В частности, авторы использовали линейную модель *SVM* классификатора, с функцией предсказания следующего вида:

$$f = sign(w^T \mathbf{x} + b) \tag{24}$$

Где w – весовые коэффициенты, полученные на основе обучающей кол-

лекции; b – поправочный коэффициент.

Авторы статьи предлагают следующий подход автоматического построения лексикона и его использования для создания классификационной модели:

1. Составление неразмеченного корпуса сообщений C сети *Twitter*.

2. Для каждого сообщения $t_i \in C$ использовать подсказки (хэштеги, эмотиконы) для получения метки (*positive* и *negative*) $y_i \in \{-1; +1\}$. Использование эмотиконов вида «:-)», «:-(» в качестве индикатора выражения автора сообщения в целом.

3. Извлечение биграмм и униграмм особенности сообщения t_i в вектор $\mathbf{x}_i \in R^{|L|}$, где L – лексикон, состоящий из термов формата биграмм и униграмм;

4. Построить классификационную модель $\mathbf{w}$ на основе корпуса $C = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{N}$ следующим образом:
$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \tag{25}$$
Здесь $\mathbf{x}_i$ выступают в качестве опорных векторов; y_i – их метки; α_i – параметр краевой задачи, который вносит вклад в w в случае когда положителен.

5. Каждый компонент w_j обученной модели w, соответствует компоненту l_i лексикона L, что устанавливает связи с ассоциативной оценкой.

Используемый лексикон составлен на основе *Twitter* корпуса *Emoticon140*. Метки для корпуса расставлялись на основе эмотиконов, содержащихся в тексте сообщений. Так, сообщения содержащие эмотиконы типа «:)» считались положительными, а «:(» – отрицательными. Объем корпуса составляет 1.6 *млн. сообщений* с одинаковым распределением положительных и негативных сообщений.

Для составления лексикона используется подход на основе вычисления точечной меры взаимоинформации (см. п. 1.3.3). Дополнительно авторами были составлены собственные лексиконы: MPQA, BingLiu, NRC. На этапе

предварительного тестирования и настройки модели, отмечается прирост качества при увеличении числа используемых лексиконов.

Подход демонстрирует хорошие результаты качества работы классификационной модели. На соревнованиях *Semeval-2014* такой подход занял второе место. Применительно к коллекциям SMS и $Twitter$, оценка качества работы на основе $F-$меры колеблется в диапазоне 66.8 – 71%.

1.5.2 Подход к решению задачи тональной классификации, предложенный: Saif M. Mohammad, Svetlana Kiritchenko, Xiaodan Zhu

Статья [11] предлагает способы построения SVM классификаторов для решения следующих задач классификации:

- Определения тональной оценки для всего сообщения в целом;
- Выявления тональности термов сообщения.

Ключевой идеей повышения качества классификации являются лексиконы, которые созданы автоматически на основе коллекции сети *Twitter*. Для этого, авторы разделили все сообщения корпуса на тональные классы с помощью такой метаинформации в сообщениях, как хэштеги. Для этого были составлены два множества хэштегов: *positive* и *negative*.

Объем коллекции, на основе которой составлен лексикон, составляет 775 *тыс. сообщений*. Для распределения сообщений на классы, авторы использовали следующую логику:

1. Если в сообщении встречался хотя бы один хэштег из множества *positive*, то сообщение считается положительным;
2. Если в сообщении встречается хотя бы один хэштег из *negative* множества, то сообщение считается отрицательным.

Для построения лексиконов, авторы использовали метод *«тональности словосочетаний»* (см. п. 1.3.3, формула 12) [7].

В качестве классификатора, авторы использовали SVM с линейным ядром, и параметром штрафной функции $C = 5 \cdot 10^{-3}$. Векторизация сообщения включала в себя набор дополнительных признаков:

- **Учет регистра:** количество слов, записанных в верхнем регистре;
- **Учет хэштегов:** число входящих в сообщение хэштегов (слов с префиксом «#»);
- **Символьные n-граммы:** присутствие или отсутствие последовательности подряд идущих одинаковых символов длиной в 3, 4, и 5 символов;
- **На основе лексиконов:** Множество всех лексиконов включает в себя три лексикона, созданных в ручную (*NRC Emotion Lexicon, MPQA, Bing Liu Lexicon)*, а также два автоматически сгенерированных (*Hashtag Sentiment Lexicon, Sentiment140*). В качестве термов выступали биграммы, униграммы, Пусть w – рассматриваемый токен, p – полярность. Тогда, авторами были составлены следующие признаки:
  - Число токенов, для которых выполнено: $score(w, p) > 0$;
  - Суммарное значение $\sum_{w \in tweet} score(w, p)$;
  - Вычисление максимума $\max_{w \in tweet} score(w, p)$;
  - Учет оценки последнего токена, при условии: $score(w, p) > 0$.
- **Пунктуация:**
  - Подсчет числа последовательностей символов «!» и «?», а также подсчет случаев когда они оба символа встречаются в одной последовательности;
  - Учет содержания знаков «?» или «!» в последнем терме.
- **Эмотиконы:**
  - Присутствие или отсутствие положительных и негативных эмотиконов в любой позиции сообщения;
  - Признак, указывающий на наличие эмотикона (положительного или негативного) в конце сообщения.
- **Удлиненные слова:** число слов, в которых символ повторяется более двух раз, например «sooooo»;

- **Суффикс отрицания:** добавляет к слову суффикс *«_NEG»*, в случае, если перед ним имеется конструкция: *отрицание + знак пунктуации*. Под отрицанием понимаются слова вида: *«no»*, *«shouldn't»*. В качестве знаков пунктуации рассматриваются символы: «,», «.», «:», «;», «!», «?». Пример преобразования:

$$\underline{shouldn't,}\ perfect$$
$$perfect\_NEG$$

Среди всех команд, принимавших участие в соревновании *SemEval-2013 'Detecting Sentiment in Twitter'*, описанный подход занял первое место как в задаче определения тональности отдельного терма, так и сообщения в целом. По задаче оценки термов, авторам подхода удалось добиться оценки $F_{score} = 69.02\%$. На задаче тонального анализа на уровне сообщения, лучшая оценка несколько выше: 88.93%.

1.6 Соревнования в области тональной классификации сообщений

За последние несколько лет, в области автоматической обработки текстовых сообщений активно набирает популярность платформа открытого тестирования систем тонального анализа русскоязычных сообщений *SentiRuEval* [12]. тестирование проводится в формате соревнований, к которым организаторы заранее подготавливают коллекции данных. Данные представляют собой сообщения сети *Twitter*, каждое из которых выражает положительное либо негативное мнение автора по отношению к рассматриваемой в сообщении организации. Среди предметных областей доступны следующие коллекции данных:

- Сообщения о банковских компаниях;
- Сообщения о телекоммуникационных компаниях.

Тональная классификация заключается в определении оценки по отношению к рассматриваемой в сообщении компании. сообщения, для которых пользователям необходимо проставить оценку, представляются в *тестовой коллекции*. Изначально все сообщения тестовой коллекции отмечены как

«нейтральные», и имеют оценку «0». Участнику предлагается изменить значение оценки на «1» если отношение к рассматриваемой организации «положительное», или на «-1» в случае «негативного» отношения.

Помимо тестовых коллекций, организаторы также подготавливают *обучающие* и *эталонные* коллекции для каждой из областей. последние, в свою очередь, доступны после окончания соревнований. данные для коллекций собираются на основе *Streaming API Twitter*. Отмечается, что при сборе данных отсутствует искусственное завышение классов с малым числом сообщений [12]. Так, во всех предметных областях преобладают сообщения с нейтральной оценкой (см. таблицы 2 и 3).

Таблица 2: Распределение объемов классов сообщений в обучающих коллекциях

Область	Нейтральные %	Тональные %	Число сообщений [13]
ТКК	47.59	52.42	$\tilde{4}$ 800
Banks	58.35	41.65	$\tilde{4}$ 900

Таблица 3: Распределение объемов классов сообщений в тестовых коллекциях

Область	Нейтральные %	Тональные %
ТКК	67.70	32.30
Banks	77.90	22.10

Для *оценки качества* классификатора, используются макро- и микроусреднения F_1-меры для классов положительных и негативных сообщений. Вычисление $F_{macro(pos,neg)}$ и $F_{micro(pos,neg)}$ производится на основе формул 22 и 23 соответственно. Нейтральный класс сообщения в оценке не участвует. Однако в случае, если классификатор некорректно определяет тональное сообщение, то это выявляется в расхождении мнения классификатора и эксперта в таблицах контингентностей, составленных для классов pos и neg, на основании которых высчитываются параметры полноты и точности.

Результаты участников сравниваются относительно нижнего порогового значения – *baseline*. В качестве такого порога рассматривается классификатор, который для каждого сообщения проставляет тональную оценку наиболее частотного класса. В таблицах 4 и 5 рассматривается прогоны, де-

монстрирующие лучшие результаты в сравнении с нижним пороговым значением. Параметры прогонов следующие [12]:

1. Классификатор *SVM*, с учетом признаков: нормализованные леммы, установление связи между леммами;

2. Метод максимальной энтропии с учетом признаков: n-граммы (слов и символьные), дополнительные результаты моделирования;

3. Классификатор *SVM* с учетом: n-граммы (словесные и буквенные), знаки пунктуации, наличие ссылок, символы «ретвита», использование эмоционально окрашенного словаря.

Таблица 4: Результаты качества работы лучшего прогона в сравнении с *baseline*. *SentiRuEval-2015*, коллекция сообщений о банках

№	$F_{macro(pos,neg)}$	$F_{micro(pos,neg)}$
baseline	0.1267	0.2377
1	0.3354	0.3656
2	0.3598	0.3430
3	0.3520	0.3370

Таблица 5: Результаты качества работы лучшего прогона в сравнении с *baseline*. *SentiRuEval-2015*, коллекция телекоммуникационных сообщений

№	$F_{macro(pos,neg)}$	$F_{micro(pos,neg)}$
baseline	0.1823	0.3370
1	0.4882	0.5355
2	0.4670	0.5060
3	0.4477	0.5282

Как можно заметить, показания таблиц 4 и 5 демонстрируют значительную разницу в максимумах для каждого из прогонов двух областей. Для выявления причины, вызвавшей такое расхождение, в [12] проводится подсчет разницы между обучающей и тестовых коллекцией для каждой из областей на основе:

- Несимметричная мера удаленности – дивергенция Кульбака-Лейблера;
- Симметричная мера Йенсена-Шеннона.

Результаты показали, что наибольшее расхождение между коллекциями наблюдалось в области банков. Такое расхождение может быть объяснено тем, что коллекции для каждой из областей были подготовлены в разные периоды времени. Обучающие коллекции составлены в период июнь-август 2014 года, в момент разгара боевых действий на Украине. В тоже время, данные для тестовых коллекций были получены *на пол года раньше*, в период с декабря 2013 по февраль 2014.

Несмотря на то, что в сообщениях могло упоминаться отношение к нескольким компаниям одновременно, большинство участников все равно предпочли оценивать сообщение в целом. Участники с такой стратегией показали более высокие результаты чем те, кто пытался производить оценку для каждой компании в отдельности. Отсюда можно сделать вывод, что на сегодняшний момент задача по тональной оценке отдельных объектов сообщения является весьма ограниченной [12].

2 Постановка задачи

Задачей данной работы является разработка методов анализа тональности сообщений Твиттера на основе комбинирования методов машинного обучения и словарей оценочной лексики. Каждое сообщение представляет собой высказывание автора относительно компаний определенной области. Задача анализа тональности ставится как задача классификации сообщений на три класса: позитивный, негативный, и нейтральный. Классификация производится по отношению к компаниям, которые упоминаются в сообщении.

Для решения этой задачи необходимо:

- Исследовать представление текстов сообщений в виде набора признаков для применения методов машинного обучения;
- Исследовать возможности автоматического порождения словарей оценочной лексики;
- Исследовать способы представления информации из словарей оценочной лексики в виде признаков для системы машинного обучения;
- Экспериментально проверить качество созданной системы анализа тональности на данных тестирований систем анализа тональности для русского языка *SentiRuEval-2015* и *SentiRuEval-2016*.

3 Описание подхода к решению задачи тонального анализа сообщений

Строится тональный классификатор на основе методов машинного обучения. Основным параметром для классификаторов на основе таких методов являются *вектора*, описывающие исходные сообщения. Рассмотрим процесс преобразования сообщений в вектор, а также добавим дополнительные признаки которые бы способствовали повышению качества классификации.

3.1 Обработка сообщений

Процесс обработки сообщений коллекции сообщений состоит из выполнения следующих этапов:

1. Лемматизация слов сообщений с целью получения списка термов;
2. Все термы сообщения можно разделить на два класса: *основные* и *дополнительные*. Ко второму классу относятся следующие термы:
   - Символы «Ретвита» — термы со значением «RT»;
   - Ссылки на ресурсы сети Интернет — *URL*-адреса;
   - Имена пользователей сети *Twitter* — термы с префиксом «@»;
   - Хэштеги — термы с префиксом «#».

   Все остальные термы относятся к классу *основных*, и на текущем этапе термы этого класса остаются в сообщении. Среди множества дополнительных термов, в сообщении остаются только хэштеги и *URL*-адреса;
3. Применение предварительно составленного *списка стоп слов* — это термы, которые должны быть исключены из сообщения.
4. Замена некоторых биграмм и униграмм на тональные префиксы.

Рассмотрим последний пункт обработки сообщения подробнее. Для выполнения этого этапа, предполагается что предварительно составлен список пар $L_{tone} = \langle t, s \rangle$, где t – терм, а s – тональная оценка («+» или «–»). На

этом этапе, для каждого терма сообщения $t_i \in L_{tone}$ выполняется замена на соответствующую оценку s, которая становится префиксом следующего терма t_{i+1}. Пример преобразования:

Сейчас хорошо работать, плохо было раньше.

Сейчас +работать, -было раньше.

Среди описанных выше этапов обработки сообщений, обязательным и неизменным является только лемматизация слов сообщений. Использование и настройки остальных этапов могут быть изменены в зависимости от предпочтений пользователя.

Для получения весовых коэффициентов термов, предполагается использовать меру *tf-idf* (п. 1.2.1).

3.2 Вспомогательные признаки классификации

Помимо термов, составляющих вектор сообщения, предполагается внести все признаки, перечисленные в п. 1.2.2:

- Преобразование эмотиконов сообщения в числовой коэффициент. Предварительно составляются два множества эмотиконов: отрицательные и положительные. Для каждого множества определяется C – суммарное число вхождений его элементов в рассматриваемое сообщение. Результирующий числовой коэффициент вычисляется по формуле: $C_+ - C_-$;
- Подсчет количества термов, записанных в верхнем регистре;
- Подсчет числа знаков препинания: «?», «...», «!»;
- Относительно каждого из предварительно составленных лексиконов, вычисляется сумма численных коэффициентов лексикона для термов входящих в рассматриваемое сообщение. Если терм отсутствует в лексиконе, то в качестве коэффициента рассматривается нулевое значение.

  Дополнительно производится нормализация полученного значения в диапазоне $[-1, 1]$ на основе следующего преобразования:

$$\begin{cases} s = 1 - e^{-|x|}, x > 0 & (26) \\ s = -(1 - e^{-|x|}), x < 0 & (27) \end{cases}$$

3.3 Коллекции данных для обучения

Для обучения классификатора предполагается использовать соответствующие коллекции данных соревнований *SentiRuEval*. Поскольку в предоставляемых данных число тональных сообщений существенно уступает объему класса нейтральных сообщений, то дополнительно планируется создать сбалансированную обучающую коллекцию. В работе [14], применительно к классификаторам типа *Наивного Байеса* и *SVM*, отмечается существенный прирост качества при использовании коллекций сбалансированного типа.

Предоставляемые коллекции для обучения достаточно велики (см. таблицу 2) по объему для проведения ручной балансировки. Для решения подобной задачи можно воспользоваться готовыми корпусами, в котором сообщения автоматически распределены на две тональные группы: положительные и негативные. В целях исследования улучшения качества, в качестве источника дополнительных тональных сообщений можно использовать корпус коротких сообщений сети *Twitter*, предоставляемый Ю. Рубцовой [15]. Характеристики такого корпуса представлены в таблице 6.

Таблица 6: Параметры корпуса коротких сообщений сети *Twitter*, Ю.Рубцова

Коллекция	Число сообщений
positive	114 991
negative	111 923

Среди всех сообщений каждого класса корпуса коротких сообщений, необходимо отобрать небольшой процент тех, которые являются наиболее эмоциональными. В общем случае, можно сказать, что требуется функция, которая бы на основе слов сообщения, а также эмоциональных коэффициентов каждого из слов, позволила бы оценить рассматриваемое предложение.

Вычисление эмоциональных коэффициентов можно производить с использованием лексиконов, на основе подхода, рассмотренного в п. 1.3.3. Определение оценки сообщения можно произвести одним из следующих способов:

- Вычисление *суммы эмоциональных коэффициентов* всех входящих в сообщение слов;
- Вычисление *максимального коэффициента* среди всех слов сообщения.

Если полученные значения рассматривать в формате абсолютных величин, то введение нижнего порогового значения позволяет отобрать наиболее подходящие сообщения. Таким образом можно получить сообщения, которые являются наиболее предпочтительными для балансировки обучающих коллекций.

4 Реализация предложенного подхода

Приложение предоставляет пользователю следующий набор возможностей:

1. Определение тональности сообщений тестовой коллекции по отношению к рассматриваемым компаниями. Для каждого сообщения тестовой коллекции, классификатор проставляет одну оценку из множества значений $\{1, 0, -1\}$;

2. Составление собственной тестовой коллекции сообщений, по тематике схожих с отзывами о банках или телекоммуникационных компаниях (см. п. 1.6).

3. Вычисление оценки работы классификатора на сообщениях тестовой коллекции с помощью сравнения полученных результатов с эталонными значениями.

4. Извлекать актуальные данные из сети *Twitter* с помощью *Streaming API*, а также набор инструментов для построения лексиконов на их основе.

4.1 Формат представления коллекций

Изначально, для описания коллекций используется документ формата *XML*, который представляет собой экспорт таблицы с сообщениями из СУБД с поддержкой *SQL* синтаксиса. Именно в таком формате, организаторы соревнований *SentiRuEval* распространяют все типы коллекций по каждой тематике. Структура документа поделена на две части:

1. Описание таблицы, используемой для хранения текстовых сообщений с тональными оценками (см. листинг 1);

2. Сообщения с заполненными в соответствии с форматом таблицы параметрами (см. листинг 2);

Листинг 1: "Формат описание таблицы для хранения сообщений"

```
<!--
- Structure schemas
-->
<pma:structure_schemas>
    <pma:database name="RomipData" charset="utf8">
        <pma:table name="bank_train">
            CREATE TABLE `bank_train` (
              `id` int(11) NOT NULL AUTO_INCREMENT,
              `twitid` bigint(32) NOT NULL DEFAULT '0',
              `date` varchar(128) DEFAULT NULL,
              `text` varchar(256) DEFAULT NULL,
                                ...
            );
        </pma:table>
    </pma:database>
</pma:structure_schemas>
```

Листинг 2: "Формат представления сообщений"

```
<!--
- Database: 'RomipData'
-->
<database name="RomipData">
    <!--
    - Table: bank_train
    -->
    <table name="bank_train">
        <column name="id">1</column>
        <column name="twitid">492367586156630000</column>
        <column name="date">1406224554</column>
        <column name="text"></column>
                    ...
    </table>
     ...
</database>
```

Все параметры сообщения можно разделить на два класса: *обязательные* и *дополнительные*. Каждое сообщение содержит следующий набор обязательных параметров (см. листинг 2):

1. `id` — идентификатор сообщения, уникальный в пределах рассматриваемой коллекции;

2. `twitid` — уникальный идентификатор сообщения сети *Twitter*;

3. `date` — дата появления сообщения в сети;

4. `text` — текст сообщения.

Набор дополнительных параметров зависит от предметной области, в которой рассматривается сообщение. В общем случае, это организации, по отношению к которым рассматривается сообщение.

4.2 Формат выходных данных

Одним из результатов работы классификатора является *XML* документ, формат которого совпадает с представлением листинга 2, и в котором для каждого сообщения во все дополнительные поля проставлена тональная оценка.

Оценка качества работы классификатора представляет собой текстовое сообщение, содержащее результаты показаний $F_{macro}(neg, pos)$ и $F_{micro}(neg, pos)$, а также значения вспомогательных параметров, на основе которых они были получены. Пример вывода результата качества работы классификатора представлен в листинге 3.

Листинг 3: "Формат описания результатов классификатора."

```
1 Counts    -  positive { tp: 91, tn: 34453, fp: 121, fn: 259 }
2              negative { tp: 298, tn: 32268, fp: 191, fn: 372 }
3 Precision -  { positive: 0.42924528301886794, negative: 0.6094069529652352 }
4 Recall    -  { positive: 0.26, negative: 0.44477611940298506 }
5 F_R_macro -  0.41903991353449255
6 F_R_micro -  0.45206275421266706
```

Разберем структуру листинга 3 подробнее. В строках 1-2 указываются значения параметров таблицы контингентности (см. таблицу 1) для классов *positive* и *negative*. Значения параметров точности и полноты для каждого из тональных классов указаны в строках 3-4. Результаты *макро-* и *микрооценок* $F-$меры представлены в строках 5-6.

Для оценки качества работы классификатора, организаторами соревнований *SentiRuEval* предоставляется сценарий реализованный на языке *Javascript*. Аргументами сценария являются:

- **Тип задачи** — какой сфере компаний принадлежат сообщения. Используется для определения полей, которые будут использованы для подсчета оценок;
- **Тестовая коллекция** соответствующей задачи;
- **Эталонная коллекция** соответствующей задачи.

Тестовая коллекция должна содержать оценки, проставленные классификатором. Обе коллекции должны соответствовать формату представления данных п. 4.1.

4.3 Разработка тонального классификатора

В качестве основы для разработки приложения был выбран язык *Python*, ввиду следующих особенностей:

1. Отсутствие временных ограничений на обработку сообщений, классификацию;
2. Возможность быстро подстраиваться под изменение архитектуры проекта и введения новых возможностей;
3. Наличие библиотеки LibSVM реализующей классификацию методом Опорных Векторов. [16]

Что касается вопроса хранения коллекций и вспомогательной информации для решения задачи классификации, для этих целей используется СУБД *PostgreSQL*. Такой выбор обусловлен удобным интерфейсом взаимодействия через сценарии языка *Python*, а также возможностью удобного консольного администрирования хранилища. Как и в случае с выбором языка для реализации задачи, на выбор хранилища не накладываются временные ограничения.

Под разработкой проекта понимается реализация сценариев на языке *Python*, для выполнения следующих подзадач:

1. Подготовка коллекций — импорт XML данных в хранилище;
2. Построение модели на основе обучающей коллекции;
3. Применение модели к данным тестовой коллекции;
4. Экспорт размеченных классификатором сообщений из хранилища в *XML* формат;
5. Вычисление оценки качества работы модели.

4.3.1 Импорт/Экспорт сообщений

Для хранения исходных коллекций, данные формата *XML* экспортировались в СУБД *PostrgreSQL*. В п. 4.1 рассматривается структура данных коллекции, с указанием на классы параметров. В зависимости от рассматриваемой задачи (см. п. 1.6), изменяется только набор *дополнительных параметров*, а *основные* остаются неизменными. Формат таблиц описывается с помощью *SQL* синтаксиса (см. п. 4.1). В листинге 4 приведен пример сценария создания таблицы для сообщений в банковской сфере.

Листинг 4: "Сценарий создания таблицы для хранения тестовой коллекции сообщений банковской сферы"

```
CREATE TABLE `bank_test_2016` (
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `twitid` bigint(32) NOT NULL,
    `date` varchar(128) DEFAULT NULL,
    `text` varchar(256) DEFAULT NULL,
    `sberbank` int(10) DEFAULT NULL,
    `vtb` int(10) DEFAULT NULL,
    `gazprom` int(10) DEFAULT NULL,
    `alfabank` int(10) DEFAULT NULL,
    `bankmoskvy` int(10) DEFAULT NULL,
    `raiffeisen` int(10) DEFAULT NULL,
    `uralsib` int(10) DEFAULT NULL,
    `rshb` int(10) DEFAULT NULL
);
```

Дополнительными полями в листинге 4 являются те, которые записаны в строках 6-13 и указывают на рассматриваемые компании. Сценарий создания таблицы для сферы ТКК отличается лишь набором дополнительных полей.

Работа с таблицами *PostgreSQL* в языке *Python* производится с помощью библиотеки `psycopg`. Для удобного чтения *XML* файла используется библиотека `libxml2`. Преобразование *XML* формата в хранилище рассматривается в листинге 5.

Листинг 5: "Импорт данных из формата *XML* в таблицу *PostgreSQL*"

```
1  # Parse XML file
2  doc = libxml2.parseFile(xmlFile)
3  ctx = doc.xpathNewContext()
4  # Create connection
5  conn = psycopg2.connect(connSettings)
6  cursor = conn.cursor()
7  # Reading data from XML file
8  tables = ctx.xpathEval("/pma/database/table")
9  for table in tables:
10     row = getRow(table)
11     insertRow(cursor, table.prop("name"), row)
```

Экспорт сообщений из *PostgreSQL* в *XML* формат (см. листинг 2) реализован с помощью библиотеки `etree` для построения *XML* дерева. В листинге 6 приведен процесс построения дерева с помощью курсора, созданного для обхода таблицы СУБД.

Листинг 6: "Экспорт данных из таблицы *PostgreSQL* в *XML* формат"

```
1  import xml.etree.cElementTree as ET
2  row = data_cursor.fetchone()
3  while row is not None:
4      # create table element
5      table_node = ET.SubElement(database_node, "table", name=table_name)
6      for i in range(0, len(row)):
7          text = str(row[i])
8          if row[i] is None:
9              text = 'NULL'
10         if (row_cols[i] != 'text'):
11             ET.SubElement(table_node, "column", name=row_cols[i]).text = text
```

```
12      # go to next line
13      row = data_cursor.fetchone()
```

4.3.2 Обработка сообщений сети Twitter

Прежде чем перейти к описанию процесса преобразования, введем следующие понятия:

- **Слово** — это последовательность символов, которая ограничена пробельными символами, либо символами перевода строки.
- **Терм** — слова сообщения, к которым либо была применена лемматизация, либо применение лемматизации не требуется.

Процесс обработки сообщений начинается с этапа разбиния всех слов сообщения на следующие классы (согласно п. 3.1):

- Имена пользователей сети *Twitter* — термы с префиксом «@»;
- Хэштеги — термы с префиксом «#».
- Символы «Ретвита» — термы со значением «RT»;
- Ссылки на ресурсы сети Интернет — *URL*-адреса;
- Основные слова — множество слов, не вошедших ни в один из четырех выше перечисленных классов.

Код программы, отвечающий за выбор класса, к которому относится каждое из слов сообщения, представлен в листинге 7.

Листинг 7: "Определение класса для каждого из слов сообщения"

```
1  for word in words:
2      if (word[0] == '@'):
3          users.append(word)
4      elif (word[0] == '#'):
5          hash_tags.append(word)
6      elif (word.find('http:') == 0):
7          urls.append(word)
8      elif(word == 'RT'):
```

```
9           has_retweet = True
10
11  for f in urls + users + hash_tags + 'RT':
12      if f in words:
13          words.remove(f)
```

Слова, не относящиеся ко множеству *основных*, не подвергаются дальнейшей обработке и лемматизации. Поэтому, ввиду выше введенных определений, слова этих множеств можно считать термами.

На следующем этапе обработки сообщения выполняется *лемматизация основных слов*. Результатом этого этапа должны стать леммы. Каждая лемма представляет собой последовательность буквенных символов. Другими словами, леммы должны быть очищены от знаков препинания, эмотиконов, и т.п.

Для преобразования слова в лемму, используется пакет *Mystem* компании *Yandex* [17]. Такой пакет предоставляет одноименный класс *Mystem*, объект на основе которого используется для решения такой задачи. Лемматизация производится на основе словарей русского языка, которые добавляются в качестве компонента к пакету при первом его использовании.

Лемматизация основных слов сообщения приведена в листинге 8. Для, того чтобы игнорировать знаки препинания, а также эмотиконы которые могут встречаться в конце слов, пакет предусматривает в качестве аргумента параметр *entire_input* со значением *false* (строка 1, листинг 8).

Листинг 8: "Лемматизация основных слов сообщения"

```
1  mystem = Mystem(entire_input=False)
2  lemmas = [unicode(w.strip(), 'utf-8') for w in
3      self.mystem.lemmatize(' '.join(self.words)) if
4      not(w in ['\n', ' ', '\t', '\r'])]
```

После того как сообщение преобразовано в список лемм, производится *опциональное*[1] *отбрасывание из этого списка стоп-слов*. Предусматривается два типа списков:

- Абсолютный список стоп-слов — является общим для всех задач. (см. п. 1.6);

[1] Разрешение на выполнение этого этапа, как и любых других опциальных, указывается в конфигурационных файлах приложения.

- Список стоп-слов конкретной задачи.

Реализация этой операции приведена в листинге 9.

Листинг 9: "Применение списка стоп слов для фильтрации лемм"

```
if (use_stop_words):
    unicode_terms = [t for t in unicode_terms if
        not(t in abs_stop_words) and not(t in stop_words)]
```

Последний этап, который также выполняется опционально, реализует замену некоторых лемм на тональные префиксы «+» и «−», а также дальнейшую конкатенацию префиксов со следующими термами списка. Тональные префиксы могут представлять собой список биграмм и униграмм. В листинге 38 представлена реализация такого преобразования.

Листинг 10: "Преобразование термов в тональные префиксы"

```
if (use_tone_prefixes):
    to_remove = [], i = 0
    while i < len(terms)-1:
        bigram = terms[i] + ' ' + terms[i+1]
        if (bigram in prefix) and (i < len(terms)-2):
            terms[i+2] = prefix[bigram] + terms[i+2]
            to_remove.append(i), to_remove.append(i+1), i += 3
        else:
            unigram = terms[i]
            if (unigram in prefix):
                terms[i+1] = prefix[unigram] + terms[i+1]
                to_remove.append(i), i += 2
            else:
                i += 1
```

После завершения всех описанных выше этапов выполняется векторизация сообщения.

4.3.3 Использование LibSVM для классификации методом опорных векторов

Сообщения в векторизованном виде используются как для обучения модели на основе классификатора SVM, так и для тестирования. Обучение

или применение построенной модели к данным, зависит от типа коллекции, которая подверглась векторизации. Так, векторизация сообщений обучающей коллекции необходима для построения обучающей модели, а векторизация сообщений тестовой — для разметки сообщений на основе уже построенной модели.

Рассмотрим формат представления коллекций, с которым работает пакет *LibSVM*. Чтобы создать классификационную модель, сообщения для обучения должны быть представлены в формате векторов, которые записываются следующим образом:

<label> <$index_1$>:<$value_1$> <$index_2$>:<$value_2$> ...

Рассмотрим параметры, входящие в шаблон:

- **Метка** (*от англ. Label*) — применительно к задаче тональной классификации, указывает на оценку которая была присвоена соответствующему векторизованному сообщению;
- **Индекс терма** — уникальное числовое значение, используемое для идентифицирования соответствующего терма среди всех термов, встречающихся в коллекции;
- **Значение** (*от англ. Value*) — числовое значение, соответствующее терму.

По аналогии с форматом описания данных для обучения классификатора, вектора тестовой коллекции описываются аналогичным образом, с той лишь разницей, что в качестве *«метки»* может быть передано любое значение. Значение этого параметра не используется при классификации. В текущей реализации, этот параметр хранит идентификатор сообщения (поле *id*, см. п. 4.3.1).

Поскольку индексы термов тестовой и обучающих коллекций должны быть синхронизованы, то индексация должна производится для объединенного множества термов каждой из коллекций. Модуль генерации индексов[2] работает с коллекциями сообщений сети *Twitter*, представленных в формате таблиц хранилища. В Листинге 11 реализовано составление словаря на основе списка таблиц с коллекциями.

[2] github.com/nicolay-r/tone-classifier/blob/2016_jan_contest/models/aux/indexer.py

Листинг 11: "Получения словаря индексов для всех термов обучающей и тестовой коллекций"

```
vocabulary = TermVocabulary()
mystem = Mystem(entire_input=False)
for table in config['tables']:
    cursor.execute("""SELECT text FROM %s;"""%(table))
    row = cursor.fetchone(), current_row = 0, rowcount = cursor.rowcount
    while (row is not None):
        message = Message(text=row[0], mystem=mystem, configpath="msg.conf")
        message.process()
        terms = message.get_terms()
        for t in terms:
            vocabulary.insert_term(t)
        row = cursor.fetchone(), current_row += 1
```

Вычисление параметра *«значение»* производилось следующими способами:

1. На основе метрики *tf-idf* (см. формулу 5);
2. Искусственное повышение точности метрики *tf-idf* засчет информации о числе вхождении наиболее популярных термов в корпус из 1 миллиона документов.

Во втором случае, информация числе о документов, в которых содержится тот или иной терм, представлена в текстовом файле в формате списка:

«терм» «число документов, содержащих терм»

Пусть K — основной корпус документов. Обозначим E — корпус, на основе которого для каждого терма была построена информация о числе его вхождений в документы. Тогда, формула вычисления *idf меры* будет выглядеть следующим образом:

$$idf(t, K \cup E) = log\left[\frac{|K \cup E|}{|\{d_i \in K \cup E | t_i \in d_i|\}|}\right] \tag{28}$$

После того, как были подготовлены тестовые и обучающие коллекции к описанному выше формату, выполняются действия, за которые отвечает

модуль классификации[3] (реализация приведена в листинге 12):

1. Построение модели на основе обучающих данных (строки 4-5);

2. Применение построенной модели к тестовой коллекции (строки 6-9).

Листинг 12: "Построение модели и ее применение к тестовым данным"

```
1  # importing libraries
2  from svm import *
3  from svmutil import *
4  # preparing a model
5  model = svm_load_model('train_collection.txt')
6  # problem reading
7  ids, x = svm_read_problem('test_collection.txt')
8  # predicting
9  p_label, p_acc, p_val = svm_predict(y, x, model)
10 # show class volumes
11 print "-1: %s "%(p_label.count(-1))
12 print "0: %s "%(p_label.count(0))
13 print "1: %s "%(p_label.count(1))
```

4.3.4 Дополнительные признаки классификации

Рассмотрим реализацию дополнительных признаков[4] п. 3.2. Признаки добавлялись в формате термов к сообщению. Такие термы представляли собой ключевые слова, префиксом которых является символ «$».

Для обозначения признаков используются следующие ключевые слова (термы):

- `$emoticons` — вычисление значения признака на основе множеств положительных и негативных эмотиконов. Для этого признака используется исходный текст сообщения (см. листинг 13);

- `$capitals` — признак, используемый для указания числа термов, записанных в верхнем регистре (см. листинг 14);

- `$signs` — подсчет числа знаков препинания (см. листинг 15).

[3] github.com/nicolay-r/tone-classifier/blob/2016_jan_contest/svm/python/predict.py
[4] github.com/nicolay-r/tone-classifier/blob/2016_jan_contest/models/aux/features.py

Ввиду того, что лексиконов может быть использовано несколько, а также на основе каждого лексикона реализуется несколько признаков, ключевое слово задается в следующем формате:

`${имя признака на основе лексикона}`

Таким образом, имя признака включает в себя как имя лексикона, так и имя функции, на основе которой вычисляется значение.

Листинг 13: "Вычисление значения признака на основе эмотиконов"

```
def smiles_feature(unicode_message, unicode_smiles):
    total = 0
    for smile in unicode_smiles:
        total += unicode_message.count(smile)
        unicode_message.replace(smile, '')
    return total
# usage
positive = Features.smiles_feature(unicode_message, smiles['positive_values'])
negative = -Features.smiles_feature(unicode_message, smiles['negative_values'])
```

Листинг 14: "Учет числа термов записанных в верхнем регистре"

```
def uppercase_words(msg):
    uppercase_words = 0
    for word in msg.split(' '):
        if len(word) > 0 and word.upper() == word:
            uppercase_words += 1
    return uppercase_words
```

Листинг 15: "Подсчет числа знаков препинания"

```
def signs_feature(msg, chars):
    signs = 0
    for char in chars:
        signs += msg.count(char)
    return signs
```

4.4 Разработка вспомогательных инструментов

В роли вспомогательных инструментов выступают компоненты, которые в совокупности решают две задачи:

- Составление сбалансированной коллекции для обучения классификатора;
- Построения тонального лексикона на основе сообщений сети *Twitter* .

4.4.1 Прием текстовых сообщений сети Twitter

Прием тестовых сообщений реализован на основе библиотеки `tweepy`, которая предоставляет интерфейс взаимодействия с *Streaming Twitter API*. Сначала требуется пройти авторизацию в социальной сети Twitter, основанную на открытом протоколе *OAuth*. От пользователя требуются предварительно зарегистрировать новое приложение, после чего получить необходимые ключи авторизации.

Для приема и сохранения сообщений, реализован класс `StdOutListener` (см. *приложение А*) который наследует класс `tweepy.StreamListener`, с переопределением обработчиков следующих событий:

- `on_data` — срабатывает в случае поступления новых данных с сервиса *Twitter Streaming API*;
- `on_error` — обработчик события возникновения исключения в процессе трансляции сообщений.

Все принятые сообщения сохраняются в текстовый файл формата `.csv`. После того как объект-слушатель создан, необходимо подключиться к трансляции с параметрами фильтрации (строка 6, листинг 16):

- «`track=["twitter"]`» — указывает на извлечение всех сообщений сети (все сообщения отмечены тегом `twitter`);
- «`languages=["ru"]`» — определяет регион, сообщения которого попадут в трансляцию.

Листинг 16: "Прием сообщений сети Twitter"

```
1 stream = tweepy.Stream(auth, listener)
2 with open(out_log, 'w') as log:
3     while (True):
4         log.write("Reconnecting ...")
```

```
5           try:
6               stream.filter(track=['twitter'], languages=['ru'])
7           except Exception as e:
8               log.write("Exception: %s"%(str(e)))
```

4.4.2 Создание лексиконов методом определения тональности словосочетаний

Для построения лексиконов используется подход, рассмотренный в п. 1.3.3. Подход основан на определении семантической ориентации словосочетаний, которая определяется метрикой *точечной взаимной информации* (на основе формулы 11):

$$PMI(t_1, t_2) = \log_2 \frac{P(t_1 \wedge t_2)}{P(t_1) \cdot P(t_2)} \tag{29}$$

Поскольку для каждого терма t, содержащегося в лексиконе необходимо сопоставить оценку тональности, то в качестве одного из аргументов метрики PMI можно рассмотреть один из двух маркеров:

- `«Excellent»` — положительный оттенок;
- `«Poor»` — негативный оттенок.

Степень принадлежности терма двум маркерам называется его *семантической ориентацией*, и определяется формулой (на основе формулы 12):

$$SO(t) = PMI(t, \texttt{Excellent}) - PMI(t, \texttt{Poor}) \tag{30}$$

Пусть K — это коллекция сообщений. Тогда лексикон составляется следующим образом:

$$S : \{\langle t, SO(t)\rangle \mid t \in K_{Excellent} \cup K_{Poor}\} \tag{31}$$

Где $K_{Excellent}$ и K_{Poor} – разделение исходной коллекции K на не пересекающиеся тональные классы сообщений с положительным и негативным

оттенками соответственно. Для построения оценочных лексиконов по сообщениям Твиттера вместо сочетаемости с конкретными словами используется встречаемость слова с положительными (отрицательными) эмотиконами.

Компонент, который реализует построение лексикона[5], представляет собой сценарий, который в качестве аргументов принимает следующие параметры (оба параметра представляют собой названия таблиц СУБД): [6]

1. Тонально положительную коллекцию сообщений;
2. Тонально негативную коллекцию сообщений.

Реализация операции вычисления *точечной взаимной информации*, представлена в листинге 17. Аргументами функции *PMI* листинга 17, являются параметры:

1. `term` — является параметром t_1 формулы 29;
2. `dv1` — словарь, содержащий все документы положительной коллекции;
3. `dv2` — словарь, содержащий все документы негативной коллекции сообщений.

Вычисление результирующего значения функции производится на основе формулы 13.

Листинг 17: "Вычисление точечной меры взаимной информации (*PMI*)."

```
1  def PMI(term, dv1, dv2):
2      t1 = dv1.get_term_in_docs_count_safe(term) + 1
3      t2 = dv2.get_term_in_docs_count_safe(term) + 1
4      N1 = dv1.get_docs_count()
5      N2 = dv2.get_docs_count()
6      return log(p(t1, N1 + N2) * float(N1 + N2) /
7          (p(t1 + t2, N1 + N2) * p(N1, N1 + N2)), 2)
8
9  def p(docs_with_term, total_docs):
10     return float(docs_with_term) / total_docs
```

[5] github.com/nicolay-r/tone-classifier/blob/2016_jan_contest/tools/pmieval/pmieval.py

[6] Сценарий использует дополнительные параметры, связанные с подключением к хранилищу, а также обработкой текстовых сообщений. Формат конфигурационных файлов рассматривается в *Руководстве Пользователя.*

Вычисление *семантической ориентации* приводится в листинге 18. Помимо ранее рассмотренных аргументов для вычисления функции *PMI* листинга 17, добавляются следущие параметры:

- `tv1` — словарь, содержащий все термы положительной коллекции;
- `tv2` — словарь, содержащий все термы негативной коллекции сообщений.

Листинг 18: "Вычисление семантической ориентации терма"

```
def SO(tv1, dv1, tv2, dv2):
    N = dv1.get_docs_count() + dv2.get_docs_count()
    r1 = {}
    all_terms = merge_two_lists(tv1.get_terms(), tv2.get_terms())
    for term in all_terms:
        r1[term] = PMI(term, dv1, dv2) - PMI(term, dv2, dv1)
    r1 = sorted(r1.items(), key=operator.itemgetter(1), reverse=True)
    return r1
```

Результатом работы сценария является таблица со столбцами `term` и `tone`, которые используются для хранения термов и их семантической ориентации соответственно. В листинге 19 представлена реализация сохранения результата

Листинг 19: "Сохранение результата"

```
def save(connection, out_table, result):
    cursor = connection.cursor(), max_term_length = 50
    cursor.execute("""CREATE TABLE IF NOT EXISTS {table} (
        term VARCHAR({max_length}),
        tone REAL)""".format(table = out_table, max_length = max_term_length));
    cursor.execute("DELETE FROM {table}".format(table = out_table));
    for pair in result:
        if (len(pair[0]) < max_term_length):
            cursor.execute("""INSERT INTO {table} ({term}, {value}) VALUES
                (\'{t}\', {v})""".format(term="term", value='tone',
                table=out_table, t=pair[0].encode('utf-8'), v=pair[1]));
```

4.4.3 Балансировка исходных обучающих коллекций

Согласно п. 1.6, обучающие коллекции, предоставленные организаторами являются несбалансированными. Число нейтральных сообщений существенно превышает число сообщений в положительном и негативном классах. Модуль для балансировки обучающих коллекций основан на формате «`.csv`». Именно в одном из таких форматов распространяется *тональный корпус коротких текстов Ю. Рубцовой* [15].

Корпус представляет собой набор сообщений, разбитый на классы *положительных* и *отрицательных*. На основе корпуса коротких текстов был построен лексикон L, который используется в дальнейшем для определения *наиболее эмоциональных слов* w_i:

$$w_i : |L(w_i)| > K \tag{32}$$

Где K – пороговое значение. Списки наиболее эмоциональных слов лексикона L представлены в *Приложении Б* (листинги 36–37). Процесс построения сбалансированной коллекции на основе существующей коллекции реализуется выполнением следующих действий:

1. Создание обучающей коллекции на основе уже существующей, несбалансированной коллекции (см. п. 4.3.1);
2. Составление лексикона на основе корпуса коротких текстов;
3. Фильтрация сообщений: в обучающую коллекцию добавляются только те сообщения, которые содержат наиболее эмоциональные слова из составленного лексикона.

Что касается последнего пункта, то в зависимости от знака такого значения, определяется тональный класс в обучающей коллекции, в который будет произведено добавление. Если в сообщении встречается несколько эмоциональных слов, и при этом принадлежащих разным классам, то сообщение игнорируется.

4.5 Руководство пользователя

4.5.1 Настройка компонентов приложения

Сборка и эксплуатация приложения[7] проверялась под операционной системой *Ubuntu 14.04 x64*. Изначально необходимо выполнить установку зависимостей. Список команд представлен в листинге 20.

Листинг 20: "Установка зависимостей проекта"

```
# Установка зависимостей.
sudo apt-get install python-libxml2 python-psycopg2 python-pip postgresql-9.3 \
    g++ nodejs npm nodejs-legacy unzip
sudo pip install pymystem3
# Загрузка и компиляция библиотеки LibSVM.
git clone https://github.com/cjlin1/libsvm
make -C libsvm
make -C libsvm/python
# Установка пакета eval - скрипт оценки качества работы модели.
cd eval
npm install
```

По-умолчанию, приложение включает в себя обучающую/тестовую/-эталонную коллекции соревнований *SentiRuEval* за 2015 и 2016 года. Для инициализации коллекций в хранилище, пользователю необходимо выполнить последовательность команд, представленных в листинге 21.

Листинг 21: "Инициализация обучающих коллекций"

```
make sentiRuEval2015_collection
make sentiRuEval2016_collection
```

Применение балансировки к обучающей повышает качество оценки сообщений. [14] В листинге 22 приводится последовательность команд, позволяющая создать сбалансированные коллекции на основе обучающих коллекций *SentiRuEval*, дополненных сообщениями из *корпуса коротких текстов* [15].

Листинг 22: "Создание сбалансированных коллекций."

```
make sentiRuEval2015_collection_balanced
make sentiRuEval2016_collection_balanced
```

[7] github.com/nicolay-r/tone-classifier/tree/2016_jan_contest

Приложение содержит по-умолчанию лексиконы[8], представленные в текстовом формате. Для применения лексиконов в классификаторе, необходимо преобразовать текстовые данные в хранилище. Для этого пользователю достаточно выполнить следующие действия, представленные в листинге. 23

Листинг 23: "Инициализация лексиконов по-умолчанию"

```
1 cd data/lexcons/
2 ./extract.py experts_lexicon.txt
3 ./extract.py jan16_lexicon.txt
4 ./extract.py rubtsova_lexicon.txt
```

4.5.2 Работа с приложением

Для работы с приложением, пользователю необходимо перейти в каталог проекта `svm`. Каталог содержит настройки векторизации сообщений:

- `msg.conf` — описывает настройки семантической обработки текста сообщений в формате *JSON*.
- `features.conf` — предоставляет настройку списка дополнительных признаков векторизации в формате *JSON*.

Настройки параметров данных для классификации описаны в файле *Makefile*. Каждая из настроек описывается именем, которое включает в себя характеристику следующих параметров (формат определения значений необходимых параметров представлен в листинге 24):

1. `TASK_TYPE` – тип решаемой задачи (`bank` или `tkk`);
2. `TEST_TABLE` – имя тестовой коллекции в хранилище;
3. `TRAIN_TABLE` – имя обучающей коллекции в хранилище;
4. `MODEL_NAME` – имя используемой модели векторизации сообщений;
5. `ETALON_RESULT` – ссылка на эталонную коллекцию.

[8] `github.com/nicolay-r/tone-classifier/tree/2016_jan_contest/data/lexicons`

Листинг 24: "Пример настройки параметров классификации в *Makefile*."

```
#
# Bank task, SentiRuEval 2015
#
bank_imbalanced: TASK_TYPE = bank
bank_imbalanced: TEST_TABLE = bank_test_2015
bank_imbalanced: TRAIN_TABLE = bank_train_2015
bank_imbalanced: MODEL_NAME = tf_idf
bank_imbalanced: ETALON_RESULT = ../data/2015/$(TASK_TYPE)_etalon.xml
bank_imbalanced: core
```

Чтобы векторизовать содержимое обучающей и тестовых коллекций, а также произвести оценку качества работы классификатора, необходимо выполнить команду:

```
make {название_настройки}
```

По завершении процесса, вся необходимая информация по оценке качества работы будет отображена на экране в формате, представленном в листинге 3.

4.5.3 Формат описания настроек в конфигурационных файлах

Рассмотрим формат описания настроек в конфигурационных файлах. В проекте используются следующие файлы с настройками:

- `msg.conf` — описывает настройки семантической обработки текста сообщений в формате *JSON*.
- `features.conf` — предоставляет настройку списока дополнительных признаков векторизации в формате *JSON*.

Файл с семантической обработкой теста включает в себя настройки следующих параметров:

- Настройки используемых термов, входящие в векторизацию согласно п. 3.1. В листинге 25 описан формат задания настроек. Для активации/отключения термов каждого из типов необходимо в качестве значений указываются `True` или `False` соответветственно;

Листинг 25: "Указание используемых в векторизации термов"

```
1 "urls_used" : "False",
2 "ht_used" : "True",
3 "users_used" : "False",
4 "retweet_used" : "False",
5 "use_stop_words" : "False",
```

- Преобразование списка термов в тональные префиксы «+»/«-» (см. листинг 26). Среди термов могут быть указаны униграммы либо биграммы. Полный список используемых тональных префиксов (строка 2) приведен в *«Приложении В»*.

Листинг 26: "Настройка преобразования термов в тональные префиксы"

```
1 "use_prefix_processor" : "False",
2 "tone_prefix" : {...}
```

- Списки термов, которые удаляются из векторизации (стоп слова) представлены в листинге 27 для каждой задачи в отдельности. Значением этих списков являются строки, перечисленные через зяпятую.

Листинг 27: "Настройка используемых списков стоп слов"

```
1 "bank_stop_words" : [ ... ],
2 "tkk_stop_words" : [ ... ]
```

Настройки дополнительных признаков указываются в конфигурационном файле следующим образом:

- Настройка используемых лексиконов рассматривается в листинге 28. Значением параметра «`lexicons`» является список используемых лексиконов. Каждый лексикон включает в себя следующие настройки:
  - `table` — имя таблицы, в которой описан лексикон;
  - `name` — терм, который будет использоваться при добавлении в словарь с термами;
  - `enabled` — указывает на использование признака; на основе этого лексикона в векторизации.

Листинг 28: "Настройка используемых лексиконов"

```
"lexicons" : [ {
        "lexicon_configpath" : "lexicon.conf",
        "table" : "rubtsova_lexicon",
        "name" : "$rubtsova_lexicon",
        "enabled" : "True"
    }, ... ]
```

- Параметры настройки признака на основе лексиконов приведены в листинге 29. Из параметров, отличных от настройки лексикона следует отметить:
  - `positive_values` — список эмотиконов с положительной тональностью;
  - `negative_values` — список эмотиконов с негативной тональностью.

Листинг 29: "Настройка признака на основе эмотиконов"

```
"smiles" : {
    "enabled"  : "False",
    "name" : "$smiles",
    "positive_values" : [ ... ],
    "negative_values" : [ ... ]
}
```

- Настройка признака на основе знаков пунктуации представлена в листинге 30.

Листинг 30: "Настройка признаков на основе знаков пунктуации"

```
"signs" : {
    "enabled" : "False",
    "name" : "$signs",
    "chars" : ["?", "!", "..."]
},
```

- Настройка признака подсчета числа слов в верхнем регистре описана в листинге 31.

Листинг 31: "Настройка признака подсчета числа слов в верхнем регистре"

```
"uppercase_words" : {
    "enabled" : "False",
    "name" : "$uppercase_words"
}
```

5 Тестирование

5.1 Подготовка данных для построения классифицирующей модели

5.1.1 Данные для обработки сообщений и составления признаков

- Для составления *признаков на основе эмотиконов*, использовались следующие списки положительных и негативных термов (см. листинги 32-33).

Листинг 32: "Список положительных эмотиконов"

```
":)", ":*", ":P", ":D", ":-)", ":-D", "=)", "x)", "xD", "xД"
```

Листинг 33: "Список негативных эмотиконов"

```
":(", "D:", ":'(", ":/", ":-(", "D-:", ":-'(", "=(", "='(", "TT", "x(",
"Dx"
```

- *Список используемых стоп слов*, в зависимости от рассматриваемой задачи, представлен в листингах 34-35.

Листинг 34: "Список стоп слов для задачи *BANK*"

```
"пол", "идти", "бы", "со", "в", "работа", "во", "вот", "грн", "три", "Ъ"
```

Листинг 35: "Список стоп слов для задачи *TKK*"

```
"о", "по", "из", "и", "в", "мочь"
```

- В *«Приложении В»* (листинг 38) рассматриваются лемматизированные слова и словосочетания, которые используются *для получения тональных префиксов.*

5.1.2 Построение лексиконов

Лексиконы были составлены 5 на основе следующих данных (параметры представлены в таблице 7):

1. Корпуса коротких текстов на русском языке[9] [15];

2. Сообщений сети *Twitter* за январь 2016 года (подключение к трансляции сообщений на русском языке с помощью *Streaming API Twitter*);

3. Обучающая коллекция *SentiRuEval-2015* года [13];

4. Тональный словарь созданный вручную экспертами [18].

Таблица 7: Параметры созданных лексиконов (количество термов)

№	Задачи	Положительных	Негативных	Всего
1	Для всех	62 637 (55.5%)	50 177 (44.5%)	112 814
2	Для всех	7 370 (3.12%)	228 721 (96.8%)	236 091
3	BANK	1 748 (41.51%)	2 466 (58.56%)	4 211
	TKK	2 460 (38.47%)	3 934 (61.53%)	6 394
4	Для всех	2 774 (26.0%)	7 148 (67.0%)	10 668

5.1.3 Составление обучающих коллекций

Одно из последних соревнований в этой области проводилось в 2015 году (*SentiRuEval-2015*) [13], данные которого находятся в открытом доступе и содержат эталонную коллекцию. Поэтому можно использовать коллекции *SentiRuEval-2015* для предварительного тестирования.

Обучающие коллекции не являются сбалансированными, и содержат преобладающий по объему класс нейтральных сообщений (отмечается в п. 1.6). В связи с этим, дополнительно была произведена балансировка сообщениями, содержащих термы t с высокими по модулю значениями $SO(t)$ лексикона №1 (см. таблицу 7). Параметры коллекций для предварительного тестирования представлены в таблице Параметры коллекций *SentiRuEval-2016* представлены в таблице 9.

[9] `https://github.com/nicolay-r/tone-classifier/tree/2016_jan_contest/data/lexicons`

Таблица 8: Параметры обучающих коллекций для предварительного тестирования *SentiRuEval-2015* (число сообщений в классах)

Несбалансированная обучающая коллекция *SentiRuEval-2015*				
Коллекция	*Positive*	*Neutral*	*Negative*	Всего
BANK	356 (7,2%)	3482 **(70.84%)**	1077 (21.29%)	4915
TKK	956 (19.67%)	2269 **(46.69%)**	1634 (33.62%)	4859
Сбалансированная коллекция				
Коллекция	Объем класса			Всего
BANK	3482			10446
TKK	2296			6888

Таблица 9: Параметры обучающих коллекций соревнования *SentiRuEval-2016* (число сообщений в классах)

Коллекция	*Positive*	*Neutral*	*Negative*	Всего
BANK	1354 (15.41%)	4870 **(55.4%)**	2550 (29.03%)	4915
TKK	704 (7.7%)	6756 **(74.22%)**	1741 (19.12%)	4859

5.2 Тестирование на коллекции SentiRuEval-2015

Предварительное тестирование классификатора производилось на данных соревнований 2015 года. Настройки векторизации сообщений в предварительных прогонах следующие:

1. Использование русскоязычных термов и хэштегов;
2. Прогон №1 + применение тональных префиксов, использование лексиконов 1 и №2, а также учет всех признаков;
3. Прогон №2 + использование всех лексиконов (кроме №3)[10].

В таблице 10-11 приведены оценки качества работы классификаторов в зависимости от настроек. Процентный прирост качества вычисляется как отношение наибольшего значения оценки по соответствующей метрике ($F_{macro}(neg, pos)$ или $F_{micro}(neg, pos)$) к наименьшему.

На основе полученных результатов было принято решение о создании **расширенной сбалансированной коллекции**: дополнение положительных и негативных классов коллекции 2016 года соответствующими классами

[10] Применение лексикона, составленного на обучающей коллекции *SentiRuEval-2015* года не привело к повышению качества (ввиду малого объема).

Таблица 10: Результаты тестирования (Коллекция BANK, *SentiRuEval-2015*)

№	BANK — сообщения о банковских компаниях			
	Не сбалансированная коллекция		Сбалансированная коллекция	
	$F_{macro}(neg, pos)$	$F_{micro}(neg, pos)$	$F_{macro}(neg, pos)$	$F_{micro}(neg, pos)$
1	0.3659	0.4	**0.4206 (+15.0%)**	**0.458 (+14.5%)**
2	0.3933	0.4128	**0.4305 (+9.4%)**	**0.4718 (+14.2%)**
3	0.4119	0.4394	**0.4349 (+5.5%)**	**0.4792 (+9.0%)**

Таблица 11: Результаты тестирования (Коллекция ТКК, *SentiRuEval-2016*)

№	ТКК — сообщения о телекоммуникационных компаниях			
	Не сбалансированная коллекция		Сбалансированная коллекция	
	$F_{macro}(neg, pos)$	$F_{micro}(neg, pos)$	$F_{macro}(neg, pos)$	$F_{micro}(neg, pos)$
1	**0.4608 (+0.5%)**	**0.5172 (+2.5%)**	0.4583	0.5045
2	**0.4701 (+0.26%)**	**0.5207 (+2.0%)**	0.4689	0.5104
3	**0.4925 (+3.3%)**	**0.5378 (+3.7%)**	0.4767	0.5184

коллекции 2015 года, и дальнейшая балансировка сообщениями. Параметры расширенной сбалансированной коллекции (см. таблицу 12).

Таблица 12: Расширенная обучающая сбалансированная коллекция (количество сообщений)

Коллекция	Объем класса	Всего
BANK	6765	20295
ТКК	4894	14682

5.3 Участие в соревнованях SentiRuEval-2016

5.3.1 Результаты

В таблице 11 приведены оценки качества работы классификатора для тестовой коллекции *SentiRuEval-2016* [19] при использовании настроек предварительного тестирования. Прогоны с такими настройками показали лучшие результаты среди других вариаций настроек предложенного подхода (см. таблицы 13-14).

Таблица 13: Результаты прогонов соревнования (задача BANK, *SentiRuEval-2016*)

№	BANK — сообщения о банковских компаниях			
	Не сбалансированная коллекция (2015 год)		Расширенная сбалансированная коллекция	
	$F_{macro}(neg, pos)$	$F_{micro}(neg, pos)$	$F_{macro}(neg, pos)$	$F_{micro}(neg, pos)$
1	0.384	0.4203	**0.4536 (+18.1%)**	**0.4982 (+18,53%)**
2	0.3849	0.415	**0.4672 (+20.9%)**	**0.5029 (+21,1%)**
3	0.3862	0.4218	**0.4683 (+21.25%)**	**0.5022(+19.06%)**

Таблица 14: Результаты прогонов соревнования (задача ТКК, *SentiRuEval-2016*)

№	ТКК — сообщения о телекоммуникационных компаниях			
	Не сбалансированная коллекция (2015 год)		Расширенная сбалансированная коллекция	
	$F_{macro}(neg, pos)$	$F_{micro}(neg, pos)$	$F_{macro}(neg, pos)$	$F_{micro}(neg, pos)$
1	0.4849	0.641	**0.5103 (+5.2%)**	**0.6509 (+1.5%)**
2	0.4832	0.6473	**0.5231 (+8.2%)**	**0.6508 (+0.5%)**
3	0.5099	**0.677 (+2.0%)**	**0.5286 (+3.6%)**	0.6632

5.3.2 Улучшение результатов

После проведения соревнований, в целях повышения качества классификации, настройки прогонов изменялись в следующих направлениях:

1. **Настройка параметра** C штрафной функции SVM классификатора. По умолчанию $C = 1$. Среди множества протестированных значений $\{1, 0.75, 0.5, 0.25\}$, наибольший прирост достигается при $C = 0.5$ (см. таблицу 15).

2. **Добавление новых признаков:** вычисление *максимальных* и *минимальных* значений (с учетом нормализации на основе формул 26-27) среди всех термов сообщения по каждому из лексиконов. Пусть l – произвольный лексикон из всего множества L. Тогда относительно рассматриваемого лексикона l, для каждого сообщения $m = \{t_i\}_{i=1}^{n}$ вычисляются следующие признаки:

$$f_{max_l} = \max_{i=\overline{1...n}} l(t_i), t_i \in l$$

$$f_{min_l} = \min_{i=1...n} l(t_i), t_i \in l$$

Таблица 15: Влияние настройки параметра Cost (C=0.5) (*SentiRuEval-2016*)

№	BANK (Расширенная сбалансированная коллекция, C=0.5)		TKK (Расширенная сбалансированная коллекция, C=0.5)	
	$F_{macro}(neg, pos)$	$F_{micro}(neg, pos)$	$F_{macro}(neg, pos)$	$F_{micro}(neg, pos)$
1	0.4558 (+0.4%)	0.5037 (+1.1%)	0.5235 (+2.5%)	0.6612 (+1.5%)
2	**0.4795 (+2.6%)**	**0.5167 (+2.7%)**	0.5338 (+2.0%)	0.6610 (+1.5%)
3	0.4768 (+1.8%)	0.5135(+2.2%)	**0.5452 (+3.1%)**	**0.6733 (+1.5%)**

Комбинация рассмотренных выше улучшений привела к настройке *финальных прогонов* (результаты представлены в таблице 16). Во всех прогонах использовались русскоязычные термы и хэштеги, применялись тональные префиксы, а также учитывались все признаки. Изменения в настройках касались только числа используемых лексиконов, а также признаков построенных на их основе (настройки прогонов):

1. Вычисление суммы, минимума, максимума на основе лексикона №1 (см. таблицу 7).

2. Прогон №1 + признаки суммы, минимума, максимума на основе лексикона №2.

3. Прогон №2 + признаки суммы, минимума, максимума на основе лексикона №4.

4. Прогон №3 + признаки минимума и максимума на основе лексиконов №3.

5.3.3 Вывод

Использование метаинформации на основе лексиконов стабильно повышает качество классификации. Наибольший прирост качества достигается в

Таблица 16: Результаты финального тестирования *SentiRuEval-2016*

№	BANK (Расширенная сбалансированная коллекция, C=0.5)		TKK (Расширенная сбалансированная коллекция, C=0.5)	
	$F_{macro}(neg, pos)$	$F_{micro}(neg, pos)$	$F_{macro}(neg, pos)$	$F_{micro}(neg, pos)$
1	0.4955	0.5388	0.5259	0.6662
2	0.5012	0.5379	0.5283	0.6720
3	**0.5239**	**0.5514**	**0.5453**	**0.6970**
4	0.4818	0.5238	0.5356	0.6659

случае, если классификатор был обучен на коллекции несбалансированного типа (см. таблицу 17[11])[12].

Таблица 17: Рост качества при использовании признаков на основе лексиконов в зависимости от типа обучающей коллекции

Параметры обучающей коллекции		BANK		TKK	
Год	Тип	$F_{macro}(neg, pos)$	$F_{micro}(neg, pos)$	$F_{macro}(neg, pos)$	$F_{micro}(neg, pos)$
2015	*A*	+12.57%	+9.8%	+6.8%	+3.9%
	B	+3.3% (+19.0%)	+4.6% (+19.8%)	+4% (+3.4%)	+2.7% (+1.9%)
2016	*A*	—	—	+5.1%	+4.6%
	B	+0.5%	+0.03%	—	—
	C	+4.6% (+21.95)	+1.9% (+19.48%)	+4.1% (+9.0%)	+1.8% (+3.4%)

В таблице 17, значения (+21.95), и (+19.48) последней строки указывают на общий прирост качества с учетом расширенной балансировки по отношению к обычной балансировке (тестирование в этих случаях на несбалансированной коллекции не проводилось, ввиду результатов п. 5.2, таблица 10). Увеличение числа признаков по каждому из лексиконов позволяет повысить показания таблицы 17. В совокупности с использованием сбалансированной обучающей коллекции и настройкой классификатора, в рамках этой работы были получены максимальные результаты (см. таблицу 16, прогон №3). В

[11] Тип обучающей коллекции обозначается следующим образом: *A* — не сбалансированная; *B* — сбалансированная; *C* — расширенная.

[12] В таблице рассматривается прирост качества 3-его прогона по отношению к 1-ому (согласно таблицам 10-11, и 13-14). В скобках указывается общий прирост качества с учетом балансировки.

таблице 18 представлен прирост качества в результате использования расширенной сбалансированной коллекции в сочетании с признаками на основе лексиконов. Наибольший прирост достигается для задачи *BANK*.

Таблица 18: Прирост качества для каждой из задач (сравнение лучшего финального результата с результатами прогона №1, *SentiRuEval-2016*)

Прирост качества	BANK	TKK
Общий	+36.4%	+12.4%

Результаты всех участников соревнований представлены в таблице 19. Про некоторых из участников известны настройки прогонов, которые они использовали для получения соответствующих оценок. Описание настроек рассматривается в таблице 20.[13] Участник под номером 1 соответствует текущей работе. Соответственно, в первой строке таблицы 19 рассматривается максимально полученный в рамках этой работы результат. Вторая строка таблицы содержит результат, полученный во время проведения соревнований.[14]

Таблица 19: В таблице указаны наилучшие результаты для каждой задачи каждого из участников по каждой из задачи относительно показания $F_{macro}(neg, pos)$. Жирным шрифтом выделены результаты участника, который достиг максимальных результатов по метрикам $F_{macro}(neg, pos)$.

Номер участника	BANK		TKK	
	$F_{macro}(neg, pos)$	$F_{micro}(neg, pos)$	$F_{macro}(neg, pos)$	$F_{micro}(neg, pos)$
1	0.5239 (-5.3%) 0.4683 (-17.8%)	0.5514 (-6.6%) 0.5022 (-17.1%)	0.5453 (-2.5%) 0.5286 (-5.8%)	0.6970 (+5.7%) 0.6632 (+0.9%)
2	**0.5517**	**0.5881**	**0.5594**	**0.6569**
3	0.3423	0.3524	0.3994	0.3994
4	0.3730	0.3967	0.4955	0.6252
5	0.3859	0.4640	0.3499	0.4044
6	0.2398	0.3127	0.3545	0.5263
7	0.471	0.5128	0.4842	0.6374
8	0.4492	0.4705	0.4871	0.5745
9	0.5195	0.5595	0.5489	0.6822
10	0.4659	0.5053	0.5055	0.6254

[13] Таблица с результатами прогонов всех участников соревнований, а также настройками прогонов: `https://docs.google.com/spreadsheets/d/1rCaklClawfnnSnyk4q8CW4zWuO3P38DSrLw_f2wyyjg/edit#gid=0`

[14] Разница в результатах, записанная в формате процентов, вычисляется как отношение результата победителя соревнований к соответвующему резульату участика №1.

Таблица 20: Настройки прогонов участников соревнований

Номер участника	Настройки прогона
1	Текущая работа (опубликована в формате статьи конференции *Диалог-2016* [20]). Описание подхода рассмотрено как в статье [20], так и в п. 3.
2	рекуррентная нейронная сетка (*LSTM*). В качестве признаков *WORD2VEC*, обученный на внешней коллекции. (Посты и комментарии из социальных сетей)
4	Словарные признаки + признаки мета-классификаторов (логистическая регрессия, ридж-регрессия, классификатор на основе градиентного бустинга и классификатор на основе нейронной сети) и линейный *SVM* в качестве классификатора.
8	поиск эмоциональных слов по словарю (200 тыс. словоформ), правила их комбинирования на основе синтаксического анализа; применение онтологических правил, характерных для данной предметной области
9	*SVM*: униграммы, биграммы, словарь РуСентиЛекс, учет частей речи, многозначных слов (автоматический словарь коннотаций по новостям для ТКК задачи)
10	*SVM*, в качестве признаков использовались униграммы, подвергшиеся преобразованиям (*не + слово = один признак*, множественные повторения символов заменяются двукратным; ссылки, ответы, даты, числа – заменяются паттернами и другие преобразования). Подключение словаря РуСентиЛекс

6 Технико-экономическое обоснование

По степени новизны разрабатываемый проект относится к *группе новизны A* – разработка программных комплексов, требующих использования принципиально новых методов их создания, проведения НИР и т.п.

По степени сложности алгоритма функционирования проект относится к *2 группе сложности* - программная продукция, реализующая учетно-статистические алгоритмы.

По виду представления исходной информации и способа ее контроля программный продукт относится к *группе 12* - исходная информация представлена в форме документов, имеющих различный формат и структуру и *группе 22* - требуется печать документов одинаковой формы и содержания, вывод массивов данных на машинные носители.

6.1 Трудоемкость разработки программной продукции

Трудоемкость разработки программной продукции (τ_{PP}) может быть определена как сумма величин трудоемкости выполнения отдельных стадий разработки программного продукта из выражения:

$$\tau_{PP} = \tau_{TZ} + \tau_{EP} + \tau_{TP} + \tau_{RP} + \tau_{V},$$

где τ_{TZ} — трудоемкость разработки технического задания на создание программного продукта; τ_{EP} — трудоемкость разработки эскизного проекта программного продукта; τ_{TP} — трудоемкость разработки технического проекта программного продукта; τ_{RP} — трудоемкость разработки рабочего проекта программного продукта; τ_{V} — трудоемкость внедрения разработанного программного продукта.

6.1.1 Трудоемкость разработки технического задания

Расчёт трудоёмкости разработки технического задания (τ_{PP}) [чел.-дни] производится по формуле:

$$\tau_{TZ} = T^{Z}_{RZ} + T^{Z}_{RP},$$

где T_{RZ}^{Z} — затраты времени разработчика постановки задачи на разработку ТЗ, [чел.-дни]; T_{RP}^{Z} — затраты времени разработчика программного обеспечения на разработку ТЗ, [чел.-дни]. Их значения рассчитываются по формулам:

$$T_{RZ}^{Z} = t_Z \cdot K_{RZ}^{Z}$$

$$T_{RP}^{Z} = t_Z \cdot K_{RP}^{Z}$$

где t_Z – норма времени на разработку ТЗ на программный продукт (зависит от функционального назначения и степени новизны разрабатываемого программного продукта), [чел.-дни]. В нашем случае по таблице получаем значение (группа новизны – А, функциональное назначение – технико-экономическое планирование):

$$t_Z = 79.$$

K_{RZ}^{Z} — коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком постановки задачи на стадии ТЗ. В нашем случае (совместная разработка с разработчиком ПО):

$$K_{RZ}^{Z} = 0.65.$$

K_{RP}^{Z} — коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком программного обеспечения на стадии ТЗ. В нашем случае (совместная разработка с разработчиком постановки задач):

$$K_{RP}^{Z} = 0.35.$$

Тогда:

$$\tau_{TZ} = 79 \cdot \ (0.35 + 0.65) = 79.$$

6.1.2 Трудоемкость разработки эскизного проекта

Расчёт трудоёмкости разработки эскизного проекта (τ_{EP}) [чел.-дни] производится по формуле:

$$\tau_{EP} = T^E_{RZ} + T^E_{RP},$$

где T^E_{RZ} — затраты времени разработчика постановки задачи на разработку эскизного проекта (ЭП), [чел.-дни]; T^E_{RP} — затраты времени разработчика программного обеспечения на разработку ЭП, [чел.-дни]. Их значения рассчитываются по формулам:

$$T^E_{RZ} = t_E \cdot K^E_{RZ},$$

$$T^E_{RP} = t_E \cdot K^E_{RP},$$

где t_E – норма времени на разработку ЭП на программный продукт (зависит от функционального назначения и степени новизны разрабатываемого программного продукта), [чел.-дни]. В нашем случае по таблице получаем значение (группа новизны – А, функциональное назначение – технико-экономическое планирование):

$$t_E = 175.$$

K^E_{RZ} — коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком постановки задачи на стадии ЭП. В нашем случае (совместная разработка с разработчиком ПО):

$$K^E_{RZ} = 0.7.$$

K^E_{RP} — коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком программного обеспечения на стадии ТЗ. В нашем случае (совместная разработка с разработчиком постановки задач):

$$K^E_{RP} = 0.3.$$

Тогда:

$$\tau_{EP} = 175 \cdot \ (0.3 + 0.7) = 175.$$

6.1.3 Трудоемкость разработки технического проекта

Трудоёмкость разработки технического проекта (τ_{TP}) [чел.-дни] зависит от функционального назначения программного продукта, количества разновидностей форм входной и выходной информации и определяется по формуле:

$$\tau_{TP} = (t^T_{RZ} + t^T_{RP}) \cdot K_V \cdot K_R,$$

где t^T_{RZ} — норма времени, затрачиваемого на разработку технического проекта (ТП) разработчиком постановки задач, [чел.-дни]; t^T_{RP} — норма времени, затрачиваемого на разработку ТП разработчиком ПО, [чел.-дни]. По таблице принимаем (функциональное назначение — технико-экономическое планирование, количество разновидностей форм входной информации — 1 (результаты качества работы классификатора), количество разновидностей форм выходной информации — 2 (тональная оценка сообщений, оценка работы классификатора)):

$$t^T_{RZ} = 38, \qquad t^T_{RP} = 9$$

K_R — коэффициент учета режима обработки информации. По таблице принимаем (группа новизны — А, режим обработки информации — реальный масштаб времени):

$$K_R = 1.45$$

K_V — коэффициент учета вида используемой информации, определяется по формуле:

$$K_V = \frac{K_P \cdot n_P + K_{NS} \cdot n_{NS} + K_B \cdot n_B}{n_P + n_{NS} + n_B},$$

где K_P — коэффициент учета вида используемой информации для переменной информации; K_{NS} — коэффициент учета вида используемой информации для нормативно-справочной информации; K_B — коэффициент учета вида используемой информации для баз данных; n_P — количество наборов данных переменной информации; n_{NS} — количество наборов данных нормативно-справочной информации; n_B — количество баз данных. Коэффициенты на-

ходим по таблице (группа новизны - А):

$$K_P = 1.70, \qquad K_{NS} = 1.45, \qquad K_B = 4.37.$$

Количество наборов данных, используемых в рамках задачи:

$$n_P = 3, \qquad n_{NS} = 0, \qquad n_B = 1.$$

Находим значение K_V:

$$K_V = \frac{1.70 \cdot 3 + 1.45 \cdot 0 + 4.37 \cdot 1}{3 + 0 + 1} = 2.3675.$$

Тогда:

$$\tau_{TP} = (38 + 9) \cdot 2.3675 \cdot 1.67 = 185.2$$

6.1.4 Трудоемкость разработки рабочего проекта

Трудоёмкость разработки рабочего проекта (τ_{RP}) [чел.-дни] зависит от функционального назначения программного продукта, количества разновидностей форм входной и выходной информации, сложности алгоритма функционирования, сложности контроля информации, степени использования готовых программных модулей, уровня алгоритмического языка программирования и определяется по формуле:

$$\tau_{RP} = (t^R_{RZ} + t^R_{RP}) \cdot K_K \cdot K_R \cdot K_Y \cdot K_Z \cdot K_{IA},$$

где t^R_{RZ} — норма времени, затраченного на разработку рабочего проекта на алгоритмическом языке высокого уровня разработчиком постановки задач, [чел.-дни]. t^R_{RP} — норма времени, затраченного на разработку рабочего проекта на алгоритмическом языке высокого уровня разработчиком ПО, [чел.-дни]. По таблице принимаем (функциональное назначение — технико-экономическое планирование, количество разновидностей форм входной информации — 1, количество разновидностей форм выходной информации — 2:

$$t^R_{RZ} = 11, \qquad t^R_{RP} = 68.$$

K_K — коэффициент учета сложности контроля информации. По таблице принимаем (степень сложности контроля входной информации — 11, степень сложности контроля выходной информации — 22):

$$K_K = 1.07.$$

K_R — коэффициент учета режима обработки информации. По таблице принимаем (группа новизны — А, режим обработки информации — реальный масштаб времени):

$$K_R = 1.75.$$

K_Y — коэффициент учета уровня используемого алгоритмического языка программирования. По таблице принимаем значение (интерпретаторы, языковые описатели):

$$K_Y = 0.8.$$

K_Z — коэффициент учета степени использования готовых программных модулей. По таблице принимаем (использование готовых программных модулей составляет около 50

$$K_Z = 0.6.$$

K_{IA} — коэффициент учета вида используемой информации и сложности алгоритма программного продукта, его значение определяется по формуле:

$$K_I A = \frac{K'_P \cdot n_P + K'_{NS} \cdot n_{NS} + K'_B \cdot n_B}{n_P + n_{NS} + n_B},$$

где K'_P — коэффициент учета сложности алгоритма ПП и вида используемой информации для переменной информации; K'_{NS} — коэффициент учета сложности алгоритма ПП и вида используемой информации для нормативно-справочной информации; K'_B — коэффициент учета сложности алгоритма ПП и вида используемой информации для баз данных. n_P — количество наборов данных переменной информации; n_{NS} — количество наборов данных нормативно-справочной информации; n_B — количество баз данных. Коэффициенты находим по таблице (группа новизны - А):

$$K'_P = 2.02, \qquad K'_{NS} = 1.21, \qquad K'_B = 1.05.$$

Количество наборов данных, используемых в рамках задачи:

$$n_P = 3, \qquad n_{NS} = 0, \qquad n_B = 1.$$

Находим значение K_{IA}:

$$K_{IA} = \frac{2.02 \cdot 3 + 1.21 \cdot 0 + 1.05 \cdot 1}{3 + 0 + 1} = 1.7775.$$

Тогда:

$$\tau_{RP} = (11 + 68) \cdot 1.00 \cdot 1.75 \cdot 0.8 \cdot 0.6 \cdot 1.7775 = 126.2$$

6.1.5 Трудоемкость выполнения стадии «Внедрение»

Расчёт трудоёмкости разработки технического проекта (τ_V) [чел.-дни] производится по формуле:

$$\tau_V = (t^V_{RZ} + t^V_{RP}) \cdot K_K \cdot K_R \cdot K_Z,$$

где t^V_{RZ} — норма времени, затрачиваемого разработчиком постановки задач на выполнение процедур внедрения программного продукта, [чел.-дни]; t^V_{RP} — норма времени, затрачиваемого разработчиком программного обеспечения на выполнение процедур внедрения программного продукта, [чел.-дни]. По таблице принимаем (функциональное назначение — технико-экономическое планирование, количество разновидностей форм входной информации — 1, количество разновидностей форм выходной информации — 2):

$$t^V_{RZ} = 13, \qquad t^V_{RP} = 15.$$

Коэффициент K_K и K_Z были найдены выше:

$$K_K = 1.07, \qquad K_Z = 0.6.$$

K_R — коэффициент учета режима обработки информации. По таблице принимаем (группа новизны — А, режим обработки информации — реальный

масштаб времени):

$$K_R = 1.60.$$

Тогда:

$$\tau_V = (17 + 19) \cdot 1.07 \cdot 1.60 \cdot 0.6 = 36.9$$

Общая трудоёмкость разработки ПП:

$$\tau_{PP} = 79 + 175 + 185.24 + 126.21 + 36.98 = 602.43$$

6.2 Расчет количества исполнителей

Средняя численность исполнителей при реализации проекта разработки и внедрения ПО определяется соотношением:

$$N = \frac{Q_p}{F},$$

где t — затраты труда на выполнение проекта (разработка и внедрение ПО); F — фонд рабочего времени. Разработка велась 5 месяцев с 1 января 2016 по 31 мая 2016. Из таблицы получаем, что фонд рабочего времени

$$F = 664.$$

Получаем число исполнителей проекта:

Таблица 21: Количество рабочих дней по месяцам

Номер месяца	**Интервал дней**	**Количество рабочих дней**
1	01.01.2016 - 31.01.2016	15
3	01.02.2016 - 29.02.2016	20
4	01.03.2016 - 31.03.2016	21
5	01.04.2016 - 30.04.2016	21
6	01.05.2016 - 31.05.2016	19
Итого		96

$$N = \frac{4816}{664} = 7$$

Для реализации проекта потребуются 3 старших инженеров и 4 простых инженеров. Исходя из того, что в месяце в среднем 22 рабочих дня, то для выполнения всего проекта потребуется около 4,7 месяца. На выполнение всего проекта, требуется около 22 недель.

6.3 Ленточный график выполнения работ

На основе рассчитанных в главах 6.1, 6.2 трудоёмкости и фонда рабочего времени найдём количество рабочих дней, требуемых для выполнения каждого этапа разработка. Результаты приведены в таблице 21. Планирование и контроль хода выполнения разработки проводится по ленточному графику выполнения работ (см. таблицу 22).

Таблица 22: Ленточный график выполнения работ

Номер стадии	Продолжительность [раб.-дни]	Календарные дни																						
		01.01.2016 - 03.01.2016	04.01.2016 - 10.01.2016	11.01.2016 - 17.01.2016	18.01.2016 - 24.01.2016	25.01.2016 - 31.01.2016	01.02.2016 - 07.02.2016	08.02.2016 - 14.02.2016	15.02.2016 - 21.02.2016	22.02.2016 - 28.02.2016	29.02.2016 - 06.03.2016	07.03.2016 - 13.03.2016	14.03.2016 - 20.03.2016	21.03.2016 - 27.03.2016	28.03.2016 - 03.04.2016	04.04.2016 - 10.04.2016	11.04.2016 - 17.04.2016	18.04.2016 - 24.04.2016	25.04.2016 - 01.05.2016	02.05.2016 - 08.05.2016	08.05.2016 - 15.05.2016	16.05.2016 - 22.05.2016	23.05.2016 - 29.05.2016	30.05.2016 - 31.05.2016
		Количество рабочих дней																						
		0	0	5	5	5	5	5	6	5	5	3	5	5	5	5	5	5	5	5	4	4	5	5	2
1	26			5	5	5	5	5	1																
2	25								4	5	4	5	5	2											
3	27													3	5	5	5	4							
4	18																	1	4	4	4	4	1	
5	9																						4	5

6.4 Определение себестоимости программной продукции

В работе над проектом используется специальное оборудование — персональные электронно-вычислительные машины (ПЭВМ) в количестве 1 шт.

Стоимость одной ПЭВМ составляет 45000 рублей. Месячная норма амортизации К = 2,7%. Тогда за 4 месяцев работы расходы на амортизацию составят

$$P = 45000 \cdot 1 \cdot 0.027 \cdot 4 = 4860 \text{ рублей}$$

Вывод (см. таблицу 23): затраты на разработку программы составляют:

Таблица 23: Итоговая смета затрат

Номер стадии	**Название стадии**	**Затраты руб.**
1	Затраты на оплату труда	2 189 265
2	Дополнительная заработная плата	218 926
3	Отчисления в ФСС	842 867
4	Амортизация оборудования	5 512
5	Накладные расходы	4 597 456
Итого		7 854 027

7854027 рублей.

6.5 Определение стоимости программной продукции

Для определения стоимости работ необходимо на основании плановых сроков выполнения работ и численности исполнителей рассчитать общую сумму затрат на разработку программного продукта. Если ПП рассматривается и создается как продукция производственно-технического назначения, допускающая многократное тиражирование и отчуждение от непосредственных разработчиков, то ее цена P определяется по формуле:

$$P = K \cdot C + Pr,$$

где C — затраты на разработку ПП (сметная себестоимость); K — коэффициент учёта затрат на изготовление опытного образца ПП как продукции производственно-технического назначения ($K = 1.1$); Pr — нормативная прибыль, рассчитываемая по формуле:

$$Pr = \frac{C \cdot \rho_N}{100},$$

где ρ_N — норматив рентабельности, $\rho_N = 30\%$; Получаем стоимость программного продукта:

$$P = 1.1 \cdot 7854027 + 2356208 \cdot 0.3 = 10\ 995\ 638 \text{ рублей}$$

6.6 Расчет экономической эффективности

Основными показателями экономической эффективности является чистый дисконтированный доход (NPV) и срок окупаемости вложенных средств. Чистый дисконтированный доход определяется по формуле:

$$NPV = \sum_{t=0}^{T} (R_t - Z_t) \cdot \frac{1}{(1+E)^t},$$

T — горизонт расчета по месяцам;

t — период расчета;

R_t — результат, достигнутый на t шаге (стоимость);

Z_t — текущие затраты (на шаге t);

E — приемлемая для инвестора норма прибыли на вложенный капитал.

На момент начала 2015 года, ставка рефинансирования 8.25% годовых (ЦБ РФ), что эквивалентно 0.68% в месяц. В виду особенности разрабатываемого продукта он может быть продан лишь однократно. Отсюда получаем: $E = 0.0068$.

В таблице 24 находится расчёт чистого дисконтированного дохода. График его изменения приведён на рисунке 1.

Таблица 24: Расчёт чистого дисконтированного дохода

Месяц	**Текущие затраты, руб.**	**Затраты с начала года, руб.**	**Текущий доход, руб.**	**ЧДД, руб.**
Январь	939 352	939 352	0	-939 352
Февраль	1 783 352	2 712 705	0	-2 700 853
Март	1 783 352	4 486 057	0	-4 450 492
Апрель	1 783 352	6 259 410	0	-6 188 439
Мая	1 353 798	7 613 208	10 995 638	3 197 585

Согласно проведенным расчетам, проект является рентабельным. Раз-

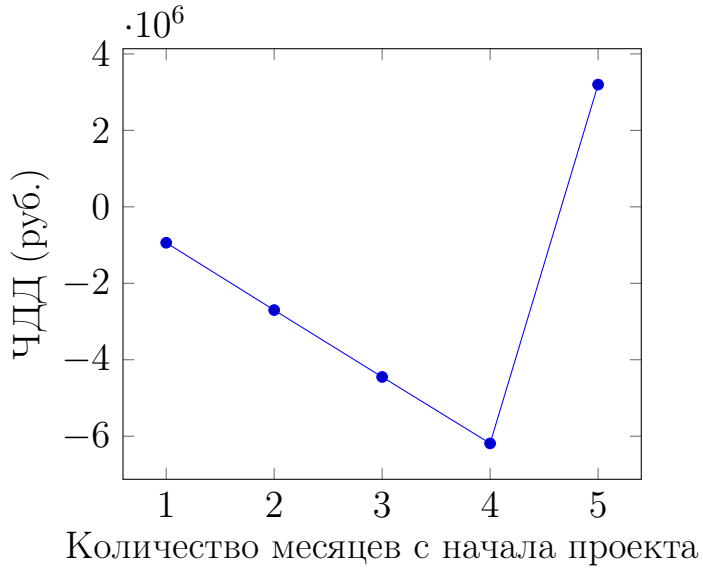

Рисунок 1 – График изменения чистого дисконтированного дохода

рабатываемый проект позволит превысить показатели качества существующих систем и сможет их заменить. Итоговый ЧДД составил: 3 197 585 рублей.

6.7 Результаты

В рамках организационно-экономической части был спланирован календарный график проведения работ по созданию подсистемы поддержки проведения диагностики промышленных, а также были проведены расчеты по трудозатратам. Были исследованы и рассчитаны следующие статьи затрат: материальные затраты; заработная плата исполнителей; отчисления на социальное страхование; накладные расходы.

В результате расчетов было получено общее время выполнения проекта, которое составило 105 рабочих дней, получены данные по суммарным затратам на создание системы для тональной классификации сообщений сети *Twitter*, которые составили 7 613 208 рублей.

Согласно проведенным расчетам, проект является рентабельным. Цена данного программного проекта составила 10 995 638 рублей, итоговый ЧДД составил 3 197 585 рублей.

ЗАКЛЮЧЕНИЕ

В данной работе исследован метод машинного обучения в качестве подхода к решению задачи тональной классификации. Для построения боллее точной классификационной модели применялись признаки на основе словарей оценочной лексики. Исследована возможность австоматического порождения таких словарей на основе сообщений сети *Twitter*.

Качество работы классификатора было протестировано в системах анализа тональности русского языка *SentiRuEval-2015* и *SentiRuEval-2016*. На последней демонстрируется довольно высокий результат (3-е место) относительно остальных участников тестирования. После проведения настройки классификатора и введения дополнительных признаков на основе словарей оценочной лексики, рассматриваемый в статье подход можно считать одним из наиболее успешных на сегодняшний день для проведения тональной классификации сообщений различных областей русскоязычной сети *Twitter*.

Приложение А. Извлечение сообщений из социальной сети Twitter

```
#!/usr/bin/python
import tweepy, json, sys, time, os, csv

class StdOutListener(tweepy.StreamListener):
    def on_data(self, data):
        t = json.loads(data)
        with open(self.filepath, 'a') as out: # how to save
            self.csv_writer = csv.writer(out, delimiter=';',
                quoting=csv.QUOTE_MINIMAL)
            self.csv_writer.writerow([t['id'], t['created_at'],
                t['user']['screen_name'].encode('utf-8'),
                t['text'].encode('utf-8'), t['retweet_count'],
                t['user']['favourites_count'], t['user']['statuses_count'],
                t['user']['followers_count'], t['user']['friends_count'],
                t['user']['listed_count']])
        return True
    def on_error(self, status):
        print status
    def __init__(self, filepath): # where to save
        self.filepath = filepath

# Create listener object
listener = StdOutListener(data_filepath)
auth = tweepy.OAuthHandler(keys['consumer_key'], keys['consumer_secret'])
auth.set_access_token(keys['access_token'], keys['access_secret'])
# https://dev.twitter.com/docs/streaming-apis
stream = tweepy.Stream(auth, listener)
# Setting up streamer
with open(log_file, 'w') as log:
    while (True):
        log.write("Reconnect ...")
        try:
            stream.filter(track=['twitter'], languages=['ru'])
        except Exception as e:
            log.write("Exception: %s"%(str(e)))
```

Приложение Б. Наиболее эмоциональные термы корпуса коротких текстов

В листингах 36-37 представлены наиболее эмоциональные слова *корпуса коротких текстров Ю. Рубцовой* [15]. Степень эмоциональности определяется на основе подхода, описанного в п. 1.3.3.

Листинг 36: "Эмоционально положительные термы"

```
"D", "DDD", "DDDD", "XD", "DD", "царевич", "xD", "DDDDD", "DDDDDD", "уругвай",
"#улыбнуло", "хрещатик", "баярлалаа", "#ЛУЧИРАДОСТИОТРАДОСТИ", "конгениальность",
"#рубль", "аге", "XDD", "позаимствовать", "ржач", "листаться", "бесподобный",
"Microsoft", "позитив", "реквестировать", "улыбнуть", "#музыка", "бугага",
"ехууу";
```

Листинг 37: "Эмоционально негативные термы"

```
"теракт", "погибший", "еб**ат", "цымбаларь", "#сми", "пострадавший", "гренобль",
"цег", "#Вконтакте", "траур", "критический", "однобокость", "михаэль", "поч",
"хныk", "блинн", "таскание", "пичаливать", "грусный", "пичаливать", "печалька",
"почемууу", "покоиться", "пращать", "смертница", "навидеть", "разочарованный",
"мазерать", "о", "скорбеть";
```

Приложение В. Список используемых тональных префиксов

Листинг 38: "Словосочетания используемые для составления тональных префиксов"

```
"имитировать": "-", "даже если": "-", "снижение": "-", "не назвать": "-",
"уменьшение": "-", "много": "+", "весьма": "+", "просто": "+", "сильно": "+",
"спад": "-", "все время": "+", "явный": "+", "снизить": "-", "совершенно": "+"
, "снизиться": "-", "ликвидировать": "-", "значительный": "+","ослабление": "-",
"разрушать": "-", "сильнейший": "+", "нельзя назвать": "-", "разительный": "+",
"колоссально": "+", "ничего": "-", "острый": "+", "повышать": "+",
"ослаблять": "-", "пресекать": "-", "масштабный": "+","повысить": "+",
"невообразимый": "+", "настолько": "+", "якобы": "-", "вырасти": "+",
"редкостный": "+", "сильный": "+", "чрезвычайный": "+", "имитация": "-",
"намного": "+", "заметный рост": "+", "жуть как": "+", "увеличить": "+",
"необычайно": "+", "безусловный": "+", "противодействовать": "-",
"большой": "+", "крайне": "+", "перестать": "-","увеличение": "+",
"масштабность": "+", "разрушение": "-", "безумно": "+", "абсолютный": "+",
"особенно": "+", "жутко": "+", "преодолеть": "-", "запросто": "+",
"отсутствие": "-", "рост": "+", "порядочный": "+", "усилить": "+","вполне": "+",
"не": "-", "недостаточно": "-", "избыток": "+", "лишиться": "-","отменить": "-",
"абсолютно": "+", "дикий": "+", "совсем": "+", "невероятно": "+", "очень": "+",
"лишить": "-", "утратить": "-", "нет никакой": "+", "ослабить": "-",
"запредельный": "+", "утрачивать": "-", "полный": "+", "нарастание": "+",
"по-настоящему": "+", "немыслимый": "+", "гораааздо": "+", "не просто": "+",
"ликвидация": "-", "снять": "-", "преодоление": "-", "избавиться": "-",
"повышение": "+", "падение": "-", "полностью": "+", "вырастать": "+",
"предельно": "+", "нереальный": "+", "противодействие": "-", "разрушить": "-",
"совершенный": "+", "рекордно": "+", "серьезный": "+", "снижаться": "-",
"снимать": "-", "нет": "-", "достаточно": "+", "уменьшать": "-", "отмена": "-",
"поистине": "+", "никакой": "-", "наибольший": "+", "неимоверно": "+",
"уменьшить": "-", "стопроцентный": "+", "гораздо": "+", "нереально": "+",
"отсутствовать": "-", "невиданный": "+", "запрет": "-", "великий вероятность":
"+", "увеличивать": "+", "конец": "-", "лишаться": "-", "защита от": "-",
"избавление": "-", "без": "-", "потеря": "-", "жгучий": "+", "колоссальный":
"+", "терять": "-", "утрата": "-", "самый": "+", "лишать": "-", "потерять": "-",
"совсем-совсем": "+", "дефицит": "-", "чрезвычайно": "+", "нейтрализация": "-"
, "усиливать": "+", "колоссальнейший": "+", "избавляться": "-", "усиление": "+",
```

Список литературы

[1] Pang B., Lee L., Vaithyanathan S. Thumbs up: sentiment classification using machine learning techniques // In Proceedings of the ACL-02 conference on Empirical methods in natural language processing, Association for Computational Linguistics. 2002. Т. 10. С. 79–86.

[2] Domingos P., Michael J. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss // Machine Learning, 29(2-3). 1997. С. 103–130.

[3] J. Thorsten. Text categorization with support vector machines: Learning with many relevant features // In Proc. of the European Conference on Machine Learning (ECML). С. 137–142.

[4] Rayson P., Garside R. Comparing corpora using frequency profiling. // WCC '00 Proceedings of the workshop on Comparing corpora. 2000. Т. 9. С. 1 – 6.

[5] Hatzivassiloglou V., Kathleen R. McKeown. Predicting the Semantic Orientation of Adjectives // ACL '98 Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics. 1997. С. 174 – 181.

[6] I-Iatzivassiloglou V., Kathleen R. MeKeown. A quantitative evaluation of linguistic tests for the automatic prediction of semantic markedness. // In Proceedings of the 83rd Annual Meeting of the ACL. 1995. С. 197 – 204.

[7] Turney P. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews // ACL '02 Proceedings of the 40th Annual Meeting on Association for Computational Linguistics. 2002. С. 417 – 424.

[8] Asch Vincent Van. Macro- and micro-averaged evaluation measure. 2013.

[9] Manning C., Raghavan P., Scutze H. Introduction to Information Retrieval. // Cambridge, UK: Cambridge University Press. 2008.

[10] Severyn A., Moschitti A. On the Automatic Learning of Sentiment Lexicons // Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL. C. 1397–1402.

[11] Saif M., Kiritchenko S., Xiaodan Z. NRC-Canada: Building the State-of-the-Art in Sentiment Analysis of Tweets // Second Joint Conference on Lexical and Computational Semantics. 2015. T. 2. C. 321–327.

[12] Лукашевич Н., Рубцова Ю. Объектно-ориентированный анализ твитов по тональности: резульаты и проблемы // Труды XVII Международной конференции DAMDID/RCDL'2015 «Аналитика и управление данными в областях с интенсивным использованием данных». 2015.

[13] SentiRuEval: testing object-oriented sentiment analysis systems in Russian / N. Loukachevitch, P. Blinov, E. Kotelnikov [и др.] // Proceedings of International Conference Dialog-2015. 2015. T. 2. C. 3 – 13.

[14] А. Завгородний. Анализ тональноасти предложений на материале сообщений из Твиттера. 2015.

[15] Ю. Рубцова. Корпус коротких текстов на русском языке на основе «постов» сети Twitter. URL: study.mokoron.com.

[16] Chih-Chung Chang, Chih-Jen Lin. LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology. 2011. URL: http://www.csie.ntu.edu.tw/ cjlin/papers/libsvm.pdf.

[17] Mystem — Технологии Яндекса. URL: https://tech.yandex.ru/mystem/.

[18] Loukachevitch N., Levchik A. Создание лексикона основных слов русского языка RuSentileks // misedings of Conference OSTIS-2016. 2016. C. 377–382.

[19] Loukachevich N., Rubtsova Yu. SentiRuEval-2016: Overcoming Time Gap and Data Sparsity in Tweet Sentiment Analysis, Proceedings of International Conference Dialog-2016 // Proceedings of International Conference Dialog-2016.

[20] Rusnachenko N. L. Use of lexicons to improve quality of sentiment classification. URL: http://www.dialog-21.ru/media/3469/rusnachenko.pdf.