САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Математико-механический факультет

Кафедра Системного Программирования

Лебедева Екатерина Андреевна

# Анализ эмоциональной окраски сообщений в микроблогах с помощью вероятностных моделей

Дипломная работа

Допущена к защите.
Зав. кафедрой:
д. ф.-м. н., профессор Терехов А. Н.

Научный руководитель:
ст.преподаватель Луцив Д. В.

Рецензент:
Тузова Е. А.

Санкт-Петербург
2014

SAINT-PETERSBURG STATE UNIVERSITY
Mathematics & Mechanics Faculty

Software Engineering Chair

Ekaterina Lebedeva

# Sentiment analysis of microblog data via probabilistic models

Graduation Thesis

Admitted for defence.
Head of the chair:
professor Andrey Terekhov

Scientific supervisor:
senior lecturer Dmitry Luciv

Reviewer:
Ekaterina Tuzova

Saint-Petersburg
2014

# Оглавление

# Введение

Не так давно грань между потребителями и создателями информации в интернете исчезла: на смену статическим страницам у всех пользователей появилась возможность публиковать свою информацию. Сейчас мы наблюдаем огромное количество видов создаваемых материалов: это может быть запись в блоге или на форуме, фотография или видеозапись на соответствующем ресурсе, отзыв в интернет-магазине,“статус” в социальной сети и многое другое. Совершенная простота размещения текстов от разных людей в одном месте в Интернете стала поводом для появления всевозможных веб-сайтов, собирающих мнения пользователей, например, о книгах, фильмах, товарах, и вот некоторые из них: Epinions[1], Rotten Tomatoes [2], Amazon[3], Яндекс.Маркет [4]. Прежде, чем что-то приобрести, покупатель ищет отзывы о серии необходимых товаров в интернете, читает десятки мнений различных людей, на основании этих мнений делает вывод о том, какой же продукт ему действительно подходит, и только после этого что-то покупает. Со временем текстов стало так много, что обработать их все за разумное время человеку просто не по силам. Именно такая ситуация стала причиной возникновения задачи анализа мнений: появилась необходимость в создании системы для автоматического поиска, классификации и представления точек зрения.

Анализ мнений – одно из направлений области обработки текстов на естественных языках. Саму задачу можно определить как вычислительное выявление субъективности в текстах и отношения авторов этих текстов к некоторым объектам. Изначально в качестве исследуемых данных использовались большие записи, состоящие из нескольких предложений, в которых явно прослеживались связь и контекст. Позже, с развитием социальных сетей, с появлением в них комментариев, “статусов” и коротких сообщений, пользовательский контент стал менее ёмким, но при этом более субъективным и превратился в бесконечный поток поступающей информации. Ярким примером этому является сервис микроблогов Twitter (Твиттер)[5]. С помощью этого сервиса пользователи распространяют свои взгляды на актуальные новости, связанные с разными интересными другим людям областями, такими как политика, экономика, бизнес и другие, рассказывают о купленных товарах, а также публикуют личную информацию, например, что они сейчас делают и в каком настроении находятся.

В этой работе речь пойдёт именно о сообщениях, характерных для Твиттера. Отличитель-

[1] epinions.com
[2] rottentomatoes.com
[3] amazon.es
[4] market.ya.ru
[5] twitter.com

ная особенность этой платформы в том, что у пользователя есть только 140 символов, чтобы выразить свои мысли или отношение к чему-либо. Каждое сообщение, называемое здесь “твит” и публикуемое пользователем в Твиттере, могут увидеть его подписчики – люди, которые связаны с ним в этой социальной сети. Подписчики (или иначе “читатели”) могут быть как односторонними, так и взаимными. Если человек увидел заинтересовавший его твит и разместил его на своей странице, то говорят, что он ретвитнул запись другого пользователя. В этом случае информация распространяется не только на подписчиков первоначального автора, но и на читателей того, кто сделал ретвит. Есть и другой вид взаимодействия пользовательской информации: упоминания. Если читатель захотел ответить на какой-либо твит, он это делает, вставив в начало своего сообщения псевдоним автора в Твиттере при помощи символа @ (@username), тем самым, упоминая его. В этом случае ответ увидят только те, кто читает обоих дискутирующих пользователей. Если упоминание происходит в середине твита, то он доступен точно так же, как и обычный, только упомянутому пользователю приходит отдельное оповещение. Механизмов социального взаимодействия в Твиттере больше нет, но этого достаточно, чтобы информация распространялась очень быстро и охватывала большую аудиторию.

Основной способ представления твитов – это представление в виде ленты. Пользователь, войдя на сайт, видит сообщения от всех читаемых им людей, отсортированные в порядке удаления времени от настоящего момента. Дальше он может перейти на страницу конкретного человека и прочитать только его сообщения, но обычно информация воспринимается именно в форме потока, уходящего назад, до момента регистрации читающего пользователя на сайте. Так он узнаёт, что нового произошло в жизни его знакомых, о чём рассказывают интересные ему аккаунты и какие события обсуждаются в мире. При помощи ретвитов информация действительно распространяется очень быстро.

Кроме чтения релевантных сообщений от читаемых людей можно пользоваться поиском по хештегу. Хештег – специальное слово, перед которым стоит символ # (#хештег). Наличие хештега подразумевает, что твит имеет какое-то отношение к объекту, обозначаему этим словом. Поиск по хештегам учитывает записи всех пользователей, поэтому количество получаемой информации здесь не просто большое, оно настолько велико, что страница с выдачей по популярным запросам почти никогда не является актуальной. Твиты в выдаче также организованы по принципу временной ленты, отдаляющейся от настоящего момента, поэтому, как только пользователь пишет новый запрос, кто-то может в то же время опубликовать запись с таким же хештегом. Проблема даже не в том, что появляются новые твиты, а в том, что

пользователь не успевает обработать все существующие. Хештег – это способ явно указать объект, о котором идёт речь в сообщении, но не все пользователи их ставят, поэтому поиск по ключевым словам даёт более полную информацию, хотя иногда и не носящую смысла, а её количество уж тем более становится неподъёмным.

Что же можно делать с огромным количеством коротких текстов на определённую тему, носящих, в основном, субъективный характер и не помещающихся вместе в голове обычного человека? Точнее, что можно хотеть с ними делать? Вот пример: выходит обновление какого-нибудь известного ПО, и компания публикует об этом новость на своей странице в Твиттере. Читатели Твиттера этой компании воспринимают сообщение о новой версии продукта и, во-первых, сами о ней узнают, во-вторых, могут ретвитнуть её для своих подписчиков, и к аудитории новости присоединятся другие пользователи, в-третьих, могут прокомментировать и показать тем самым своё отношение к событию. На всех этапах распространения информации о продукте компании важно, какую эмоциональную окраску она несёт. В такой ситуации понятно, в какой момент и что надо отслеживать. Но бывает, что человек написал в своём Твиттере мнение о продукте независимо от публикаций компании или её представителей. Тут уже начинается исследование эмоциональной окраски не среди комментариев к твитам и ретвитам конкретной записи, а в целом среди текстов, имеющих отношение к целевому объекту. Такая же задача встаёт, когда речь идёт об объектах из других областей: всё те же политика, экономика, события в обществе и прочее.

Таким образом, ставится задача разметить в соответствии с эмоциональной окраской множество твитов, имеющих отношение к конкретному объекту, заданному словом или словосочетанием, то есть найденных по поисковому запросу, с использованием особенностей именно этой социальной сети. Подобную задачу решают и для больших текстов при помощи лингвистического словарного подхода и вычислительно, методами машинного обучения. Цель данной работы – исследовать проблему для Твиттера и предложить вариант её решения с использованием аппарата вероятностных моделей. Для достижения этой цели можно сформулировать следующие шаги:

- проанализировать особенности задачи для микроблогов;
- сравнить базовые методы обучения с учителем для данных из микроблогов и выбрать лучший по параметрам точности, полноты результатов и времени обучения;
- предложить, обосновать и реализовать новый метод на основе выбранного;
- оценить результаты работы нового метода.

Кроме Твиттера сервисами микроблогов отчасти являются и другие социальные сети, например, ВКонтакте [6], Facebook [7], FourSquare [8], Instagram [9], поэтому задача, в целом, распространяема и на них, но, так как эти платформы предоставляют много других возможностей, для ведения микроблогов они используются гораздо меньше, чем Твиттер, и в этой работе рассматриваться не будут.

[6] vk.com
[7] facebook.com
[8] foursquare.com
[9] instagram.com

# 1. Существующие подходы

## 1.1. Конкретизация задачи

Задача анализа эмоциональной окраски текстов сводится к задаче классификации. В нашем случае имеется набор твитов, каждый из которых нужно отнести к одной из трёх категорий: положительные, нейтральные или отрицательные.

Иногда классификация происходит в два этапа и на обоих этапах является бинарной. На первом отделяются субъективные сообщения от объективных. Объективными в этом случае называются как раз те, которые не несут эмоциональной окраски и являются нейтральными в варианте с тремя классами. Второй этап делит субъективные тексты на положительные и отрицательные. В случае с Твиттером, где почти все сообщения субъективны, а критерии нейтральности можно сформулировать только в смысле “не положительное” и “не отрицательное”, будем для простоты рассматривать разделение на два класса.

Твит – это строка, состоящая из не более чем 140 символов. Она может содержать специальные слова, начинающиеся с определённых знаков: сразу после “@” пишется имя пользователя, с которым сообщение связано или к которому оно обращено, а после “#” находится так называемый хештег – слово, которое явно указывает на связь твита с объектом, который этим словом обозначается. Все твиты создаются пользователями, поэтому могут содержать опечатки, ошибки, сокращения, особую пунктуацию и прочие способы выражения мысли в коротком тексте. У каждого сообщения в Твиттере есть время, когда оно опубликовано, и автор. Если один твит является ответом на другой, то у первого есть ещё и ссылка на второй, то есть на “родительский”. Ретвиты содержат также данные о первоначальном размещении.

Вычислительно поставленная задача решается при помощи техник машинного обучения. Первое упоминание задачи анализа мнений относится к 2002 году. Тогда были рассмотрены стандартные решения методом обучения без учителя [18] и методом обучения с учителем [11]. В обеих статьях исследовались отзывы на специализированном ресурсе: хотелось выяснить, рекомендует или нет пользователь, оставивший отзыв, то, о чём он написал.

## 1.2. Методы обучения без учителя для задачи анализа мнений

В статье [11] автор предлагает алгоритм обучения без учителя для классификации отзывов на две категории: “рекомендует” и “не рекомендует”. Алгоритм состоит из трёх этапов.

1. Поиск словосочетаний с прилагательными или наречиями. Для дальнейшей работы

алгоритма нужны будут фразы, где одно из слов – прилагательное или наречие, а другое указывает на контекст. Если говорить про английский язык, то обычно для поиска второго слова достаточно взять соседнее.

2. Определение семантической ориентации словосочетания: положительное или отрицательное. На этом этапе используется PMM-IR алгоритм для выявления семантических ассоциаций [12]. При помощи этого алгоритма автор определяет схожесть словосочетания (*phrase*) с “excelent” и с “poor” и вычисляет его семантическую ориентацию (SO) по формуле

$$\mathrm{SO}(phrase) = \mathrm{PMI}(phrase, \text{“excelent”}) - \mathrm{PMI}(phrase, \text{“poor”}) \tag{1}$$

где функция $\mathrm{PMI}(x, y)$ как раз определяет, есть ли семантическая ассоциация между $x$ и $y$. Для уточнения этой формулы автор вводит отношение NEAR и функцию $\mathrm{hits}(x\,\mathrm{NEAR}\,y)$ на основе $\mathrm{PMI}(x, y)$, которая показывает, попадает ли $x$ в класс близких по смыслу к $y$ и считает семантическую ориентацию по новой формуле:

$$\mathrm{SO}(phrase) = \log_2 \frac{\mathrm{hits}(phrase\,\mathrm{NEAR}\,\text{“excelent”})\,\mathrm{hits}(\text{“poor”})}{\mathrm{hits}(phrase\,\mathrm{NEAR}\,\text{“poor”})\,\mathrm{hits}(\text{“excelent”})} \tag{2}$$

3. Определение семантической ориентации отзыва. Здесь считается средняя семантическая ориентация по всем словосочетаниям, найденным в отзыве, и определяется метка: “рекомендует”, если среднее получилось положительным, и “не рекомендует”, если оно получилось отрицательным.

В результате алгоритм показывает точность около 80% на отзывах, состоящих из нескольких предложений, то есть представляющих собой полноценный текст. Сложность этого подхода в том, что для работы второго этапа необходим корпус, собранный лингвистами вручную, то есть появляется безусловный человеческий фактор. Если вернуться к задаче для Твиттера, то особенности данных: опечатки, зачастую отсутствие контекста, пролонгирование гласных и прочее – обязывают постоянно расширять словари для определения семантической ориентации, а раз это делает человек, то либо это невозможно, либо составление такой или подобной базы нужно автоматизировать.

## 1.3. Методы обучения с учителем для задачи анализа мнений

### 1.3.1. Общая формулировка

Методы обучения с учителем предсказывают, к какому классу относится объект, на основании уже размеченного набора данных, который также называется тренировочным. Каждый метод такого вида должен уметь делать две вещи: обучаться на тренировочных данных и делать предсказание для новых. Слово “обучиться” здесь означает “построить функцию, которая для примеров из тренировочного набора сделает разметку, максимально близкую к действительной”. Другими словами, классификацию нужно смоделировать.

В статье [11] авторы рассматривают три таких подхода: метод опорных векторов, наивный байесовский классификатор и метод мультиномиальной регрессии. На каждом из них сперва остановимся подробнее, а затем сравним их на данных из Твиттера.

### 1.3.2. Наивный байесовский классификатор

Наивный байесовский классификатор [8] работает с условными вероятностями, наивно предполагая, что слова в предложении независимы. Этот простой классификатор хорошо показывает себя в решении задачи классификации текстов [7]. Сперва необходимо выбрать закон, по которому, как предполагается, распределены данные. Затем по размеченным примерам вычисляются параметры этого распределения, которые в дальнейшем используются для разметки. Предположим, что данные распределены по закону Бернулли. В таком случае класс $c^*$, к которому относится неизвестное сообщение $t$, вычисляется по формуле:

$$c^* = \underset{c}{\operatorname{argmax}} \frac{P(c) \sum_{i=1}^{m} P(x|c)^{x_i(t)}}{P(t)} \tag{3}$$

Здесь $x$ – это характеристики, по которым оцениваются сообщения, и всего их $m$, а $x_i(t)$ – величины, которые показывают, как $i$-ая характеристика представлена в сообщении $t$, $c$ – метка класса из множества всех меток $\mathscr{C}$. $P(c)$ и $P(x|c)$ – параметры модели, найденные при обучении классификатора.

### 1.3.3. Классификация методом опорных векторов

Метод опорных векторов (SVM) [17] работает по принципу разделения пространства на подпространства, соответствующие классам. Здесь тоже выбираются признаки, по которым измеряются примеры и согласно измерениям преобразуются в числовые векторы. Дальше

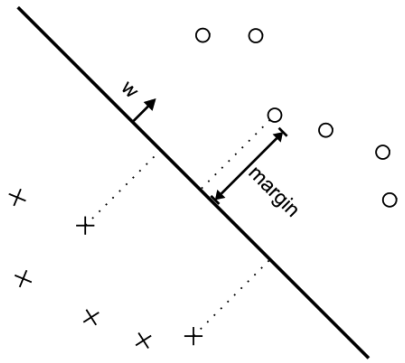


Рис. 1: Двоичная классификация SVM с линейным оператором ядра. $margin$ – расстояние от гиперплоскости до каждого из классов. $w$ – вектор нового примера, для которого делается предсказание.

работа идёт уже с этими векторами и пространством, в котором они располагаются. На этапе обучения задача метода – преобразовать пространство при помощи оператора ядра так, чтобы нашлись такие гиперплоскости, которые разделяют примеры из разных классов обучающей выборки. Предсказание делается согласно тому, в какую часть пространства относительно найденных гиперплоскостей попадает вектор, соответствующий новому примеру.

Иллюстрация разделения на два класса при помощи линейного ядра изображена на рисунке 1. Здесь показано, как строится равноудалённая от обоих множеств гиперплоскость и как новый вектор попадает в одно из них в зависимости от расположения относительно этой гиперплоскости.

### 1.3.4. Метод максимальной энтропии

Следующей рассмотрим классификацию при помощи метода максимальной энтропии [9]. В случае с разбиением на два класса это использование логистической регрессии для поиска распределения данных по классам. В отличие от наивного байесовского классификатора этот метод не предполагает независимости признаков. Это значит, что можно использовать для предсказания признаки разной природы, например, измерять $n$-граммы[10] и словосочетания в сообщении одновременно. Суть этого метода в том, что надо выбрать самую подходящую модель, удовлетворяющую всем естественным ограничениям. Модель описывается формулой

[10] $n$-буквенные сочетания, например, разбиение предложения **“Мама мыла раму.”** на 3-граммы выглядит так: |**Мам**|**а м**|**ыла**| **ра**|**му.**|

$$P(c|t,\lambda) = \frac{\exp\left(\sum_{i=1}^{N} \lambda_i x_i(c,t)\right)}{\sum_{c\in\mathscr{C}} \exp\left(\sum_{i=1}^{N} \lambda_i x_i(c,t)\right)} \tag{4}$$

Здесь $c$ – метка класса, $t$ – рассматриваемое сообщение, $x_i(c,t)$ – совместная представленность $i$-ого признака в классе $c$ и в примере $t$ , $N$ – количество признаков, $\lambda$ – вектор весов для всех признаков: чем больше вес, тем больше значимость этого признака для классификатора. На этапе обучения при помощи методов оптимизации вычисляется именно вектор весов признаков. При предсказании класса для нового примера снова нужно найти такое $c^*$ из множества меток, что рассматриваемая величина $P(c|t,\lambda)$ будет максимальной.

### 1.3.5. Сравнение работы методов на данных из Твиттера

Сравнение проводится на данных, собранных из Твиттера в 2009 году [6] и расширенных собранными из Твиттера самостоятельно. Здесь мы берём текст в сыром виде, без дополнительной обработки, и передаём алгоритму. В таблицах 1, 2 и 3 представлены результаты работы наивного байесовского классификатора, классификатора на основе метода опорных векторов и классификатора по принципу максимальной энтропии соответственно. Алгоритмы обучались на 1000000 размеченных примеров и предсказывали результаты для 386 новых.

Чтобы оценить качество классификации, обычно используют $F1-score$ – гармоническое среднее двух других: $precision$ (точность) и $recall$ (полнота). На языке вероятностей можно определить эти величины следующим образом: $precision$ – это вероятность того, что случайно выбранный твит попал в тот класс, которому он принадлежит на самом деле; $recall$ – это вероятность того, что случайно выбранный твит из класса при классификации в него и попадёт. Покажем, что значат эти величины более формально. Пусть зафиксирован класс $c$ и есть множество всех классифицируемых твитов $\mathscr{T}$ , которое делится на два множества: $\mathscr{T}_c$ – те, что на самом деле относятся к классу $c$, и $\mathscr{T} \setminus \mathscr{T}_c$ – те, у которых должны стоять другие метки. По результатам эксперимента определяются следующие величины:

$TP$ – количество твитов из $\mathscr{T}$, которым алгоритм поставил метку $c$;

$FP$ – количество твитов из $\mathscr{T} \setminus \mathscr{T}_c$, которым алгоритм поставил метку $c$;

$TN$ – количество твитов из $\mathscr{T} \setminus \mathscr{T}_c$, которым алгоритм поставил метку не $c$;

$FN$ – количество твитов из $\mathscr{T}_c$, которым алгоритм поставил метку не $c$.

Для проверки работы методов используются реализации из библиотеки Scikit-learn [16]. Для наивного байесовского классификатора предполагается, что данные распределены по закону Бернулли. В качестве характеристик берутся все слова, встретившиеся в обучающей выборке, и каждый твит преобразуется в вектор из целых чисел, где на месте $i$-ого слова ставится 0, если слово встретилось в сообщении, и 1, если нет.

| **Метка класса** | **Precision** | **Recall** | **F1-score** | **Количество** |
|---|---|---|---|---|
| -1.0 | 0.82 | 0.75 | 0.78 | 204 |
| 1.0 | 0.74 | 0.82 | 0.78 | 182 |
| avg / total | 0.79 | 0.78 | 0.78 | 386 |

Таблица 1: Классификация наивным байесовским классификатором. Время обучения – 1 с.

| **Метка класса** | **Precision** | **Recall** | **F1-score** | **Количество** |
|---|---|---|---|---|
| -1.0 | 0.86 | 0.73 | 0.79 | 204 |
| 1.0 | 0.74 | 0.87 | 0.80 | 182 |
| avg / total | 0.80 | 0.80 | 0.79 | 386 |

Таблица 2: Классификация методом опорных векторов. Время обучения – 750 с.

| **Метка класса** | **Precision** | **Recall** | **F1-score** | **Количество** |
|---|---|---|---|---|
| -1.0 | 0.87 | 0.71 | 0.78 | 204 |
| 1.0 | 0.73 | 0.88 | 0.80 | 182 |
| avg / total | 0.80 | 0.79 | 0.79 | 386 |

Таблица 3: Классификация методом максимальной энтропии. Время обучения – 437 с.

Как уже было сказано, всего в тестовой выборке было 386 сообщений, из которых 182 были помечены “+”, а 204 – “−”. Из таблиц 1, 2 и 3 видно, что все методы показали примерно одинаковую точность и полноту работы, но время обучения при этом у наивного байесовского классификатора отличается на порядок от двух других. Для постановки задачи, когда предсказание делается для данных из микроблогов, время обучения при равных показателях $F1 - score$ является решающим, так как меняются темы, о которых пишут в Интернете, а значит меняются лексика и способы выражения отношения к ним – классификатору нужно подстраиваться под эти обстоятельства, постоянно переобучаясь.

Как итог сравнения за основу для улучшения стоит взять наивный байесовский классификатор. Стоит сказать, что для моделирования данных можно было без дополнительных усилий выбрать и закон мультиномиального распределения. В случае с твитами это означает, что в векторе, в который этот твит переводится, каждому слову из обучающей выборки сопоставляется количество раз, которое оно встретилось в сообщении. Когда тексты короткие,

в случае с мультиномиальной моделью векторы получаются почти всегда из 0 и 1, поэтому отдельно его можно и не рассматривать.

## 1.4. Связанные идеи

### 1.4.1. Анализ графов слов

Идея, предложенная в статье [3], основывается на построении графа для получения информации о классах. Для положительных и отрицательных слов строятся два графа соответственно. Их структуры восстанавливаются из обучающей выборки. Для присваивания метки новому примеру предлагается использовать ещё и словарь синонимов: вероятность слова оказаться в классе учитывает количество попаданий его самого и всех его синонимов в этот класс.

### 1.4.2. Использование онтологий

В статье [10] предлагается для каждой конкретной темы строить онтологии[11], которые уточняют запросы, сужая все найденные в поиске твиты до тех, в которых действительно говорится об этом объекте. Тема запроса заменяется на пару “корень онтологии” и “свойство”, например, если исходный запрос – “смартфон”, то это и есть тема онтологии, а свойствами могут быть “android”, “iphone” и “батарейка”. В этом случае вместо одной попытки поиска будет уже три, но с более релевантной выдачей: “смартфон android”, “смартфон iphone” и “смартфон батарейка”. При помощи коммерческой программы с закрытым исходным кодом OpenDover[12] авторами статьи производится дальнейший анализ окраски полученных результатов выдачи.

Авторы статьи [15] предлагают использовать онтологии в момент подготовки найденных данных к разметке. Авторы предлагают три варианта: ставить категорию с предыдущего уровня[13] рядом со словом в сообщении, для которого эта категории найдена; заменять слово на более общую категорию; рассматривать в примерах распределение слов как условное распределение от категорий. Статья на самом деле про последний способ, потому что здесь вероятность попадания примера в класс считается немного необычно: учитываются не только распределения слов, но и дополнительные признаки. Точнее, от величин дополнительных

[11] Онтология – схема области знаний.

[12] opendover.nl

[13] Для построения онтологии область знаний разбита на категории, которые сформированы в направленный ациклический граф. Для каждой категории, если это не корень онтологии, есть обощающая категория, находящаяся на предыдущем уровне.

признаков зависит распределение слов в сообщении. Предложенный метод является уточнением наивного байесовского классификатора при помощи онтологий.

### 1.4.3. Расширение модели темами

Другой взгляд на уточнение модели описан в статье [2]. Здесь предлагается расширять стандартную модель наивного байесовского классификатора условным распределением слов, зависящим от темы. Множество тем не фиксировано, но сказано, сколько их вообще. Перед обучением модели говорится, что есть какое-то количество тем, по которым нужно разделить слова из обучающей выборки. Например, вероятность встретить слово “Шекспир” в той же теме, что и “Пушкин”, больше, чем в той же теме, что и “телефон”.

# 2. Особенности задачи для данных из микроблогов

## 2.1. Основные характеристики данных

Микроблоги – это, в первую очередь, сервисы для упрощения публикации и восприятия пользовательских данных. Обычно сообщения в микроблогах состоят из одного или пары предложений, а для Твиттера есть строгое ограничение на длину твита – 140 символов. В 140 символов пользователям платформы необходимо уместить контекст, своё отношение к теме и, возможно, ссылку на фотографию, Интернет-ресурс или другой медиа объект. Часто контекст восстанавливается из окружающего мира, то есть пользователь пишет о том, что волнует Интернет в этот момент, и люди, владея этой информацией, сопоставляют высказывание с реальными событиями. У компьютера так просто быть в курсе обсуждаемых тем не получается, поэтому на восстановление контекста рассчитывать не приходится.

Платформы для ведения микроблогов также являются социальными сетями, где пользователи могут взаимодействовать друг с другом. В Твиттере, например, кроме социальных графов можно наблюдать графы, в которые выстраиваются сами сообщения: пользователи могут отвечать на твиты, а также размещать у себя твиты других пользователей, в терминологии платформы это называется “ретвитить”. Информация о социальных взаимодействиях может уточнять результаты классификации, например, есть интуитивное предположение, что ответ на отрицательно окрашенное сообщение тоже попадёт в класс негативных.

## 2.2. Особенности текстов

### 2.2.1. Смайлы

Для выражения эмоций в тексте пользователи ставят смайлы. Смайл – это набор символов, условно иллюстрирующий выражение лица автора, а точнее его настроение. Все смайлы можно поделить на восточные и западные по географии их использования, последние приведены в таблице 4 с метками, соответствующими их эмоциональной окраске. В случае с короткими текстами нет более простого способа отметить своё отношение к теме, чем поставить смайл, но не все пользователи так делают, поэтому размечать сообщения с их помощью в общем случае не получится. Есть и более сложные конструкции из скобок, двоеточий и других симовлов, но они используются не так часто и обычно означают уже не просто отношение, а какие-то действия или объекты, то есть эмоциональной окраски не несут.

| Смайл | Метка | Смайл | Метка | Смайл | Метка | Смайл | Метка | Смайл | Метка |
|---|---|---|---|---|---|---|---|---|---|
| :-) | + | :) | + | :o) | + | :] | + | :3 | + |
| :c) | + | :> | + | =] | + | 8) | + | =) | + |
| :} | + | :^) | + | :>) | + | :-D | + | :D | + |
| 8-D | + | 8D | + | x-D | + | xD | + | X-D | + |
| XD | + | =-D | + | =D | + | =-3 | + | =3 | + |
| B^D | + | :-)) | + | >:[ | - | :-( | - | :( | - |
| :-c | - | :c | - | :-< | - | :>C | - | :< | - |
| :-[ | - | :[ | - | :{ | - | ;( | - | :-‖ | - |
| :@ | - | >:( | - | :’-( | - | :’( | - | :’-) | + |
| :’) | + | D:< | - | D: | - | D8 | - | D; | - |
| D= | - | DX | - | v.v | - | D-‘: | - | :* | + |
| :^* | + | ( | + | }{ | + | ) | + | ;-) | + |
| ;) | + | *-) | + | *) | + | ;-] | + | ;] | + |
| ;D | + | ;^) | + | :-, | + | >:P | + | :-P | + |
| :P | + | X-P | + | x-p | + | xp | + | XP | + |
| :-p | + | :p | + | =p | + | :-Þ | + | :Þ | + |
| :þ | + | :-þ | + | :-b | + | :b | + | d: | + |
| >:\ | - | >:/ | - | :-/ | - | :-. | - | :/ | - |
| :\ | - | =/ | - | =\ | - | :L | - | =L | - |
| :S | - | >.< | - | :\| | - | :-\| | - | :$ | - |
| O:-) | + | 0:-3 | + | 0:3 | + | 0:-) | + | 0:) | + |
| 0;^) | + | O_O | - | ø/ | + | <3 | + | </3 | - |

Таблица 4: Эмоциональная окраска смайлов.

Кроме ASCII смайлов есть ещё и графические – это картинки, которые вставляются в текст. В современных веб-сервисах и мобильных приложениях используется графический язык Emoji[14] для записи слов, эмоций и действий. На рисунке 2 изображены некоторые известные графические смайлы, которые используются в социальной сети Facebook[15]. Обычно для каждого из них есть ASCII аналог, причём не один. Набирая сообщение на клавиатуре компьютера или ноутбука, удобнее поставить двоеточие со скобкой, но смартфоны и планшеты предоставляют все удобства для вставки улыбчивых картинок: наряду с русской и английской клавиатурой, например, на них можно подключить и клавиатуру графического языка Emoji.

Так как смайлы являются своего рода разметкой сообщений самими пользователями, их необходимо использовать при анализе эмоциональной окраски. В этой работе будет рассмотрено применение символьных и графических улыбок для сбора корпуса твитов и для предобработки данных непосредственно перед классификацией.

[14] www.emoji-cheat-sheet.com
[15] facebook.com

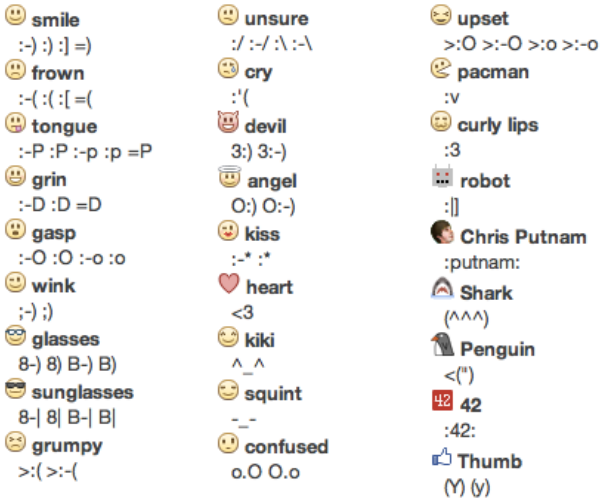


Рис. 2: Некоторые графические смайлы, используемые в Facebook.

### 2.2.2. Хештеги

Ещё одна особенность общения в микроблогах – хештеги. Пользователь помечает в своём сообщении слово, ставя перед ним “#”, тем самым показывая связь объекта, обозначаемого этим словом, и всего твита. Платформы для микроблогов предлагают возможность искать по хештегам, выбирать из них популярные и следить за потоками актуальной информации. Многие хештеги используются в течение короткого периода времени, но затем именно по ним можно найти информацию, которая когда-то была актуальной и понадобилась через несколько месяцев. Например, организаторы мероприятий стараются придумывать уникальный хештег, размещать его на информационных стендах, чтобы участники следили за твитами друг друга и распространяли информацию по всему Интернету. Можно сказать, что это повествовательная функция хештегов, точнее, тех из них, которые указывают на объект, – они могут помочь осуществлять поиск сообщений на определённую тему.

Другую функцию этих специальных слов-ассоциаций можно назвать описательной. Именно такие хештеги можно использовать в определении эмоциональной окраски текстов. В работе [14] предложен способ классификикации хештегов по их эмоциональной окраске. Авторы предлагают читателям посмотреть на 20 самых популярных хештегов из каждой группы. Для наглядности в таблице 5 приведены первые пять для каждой эмоции.

Использование хештегов непосредственно для оценки эмоциональной окраски можно считать примерно таким же, как и у смайлов, но лишь тогда, когда слово однозначно относится либо к положительным, либо к отрицательным. В противном случае они либо становятся обычными словами: без символа “#” они участвуют в классификации наравне с другими,

| Привязанность | Ярость | Страх | Наслаждение | Грусть |
|---|---|---|---|---|
| #youthebest | #godie | #hatespiders | #thankinggod | #catlady |
| #yourthebest | #donttalktome | #freakedout | #thankyoulord | #buttrue |
| #hyc | #fuckyourself | #creepedout | #thankful | #singleprobs |
| #yourethebest | #getoutofmylife | #sinister | #superexcited | #singleproblems |
| #alwaysandforever | #irritated | #wimp | #tripleblessed | #lonelytweet |

Таблица 5: Самые популярные хештеги для пяти чувств: привязанности, ярости, страха, наслаждения и грусти.

либо уточняют вероятность сообщения попасть в тот или иной класс при помощи подсчёта условных вероятностей, где условием и является хештег.

### 2.2.3. Сокращения, пролонгирования и пунктуация

Тексты в микроблогах содержат не только уточняющую информацию, но и отчасти мешающую. Её нужно научиться использовать, так как специфика сообщений не позволяет хоть что-то выкидывать.

Ограничение в 140 символов заставляет людей сокращать слова, причём как при помощи общеизвестных аббревиатур, например, “СПбГУ” – это Санкт-Петербургский Государственный Университет, так и при помощи жаргонных конструкций: “h8” – это на самом деле hate. На примере последнего видно, что избавиться от этого слова было бы расточительно, но вряд ли “h8” внесло бы вклад в вероятность сообщения попасть в класс отрцательных такой же, как и слово “hate”. Получается, сокращения нужно уметь переводить.

Когда пользователям кажется, что длина сообщения не такая уж и маленькая, они используют пролонгирования гласных – ещё один способ выражать обеспокоенность темой твита. Автор преумножает гласную в слове, изображая её продолжительное звучание, то есть, например, крик. Так “nooooooo” будет, скорее всего, означать категорическое несогласие, а “so cuuuute” – умиление. Таким образом, каждое такое слово что-то значит, но классификатор может об этом не знать, значит, нужно рассказывать классификатору какими-то другими способами, что это важное слово и какое из известных является его менее эмоциональным аналогом.

Авторская пунктуация может рассказать об эмоциональной окраске сообщения не меньше, чем смайлы. Например, в нейтральных твитах крайне редко встречаются восклицательные знаки. Впрочем, однозначно классифицирующих особенностей пунктуации не так и много: наличие восклицательных знаков указывает на наличие эмоциональной окраски, при этом нельзя без дополнительного анализа сказать, какой именно; сочетание “?!”, скорее все-

го, будет означать недоумение, то есть классифицируется как отрицательное; многоточия обычно говорят о нейтральности.

## 2.3. Использование особенностей текстов для предобработки

Смайлы, хештеги, сокращения, пролонгирования и пунктуация – это то, про что классификатор уже не знает, то есть перед подачей ему сообщения необходимо преобразовать это сообщение так, чтобы все перечисленные особенности не выбивались и превратились в обычные слова.

Смайлы, перечисленные в таблице 4, заменяются в тексте на соответствующую им метку. Это делается для того, чтобы в обучающей выборке слово “+” встретилось больше раз среди положительных твитов, тем самым, в вероятность попасть в класс положительных “+” даст больший вклад, чем просто “:)”. Так же, заменой на “+” и “–”, обрабатывается пунктуация.

К смайлам, заменяемым на метки, добавляется замена некоторых однозначно классифицирующихся хештегов. Происходит это по той же причине, что и со смайлами. Если замена хештега не произошла, то считается, что он должен стать обычным словом, то есть “#” из начала пропадает, и дальше работа происходит уже без учёта того, что это хештег.

Если в сообщении встречается неизвестное слово, его стоит проверить на наличие в словаре сокращений. В данной работе используется словарь “No slang”[16], к которому программа обращается во время подготовки данных к подаче классификатору. Запрос к словарю происходит в онлайн-режиме, и для обработки сокращений нужно подключение к Интернету.

Повторения гласных убирать совсем не нужно: достаточно сократить количество повторяющихся гласных до двух, то есть “nooooooo” заменится на “noo”. В этом случае слово “noo” может встретиться в обучающей выборке, в отличие от “nooooooo”, где именно семь, а не восемь или девять букв “o”. Таким образом слово “no” уже не то же самое, что “noo”, но все, сколько угодно длинные продолжения гласной “o” сведутся к одному и тому же эмоциональному “noo”, которое даст каждому из таких слов с продолжениями одинаковый повод попасть в класс “–”.

[16] http://www.noslang.com/

# 3. Модель классификатора

## 3.1. Наивный байесовский классификатор

### 3.1.1. Описание классификатора

Сперва рассмотрим простой случай преобразования твита в числовой вектор. Для этого построим словарь на основе обучающих данных: каждое слово – это отдельный признак, который измеряется в конкретном сообщении. Пусть в тренировочной выборке было $D$ уникальных слов, тогда числовой вектор, соответствующий твиту, выглядит следующим образом: $\mathbf{x} = \{x_1, \ldots, x_D\}$. Здесь каждая компонента $x_j$ показывает, как представлено $j$-ое слово в сообщении $\mathbf{x}$. Будем считать, что данные распределены по закону Бернулли, тогда представленность слова определяется просто его наличием или отсутствием, то есть $x_j = 1$, если $j$-ое слово встретилось в сообщении, и 0 в противном случае. Задача – сопоставить каждому такому вектору наиболее правдоподобную метку класса, которую будем обозначать $y$. Множество всех классов обозначим $\mathscr{C}$. Получается, что для каждого класса $c \in \mathscr{C}$ необходимо посчитать функцию правдоподобия $p(\mathbf{x}|y = c)$. Наивный байесовский классификатор действительно наивен в том смысле, что он предполагает независимость признаков. Это позволяет считать искомую вероятность как произведение вероятностей для каждого признака:

$$p(\mathbf{x}|y = c, \boldsymbol{\theta}) = \prod_{j=1}^{D} p(x_j|y = c, \boldsymbol{\theta}_{jc}) \tag{5}$$

Здесь $\boldsymbol{\theta}$ – это матрица-параметр распределения Бернулли, который ищется на этапе обучения классификатора, то есть по известным парам $\mathbf{x}$, $y$. На самом деле элемент этой матрицы $\boldsymbol{\theta}_{jc}$ – это вероятность того, что $j$-ое слово встретится в примерах из класса $c$.

### 3.1.2. Обучение и предсказание

На этапе обучения классификтора нужно найти, как уже было сказано, параметры распределения, моделирующего данные, и априорные вероятности классов $\boldsymbol{\pi}$. Сперва обозначим тренировочные данные $\mathscr{D}$, тогда $\mathbf{x}_i \in \mathscr{D}$ – это пример из обучающей выборки, а $x_{i1}, \ldots, x_{iD}$ – количественные характеристики признаков для примера $\mathbf{x}$, и $y_i$ – метка класса для него.

Посмотрим на $i$-ый пример, тогда вычисление вероятности пары $\mathbf{x}_i$, $y_i$ при условии параметров $\boldsymbol{\theta}$ происходит по формуле:

$$p(\mathbf{x}_i, y_i|\boldsymbol{\theta}) = p(y_i|\boldsymbol{\pi}) \prod_{j=1}^{D} p(x_{ij}|\boldsymbol{\theta}_j) = \prod_{c\in\mathscr{C}} \pi_c^{\mathbb{I}(y_i=c)} \prod_{j=1}^{D} \prod_{c\in\mathscr{C}} p(x_{ij}|\boldsymbol{\theta}_{jc})^{\mathbb{I}(y_i=c)} \quad (6)$$

Переходя ко всему набору данных получается, что нужно максимизировать следующую величину

$$p(\mathscr{D}|\boldsymbol{\theta}) = \prod_{c\in\mathscr{C}} \pi_c \prod_{j=1}^{D} \prod_{c\in\mathscr{C}} \prod_{i:\, y_i=c} p(x_{ij}|\boldsymbol{\theta}_{jc}) \quad (7)$$

Теперь необходимо найти сответствующие этому максимуму параметры $\hat{\boldsymbol{\pi}}$ и $\hat{\boldsymbol{\theta}}_{ij}$. Произведя все необходимые вычисления, получим, что

$$\hat{\pi}_c = \frac{N_c}{N} \quad (8)$$

$$\hat{\theta}_{jc} = \frac{N_{jc}}{N_c} \quad (9)$$

В этих формулах $N$ – количество примеров в обучающей выборке, $N_c$ –количество элементов в классе $c$, $N_{jc}$ – количество раз, которое $j$-ое слово встретилось в примерах из класса $c$. Обучение модели происходит за $O(N\,D)$ шагов.

Для предсказания класса $y$ для нового примера $\mathbf{x}$ необходимо максимизовать следующее выражение по всем классам $c \in \mathscr{C}$

$$\hat{\pi}_c \prod_{j=1}^{D} (\hat{\theta}_{jc})^{\mathbb{I}(x_j=1)} (1 - \hat{\theta}_{jc})^{\mathbb{I}(x_j=0)} \quad (10)$$

Класс, для которого оно получилось максимальным, и считается наиболее правдоподобным для данного примера.

### 3.1.3. Проблемы подхода

Основная проблема описанного подхода заключается в том, что не обязательно все слова (признаки) встретятся во всех классах. Например, если $j$-ое слово не представлено в классе $c$, то $N_{jc} = 0$ и вероятность того, что рассматриваемое сообщение попадёт в класс $c$ тоже станет нулевой. От этой проблемы может избавить переход к байесовскому подходу, когда параметры становятся случайными величинами со своими распределениями. Ещё для устранения этой проблемы используется сглаживание Лапласа[5], когда и к числителю, и к знаменателю прибавляется по 1. Это частный случай байесовского подхода.

Другая проблема, с которой необходимо бороться, – это невозможность расширения сло-

варя: если слова не было в обучающей выборке, то классификатор снова сталкивается с нулевыми вероятностями, но теперь для всех классов.

## 3.2. Вероятностная модель нового метода

### 3.2.1. Переход к байесовскому подходу

Чтобы избавиться от проблемы с нулевыми вероятностями “честным” способом, перейдём к байесовскому наивному байесовскому классификатору. В этом случае параметры вместо точечных величин представляются случайными величинами с априорными распределениями. В модели классификатора параметры – это вероятности, то есть их значения ограничены на отрезке $[0, 1]$. Так распределение для каждого из параметров должно задавать случайную величину, ограниченную на отрезке. Для моделирования параметра $\boldsymbol{\pi}$ будем использовать распределение Дирихле $\mathrm{Dir}(\boldsymbol{\alpha_0})$, которое устанавливает закон распределения многомерной случайной величины, где все компоненты – величины из отрезка $[0, 1]$ и суммируются в 1. Считаем также, что каждый из параметров $\theta_{jc}$ имеет Бета-распределение на отрезке $[0, 1]$ $\mathrm{Beta}(\beta_0, \beta_1)$. Все параметры считаются независимыми, поэтому априорная вероятность параметров считается по формуле

$$p(\boldsymbol{\theta}) = p(\boldsymbol{\pi}) \prod_{j=1}^{D} \prod_{c \in \mathscr{C}} p(\theta_{jc}) \tag{11}$$

Задача теперь получить апостериорную вероятность на основании данных $\mathscr{D}$ из обучающей выборки и выяснить, попадёт ли она в тот же класс, что и априорная, то есть получится ли для неё выражение вида $\mathrm{Dir}(\ldots) \cdot \mathrm{Beta}(\ldots) \cdot \ldots \cdot \mathrm{Beta}(\ldots)$. По теореме Байеса искомую вероятность можно посчитать по следующей формуле

$$p(\boldsymbol{\theta}|\mathscr{D}) = \frac{p(\boldsymbol{\theta}) \cdot p(\mathscr{D}|\boldsymbol{\theta})}{p(\mathscr{D})} \tag{12}$$

Знаменатель выражения не зависит от параметров, поэтому рассмотрим отдельно числитель. Если расписать плотности распределений и сгруппировать полученные произведения, получится вывод, сокращённая версия которого приведена ниже. $\boldsymbol{M}$ здесь некоторая постоянная.

$$p(\boldsymbol{\theta}) \cdot p(\mathscr{D}|\boldsymbol{\theta}) = M \cdot \prod_{c \in \mathscr{C}} \pi_c^{\alpha_0 c - 1} \cdot \prod_{j=1}^{D} \prod_{c \in \mathscr{C}} \theta_{jc}^{\beta_0} (1 - \theta_{jc})^{\beta_1} \cdot \prod_{\mathbf{x} \in \mathscr{D}} \prod_{c \in \mathscr{C}} \pi_c \prod_{j=1}^{D} \theta_{jc}^{x_{jc}} \tag{13}$$

$$= M \cdot \prod_{c \in \mathscr{C}} \pi_c^{\alpha_{0c} + N_c} \cdot \prod_{c \in \mathscr{C}} \prod_{j=1}^{D} \theta_{jc}^{\beta_0 + N_c - N_{jc}} (1 - \theta_{jc})^{\beta_1 + N_{jc}} \tag{14}$$

В выражении записано не что иное, как произведение плотностей распределения Дирихле и Бета-распределений. Так вероятность $p(\boldsymbol{\theta}|\mathscr{D})$ осталась в том же классе, что и $p(\boldsymbol{\theta})$.

$$p(\boldsymbol{\pi}|\mathscr{D}) = \mathrm{Dir}(N_1 + \alpha_{01}, \dots, N_c + \alpha_{0c}, \dots) \tag{15}$$

$$p(\theta_{jc}|\mathscr{D}) = \mathrm{Beta}(N_c - N_{jc} + \beta_0, N_{jc} + \beta_1) \tag{16}$$

Обозначим получившиеся параметры как $\alpha_0^*$, $\beta_0^*$ и $\beta_1*$ соответственно.

Остаётся понять, как при помощи полученных данных можно предсказать класс для новых примеров. Рассмотрим пример $\mathbf{x}$. Хотим найти для него значение метки $y$. Для этого нужно максимизировать $p(y = c|\mathbf{x}, \mathscr{D})$ по всем $c \in \mathscr{C}$.

$$p(y = c|\mathbf{x}, \mathscr{D}) = \frac{p(y = c|\mathscr{D}) \cdot p(\mathbf{x}|y = c, \mathscr{D})}{p(\mathbf{x}|\mathscr{D})} \tag{17}$$

Снова будем смотреть только на числитель. Первый его множитель на самом деле является математическим ожиданием $\boldsymbol{\pi}_c$ по $\boldsymbol{\pi}$ с плотностью распределения Дирихле. Краткая версия вывода приведена ниже, в 18, 19 и 20. Здесь для преобразования используется техника маргинализации, а переход к 20 следует из определения математического ожидания.

$$p(y = c|\mathscr{D}) = \int_{\boldsymbol{\pi}\in\Pi} p(y = c, \boldsymbol{\pi}|\mathscr{D}) d\boldsymbol{\pi} \tag{18}$$

$$= \int_{\boldsymbol{\pi}\in\Pi} p(\boldsymbol{\pi}|\mathscr{D}) p(y = c|\mathscr{D}, \boldsymbol{\pi}) d\boldsymbol{\pi} \tag{19}$$

$$= \mathop{\mathbb{E}}_{\boldsymbol{\pi}\sim\mathrm{Dir}(\alpha_0^*)} \boldsymbol{\pi}_c \tag{20}$$

Аналогично первому разбираем второй множитель. Снова воспользуемся техникой маргинализации для записи 21. Теперь раскрываем вероятность в произведение вероятностей – так можно делать, потому что величины $x_i$ и $\theta_{*c}$ независимы, и получаем формулу 23. Далее переписываем интеграл в обозначениях математического ожидания и, согласно посчитанному ранее, приходим к тому, что благодаря независимости параметров получили произведение математических ожиданий случайных величин $\theta_{jc}$ с плотностью Бета-распределения.

$$p(\mathbf{x}|y=c,\mathscr{D}) = \int_{\theta\in\Theta} p(\mathbf{x},\theta_{*c}|y=c,\mathscr{D})d\theta_{*c} \tag{21}$$

$$= \int_{\theta\in\Theta} p(\mathbf{x}|y=c,\mathscr{D},\theta_{*c})p(\theta_{*c}|y=c,\mathscr{D})d\theta_{*c} \tag{22}$$

$$= \mathop{\mathbb{E}}_{p(\theta_{*c}|y=c,\mathscr{D})} p(\mathbf{x}|y=c,\mathscr{D},\theta_{*c}) \tag{23}$$

$$= \prod_{j=1}^{D} \mathop{\mathbb{E}}_{\theta_{*c}\sim\mathrm{Beta}(\beta_0^*,\beta_1^*)} \theta_{jc}^{x_{ij}} \tag{24}$$

Так как параметры вычислены на этапе обучения, а формула математических ожиданий для соответствующих распределений известна, остаётся только подставить их в выражение 17, максимум которого мы ищем.

### 3.2.2. Использование n-грамм для измерения признаков

Альтернативный взгляд на подсчёт вероятностей сообщений – это разбиение сообщения на $n$-граммы, то есть $n$-буквенные сочетания. Такое изменение позволит искусственно ослабить требование о независимости признаков. Допустим, что исходное сообщение разбивается на триграммы. Три здесь выбрано, так как вычислительно количество всех возможных триграмм ещё не так велико, но уже информативно. Переобозначим $x_i$, теперь они представляют триграммы. Пусть триграмм в сообщении всего $L$, тогда вектор $\mathbf{x} = x_1, \ldots, x_L$ описывает представленность триграмм в сообщении. Вероятность того, что сообщение попадёт в класс $c$, в этом случае подсчитывается по следующей формуле

$$p(\mathbf{x}, y=c) = \prod_{j=2}^{L} p(x_j|x_{j-1}, y=c) \tag{25}$$

Модель классификатора немного изменится, а новыми параметрами станут как раз $p(x_j|x_{j-1}, y=c)$, которые и будет искать алгоритм на этапе обучения.

Перейдём к предыдущим обозначениям и выясним, как изменятся формулы подсчёта параметров и искомой. Вектор $\mathbf{x}$ снова показывает представленность каждой триграммы из обучающей выборки в рассматриваемом примере: 1 стоит на месте присутствующей триграммы и 0 на месте отсутствующей. Введём также функцию, которая поможет понимать последовательность триграмм в примере: $n(x_j)$ показывает, какой по счёту встретилась $j$-ая триграмма в примере. Если не встретилась, то пускай значение этой функции будет $-\infty$. Для простоты считаем, что каждая триграмма встретилась только один раз. В формуле пересчёта

параметра $\boldsymbol{\pi}$ 15 ничего не изменится, так как на априорные вероятности класса изменение не повлияет. Пусть $D$ – это теперь количество всех триграмм, встретившихся в обучающей выборке. Дополнительно обозначим $N_{jkc}$ – количество пар из триграммы с номером $k$ и следующей прямо за ней триграммы с номером $j$ в классе $c$. $\theta_{jkc}$ – вероятность встретить триграмму с номером $k$ сразу за триграммой с номером $j$ в классе $c$. Тогда пересчёт параметра $\theta_{jkc}$ будет производиться по формуле, аналогичной 16, но с учётом связи триграмм.

$$p(\theta_{jkc}|\mathscr{D}) = \text{Beta}(N_c - N_{jkc} + \beta_0, N_{jkc} + \beta_1) \tag{26}$$

Новые параметры опять обозначим $\beta_0^*$ и $\beta_1^*$ для удобства. В формуле 17 снова будем смотреть только на числитель. Первый множитель не изменится, а второй будет вычисляться так

$$p(\mathbf{x}|y = c, \mathscr{D}) = \prod_{j=1}^{D}\prod_{k=1}^{D} \mathop{\mathbb{E}}_{\theta_{**c}\sim\text{Beta}(\beta_0^*,\beta_1^*)} \theta_{jkc}^{\mathbb{I}(n(x_j)=n(x_k)+1)} \tag{27}$$

Причём получится, что каждое $\theta_{jkc}$ представлено не больше, чем один раз, так как в примере за триграммой может следовать только одна другая.

# 4. Реализация

## 4.1. Онтологии для замены неизвестных слов

Способ содержательного устранения проблемы с неизвестностью слов в анализируемых примерах, предлагаемый в работе, – это использование онтологий. Онтологией называется некоторая схема области знаний, обычно она представляет собой направленный ациклический граф, вершины которого – понятия. Чем выше в графе находится вершина, тем шире соответствующее ей понятие. Таким образом, если в сообщении встретилось слово, которое классификатору неизвестно, его можно заменить на более общее понятие в соответствии с онтологией.

База знаний Википедии[17] составляется пользователями и на данный момент содержит 4515000 статей только на английском языке. Структура организации статей в Википедии похожа на то, что мы ищем, – это граф категорий. На основе категорий Википедии уже составлена онтология проектом DBpedia[4] – она и используется в работе.

Данные выкачиваются в формате Turtle[18]. У нас есть два файла: первый – отображение статей в подмножество категорий, второй – лес всех категорий. Граф категорий сильно несвязный, большинство из них не участвуют ни в какой иерархии. Согласно этим особенностям нужно решить задачу хранения онтологии и поиска нужной категории.

В первую очередь, стоит сказать, что хранится отображение категорий в номера вершин в графе, так как хранить целые строчки для такого количества данных неэкономно. Эти отображения кладутся в бор[19], в соседнем боре запоминаются отображения статей в категории. Когда появляется неизвестное слово, ищется статья, где искомое слово – префикс, так как названия статей не всегда из одного слова. Выбираем из всех категорий, к которым относится данная статья, ту, что в иерархии находится ниже всего. При равенстве уровня выбираем любую.

## 4.2. Алгоритмы подготовки данных, предсказывания и обучения

Описанные в ходе данной работы идеи были объединены в метод классификации и реализованы на языке Python с использованием его расширения Cython. Ниже приведена последовательность действий со ссылками на разделы в тексте, где про эти действия рассказывается

[17] https://www.wikipedia.org/
[18] http://www.w3.org/TR/turtle/
[19] Marisa trie: https://code.google.com/p/marisa-trie/

более подробно. Некоторые моменты не были разъяснены ранее, поэтому им уделяется чуть больше внимания. Отдельно рассмотрим три части: подготовка данных, обучение классификатора, предсказание класса.

**Подготовка данных** (некоторые пункты не относятся к данным из обучающей выборки, в этом случае в конце пункта стоит "*"):

- замена сущностей html на слово "URL", упоминаний пользователя на "USER" и чисел на "42";
- перевод сообщения в нижний регистр;
- замена смайлов, хештегов и некоторых знаков препинания на "+" или "–" (раздел 2.3);
- замена долгого повторения гласных на сочетание из двух букв (раздел 2.3);
- разбивка на слова с учётом знаков препинания при помощи Penn Treebank Tokenizer из NLTK[1];
- замена сокращений на их расшифровки (раздел 2.3);
- замена неизвестных классификатору слов на обобщения каждого из них (раздел 4.1);*
- приведение всех слов к начальной форме при помощи алгоритма Snowball Stemmer[13]
- удаление артиклей, предлогов и союзов из сообщения.

**Обучение классификатора:**

- подготовка данных и сохранение множества известных слов;
- преобразование полученных строк на перекрывающиеся пары триграмм[20], теперь одна пара триграмм – это признак, по которому измеряются сообщения;
- каждый пример из обучающей выборки преобразуется в числовой вектор: на месте соответствующей пары триграмм из набора ставится 1, если она есть в примере, и 0, если нет;
- поиск апостериорных распределений параметров согласно формулам 15 и 26.

[20]Из предложения "Мама мыла раму." получатся следующие пары триграмм: "Мам|а м", "а м|ыла", "ыла|ра", " ра|му."

**Предсказание класса для нового примера:**

- подготовка примера согласно описанному выше в “Подготовка данных”;
- преобразование полученной строки на перекрывающиеся пары триграмм;
- преобразование полученного набора в числовой вектор: на месте соответствующей пары триграмм из набора ставится 1, если она есть в примере, и 0, если нет;
- вычисление вероятности примера оказаться в каждом из классов (“1” или “−1”) согласно выражению 17 и вычислению множителей из неё по 20 и 27
- выбор класса, который дал наибольшую вероятность попадания в него.

# 5. Количественная оценка метода

Твиттер предоставляет API[21] для извлечения и поиска сообщений, в том числе по поисковому запросу. Ответом на запрос к Твиттеру является набор сущностей, каждая из которых хранит информацию о сообщении: его id, текст, имя пользователя-автора, время публикации, а также, если оно является ответом или ретвитом другого, то указывается id “родительского” твита. В таком виде не восстановить цепочки твитов: чтобы найти диалог между пользователями, нужно обойти все существующие твиты и найти из них те, которые ссылаются на определённое сообщение, – так можно найти все ответы на него или его ретвиты. Скорее всего, если найден некоторый твит про объект, то ответы на него будут про этот же объект, то есть вычислять ответы необходимо.

Предлагается делать это следующим образом. Пусть есть твит $T$, его id $T_{id}$, имя пользователя, который его опубликовал $T_{user}$, и время публикации $T_{time}$. Отличительная особенность ответов в Твиттере, как говорилось ранее, – это упоминания, то есть ответ на твит пользователя с именем username будут начинаться со строки “@username”. Тогда, чтобы найти все ответы на твит $T$ будем искать не по множеству всех возможнных сообщений, а по всем сообщениям, опубликованным позднее $T_{time}$ по поисковому запросу “@$T_{user}$”. Среди них уже можно будет выделить сообщения, которые ссылаются на твит с номером $T_{id}$ – это и будут все ответы на $T$.

Так собираются данные по слову-запросу для разметки эмоциональной окраски актуальных сообщений по этой теме. Этот метод использовался для расширения тестовой выборки, используемой для анализа алгоритмов в ходе всей работы.

Для обучения классификатора использовались 1000000 размеченных сообщений. Классификаторы сравнивались на 386 тестовых примерах, 204 из которых отрицательные, 182 – положительные.

| **Метка класса** | **Precision** | **Recall** | **F1-score** | **Количество** |
|---|---|---|---|---|
| -1.0 | 0.82 | 0.75 | 0.78 | 204 |
| 1.0 | 0.74 | 0.82 | 0.78 | 182 |
| avg / total | 0.79 | 0.78 | 0.78 | 386 |

Таблица 6: Классификация наивным байесовским классификатором

В таблицах 6 и 7 приведено сравнение базового классификатора и его изменённой версии, основанной на байесовском подходе и использовании триграмм и онтологий. Новый

[21] https://dev.twitter.com/

| Метка класса | Precision | Recall | F1-score | Количество |
|---|---|---|---|---|
| -1.0 | 0.88 | 0.74 | 0.80 | 204 |
| 1.0 | 0.75 | 0.88 | 0.81 | 182 |
| avg / total | 0.82 | 0.81 | 0.81 | 386 |

Таблица 7: Классификация байесовским наивным байесовским классификатором с триграммами и онтологиями

классификатор дал улучшение предсказания на 3%, при этом метод не перестал быть инкрементально обучающимся, то есть его можно уточнять в онлайн-режиме.

# Заключение

В ходе данной работы были решены поставленные задачи и достигнуты следующие результаты.

1. Проведено сравнение базовых методов обучения с учителем: наивный байесовский классификатор, метод опорных векторов и логистическая регрессия – для данных из микроблогов по параметрам точности, полноты результатов и времени обучения. Подробнее о сравнении написано в разделе 1.3.5. Лучше всех себя показал наивный байесовский классификатор, время обучения которого на порядок ниже двух других.

2. Проанализированы особенности задачи анализа мнений для микроблогов, найдены варианты использования этих особенностей для улучшения методов классификации. Об использовании особенностей написано в разделе 2.3.

3. На основе наивного байесовского классификатора предложен, обоснован и реализован новый метод. Для улучшения использовались: переход к байесовскому подходу (модель описана в разделе 3.2.1), триграммы для использования информации о связи слов в предложении (раздел 3.2.2) и онотологии на базе категорий из Википедии (раздел 4.1).

4. Сравнение нового метода с базовым проведено на тех же данных, что и сравнение методов обучения с учителем. Разработанный метод дал улучшение на 3% (раздел 5).

В качестве продолжения работы можно сформулировать и решить задачу классификации сообщений из микроблогов на субъективные и объективные и такой классификацией дополнить уже полученный метод. Кроме того, на основе предложенного в работе метода планируется разработать веб-сайт, собирающий и демонстирующий пользователю статистику эмоциональной окраски сообщений из Твиттера по поисковому запросу.

# Список литературы


[1] Bird Steven. NLTK: the natural language toolkit // Proceedings of the COLING/ACL on Interactive presentation sessions / Association for Computational Linguistics. — 2006. — P. 69–72.

[2] Celikyilmaz Asli. Probabilistic model-based sentiment analysis of twitter messages. — 2010. — P. 79–84.

[3] Colace Francesco, De Santo Massimo, Greco Luca. A Probabilistic Approach to Tweets' Sentiment Classification // 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction. — 2013. — P. 37–42.

[4] DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia / Jens Lehmann, Robert Isele, Max Jakob et al. // Semantic Web Journal. — 2014.

[5] Field David A. Laplacian smoothing and Delaunay triangulations // Communications in applied numerical methods. — 1988. — Vol. 4, no. 6. — P. 709–712.

[6] Go Alec, Bhayani Richa, Huang Lei. Twitter sentiment classification using distant supervision // CS224N Project Report, Stanford. — 2009. — P. 1–12.

[7] Manning Christopher D, Schütze Hinrich. Foundations of statistical natural language processing. — MIT press, 1999.

[8] Murphy Kevin P. Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series). — The MIT Press, 2012. — ISBN: 0262018020.

[9] Nigam Kamal, Lafferty John, McCallum Andrew. Using maximum entropy for text classification // IJCAI99 workshop on machine learning for information filtering. — 1999. — P. 61–67.

[10] Ontology-based sentiment analysis of twitter posts / Efstratios Kontopoulos, Christos Berberidis, Theologos Dergiades, Nick Bassiliades // Expert Systems with Applications. — 2013. — Vol. 40, no. 10. — P. 4065–4074.

[11] Pang Bo, Lee Lillian, Vaithyanathan Shivakumar. Thumbs up?: sentiment classification using machine learning techniques // Proceedings of the ACL-02 conference on Empirical methods

in natural language processing-Volume 10 / Association for Computational Linguistics. — 2002. — P. 79–86.

[12] Parsing, Word Associations and Typical Predicate-argument Relations / Kenneth Church, William Gale, Patrick Hanks, Donald Hindle // Proceedings of the Workshop on Speech and Natural Language. — HLT 89. — Association for Computational Linguistics, 1989. — P. 75–81.

[13] Porter Martin F. Snowball: A language for stemming algorithms. — 2001.

[14] Qadir Ashequl, Riloff Ellen. Bootstrapped Learning of Emotion Hashtags# hashtags4you // WASSA 2013. — 2013. — P. 2.

[15] Saif, Hassan He, Yulan Alani Harith. Semantic sentiment analysis of twitter // The 11th International Semantic Web Conference. — 2012.

[16] Scikit-learn: Machine Learning in Python / F. Pedregosa, G. Varoquaux, A. Gramfort et al. // Journal of Machine Learning Research. — 2011. — Vol. 12. — P. 2825–2830.

[17] Tong Simon, Koller Daphne. Support vector machine active learning with applications to text classification // The Journal of Machine Learning Research. — 2002. — Vol. 2. — P. 45–66.

[18] Turney Peter D. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews // Proceedings of the 40th annual meeting on association for computational linguistics / Association for Computational Linguistics. — 2002. — P. 417–424.