Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра алгоритмических языков

Паркин Александр Николаевич

Классификация коротких сообщений

КУРСОВАЯ РАБОТА

Научный руководитель:
к.ф.-м.н., доцент
Т.Ю. Грацианова

Москва, 2016

Содержание

1 Введение

С развитием сети Интернет развиваются также социальные сети. Они занимают все более важное место в нашей жизни, влияют на неё. В связи с этим появилась потребность в анализе социальных сетей, извлечение полезной информации. Одной из задач является анализ тональности сообщений социальных сетей для оценки товара или для оценки репутации компании, к примеру.

Тональность сообщений можно задать несколькими способами: разбить только на положительные и отрицательные отзывы, можно к ним добавить нейтральные отзывы, либо тональность оценить по шкале(-5 – крайне негативный отзыв, +5 – положительный). Для анализа тональности можно использовать инженерный подход, либо машинное обучение.

В связи с быстрым ростом объема информации, а так же производительности вычислительных систем, было выбрано машинное обучение с моделями классификации, а не инженерный подход, который занял бы много времени для составления правил и словарей. С этого момента под классификацией и тональностью текста будем подразумевать одно и то же, так как тональность текста (положительная тональность, отрицательная и нейтральная) мы можем представить тремя классами.

2 Постановка задачи

В данной курсовой работе рассматривается задача автоматического определения тональности короткого текста (сообщений) на русском языке по многополосной шкале, а также выявление признаков, присущих сообщениям социальных сетей.

Задачи:

- Изучить модели машинного обучения для определения тональности текста
- Выявить лучшие модели для коротких сообщений
- Выявить дополнительные признаки, улучшающие модели

3 Подходы к классификации сообщений при помощи машинного обучения

3.1 Подготовка данных

Задача классификации входит в группу так называемых задач машинного обучения с учителем, а это значит, что данные должны быть размечены. Для раметки можно использовать экспертов, либо размечать основываясь на какие-либо признаки. Например, отзывы на Яндекс.Маркете можно считать уже размеченными, так как при составлении отзыва пользователь так же ставит оценку продукту, автоматиация сбора данных подобным образом сэкономит массу времени и сил. При отсутствия подобных возможностей мы должны обратиться к помощи экспертов в разметке, но даже в данном случае стоит один и тот же текст дать оценить нескольким экспертам, а итоговую метку поставить ту, которую проставило большинство, таким образом улучшая качество разметки.

3.2 Векторизация текста

Для применения моделей машинного обучения мы должны представить наш текст в виде вектора. Есть несколько методов, но до их применения стоит подготовить данные. Текст разбивается на **токены**. Токен - это минимальная единица членения предложения, токеном может быть не только слова. Если токен является словом, то слово приводится к нормальной форме, из всех токенов составляется список. Из этого списка удаляются знаки препинания, ссылки на сайты и так называемые стоп-слова, то есть токены, которые почти не влияют на эмоциональную окраску текста(к примеру, союзы, местоимения, междометия и частицы). Теперь можно начать векторизовать текст.

3.2.1 Count vectorization

Для представления текста вектором для начала токенизируются все тексты нашей тренируемой выборки, составляется список токенов. Количество уникальных токенов и будет размерностью вектора, то есть если мы получили список из 10 000 уникальных токенов, то размерность нашего вектора будет 1x10000, где каждый столбец будет отвечать за свой токен. Для векторизации текста подобным образом рассматриваются токены, которые

встретились в данном тексте, в соответсвующих столбцах токенов вносится количество вхождения в данный текст. В конце вектор просто нормализуется.

3.2.2 TF–IDF

Более точным способом векторизации является метод $TF - IDF$, где учитывается важность слова в контексте документа, который является частью всего корпуса.

tf(term frequency – частота слова) — отношение числа вхождения слова к общему количеству слов текста. Таким образом мы можем оценить важность слова в пределах одного сообщения. Для того, чтобы не возникала путаница, под текстом/сообщением/документом будем подразумевать одно и то же.

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

где n_i – число вхождения слова в документ, а $\sum_k n_k$ – количество всех слов. Но для увеличения значимости некоторых слов Карен Спарк Джонс[1] была предложена концепция обратной частоты слов.

idf(inverse document frequency – обратная частота документа) – инверсия частоты, с которой слово встречается так же в других документах.

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

где $|D|$ - количество документов, в $|d_i \supset t_i|$ - количество документов, в которых встретилось слово. Ну а в качестве значения в столбце каждого слова будем брать произведение $tf - idf(t,d,D) = tf(t,d) * idf(t,D)$.

3.3 Модели машинного обучения

Как говорилось раннее, задача классификации объектов относится к группе задач машинного обучения с учителем.[2] Есть объекты, которые имеют признаки, а так же метки каждого объекта, к какому классу они относятся. Пусть X - множество признаков, а Y - множество меток/имен классов. Предполагаем, что существует зависимость $y^* : X \to Y$. Наша задача - построить алгоритм $A : X \to Y$, который сможет классифицировать произ-

вольный объект $x \in X$. Рассмотрим самые популярные методы машинного обучения для данной задачи.

3.3.1 kNN (метод k ближайших соседей)

kNN – простейший метрический классификатор, основывается на сходстве объектов. Объект x_i относится к тому классу y_i, которому принадлежит большинство из k его соседей из обучающей выборки x_i. В случае задачи с двумя классами берут нечетное количество соседей, чтобы не возникала неоднозначность определения класса. С тремя и более классами нечетности уже недостаточно, поэтому i-ому соседу приписывают вес w_i, который уменьшается при отдалении от объекта x, ему назначается тот класс, который набрал больший суммарный вес.

3.3.2 Naive Bayes

Наивный байесовский классификатор – алгоритм классификации, основанный на теореме Байеса с допущением о независимости признаков. В основе лежит теорема Байеса:

$$P(c|d) = \frac{P(d|c}{P(d)},$$

где

- $P(c|d)$ – вероятность, что документ d принадлежит классу c
- $P(d|c)$ – вероятность встретить документ d среди всех документов класса c.
- $P(c)$ – безусловная вероятность встретить документ класса c в корпусе всех документов
- $P(d)$ – безусловная вероятность документа d в корпусе документов.

Цель классификации – понять к какому классу принадлежит документ, поэтому нам не нужна вероятность, а наиболее вероятный класс, а так как вероятность документа остается константой, то можем все в итоге переписать как

$$c_{map} = \arg\max_{c \in C}[P(d|c)P(c)].$$

Исходя из предположения независимости слов друг от друга, мы можем апроксимировать условную вероятность принадлежность документа классу как произведение вероятностей встретить w_i слово в классе c. Взять логарифм, чтобы не произошло переполнение, тогда выражение превратиться в сумму

$$c_{map} = \arg\max_{c \in C}[\log P(c) + \sum_{i=1}^{n} \log P(w_i|c)].$$

Если на этапе классификации нам встретися слово, которое не встречали на этапе обучения, то документ с этим словом будет иметь нулевую вероятность. Чтобы избежать такие ситуации, можно представить, что мы видели каждое слово на один больше.

$$P(w_i|c) = \frac{W_i c + 1}{\sum_{i \in V}(W_i c + 1)} = \frac{W_i c + 1}{|V| + \sum_{i \in V} W_i},$$

где V - список всех уникальных слов.

В итоге мы получаем формулу, по которой происходит байесовская классификация:

$$c_{map} = \arg\max_{c \in C}[\log \frac{D_c}{D} + \sum_{i=1}^{n} \log \frac{W_{ic} + 1}{|V| + \sum_{i \in V} W_{ic}}],$$

где

D_c – количество документов принадлежащих c

D – общее количество документов

3.3.3 SVM

Метод опорных векторов (support vector machine) – набор схожих алгоритмов обучения с учителем, используются для задач классификации и регрессионного анализа. Каждый объект данных представляется как вектор в n - мерном пространстве, наша задача - найти гиперплоскость размерности $n - 1$. Гиперплоскостей будет много, естественно предположить, что если максимизируем зазор между классами, то будем иметь более уверенную классификацию. Если обучающая выборка линейно не разделима, то вводится дополнительный набор переменных штрафа за ошибку. Максимизация зазора между опорными векторами уже будет с учетов суммарной ошибки. Вместо скалярного произведения можно использовать нелинейну функцию ядра K, чтобы перейти в пространство с большей

размерностью и найти разделяющуюся гиперплоскость.

3.3.4 Решающие деревья

Решающие деревья воспроизводят логические схемы, позволяющие получить окончательное решение о классификации объекта с помощью ответов на иерархически организованную систему вопросов. Причём вопрос, задаваемый на последующем иерархическом уровне, зависит от ответа, полученного на предыдущем уровне. Пусть у нас объект описывается $x_1, x_2, ..., x_n$ признаками. Каждой вершине ставится предикат, для непрерывных признаков это может быть пороговый параметр. Так на каждой вершине в зависмости от предиката объект спускается на одну вершину ниже. Процесс распознования заканчивается при достижении концевой вершины (листа), этому листу ставится метка класса.

4 Практическая работа

Для анализа коротких сообщений были выбраны твит-сообщения социальной сети Twitter. Особенность твит-сообщений в том, что максимальная допустимая длина сообщения - 140 символов. Анализ сообщений в других социальных сетях (VK, Instagram) показал, что в большинстве случаев пользователи пишут либо очень короткие сообщения (не более 8-10 слов), либо очень длинные, которые можно разбить на маленькие сообщения и анализировать отдельно, объединив в конце результат. Таким образом, опыт анализа коротких сообщений в твиттере можно использовать при анализе других социальных сетей.

4.1 Подготовка данных

В качестве данных для анализа были уже размеченные данные, которые использовались для соревнования в Dialog Evolution 2015. Нашей задачей является классификация твитов про банки и телекомунникационные системы на отрицательные, нейтральные и положительные отзывы. Данные хранятся в виде xml файлов, хранится *id* текста, *twitter id*, , *twitter login*, и метки 8 банков (Сбербанк, ВТБ, Газпром, Альфа-Банк, Банк Москвы, Райффазен Банк, Уралсиб, РШБ), где данная метка может принимать 4 значения. Метка NULL означает, что в данном твит – сообщении не упоминался данный банк, -1 – в твит–сообщениие отмечено экспертами как отрицательный отзыв про данный банк, 0 - нейтральный отзыв/не имеет эмоциональной окраски, 1 - положительный отзыв. Аналогичным образом размечены данные связанные с телекоммуникационными компаниями (Билайн, МТС, Мегафон, Теле2, Ростелеком, Комстар, Скайлинк). Общее количество данных выглядит следующим образом:

Как говорилось уже раннее, для того, чтобы применить модели машинного обучения, мы должны представить наши сообщения в виде вектора. Было решено попробовать оба варианта из раннее предложенных: CountVectorizer и TfidfVectorizer из пакета scikit-learn python. Как видим на Рисунке 1, данные по категориям распределены неравномерно, поэтому как показатель эффективности модели была выбрана f1 – метрика. Для того, чтобы посчитать f1–macro используя только что размеченные данные и тестовые метки, в skikit-learn.metrics есть функция f1_score с параметром 'macro'.

		Нейтральный	Положительный	Отрицательный	Общее количество
Телеком	Тренировочная коллекция	4 870	1 354	2 550	8 643
	Тестовая коллекция	1 016	226	1 054	2 247
Банки	Тренировочная коллекция	6 977	704	1 734	9 392
	Тестовая коллекция	2 240	312	722	3 313

Рис. 1: Количество размеченных твитов.

4.2 Построение моделей машинного обучения.

Из моделей машинного обучения были выбраны такие модели, как Naive Bayes, kNN, SVM, Decision Tree и AdaBoost из пакета scikit-learn python. Для метода k–ближайших соседей было построено 5 моделей, где каждая из них учитывала 3, 5, 7, 10, 30 ближайших соседей, а уже потом выбиралась лучшая. SVM взял с RBF ядром (Radial basis function) и линейным ядром, оказалось, что на наших данных линейное ядро проявляет себя лучше, ансамбль алгоритмов AdaBoost использовал вместе с решающими деревьями. Результаты для набора данных по банкам и телекоммуникационным команиям указаны на рисунке 2 и рисунке 3 соответственно.

	train	test	Naive Bayes	SVM	SVM (linear)	kNN	Decision Tree	Decision Tree (depth = 5)	AdaBoost
Сбербанк			0.3487	0.2984	0.3536	0.3636 (3)	0.3216	0.3320	0.3359
Сбербанк *	3895	976	0.3470	0.2984	0.3485	0.3702 (10)	0.3375	0.3407	0.3401
ВТБ			0.3122	0.3293	0.3098	0.3128 (3)	0.3078	0.3056	0.2980
ВТБ *	1527	381	0.2980	0.3678	0.3128	0.3141 (30)	0.3080	0.3070	0.3091
РШБ			0.2894	0.2980	0.2800	0.3108 (5)	0.2824	0.2890	0.2912
РШБ *	929	99	0.2929	0.2980	0.2829	0.2912 (30)	0.2778	0.2890	0.3500
Газпром			0.2840	0.0254	0.3159	0.3003 (5)	0.2878	0.2921	0.468
Газпром *	408	101	0.2987	0.0254	0.3066	0.3160	0.3952	0.4727	0.2996
Райффайзен			0.2785	0.2785	0.2785	0.3403 (3)	0.3227	0.2868	0.3437
Райффайзен *	343	85	0.3545	0.2785	0.2750	0.2785	0.2758	0.2785	0.2785
УралСиб			0.3145	0.3145	0.3145	0.3145	0.2933	0.2933	0.3145
УралСиб *	114	28	0.3145	0.3145	0.3145	0.3145	0.2933	0.2857	0.2933
Общий			0.3042	0.2998	0.3346	0.3463 (3)	0.3203	0.3265	0.3373
Общий *	7603	1573	0.3268	0.2972	0.3229	0.3398 (10)	0.3125	0.3336	0.3202

Рис. 2: Результаты моделей на данных по банкам.

Для того, чтобы было понятно, что же именно отображено в таблицах, введем некоторые пояснения: строки, где есть в наименовании * говорит о том, что для векторизации

	train	test	Naive Bayes	SVM	SVM (linear)	kNN	Decision Tree	Decision Tree (depth = 5)	AdaBoost
Билайн			0.2604	0.2626	0.3113	0.2805 (3)	0.3355	0.2683	0.3058
Билайн *	2816	703	0.3162	0.2626	0.3178	0.2988 (5)	0.3428	0.2688	0.2960
МТС			0.3112	0.2433	0.3310	0.3646 (3)	0.3370	0.3240	0.3177
МТС *	2768	691	0.3470	0.2433	0.3352	0.3113 (5)	0.3452	0.2646	0.3024
Мегафон			0.2827	0.2548	0.3282	0.3535 (5)	0.3348	0.2736	0.2662
Мегафон *	1198	139	0.3227	0.2548	0.3476	0.3255 (3)	0.3484	0.2711	0.2765
Ростелеком			0.2619	0.2759	0.2963	0.3675 (5, 30)	0.3233	0.3675	0.3827
Ростелеком *	520	17	0.2695	0.2759	0.2963	0.2857 (7)	0.3233	0.3827	0.3827
Теле2			0.3169	0.3169	0.3111	0.3169	0.2924	0.3169	0.2857
Теле2 *	310	32	0.3111	0.3169	0.2857	0.3169	0.2988	0.3169	0.2857
Общий			0.2832	0.2802	0.3316	0.3464 (3)	0.3407	0.2914	0.3223
Общий *	7603	1573	0.3198	0.2827	0.3274	0.3297 (5)	0.3375	0.3085	0.3107

Рис. 3: Результаты моделей на данных по телеком. компаниям.

использовался метод CountVectorizer, а где нет — tf–idf; значения в столбце kNN в скобках содержат количество соседей, при котором модель добилась лучших результатов, если этого пояснения нет, значит при любом из 5 значений (3, 5, 7, 10, 30) были достигнуты одни и те же результаты, которые отображены в таблице.

До того, как приступить к выбору моделей, подготовке данных, были изучены материалы участников Диалог-21, которые уже занимались обработкой этих данных [3] [4] [5]. Решения представленные в этой курсовой работе не так точно решают поставленную задачу, как решения участников конкурса, но почти все модели преодолевают порог вхождения (baseline).

4.3 Выводы по проделанной работе.

Оценка результатов разных моделей на двух наборах данных позволяет понять, в какую сторону сторону следует двигаться для улучшения анализа тональности коротких сообщений. Разница между tf-idf и CountVectorization еле заметна, независимо от того, какая модель машинного обучения использовалась, какой размер нашего dataset'a. Однако, если векторизовать короткие сообщения при помощи tf–idf, то лучше всего использовать SVM с линейным ядром, при выборе kNN число соседей для сравнения выбирать небольшое (3 или 5 будет достаточно). Чуть иначе выглядит картина, если выбрать "количественную векторизацию в таком случае можно выбрать наивный байесовский классификатор, SVM с линейным ядром и kNN с числом соседей от 5 до 30. С обоими методами векторизации хороший результат показывают решающие деревья и AdaBoost с ними, наивный байесовский классификатор и SVM показывают примерно одинаковый результат.

5 Направления дальнейшего развития

Для улучшения качества анализа тональности коротких сообщений необходимо рассмотреть две задачи: как улучшить представление данных, какие методы машинного обучения лучше подойдут для решения данной задачи.

Сравнение одних и тех же моделей машинного обучения на данных с разной размерностью (1x4373 против 1x7323) показало, что увеличение размерности в подобных масштабах не ухудшает обучаемость (что могло произойти из-за появления шумовых признаков), а даже наоборот чуть улучшает. Поэтому можно попробовать добавить к монограммам и биграммы. Чтобы рамерность векторов не разрослась, можно использовать список разрешенных токенов, включать туда слова "лучше "больше"и т.д. В таком случае, биграммы добавлять к признакам только тогда, когда есть токен с разрешенного списка. Этот подход является инженерным, потому требует дополнительных для выделения разрешенных токенов, а также экспертных знаний.

Еще одним способом улучшения качества анализа путем добавления новых признаков является учет специфики социальных сетей. Из-за ограничения длинны сообщения люди часто в своих сообщения в социальных сетях используют смайлы, либо emoji - смайлы, которые призваны помочь в передаче эмоционального характера сообщения. Поэтому можно предположить, что добавление токенов, связанных на смайлах, к признакам текста приведет к улучшению модели.

Кроме повышения качества классификация посредством улучшения подготовки данных, следует испробовать нейронные сети и DeepBoosting. После изучения литературы на соответсвующие темы, интересными и способными улучшить нашу модель показались сверточная нейронная сеть CharSCNN [6] и DeepBoosting [7] [8] с применением SVM, который хорошо показал себя на наших данных.

6 Заключение

В рамках данной курсовой работы были получены следующие результаты:

- Изучены методы машинного обучения для определения классификации объектов и для определения тональности текста
- Выявлены лучшие модели для коротких сообщений
- Выявлены дополнительные признаки, улучшающие модели, а именно для какого способа векторизации какая модель классифиации машинного обучения лучше всего подходит.

При изучении методов анализа тональности текстов любой длинны[?] было выявлено, что применение стандартных моделей классификации и способов векторизации к коротким сообщениям недостаточно для достижения хороших результатов распознования тональности. Если рассмотреть все значения меры f-macro, то видно, что независимо от метода ветокризации и методов машинного обучения результаты тестов показывают значения от 0.27 до 0.46, а в среднем держится на уровне 0.33, однако победители конкурса SentiRuEval-2015 получали и более высокие значения f-меры. Для увеличения качества классификации необходимо как улучшить представление коротких сообщений в векторном пространстве, так и использовать более сложные методы машинного обучения.

Список литературы

[1] Karen Spärck Jones. A statistical interpretation of term specificity//Journal of Documentation Volume 60 Number 5 2004 pp. 493-502

[2] Воронцов Константин Вячеславович. Классификация в машинном обучении [Электронный ресурс].– Электрон. дан. URL: http://www.machinelearning.ru/wiki/index.php?title=Классификация (дата обращения: 16.05,2016)

[3] Loukachevitch, Blinov, Kotelnikov, Rubtsova, Ivanov, Tutubalina. Testing Object-oriented Sentiment Analysis Systems in Russian. //SentiRuEval-2015

[4] Vasilyev, Denisenko, Solovyev. Aspect Extraction and Twitter Sentiment Classification by Fragment Rules//SentiRuEval-2015

[5] Blinov, Klekovkina, Kotelnikov, Pestov. Research of lexical approach and machine learning methods for sentiment analysis//SentiRuEval-2014.

[6] Cìcero Nogueira dos Santos, Maıra Gatti. Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. //COLING-2014, pp. 69 - 78, 2014.

[7] Corinna Cortes, Mehryar Mohri, Umar Syed. Deep Boosting //ICML, pp. 1179 – 1187, 2014

[8] Vitaly Kuznetsov, Mehryar Mohri, Umar Syed. Multi-Class Deep Boosting // NIPS, pp. 2501-2509, 2014.

[9] Луис Педро Коэльо, Вилли Ричарт. Построение систем машинного обучения на языке Python. 2-е изд.// М.: ДМК Пресс, 2016. - 302 с.: ил.