Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
“Московский физико-технический институт
(государственный университет)”

Факультет радиотехники и кибернетики

Кафедра теоретической и прикладной информатики

КЛАССИФИКАЦИЯ ОТЗЫВОВ, ОТЧЕТОВ ОБ ОШИБКАХ, ЗАПРОСОВ ФУНКЦИОНАЛЬНОСТЕЙ ПРОДУКТА МЕТОДАМИ МАШИННОГО ОБУЧЕНИЯ

Выпускная квалификационная работа
(Бакалаврская работа)

Направление подготовки: 03.03.01 Прикладные математика и физика

Выполнил:

Студент 211 группы

Маракулин Роман Игоревич

Научный руководитель:

Соболев Артемий Анатольевич

г. Москва, 2016

Оглавление

Аннотация

При разработке и дальнейшем сопровождении программного продукта важную роль играет получение обратной связи от пользователей и учет их пожеланий.

Данная работа посвящена исследованию эффективности методов машинного обучения по автоматизации обработки отзывов пользователей на программный продукт. А именно, проводится

- сравнительный анализ методов машинного обучения для решения задачи классификации отзывов на категории: отчеты об ошибках, запросы функциональностей продукта, остальные суждения на примере отзывов о программном продукте.
- исследование методов машинного обучения по определению схожих запросов в каждой из выше перечисленных категорий.

Отзывы были получены из AppStore по программному продукту eBay. Полученные результаты показывают, что данная задача может решаться методами машинного обучения. Согласно экспериментам, наиболее эффективным методом оказался наивный байесовский классификатор на n-граммах с учетом мета информации.

1. Введение

В наши дни появляется огромное количество платформ для создания и запуска программных продуктов и приложений. За последние 5 лет было скачано порядка сотен миллионов приложений [1], и это только учитывая мобильные платформы. Недавние исследования показывают сильное влияние отзывов пользователей на успех программного продукта [2]. Отзывы помогают другим пользователям сориентироваться при выборе программного продукта. Также, огромное число отзывов содержат информацию об ошибках, запросах на улучшение [3][4]. Эта информация может оказаться полезной при улучшении программного продукта, его сопровождении. Количество таких отзывов на популярный продукт может достигать сотен тысяч [5], среди которых есть неинформативные и повторяющиеся. Имея такое количество отзывов, задача фильтрации и разбора полезной информации становится трудной для разработчиков и аналитиков.

Задача данной работы состоит в исследовании возможности создания эффективной модели по обработке отзывов пользователей с автоматизированной обратной связью и созданием актуальных и уникальных заданий разработчикам.

Цель исследования состоит в сопоставлении ряда существующих решений по классификации отзывов на заранее заданные категории, а также исследовании решений по автоматическому определению схожих отзывов с последующим выбором наиболее эффективных методов классификации.

При решении поставленной задачи не последнюю роль играет время работы алгоритма. Для популярных программных продуктов каждый день появляются несколько сотен новых отзывов [6]. Алгоритм должен быть способен обработать входящий поток данных за разумное время.

Решение данной задачи можно свести к решению ее составных частей,

таких как:

1) классификация всех поступающих от пользователей отзывов на категории: отчеты об ошибках, запросы функциональностей продукта, остальные суждения.
2) определение в каждой из категорий групп схожих отзывов.
3) проверка на соответствие между созданными задачами разработчиков и существенными (имеющих существенное количество схожих отзывов) задачами, сгенерированными моделью.

В работе освещены первая и вторая проблемы на примере отзывов о программном продукте eBay.

Задача классификации отзывов формулируется следующим образом: дано множество объектов – отзывов $X = \{x_1, x_2, x_3, \ldots, x_n, \ldots\}$ и множество возможных ответов – категории $Y = \{1, 2, \ldots, k\}$. Существует неизвестная целевая зависимость – отображение $y^*: X \rightarrow Y$, значения которой известны только на объектах конечной выборки: $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$. Требуется построить алгоритм $a: X \rightarrow Y$, который приближал бы целевую зависимость, как на элементах выборки, так и на всем множестве X. Определим категории. Применительно к нашему случаю, выделяем следующие категории:

1) **отчеты об ошибках**

   описывают проблемы с программным продуктом, которые должны быть исправлены в будущем, такие, например, как падение продукта, неправильное поведение, проблемы, связанные с производительностью
2) **запросы функциональностей продукта**

   запросы на добавление функциональностей, например, существующих в других продуктах, недостаток содержания, контента или идеи, как сделать приложение лучше;
3) **неинформативные отзывы**

категория комбинирует неинформативные суждения, такие, как, описание уже существующих возможностей продукта, опыт использования в конкретных ситуациях, восхищение, недовольство. Такие отзывы отражают информацию, которая может быть прочитана из рекламы или документации к программному продукту или является необоснованной критикой, похвалой.

Данные категории выделены в соответствии с [6].

Правильная классификация определяется человеком-асессором, который присваивает отзыву метку j в соответствии с выше описанным определением категорий.

Задачу определения в категориях групп схожих отзывов можно отнести к задаче выявления дубликатов [7][8]. При решении данной задачи мы не располагаем определенными категориями. Задачу можно отнести к классу задач обучения без учителя. Мы имеем описание (признаки) множества объектов (отзывов), требуется обнаружить внутренние взаимосвязи, закономерности, существующие между объектами (отзывами). Часто данная задача решается графовыми алгоритмами [7].

Разделение главной задачи на несколько этапов преследует очевидную цель улучшения точности классификации, а после, создания кластеров схожих по смыслу отзывов.

Существует ряд наиболее близких работ [11][12][13], в которых исследуется возможность создания модуля для системы отслеживания ошибок по поиску наиболее близких данному отзыву отчетов об ошибках.

Новизна данной работы состоит в создании комплексного и общего подхода к решению, сравнении существующих решений на одном наборе данных.

2. Обзор существующих методов

2.1. Задача классификации отзывов

Классическим подходом к решению данной задачи является рассмотрение каждого документа (отзыва) в виде набора слов (bag of words). Пусть n – количество документов $(d_1, d_2, \ldots, d_n)$, а m – количество уникальных терминов $(w_1, w_2, \ldots, w_m)$ (слов, или, более в широком смысле, токенов). Тогда, каждый документ d_i мы можем представить в виде строки $(x_1, x_2, \ldots, x_m)$, где x_j – количество вхождений термина w_j в документ d_i. Как правило, большинство значений x_j равны 0 (матрица является разреженной). Такой подход позволяет оперировать с векторным представлением документов. Но в таком представлении не учитывается важность отдельных слов. Если слово в некотором документе встречается часто, а в других документах редко, то оно представляет для нас больший интерес, так как в большей степени характеризует этот документ. Для учета веса слова самым популярным способом взвешивания для модели термин-документ – это применение TF-IDF формулы, по которой вес слова рассчитывается по формуле:

$$w_{ki} = \frac{\left(1 + \log\left(N_{ik}\right) * \log\left(\frac{|D|}{N_k}\right)\right)}{\sqrt{\sum_{s \neq k} \left(\log\left(N_{is}\right) + 1\right)^2}} \quad (1)$$

где N_{ik} – количество появлений k-ого термина в i-ом документе, N_k – количество появлений k-ого термина во всех документах, $|D|$ - количество документов в коллекции.

Таким образом, строка $(x_1, x_2, \ldots, x_m)$, характеризующая документ, заменяется строкой весов рассчитанных по формуле TF-IDF.

Наиболее популярные методы машинного обучения для классификации отзывов это наивный байесовский классификатор, метод опорных векторов, решающие деревья, метод максимальной энтропии.

В работе [6] были рассмотрены методы машинного обучения в применении к классификации отзывов. В работе производится сравнение таких алгоритмов, как наивный байесовский классификатор, решающее дерево, и метод максимальной энтропии. Каждый отзыв представлялся как набор слов. Показано, что наивный байесовский классификатор является наиболее точным среди представленных.

Большая роль уделена предобработке отзывов. Была исследована зависимость качества классификации от вариантов предобработки. Рассмотрены такие варианты предобработки, как: удаление стоп-слов (слова, которые встречаются во многих документах и не несут смысловой нагрузки, такие, как, a, has, once и так далее), стемминг (нахождение основы слова для заданного исходного слова: cats, catty → cat; stems, stemmer → stem и так далее), лемматизация (приведение словоформы к ее словарной форме: better → good; walking → walk и так далее). Например, удаление стоп-слов увеличивает точность, тогда как применение лемматизации ее уменьшает при определении запросов функциональностей. В целом, нужно аккуратно относиться к удалению стоп-слов и лемматизации. При удалении стоп-слов, мы не учитываем слова – want, please, can, что может отрицательно сказаться на классификации. Учет метаданных может улучшить классификатор. Были рассмотрены следующие метаданные: длина отзыва, рейтинг, поставленный пользователем продукту (1-5), времена глаголов в отзывах. Добавление метаданных положительно сказывалось на классификаторе.

В работе [9] был рассмотрен метод опорных векторов (SVM) применительно к задаче классификации текстов. Документы также были преобразованы в набор слов, но учитывались слова, которые присутствовали в документах более, чем 2 раза и не являлись стоп-словами. Метод опорных векторов чувствителен к размерности матрицы документов и учет большого

числа слов будет сильно сказываться на производительности. Один из способов борьбы с большим числом признаков документа – использование метода главных компонент (PCA) для уменьшения размерности, что неизбежно приводит к уменьшению анализируемой информации. Применен TF-IDF метод с последующей нормализацией вектора. Эксперименты сравнивали SVM классификаторы с различными ядрами (полиномиальное, радиальная базисная функция) с другими классификаторами на медицинских документах о болезнях и похожих документах. Метод опорных векторов с радиальной базисной функцией показал наилучший результат.

Работа [10] посвящена использованию n-грамм в задачах классификации текстов (n-граммы - последовательности из n элементов, символов). Данный подход избавляет от создания систем обработки ошибок в отзывах, документах. Подход устойчив к ошибкам в словах, неправильно поставленным знакам препинания, эффективен на небольших документах. Для того чтобы классификатор был чувствителен к началу и концу слов к словам в начале и в конце добавляется определенное количество '_' символов. То есть, 2-граммы слова TEXT: _T, TE, EX, XT, T_. Документы были очищены от цифр и знаков препинания, разбиты на токены. Токены образовали n-граммы с n = 1,2,3,4,5. В работе было показано, что использование n-грамм успешно решает задачу классификации текстов.

Совсем иной подход к классификации текстов представлен в работах [15] [18]. Идея состоит в превращении каждого слова в вектора одинаковой длины. В работах приводится пример того, как может быть учтен порядок слов. Каждый отзыв рассматривается не как набор слов и их количество. Каждое слово в отзыве имеет свой контекст. К примеру, для *i*-го слова w_i в отзыве, в зависимости от параметра k (размер окна), контекст можно представить как последовательность: $C = w_{i-k}, w_{i-k+1}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+k}$. Очевидно, для слова w_i

можно взять набор таких контекстов в зависимости от размера окна в некоторых пределах. Далее, используется предположение о том, что слова, имеющие схожие контексты имеют схожие значения. На основе этого строится такое пространство, что слова, имеющие схожие значения, представляются “близкими” ($|x_i-x_j| < \varepsilon$) векторами в данном пространстве. Для превращения слов в вектора строится нейронная сеть с одним скрытым слоем, используя skip-gram модель, представленную в [19] (вычисление векторов слов контекста по данному *i*-му слову и сопоставление с существующим контекстом). Настройка сети производится методом обратного распространения ошибки. Для избегания вычислительной сложности используется Hierarchical softmax [20], суть которого состоит в построении бинарного дерева для представления всех слов в словаре. Данная модель позволяет уменьшить вычислительную сложность обновления весов нейронной сети с O(V) до O(log(V)), где V – размерность вектора слова.

После обучения модели по представлению слов векторами одинаковой размерности требуется решить проблему классификации наборов полученных векторов (представление отдельного отзыва) на категории. Для разрешения этой проблемы часто используется следующий подход. Все пространство получившихся векторов из слов разбивается алгоритмом k ближайших соседей (kNearest Neighbor – kNN) на группы близких векторов. Пусть мы разбили пространство векторов на *m* групп. Тогда каждый отзыв можно представить вектором $(g_1, g_2, \ldots, g_m)$, где g_i – количество слов в отзыве, принадлежащих *i*-группе. После такого представления отзывов в виде векторов одинаковой размерности мы можем применить к ним один из известных классификаторов.

2.2. Задача поиска групп схожих отзывов

Данная задача относится к классу задач обучения без учителя. Известны

только описания множества объектов. Классическим подходом для решения этой задачи является представление объекта (отзыва) в виде набора характеристик (признаков) и учет расстояний между объектами при создании кластеров (матрица расстояний). Поэтому часто данная задача решается графовыми алгоритмами кластеризации.

В работе [7] делается попытка создания автоматического поиска дубликатов отзывов для системы отслеживания ошибок Bugzilla. В работе проводится предобработка исходного набора документов и сведение задачи к решению задачи кластеризации на графах, решение которой, в свою очередь взято из [12], где графы применяются для исследования групп в социальных сетях. Данная система отслеживания ошибок используется для таких крупных проектов, как Mozilla, Eclipse. Данная система отслеживания ошибок является открытой базой данных отчетов об ошибках, в которой вручную размечены дубликаты ошибок. Для превращения отчетов об ошибках в вектора признаков была использована модель набор слов (bag of words). Использованы такие инструменты предобработки документа, как стемминг, лемматизация, удаление стоп-слов. Стоит отметить, что было сформировано два набора текстов – заголовки и текст отчета учитывались отдельно, так как дают разный вклад в отчет. Каждый документ представлялся вектором $(w_1, w_2, \ldots, w_m)$, где $w_i = 3 + 2*\log_2$(количество i слова в документе). Данные коэффициенты были найдены на основе набора документов, важна лишь логарифмическая зависимость. Далее, рассчитывается расстояние между словами (схожесть двух векторов признаков) с помощью популярной формулы нахождения косинуса угла между ними:

$$\textit{схожесть} = \cos(\theta) = \frac{v_1 * v_2}{|v_1| * |v_2|} \quad (2)$$

Далее применяется подход к кластеризации графа, взятый из [12]. При появлении нового отчета об ошибке, вычисляется вектор его заголовка и тела и

в зависимости от близости к кластерам, принимается решение о том, является ли он дубликатом или нет. В работе [16] делаются попытки улучшить качества классификатора, используя некоторые расширения и допущения, но идея остается той же самой.

В работе [8] проблема решается иным образом. Каждый отчет об ошибке представляется набором следующих признаков:

1) заголовок
2) тело отчета (описание отзыва)
3) название продукта
4) компонент продукта (может быть указан при составлении отчета об ошибке)
5) тип
6) приоритет
7) версия продукта

На основе данного набора признаков для любых двух отчетов об ошибках вычисляются “сравнения” по следующим формулам:

$$\textit{сравнение}_1(d_1, d_2) = BM25F(d_{1x}, d_{2x}),\ x=1,2\ (\textit{униграммы}) \quad (3)$$

$$\textit{сравнение}_2(d_1, d_2) = BM25F(d_{1x}, d_{2x}),\ x=1,2\ (\textit{биграммы}) \quad (4)$$

$$\textit{сравнение}_3(d_1, d_2) = \begin{cases} 1 & \textit{если}\ d_{13}=d_{23} \\ 0 & \textit{иначе} \end{cases} \quad (5)$$

$$\textit{сравнение}_4(d_1, d_2) = \begin{cases} 1 & \textit{если}\ d_{14}=d_{24} \\ 0 & \textit{иначе} \end{cases} \quad (6)$$

$$\textit{сравнение}_5(d_1, d_2) = \begin{cases} 1 & \textit{если}\ d_{15}=d_{25} \\ 0 & \textit{иначе} \end{cases} \quad (7)$$

$$\textit{сравнение}_6(d_1, d_2) = \frac{1}{1+|d_{16}-d_{26}|} \quad (8)$$

$$\textit{сравнение}_7(d_1, d_2) = \frac{1}{1+|d_{17}-d_{27}|} \quad (9)$$

Для данных формул d_{ij} – означает j признак i документа.

В первых двух формулах вычисляется близость между заголовками и

телами отчетов двух документов по формуле [15]:

$$BM25F(d,q)=\sum_{t\in d\cap q} IDF(t)*\frac{TF_D(d,t)}{k_1+TF_D(d,t)} \quad (10), \quad где$$

$$IDF(t)=\log\frac{N}{N_d}$$

$$TF_d(d,t)=\sum_{f=1}^{K}\frac{w_f*Occur(d[f],t)}{1-b_f+\frac{b_f*l_f}{al_f}}$$

В которых, t – терм, присутствующий в обоих документах d, q; k_1 – подбираемый параметр; D – набор документов; N – количество всех документов, а N_d – количество документов, содержащих терм t; K – одинаковое для всех документов количество полей, на которые разбивается документ(например: заголовок и тело отчета). W_f – вес поля f; $Occur(d[f],t)$ – количество появлений терма t в поле f; l_f – размер поля f в термах, al_f – средний размер поля f среди всех документов; b_f – настраиваемый параметр, определяющий масштабирование длины поля.

В целом, эта формула является усовершенствованной формулой TF-IDF.

В итоге, для любых двух документов мы имеем 7 вычисляемых признаков. Далее, добавляются дополнительные признаки - порядковые номера отчетов об ошибках и схожесть по формуле косинусов (на основе вектора из 7 вычисляемых ранее признаков). Используются несколько известных алгоритмов, таких, как логистическая регрессия, наивный байес для обучения на основе уже проведенной классификации (дубликат или нет) среди всех документов с последующим сравнением качества классификации.

3. Основные идеи и методики решения задачи

Для решения задачи классификации по типам отзывов были реализованы вышеперечисленные подходы.

Перед тем, как применять упомянутые методы, требуется произвести предварительную обработку отзывов.

Классическими приемами по предварительной обработке отзывов являются приведение всех слов в отзыве к нижнему регистру, стемминг, лемматизация, удаление стоп-слов.

Приведение слов к нижнему регистру может отрицательно сказаться на качестве классификации для деловых и официальных документов. Мы неизбежно теряем часть информации, которая может быть использована при классификации. Отзывы же, напротив, не обладают строгой формой и содержат в себе много ошибок, что делает регистр букв неинформативным, поэтому далее не учитываем.

Стемминг и лемматизация могут как улучшить, так и ухудшить точность классификации. Для некоторых задач, знание о числах, глагольных временах оказывается полезной. Проверим в дальнейшем эффективность данной предобработки на нашем наборе данных. Будем использовать стеммер портера [17].

Также, дополнительно, удалим все цифры из отзывов. Для некоторого учета чисел мы можем любую последовательность цифр заменить некоторым кодовым словом _number_ для возможности учета их алгоритмом.

Не учитывая знаки препинания, отзыв можем представить как набор слов и для данного представления отзывов воспользоваться TF-IDF формулой.

Проведем сравнение различных алгоритмов обсужденных выше на полученном наборе отзывов.

Для каждого из алгоритмов есть один или ряд настраиваемых

коэффициентов, которые мы также будем варьировать для получения максимального значения по выбранной метрике.

3.1. Сбор данных для машинного обучения

Для решения данной задачи были взяты отзывы о программном продукте eBay из магазина приложений AppStore. Выбор основывался на популярности программного продукта, и соответственно, огромном числе отзывов для обучения. Но, очевидным образом, решение задачи применимо и к отзывам на другие продукты.

Были получены 39980 отзывов о eBay на английском языке. Каждый отзыв состоит из заголовка, тела отзыва, включает дату создания, имя пользователя, оценку по пятибалльной шкале.

Ключевой проблемой на данном этапе является отсутствие размеченных данных, неизбежность ручной разметки на определенные категории. Была проведена разметка выборки из 2000 отзывов на отчеты об ошибках, запросы функциональностей и неинформативные отзывы. Данные отзывы были выбраны случайно по всему множеству полученных отзывов (39980) и являются репрезентативными для всей выборки. Каждый из 2000 отзывов мог быть отнесен к одному или нескольким категориям.

Примеры отзывов:

1) "After new iPad replacement using restore it can't login or failure time out. Just an ever spinning wheel of uselessness. Multiple reboots, deletes and reinstalls are no help." (отчет об ошибке).

2) “Everything works pretty good no real problems just wish I was able to see my eBay bucks made per item. And total.” (запрос функциональности).

3) “Awesome App!!! On here all the time doing lots of shopping! Never had one

issue, well besides a couple of items not showing up” (неинформативный отзыв).

Особенностями отзывов на программный продукт являются средняя длина отзыва (как правило, 3-5 предложений), частые ошибки в словах и пунктуации, зашумленность текста знаками пунктуации, составляющих смайлы, большая доля отзывов с ярко выраженной положительной/отрицательной эмоциональной окраской.

После получения отзывов проводим следующую начальную предобработку данных:

1. оставляем отзывы, написанные только на английском языке.

   в выборку попали отзывы, написанные также и на испанском языке, несмотря на то, что страна с отзыва указана как Америка.
2. приводим все буквы к нижнему регистру.

   как правило, пользователи, пишущие отзывы с мобильных устройств не заботятся о правильном регистре и данная информация может быть учтена, но лишь после тщательного анализа.

3.2. Метрики, используемые для оценки качества

Для оценки качества работы алгоритмов будем использовать площадь под графиком ROC кривой. Данная кривая помогает оценить качество бинарной классификации. ROC кривая показывает зависимость доли верных положительных классификаций от доли ложных положительных классификаций при варьировании порога решающего правила (TPR(FPR)).
Доля ложных положительных классификаций (False Positive Rate, FPR):

$$TPR(a, X^m) = \frac{\sum_{i=1}^{m}[a(x_i) = +1][y_i = +1]}{\sum_{i=1}^{m}[y_i = +1]} \quad (11)$$

Доля верных положительных классификаций (True Positive Rate, TPR):

$$FPR(a, X^m) = \frac{\sum_{i=1}^{m}[a(x_i) = +1][y_i = -1]}{\sum_{i=1}^{m}[y_i = -1]} \quad (12)$$

где $a(x)$ – классификатор, $X^m = (x_1, x_2, \ldots, x_m)$ – выборка объектов, $(y_1, y_2, \ldots, y_m)$ – соответствующие им верные ответы.

Строится данная ROC кривая следующим образом. Пусть требуется разделить множество X на два класса: положительный (+1) и отрицательный (-1). Пусть с помощью классификатора $a(x)$ мы каким-либо образом можем получить вероятность $f(x)$ того, что объект x отнесен к положительному классу. Тогда алгоритм вычисления ROC кривой следующий:

1) Вычисляем представителей классов +1 и -1 в выборке: m_- и m_+ соответственно.
2) Упорядочим объекты $(x_1, x_2, \ldots, x_m)$ по убыванию значений $f(x)$
3) Начальная точка ROC кривой: $(FPR_0, TPR_0) := (0, 0)$
4) Для каждого i от 1 до m.

   Если $y_i = -1$, то

$$FPR_i = FPR_{i-1} + \frac{1}{m_-}; \quad TPR_i := TPR_{i-1} \quad (13)$$

   Иначе:

$$FPR_i = FPR_{i-1}; \quad TPR_i := TPR_{i-1} + \frac{1}{m_+} \quad (14)$$

Примерный результат работы алгоритма представлен на рис. 1:

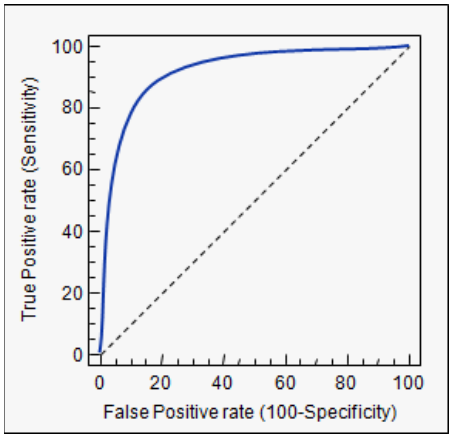


Рис. 1 AUC кривая (зависимость TPR от FPR) для 3 различных алгоритмов.

Метод оценки качества классификатора – площадь под получившейся ROC кривой. Изменяется от 0.5 (случайный классификатор) до 1.0 (идеальный классификатор) (Если значение меньше 0.5, то это значит, что мы на самом деле обозначили -1 как положительный класс, а +1 как отрицательный).

Преимущества ROC-кривой в том, что она инвариантна относительно отношения ошибок первого и второго рода.

Также, часто применяется F-мера, определяемая, как:

$$F = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (15)$$

где Precision – отношение числа правильно классифицированных объектов положительного класса к общему числу положительных объектов от классификатора, а Recall – отношение числа правильно классифицированных объектов положительного класса к общему числу положительных объектов.

Для классификации отзывов по типам: отчет об ошибке, запрос функциональности, однако, данная F-мера не совсем показательна. Связано это с тем, что в данной задаче для нас более важным оказывается отследить как можно больше отчетов об ошибках и запросов функциональностей. То есть, в данной задаче Recall для нас имеет большее значение, чем Precision. Для учета

этого, в F метрике мы можем ввести дополнительный коэффициент, показывающий “важность” Recall относительно Precision. Но для объективных результатов, мы этого не делаем, и будем явно вычислять Recall.

Площадь под графиком ROC кривой вычисляется, как правило, для бинарного классификатора. При наличии нескольких классов (отчеты об ошибках, запросы функциональностей, неинформативные отзывы) мы обучаем алгоритмы для каждого из упомянутых классов отдельно: отзыв принадлежит классу A – отзыв не принадлежит классу A. Таким образом, для одного набора данных имеем 3 значения метрики для каждого из классов. Аналогично вычисляем Recall метрику. Это может помочь нам в дальнейшем увидеть некоторые особенности классификации по каждому классу.

3.3 Полученные результаты

Описанные алгоритмы были реализованы на языке программирования Python 2.7. Напомним, что в исследовании использовался набор отзывов пользователей о продукте eBay с AppStore. Были оставлены 1974 отзыва на английском языке, и размечены по классам:

1) отчеты об ошибках
2) запросы функциональностей
3) неинформативные отзывы

Для определения качества классификатора использовались упомянутые выше метрики. Для нахождения более объективной оценки качества классификатора использовалась перекрестная проверка (cross-validation) на 3 частях. Для этого данные разбивались на k=3 частей (причем так, чтобы соотношение разных классов в каждой части было примерно таким же, что и на всей выборке). Затем, на k-1 частях данных производилось обучение модели, а оставшаяся часть данных использовалась для тестирования и определения

значения метрики. Данная процедура повторялась k раз, и значение метрики для всей выборки находилось как среднее арифметическое значений на разных ее частях.

На выборке каждая часть содержала около 660 отзывов. Полученные результаты отражены в таблице 1.

		Метрики	
		Отчеты об ошибках (ROC AUC)	**Отчеты об ошибках (recall)**
Методы	**NB**	0.827	0.732
	NB, (4,10)-grams	**0.867**	**0.839**
	NB, (4,10)- '_'-grams	0.855	0.826
	NB, -stopwords	0.824	0.711
	NB, -stopwords, +meta features	0.843	0.799
	NB, tf-idf	**0.555**	**0.107**
	NB, (4,10)-grams, -stopwords	0.837	0.779
	SVM	0.781	0.583
	SVM,(4,10)-grams	0.771	0.584
	word to vec (NN)	0.665	0.392
		Метрики	
		Запросы функциональностей (ROC AUC)	**Запросы функциональностей (recall)**
Методы	**NB**	0.739	0.537
	NB, (4,10)-grams	0.774	0.622
	NB, (4,10)- '_'-grams	0.759	0.603
	NB, -stopwords	0.743	0.596
	NB, -stopwords, +meta features	0.774	0.636
	NB, tf-idf	**0.65**	**0.294**
	NB, (4,10)-grams, -stopwords	**0.803**	0.699
	SVM	0.677	0.381
	SVM,(4,10)-grams	0.683	0.394
	word to vec (NN)	0.671	0.449

Таблица 1: Сравнение классификаторов с различной предобработкой отзывов. NB – наивный байесовский классификатор, SVM – метод опорных векторов, word to vec (NN) – применение нейронной сети для превращения слов в вектора с дальнейшим использованием kNN алгоритма. Зеленым цветом обозначен наилучший показатель по данной метрике среди всех методов. Красным цветом обозначен наихудший показатель по данной метрике среди всех методов.

Таблица 1 отражает результаты проведенного эксперимента. Для каждого из рассматриваемых алгоритмов применяли различную предобработку. NB – наивный байесовский классификатор на словах. NB, (4-10)-grams – наивный байесовский классификатор, но были использованы n-граммы символов вместо слов. В экспериментах n могло изменяться от 2 до 14. Наилучшая

классификация (ROC метрика) для n от 4 до 10, что и отражает “(4-10)-grams” запись. Аналогично, “(4-10)-'_'-grams” запись означает использование n-грамм, но с учетом пропусков между словами в виде дополнения n-грамм справа и слева символами '_'. Запись “+ meta features” означает использование мета информации из отзыва. А именно, были учтена следующая мета информация. 1) Информация о временах глаголов. В каждом отзыве были подсчитаны количества глаголов прошедшего, настоящего и будущего времени. Получившиеся 3 числа были нормированы и добавлены к вектору отзыва. 2) Длина отзыва в виде количества символов исходного отзыва. 3) Каждый отзыв из магазина приложений AppStore имеет рейтинг в виде количества звезд (от 1 до 5). Данный рейтинг отражает удовлетворение пользователем, написавшим отзыв, продуктом. Рейтинг в виде числа 1-5 был также учтен при классификации.

Более детальная информация о поведении данных классификаторов представлена в приложении А. Представлена зависимость качества классификации от размера выборки.

Как видно из таблицы 1, в целом, качество классификации запросов функциональностей продукта ниже, чем отчетов об ошибках. Это происходит в основном из-за малого количества запросов функциональностей в исходных данных: только 10% всех отзывов являются запросами функциональностей продукта, тогда как отчетов об ошибке около 25% от всей выборки. Другая причина такого поведения может быть связана со специфичностью запросов функциональностей продукта, которые часто представляют собой нечто среднее между отчетами об ошибках и неинформативными отзывами, отражающими в целом, рейтинг продукта. Например:

1) “Everything works pretty good no real problems just wish I was able to see my eBay bucks made per item. And total.” Данный отзыв следует отнести к запросу

функциональности, тем не менее, лишь “just wish” указывает нам на это.
2) "PRICES IN THE ADVERTISING is so annoying and wrong. Every time I see, for example, C$2.98 and US $2.25, I know that I'm supposed to pay theu.s. Price but I keep getting charged the C price. I know I need to probably take this up with PayPal, however, I just want to see the U.S. Prices first and just get charged that so I don't have to deal with Paypal on that end. PLEASE CHANGE THIS!" Только часть этого отзыва относится к запросу функциональности продукта. Данный отзыв может быть отнесен алгоритмом как к отчету об ошибке, так и к неинформативному отзыву.

Также, интересно, что при всем наборе настраиваемых параметров для SVM метода, метод опорных векторов не смог оказаться лучше, чем наивный байесовский классификатор.

Использование n-грамм вместо слов в отзыве улучшает классификацию. Связано данное обстоятельство с большим количеством ошибок в отзывах, а использование n-грамм позволяет учесть ошибки в отзыве, считая только части слов.

4. Использование модуля обработки ошибок

4.1. Мотивация

По результатам экспериментов, можно заключить, что ошибки в отзывах могут критично сказываться на их классификации. Для устранения этого недостатка предлагается создать модуль обработки ошибок. Далее, сравнить улучшение качества классификации при использовании данного модуля и оценить целесообразность как внедрения его в общую схему, так и последующего улучшения модуля обработки ошибок.

4.2. Эксперимент

Реализация данного модуля следующая. Считаем, сколько раз каждое слово встречается в наборе отзывов. Введем некоторый параметр k, такой, что если некоторое слово w_i встречается в наборе отзывов более, чем k раз, то будем считать слово написанным без ошибок. (Здесь мы исходим из предположения, что в одном и том же слове разные люди делают разные ошибки). Данное определение группы слов важно для учета слов, специфичных для выбранной области (отзывы на программный продукт eBay). Например, слова “app”, “eBay”, “iPad” и так далее. Также определяем некоторый параметр m, такой, что если некоторое слово w_i встречается в наборе отзывов менее, чем m раз, то будем расценивать такое слово w_i как “подозрительное на ошибки” и в дальнейшем будем исправлять ошибки лишь в таких словах.

Для исправления ошибок в словах w_i воспользуемся априорным распределением слов в некотором большом документе, написанном без ошибок или с их минимальным числом. А именно, возможны 6 вариантов:

1) Ошибок нет

2) Пропущен пробел и w_i представляет собой 2 слова
3) Вставка лишнего символа
4) Перестановка букв
5) Замена буквы
6) Пропуск буквы

Каждое “подозрительное на ошибки” слово w_i разбивается в соответствии со всеми вариантами на слова по каждой букве и с помощью большого документа находится вероятность присутствия такого слова в тексте. Слово с наибольшей вероятностью присутствия в документе w_i' заменяет исходное слово w_i. Если вероятность присутствия слова w_i' в документе меньше, чем некоторая вероятность p, то поиск замены для слова w_i производится среди выше определенных слов, специфичных для выбранной области.

Предложенный модуль действительно позволил исправить большую часть ошибок. Площадь под графиком ROC кривой для наивного байесовского классификатора при классификации отчетов об ошибках увеличилась с 0.827 до 0.833. Данное улучшение следует воспринимать как существенное и учитывать такой модуль в дальнейшем.

5. Учет эмоциональной окраски отзывов

5.1. Мотивация

В экспериментах были получены результаты классификации с использованием мета информации и без нее. Результаты показали, что использование мета информации позволяет заметно улучшить качество классификации, тем не менее, использование информации о рейтинге о программном продукте из отзыва не всегда возможно. (Особенно в тех случаях, когда стоит задача создания собственной системы по классификации отзывов или внедрения такой системы в уже существующие системы отслеживания ошибок, например, Atlassian JIRA). Проблема состоит в определении рейтинга (эмоциональной окраски отзыва) на основе только тела и заголовка отзыва.

5.2 Эксперименты

Для учета эмоциональной окраски был обучен наивный байесовский классификатор на отзывах по 7 программным продуктам из AppStore. Как и при классификации на группы отзывов, была произведена предварительная обработка входных данных. Приведение букв к нижнему регистру, удаление пунктуации, стемминг, лемматизация, удаление стоп-слов.

После обучения классификатора на 7 программных продуктах, он способен предсказать рейтинг (эмоциональную окраску) отзывов с eBay. Для сравнения качества классификации с учетом априорного рейтинга отзывов с eBay и с учетом рейтинга, полученного с помощью классификатора эмоциональной окраски, была проведена аналогичная предобработка отзывов с добавлением мета информации. Для сравнения приведем графики зависимости площади под ROC кривой от размера выборки данных. Размер выборки приводится в доле от 70% всех отзывов пользователей, полученных о eBay. Набор этих отзывов

является тренировочными данными. Остальные 30% отзывов пользователей являются тестовыми данными, на которых мы проверяем качество классификатора.

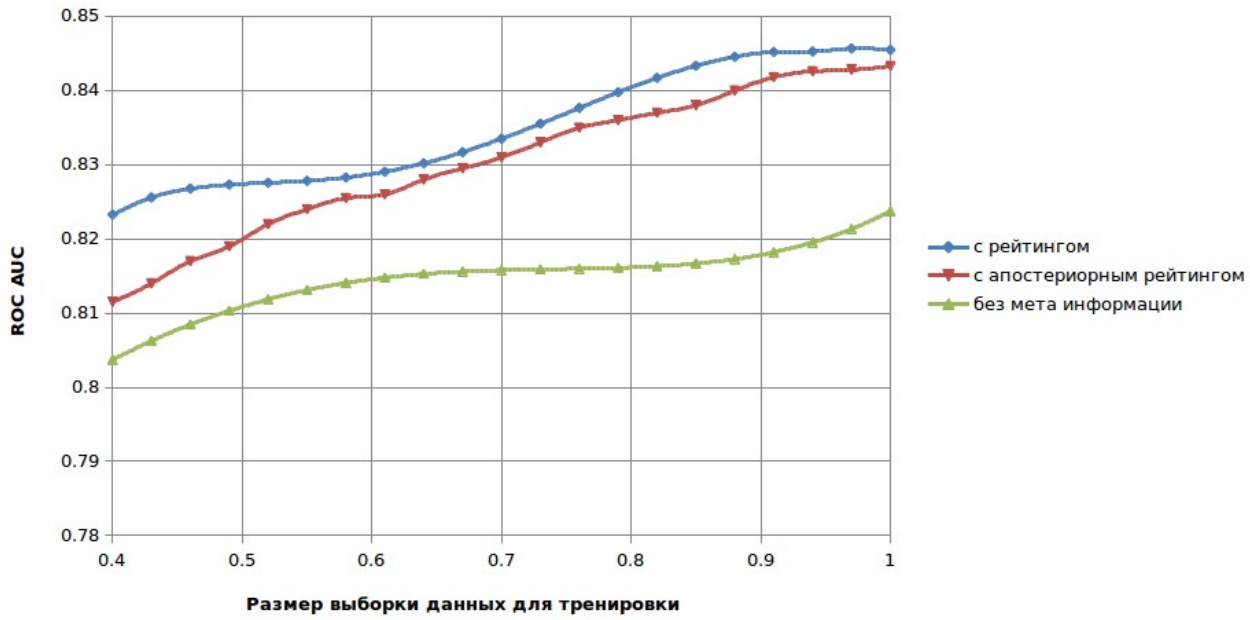


Рис. 2 Зависимость значения метрики ROC AUC от размера выборки. Размер выборки приведен в доле от тренировочного набора данных, состоящего из 1974 отзывов.

Отсутствие априорной информации об эмоциональной окраске отзыва ухудшило классификацию, тем не менее, учет информации о рейтинге, полученной косвенным образом с помощью ранее обученного классификатора позволило снизить проигрыш в классификации. Данный прием хорошо показал себя при улучшении классификации.

6. Обсуждение и выводы

По итогам обзора литературы и проведенного исследования было установлено, что данная задача может быть решена методами машинного обучения. Также было установлено, что задача классификации отзывов на запросы функциональностей оказалась для методов машинного обучения более сложной, чем задача классификации на отчеты об ошибках. Было выяснено, что наивный байесовский классификатор на n-граммах показывает лучшие результаты на ROC AUC и Recall метрике. Такое поведение может означать большую долю ошибок, опечаток при наборе отзывов. Также, интересен тот факт, что учет стоп слов ухудшает качество классификации запросов функциональностей, в то время, как классификация отчетов об ошибках улучшается. Также было показано, что учет мета информации улучшает классификацию. Был продемонстрирован способ учета эмоциональной окраски отзывов при классификации не размеченных по рейтингу данных. Был представлен модуль обработки ошибок отзывов и проведены эксперименты по влиянию его на классификацию отзывов.

По результатам исследований наилучшим классификатором является наивный байесовский классификатор на n-граммах с учетом мета информации (длина отзыва, времена глаголов), модулем обработки ошибок и на основе вычисляемой эмоциональной окраски отзыва.

Дальнейшая работа по улучшению классификатора может быть связана с учетом цифр в отзыве. Так, например, возможна корреляция между наличием чисел в отзыве и отнесением его к отчетам об ошибках или запросам функциональностей. Пользователь может указывать на недостаток/избыток каких-либо свойств программного продукта или описывать количество повторяющихся негативных событий при использовании программного продукта. Также, учет времени отправки отзыва может помочь улучшить

классификацию. С течением времени программный продукт развивается, исправляются старые ошибки, создаются новые. Относительное количество отзывов в каждой из групп меняется вместе с общей тематикой отзывов, что может быть также учтено при классификации.

Для реализации представленной двухуровневой модели требуется также экспериментально исследовать эффективность представленных методов по определению групп схожих отзывов и выбрать наилучший. Дальнейшие исследования могут быть направлены на создание общей модели на основе системы отслеживания ошибок Atlassian JIRA.

7. Заключение

В данной работе получены следующие результаты:

1) Разработан 2-х этапный метод классификации и группировки схожих отзывов пользователей о программном продукте.

2) Проведено сравнение различных алгоритмов машинного обучения с целью поиска наилучшего для решения задачи классификации отзывов.

3) Выявлен наилучший алгоритм для решения задачи классификации.

4) Предложен метод учета информации об эмоциональной окраске отзыва.

5) Показано, что применение предложенного метода по использованию эмоциональной тональности отзыва позволяет улучшить recall и AUC ROC.

8. Список литературы

[1] Gartner. Number of mobile app downloads worldwide from 2009 to 2017 (in millions). Technical report, Gartner Inc., March 2015.

[2] H. Li, L. Zhang, L. Zhang, and J. Shen. A user satisfaction analysis approach for software evolution. In Progress in Informatics and Computing (PIC), 2010 IEEE International Conference on, volume 2, pages 1093–1097. IEEE, 2010.

[3] L. V. Galvis Carreño and K. Winbladh. Analysis of user comments: an approach for software requirements evolution. In ICSE ’13 Proceedings of the 2013 International Conference on Software Engineering, pages 582–591. IEEE Press, 2013

[4] D. Pagano and W. Maalej. User feedback in the appstore: an empirical study. In Proc. of the International Conference on Requirements Engineering - RE ’13, pages 125–134, 2013.

[5] L. Hoon, R. Vasa, J.-G. Schneider, and J. Grundy. An analysis of the mobile app review landscape: trends and implications. Technical report, Swinburne University of Technology, 2013.

[6] W. Maalej, H. Nabil Bug Report, Feature Request, or Simply Praise? On Automatically Classifying App Reviews. IEEE RE, 2015.

[7] N. Jalbert and W. Weimer. Automated duplicate detection for bug tracking systems. In DSN '08, pages 52-61, 2008.

[8] A. Alipour. A Contextual Approach Towards More Accurate Duplicate Bug Report Detection. Master’s thesis, University of Alberta, Canada, 2013.

[9] Thorsten Joachims, “Text Categorization with Support Vector Machines: Learning with Many Relevant,” in Proceedings of the 10th European Conference on Machine Learning, London, 1998, pp. 137-142.

[10] CAVNAR,W.B.AND TRENKLE, J. M. 1994. N-gram-based text categorization.

In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval (Las Vegas, NV, 1994), 161-175.

[11] Rocha, H, Oliveira G, Maques-Neto H, Valente MT (2014) NextBug: A Tool for Recommending Similar Bugs in Open-Source Systems In: V Brazilian Conference on Software: Theory and Practice – Tools Track (CBSoft Tools), 53–60.. SBC, Maceio, AL, Brazil,.

[12] Xiaoyin Wang , Lu Zhang , Tao Xie , John Anvik , Jiasu Sun, An approach to detecting duplicate bug reports using natural language and execution information, Proceedings of the 30th international conference on Software engineering, May 10-18, 2008, Leipzig, Germany.

[13] John Anvik , Lyndon Hiew , Gail C. Murphy, Who should fix this bug?, Proceedings of the 28th international conference on Software engineering, May 20-28, 2006, Shanghai, China.

[14] N. Mishra, R. Schreiber, I. Stanton, and R. E. Tarjan. Clustering social networks. In Workshop on Algorithms and Models for the Web-Graph (WAW2007), pages 56-67, 2007.

[15] C. Sun, D. Lo, S.-C. Khoo, and J. Jiang. Towards more accurate retrieval of duplicate bug reports. In ASE'11, pages 253-262. IEEE CS, 2011.

[16] Y Tian, D. Lo, and C. Sun, "Improved duplicate bug report identification," in CSMR, 2012.

[17] C.J. van Rijsbergen, S.E. Robertson and M.F. Porter, 1980. New models in probabilistic information retrieval. London: British Library (British Library Research and Development Report, no. 5587).

[18] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their composi-
tionality. In Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a

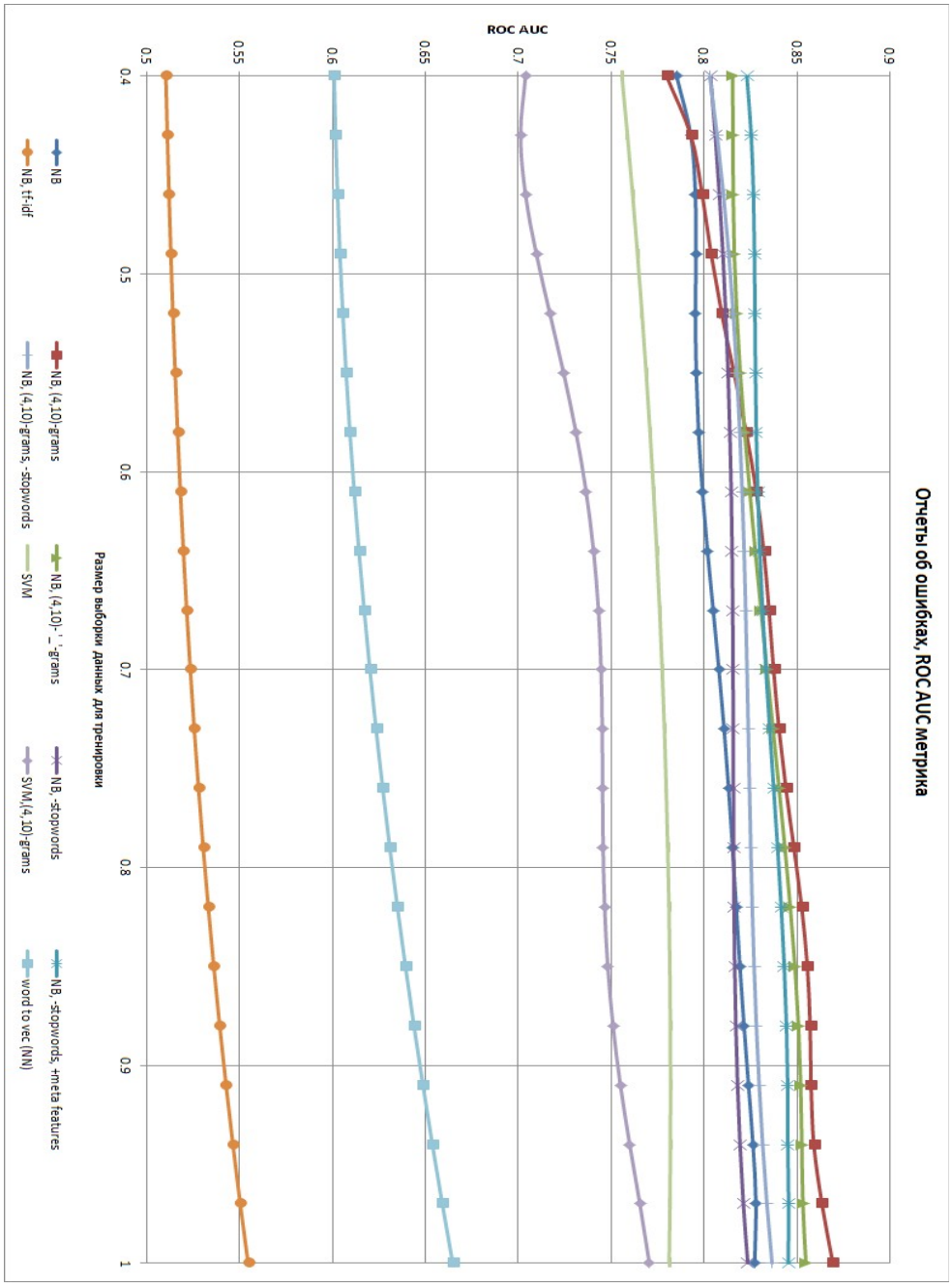
meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States, pages 3111–3119, 2013.

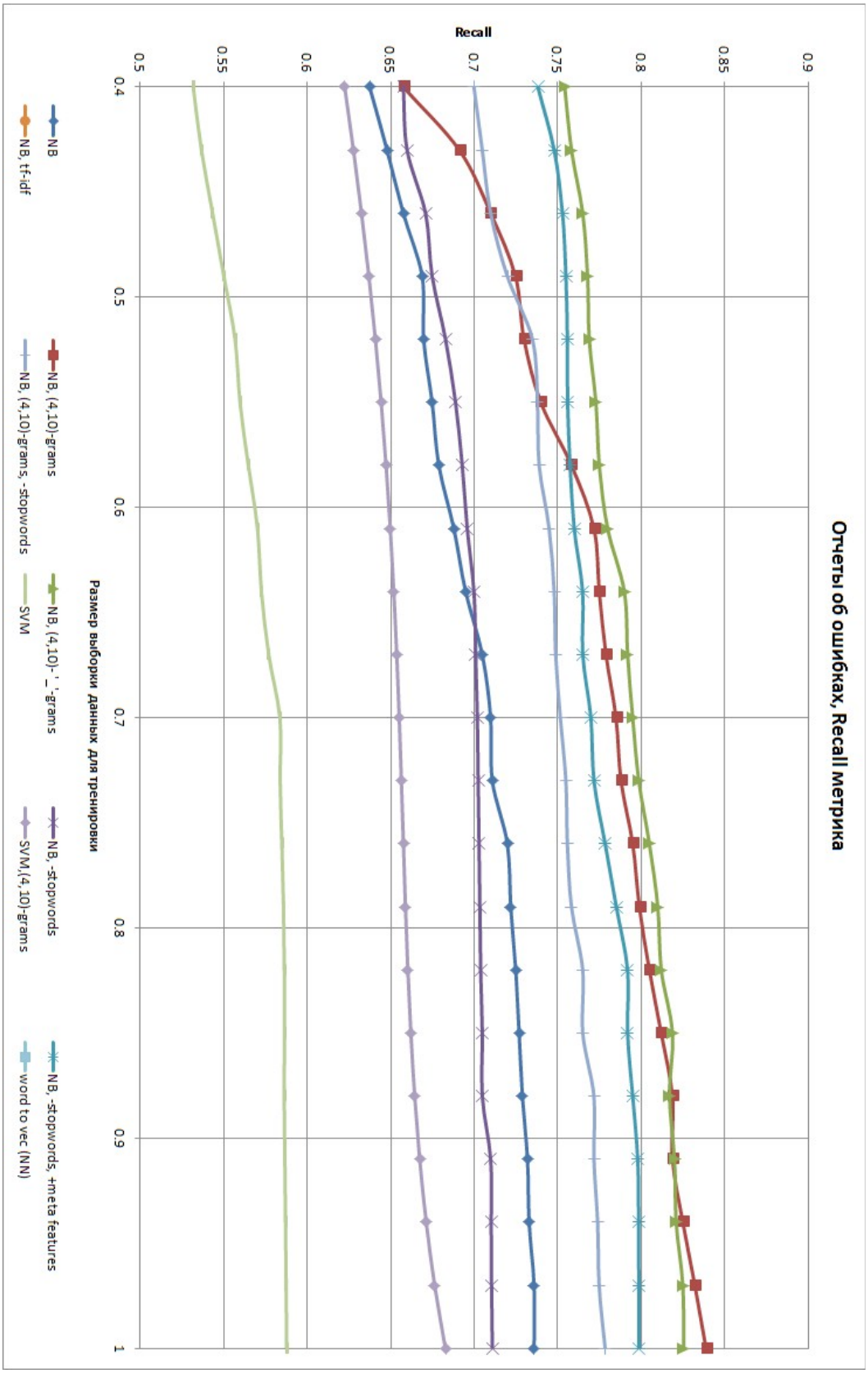
[19] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

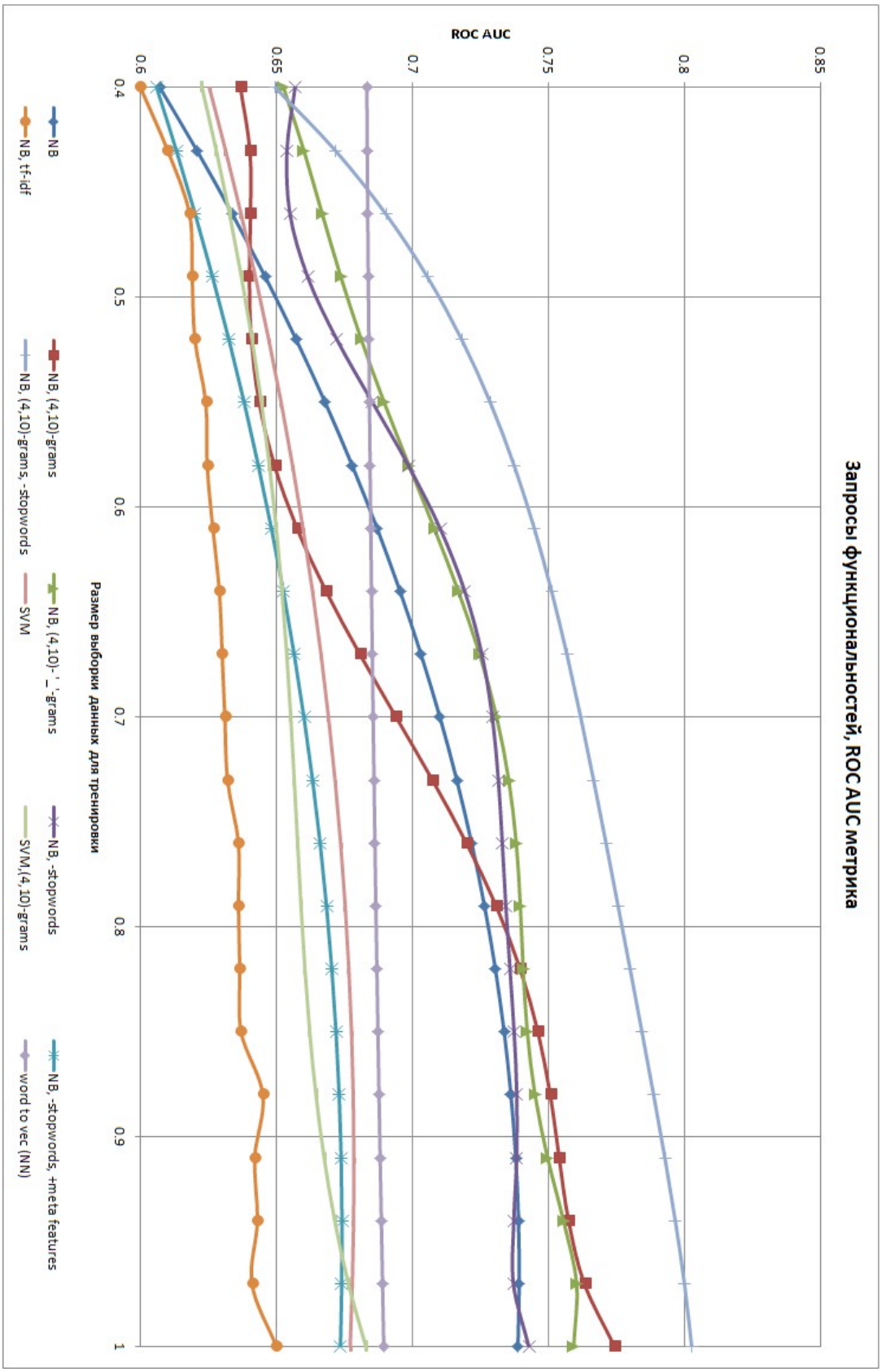
[20] Morin, F. and Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In AISTATS, volume 5, pages 246-252. Citeseer.

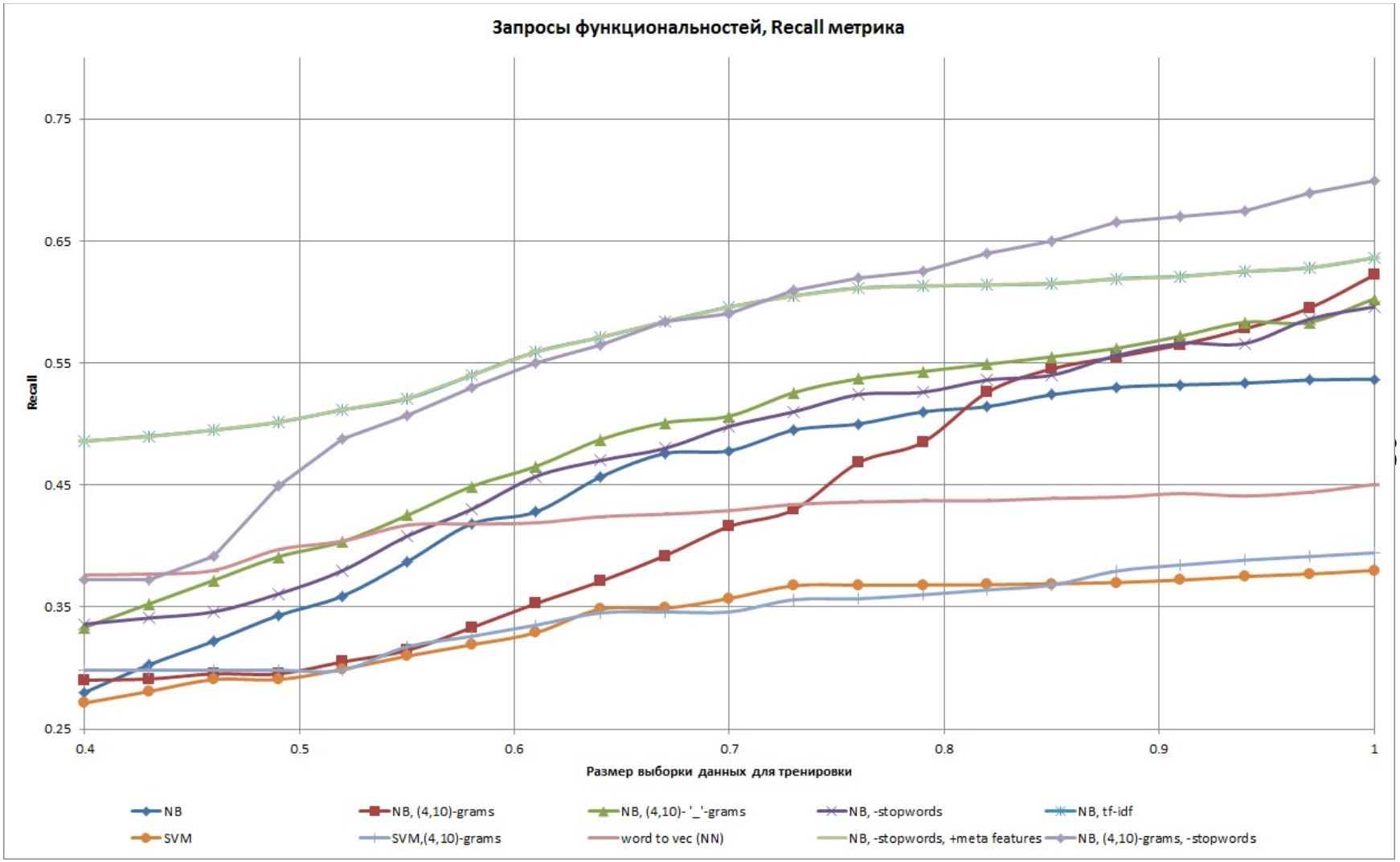
Приложение А. Зависимость метрик от размера выборки

Ниже приведены зависимости метрик ROC AUC и Recall от размера выборки при классификации отдельно запросов функциональностей и отчетов об ошибках. Размер выборки представлен в виде доли от количества тренировочных данных (около 1400 отзывов). Тестовая выборка состоит из около 600 отзывов.




Отчеты об ошибках, Recall метрика
Recall
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0.4
0.5
0.6
0.7
0.8
0.9
1
Размер выборки данных для тренировки
NB
NB, (4,10)-grams
NB, (4,10)-_-grams
NB, -stopwords
NB, -stopwords, +meta features
NB, tf-idf
NB, (4,10)-grams, -stopwords
SVM
SVM,(4,10)-grams
word to vec (NN)


Запросы функциональностей, ROC AUC метрика
ROC AUC
0.6
0.65
0.7
0.75
0.8
0.85
0.4
0.5
0.6
0.7
0.8
0.9
1
Размер выборки данных для тренировки
NB
NB, tf-idf
NB, (4,10)-grams
NB, (4,10)-grams, -stopwords
NB, (4,10)-' '-grams
SVM
NB, -stopwords
SVM,(4,10)-grams
NB, -stopwords, +meta features
word to vec (NN)


Запросы функциональностей, Recall метрика
Recall
0.75
0.65
0.55
0.45
0.35
0.25
0.4
0.5
0.6
0.7
0.8
0.9
1
Размер выборки данных для тренировки
NB
NB, (4,10)-grams
NB, (4,10)- '_'-grams
NB, -stopwords
NB, tf-idf
SVM
SVM,(4,10)-grams
word to vec (NN)
NB, -stopwords, +meta features
NB, (4,10)-grams, -stopwords