САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

КАФЕДРА ИНФОРМАЦИОННЫХ СИСТЕМ

Чернобай Юлия Владимировна

Магистерская диссертация

Разработка рекомендательной системы на основе извлечения и анализа интересов пользователей

Направление 01.04.02

Прикладная математика и информатика

Магистерская программа Математическое моделирование в задачах естествознания

Научный руководитель,
кандидат физ.-мат. наук,
доцент
Добрынин В. Ю.

Санкт-Петербург
2017

Содержание

Введение

Задача разработки рекомендательных систем появилась относительно недавно – в век развития Интернета и информационных технологий количество доступной информации увеличилось настолько, что человек не способен просмотреть ее всю, чтобы выбрать только интересную ему.

Поэтому многие современные сервисы создают рекомендательные системы, которые на основе информации о профиле пользователя и его предыдущему поведению в системе пытаются определить, какие объекты, товары или услуги могут быть ему интересны. Объектами могут быть товары, книги, музыка, фильмы, новости и т. д. Яркими примерами могут служить такие сервисы или сайты, как «Юлмарт», «КиноПоиск», «Яндекс.Дзен» и многие другие. «Юлмарт» — российский онлайн-магазин по продаже непродовольственных товаров и цифрового контента. При просмотре на сайте компании информации о товаре он также отображает список дополнительно рекомендованных товаров на основе просмотров пользователя. Кинопоиск – российский сайт о фильмах, который предлагает пользователю фильмы на основе его предпочтений. «Яндекс.Дзен» – расширение для браузера от компании «Яндекс», ставшее популярным, ищущее в Интернете информацию, которая может быть интересна пользователю, и собирающее ее в персональную ленту.

Как видно из примеров, рекомендательные системы могут служить инструментом для увеличения продаж, продажи более разнообразных объектов, увеличения лояльности пользователей, а также улучшения понимания пользовательских потребностей и желаний. Поэтому они быстро набирают популярность и начинают широко применяться в электронной коммерции, при поиске фильмов, музыки, ПО, научных статей, а также на новостных сайтах и в справочных центрах, а задача разработки эффективных рекомендательных систем является актуальной.

Выделяют два основных метода построения рекомендательных систем — метод фильтрации на основе содержания и метод коллаборативной фильтрации.

Методы фильтрации на основе содержания основаны на описании объекта и профиле предпочтений пользователя. Описанием объекта является конечное множество его дескрипторов, таких как ключевые слова, бинарные дескрипторы и т. д, а профиль предпочтений представляет собой взвешенный вектор дескрипторов объекта, в котором веса показывают важность каждого дескриптора для пользователя и его вклад в принятие конечного решения. Этот подход пытается подобрать объекты, похожие на те, которые нравились пользователю ранее, и опирается на методы информационного поиска и машинного обучения. Популярными сервисами, использующими этот тип рекомендательных систем, являются Rotten Tomatoes, Internet Movie Database и Pandora Radio.

Метод коллаборативной фильтрации базируется на информации об истории поведения пользователей в системе. Например, могут использоваться данные о покупках или оценках. В этом случае для пользователя находятся похожие на него по истории пользователи, и рекомендация основывается их на отношении к объекту. На данном методе основаны рекомендательные системы таких сервисов, как Last.fm и Amazon.com.

Постановка задачи

Исследование проводилось на данных с сайта www.livejournal.com. Эта платформа также известна в России под названием «Живой журнал» и имеет большое русскоязычное сообщество: ежемесячная аудитория – 15 млн., ежемесячно активных авторов – 110 тыс. Его пользователи пишут тексты на свободные темы – посты – и могут самостоятельно проставлять к ним ключевые слова – теги. У каждого пользователя есть также возможность заполнить информацию о себе в профиле, указав интересы, страну и школу. Для каждого поста можно узнать время его написания.

Данной работе предшествовал проект по сбору и предобработке данных с сайта livejournal.com, результаты которого были представлены на выступлении на конференции «SEIM-2016» и опубликованы в сборнике [1]. В ходе проекта было собрано 87597 пользователя с 2042011 постами и 522243 тегами к ним. Задачей данной работы является обработка и анализ собранной информации.

Будем считать, что теги, проставляемые пользователями к постам, характеризуют как их основное содержание, так и интересы самих пользователей. Например, если пользователь написал пост и проставил тег «Великая Отечественная война», то это значит, что, во-первых, сам текст написан о Великой Отечественной войне, во-вторых, что эта тема является интересом пользователя. Цель данной работы – на основе информации о пользователях и их постах построить рекомендательную систему, которая будет предлагать пользователям интересные им посты других пользователей. Платформа www.livejournal.com не дает информации об истории просмотров или оценках постов пользователями, поэтому метод коллаборативной фильтрации не может быть применен. Однако на основе информации из профиля пользователя и написанных им постов в дальнейшем может быть сформирован профиль интересов пользователя. У каждого поста есть один

или несколько ключевых слов – тегов, на основе которых легко создать профиль объекта. Поэтому целесообразно использовать метод фильтрации на основе содержания. При этом стоит заметить, что не все пользователи указывают ключевые слова к своим постам, поэтому имеет смысл обучить классификатор, который будет доставлять теги к таким постам автоматически. Поскольку для каждого тега можно определить частоту его упоминаний в определенный момент времени, можно выделять популярные в данный момент времени интересы. Назовем постоянными интересы, чья частота упоминаний остается постоянной или не имеет четкой структуры на всем промежутке времени. Временными интересами назовем те, чья частота изменяется циклически или направленно, то есть имея выраженные скачки. Временные интересы представляют большую ценность с точки зрения рекомендаций, потому что позволяют показать пользователю не только интересную, но и актуальную информацию. Поэтому одной из задач данной работы является составление алгоритма выделения временных интересов и использования их при составлении рекомендаций. Также важно научиться определять семантическую близость тегов с целью нахождения похожих тематик.

Таким образом, задача работы – провести исследование на наборе данных с целью определить лучший классификатор для выделения интересов из постов пользователей, а также построить рекомендательную систему по заданному набору данных с автоматическим выделением интересов постов и построением на их основе профилей интересов пользователей и постов. При этом также необходимо оценить целесообразность использования общей статистики по интересам, такой как временные интересы и семантическая близость интересов, при составлении рекомендаций.

Обзор литературы

Несмотря на то, что рекомендательные системы появились относительно недавно, проблеме их построения и применения в различных прикладных задачах посвящено огромное количество статей и книг. Авторы используют различные методы для улучшения качества рекомендаций и расширяют область применения рекомендательных систем.

Одной из первых публикаций на эту тему является статья Поллока [2], датируемая 1988 годом. В ней автор представил систему ISCREEN, которая позволяла пользователю самостоятельно определять некий набор правил, по которым ISCREEN в дальнейшем фильтровала текстовые сообщения и отбирала только релевантные. И хотя система не рекомендовала пользователю сообщения, сама работа показала актуальность задачи фильтрации и отбора релевантной информации.

В 1992 году появилась работа Голдберга [3], в которой авторы разработали экспериментальную систему электронной почты Tapestry, которая позволяла фильтровать email-рассылки. В отличие от работы Поллока, эта система отбирала сообщения полностью автоматически на основе реакций пользователей. Авторы статьи представили идею коллаборативной фильтрации и фильтрации на основе содержания.

Работа Голдберга имела сильный отклик и вызвала массу обсуждений. Примечательна статья Резника и Варьяна [4] 1997 года, в которой авторы рассуждают об отрицательных последствиях применения рекомендательных систем, таких как преследование недобросовестных целей при составлении рекомендаций, использование персональных данных пользователей, а также ухудшение производительности сервисов из-за работы рекомендательных систем.

Брусиловский в 1996 году в работе [5] представил новое направление в области адаптивных интерфейсов – адаптивный гипермедиа (adaptive

hypermedia). Такие системы строят строят модель каждого пользователя и используют её, например, для подстраивания содержания страницы сайта под знания и цели пользователя или предложения наиболее интересных ему ссылок. Автором были рассмотрены методы построения моделей пользователей, получившие широкое применение в дальнейшем.

В работе [6] Паццани большое внимание уделяется получению профиля пользовательских интересов для рекомендательных систем. Проведен сравнительный анализ двух основных методов фильтрации – коллаборативной фильтрации и фильтрации на основе содержания, а также описывается новый тип фильтрации – фильтрация на основе демографии, который учитывает социально-демографические показатели пользователя при составлении рекомендаций.

По мере накопления теоретического и исследовательского материала стали появляться статьи более прикладного характера, в которых исследователиискали новые приложения для применения рекомендательных систем. Например, Ким и Хан в работе [7] разработали ркомендательную систему методом коллаборативной фильтрации для показа мобильной рекламы. Аимер и Брассар в [8] создали систему для рекомендаций товаров для электронной коммерции, Голдбек в [9] представил рекомендательную систему для фильмов. Эти работы говорят о широком спектре применимости рекомендательных систем.

Задачу автоматического проставления тегов к постам можно рассмотреть как задачу мультикласовой классификации. В области машинного обучения эта задача ставится как сопоставление некоторого объекта одной или нескольким заранее определенным категориям. В случае, когда количество категорий равно двум, говорят о задаче бинарной классификации, если категорий больше – мультиклассификация.

Существует несколько методов классификации текста. Так, Владимир Вапник и Коринна Кортес в работе [10] представили метод машинного

обучения для проблемы классификации, известный как метод опорных векторов. Джочимс в статье [11] описал применение этого метода в задаче классификации текста. Наивный байесовский классификатор [12] известен с 1950 годов и остается популярным методом классификации текста. Дерево принятия решений [13] и различные его варианты также активно применяются для этой задачи. Все эти методы были изучены и использованы в работе, а по итогам проведено их сравнение.

В тех случаях, когда число классов, к одному из которых требуется определить принадлежность объекта, больше двух, говорят о задаче многоклассовой классификации (англ., multiclass classification). В свою очередь, многоклассовые классификаторы можно разделить на монотематические (англ. singlelabel classifier) и политематические (англ. multilabel classifier) – в зависимости о того, нужно ли однозначно определить принадлежность объекта к классу (как ранее формулировалось в постановке задачи) или требуется отнести объект к потенциально нескольким классам одновременно. Существует два главных метода решения задачи политематической классификации: методы преобразования проблемы(англ. problem transformation methods) и методы адаптации алгоритма (англ. algorithm adaptation methods). Методы преобразования проблемы превращают задачу политематической классификации во множество задач бинарной классификации, которые могут быть решены бинарными классификаторами. Методы адаптации алгоритма преобразовывают алгоритмы таким образом, чтобы они могли решать задачу политематической классификации без её преобразования. Среди наиболее известных методов преобразования проблемы можно выделить метод бинарной релевантности (англ. binary relevance) [14], цепной классификатор (англ. chain classifier) [15] и метод random k-labelsets [16]. Эти методы будут освещены подробнее и применены далее в работе.

Для определения семантической близости слов используются

алгоритмы дистрибутивной семантики, которые на основе огромного массива данных строят векторные представления слов, называемые контекстными векторами. Затем семантическое расстояние между словами вычисляется как косинусное расстояние между соответствующими векторами. Более полно задачи дистрибутивной семантики и ее место в прикладной лингвистике обсуждается в [17] и [18]. Для вычисления контекстных векторов часто используются методы машинного обучения. Наиболее примечательные результаты показывают нейронные сети. Существует огромное количество архитектур построения нейронных сетей [19]. Наиболее известная в дистрибутивной семантике модель word2vec использует многослойные нейронные сети для предсказания слова по окружающим словам, либо окружающие слова по контексту [20] и [21]. Она стала чрезвычайно популярной сразу после публикации ввиду высокой скорости обучения и лучшего качества векторных представлений по сравнению с другими популярными на тот момент методами. Многие исследователи до сих пор изучают свойства данной модели и возможности ее применения.

Глава 1. Обзор применяемых методов

1.1. Рекомендательные системы

Рекомендательные системы обычно подбирают список рекомендаций одним из двух способов: методом коллаборативной фильтрации или фильтрацией на основе содержания. Коллаборативная фильтрация строит модель исходя из истории поведения пользователя и составляет рекомендации на основе решений, принимаемых пользователями с похожей историей. Метод фильтрации на основе содержания использует характеристики объекта и затем рекомендует объекты с похожими свойствами. Эти методы часто комбинируют в гибридных рекомендательных системах.

Разница двух подходов может быть продемонстрирована сравнением двух популярных музыкальных рекомендательных систем: Last.fm и Pandora Radio.

Last.fm создает «станцию» рекомендованных песен на основе информации о том, какие группы и песни регулярно слушает пользователь, сравнивая его поведение с поведением других пользователей. Last.fm советует песни, которых нет в библиотеке пользователя, но которые слушают пользователи со схожим поведением. Это пример метода коллаборативной фильтрации.

Pandora рассматривает свойства песни или артиста и составляет «станцию», которая играет музыку с похожими свойствами. На основе его реакции определяются любимые и нелюбимые признаки песен, которые учитываются в модели весовыми коэффициентами. Это пример фильтрации на основе содержания.

У каждого подхода есть достоинства и недостатки. Last.fm требует огромного объема данных о поведении пользователя. Эта проблема называется проблемой “холодного старта”. В то же время Pandora требует довольно мало информации на старте, но ограничена в выборе и не способна удивить пользователя чем-то новым.

Существует множество метрик для оценки качества рекомендаций. Некоторые из них оценивают предсказания рейтинга, наиболее популярными примерами являются MAE (средняя абсолютная ошибка), MSE (средняя квадратичная ошибка) и RMSE (корень из средней квадратичной ошибки). Поскольку в данной работе рекомендательная система не предсказывает рейтинг, подробно они не рассматриваются. Существуют метрики как, например, intra-list similarity, которая оценивает разнообразие рекомендаций и дает высокие оценки рекомендациям с малой разнообразностью. Некоторые метрики оценивают целиком списки рекомендаций, для которых размечены значения релевантности. Наиболее распространенными из метрик такого типа являются Precision@n, Recall, MAP и nDCG. В Precision@n из всего списка рекомендаций выбираются n первые, а затем мера вычисляется как

$$precision@n = \frac{|N_{rel} \cap N_{retr}|}{|N_{retr}|}$$

где N_{rel} – множество релевантных пользователю объектов, N_{retr} – множество показанных пользователю объектов. Иными словами, эта мера показывает долю релевантных пользователю объектов относительно тех, которые ему показали. Похожей на нее является мера Recall:

$$recall = \frac{|N_{rel} \cap N_{retr}|}{|N_{rel}|}$$

Она показывает долю релевантных объектов, показанных пользователю, относительно всех релевантных объектов. Преимуществом обеих мер является простота вычислений, недостатком можно выделить то, что они никак не учитывают порядок рекомендаций. Мера оценки MAP (Mean

Average Precision) вычисляется на множестве списков рекомендаций размера Q как:

$$MAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q}$$

где AveP(q) считается для каждого списка отдельно по формуле

$$AveP = \sum_{k=1}^{n} P(k)\Delta r(k) = \frac{\sum_{k=1}^{n} (P(k) \times rel(k))}{|N_{rel}|}$$

где k – позиция объекта в списке рекомендаций длины n, P(k) – точность на первых k объектах, rel(k) – индикаторная функция, равна 1, если объект под номером k релевантен пользователю, и 0 в обратном случае. $|N_{rel}|$– количество всех релевантных пользователю объектов. Эта мера учитывает порядок документов, однако индикаторная функция может принимать значения лишь 0 и 1.

Метрика оценки DCG (Discounted Cummulative Gain) используется при следующих предположениях:

1. Документы с высокой релевантностью должны появляться в начале списка;

2. Функция релевантности изменяется по шкале от 0 до n, где n может быть больше 1.

Предпосылка меры DCG состоит в том, что документы с высокой релевантностью, попавшие на низкие позиции, должны штрафоваться, поэтому соответствующие веса уменьшаются логарифмически пропорционально позиции в списке. Мера DCG на списке длины p вычисляется по формуле :

$$DCG_p = \sum_{i=1}^{p} \frac{rel_i}{\log_2(i+1)}$$

Поскольку верхняя граница функции релевантности и соответственно DCG

не имеет верхней границы, ее часто нормализуют:

$$nDCG_p = \frac{DCG_p}{IDCG_p}$$

где $IDCG_p$ – мера, вычисленная на отсортированном по релевантности списке. Поскольку DCG учитывает порядок и позволяет задавать функцию релевантности в широком диапазоне, то в работе используется именно она.

1.2. Классификация текста

1.2.1. Обзор классификаторов

В машинном обучении задача классификации относится к способу обучения «обучение с учителем», где в качестве учителя выступает априорное знание о принадлежности объектов из обучающей выборки к тому или иному классу, которое чаще всего требует ручной разметки.

К классу задач машинного обучения «обучение без учителя» относится задача кластеризации, или кластерного анализа. Отличие от классификации состоит в том, что обучающая выборка отсутствует, и объекты разбиваются на классы, исходя только из степени их схожести друг с другом. Задача кластеризации выходит за рамки этой работы.

Формализуем общую задачу классификации: пусть X – множество описаний наблюдений, а Y – конечное множество обозначений (номеров, названий или др.) классов. Существует неустановленное отображение $y^* : X \to Y$ и обучающая выборка X^m, где $X^m = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, а (x_i, y_i) – описание x_i наблюдения i и обозначение категории y_i, к которой он относится. Требуется построить алгоритм $a : X \to Y$, способный классифицировать произвольный объект $x \in X$. Если Y состоит из двух классов, то говорят о двухклассовой (или бинарной) классификации, если мощность Y больше 2, то задача носит название мультиклассовой классификации. Если же элементы Y

– подмножества некоторого конечного множества классов, то есть объект x может принадлежать нескольким категориям одновременно, то говорят о задаче политематической классификации.

Задачу бинарной классификации решает метод опорных векторов. Он предполагает линейную разделимость классов и строит разделяющую классы гиперплоскость. Из всех возможных разделяющих поверхностей выбирается та, что дает максимально возможный зазор между классами. Метод опирается на предположение о том, что увеличение расстояния от разделяющей гиперплоскости до экземпляров разделяемых классов уменьшает среднюю ошибку классификатора. Опорными векторами выступают те примеры из обучающей выборки, которые ближе всего находятся к итоговой разделяющей гиперплоскости и участвуют в ее формировании. В случае, когда гарантировать линейную разделимость классов нельзя, используется переход из исходного n-мерного пространства, заданного вектором признаков, в пространство более высокой размерности, за которым следует поиск разделяющей гиперплоскости с максимизацией зазора. Преобразования, используемые для перехода в пространство большей размерности, называют функциями ядер.

Поскольку метод опорных векторов не может решить задачу мультиклассовой классификации в явном виде, существует ряд методов преобразования задачи мультиклассовой классификации к бинарной. Первый подход «каждый-против-остальных» (англ., one-vs-rest) заключается в обучении N бинарных классификаторов по одному на каждый класс, которые дают положительный ответ, если объект принадлежит соответствующему им классу, и отрицательным, если он принадлежит какому-либо из остальных. Таким образом, каждый бинарный классификатор определяет принадлежность к одному из классов. Другим подходом в сведении задачи мультиклассовой классификации к бинарной является «каждый против каждого» (англ., one-vs-one). В данном подходе строится по одному

бинарному классификатору на каждую пару распознаваемых классов. В результате получается $\frac{N(N-1)}{2}$ бинарных классификаторов для N классов. При классификации объект проходит через все классификаторы, и искомым считается тот класс, к которому объект был отнесен больше всего раз.

Следующие классификаторы могут решать задачу мультиклассовой классификации без дополнительных преобразований.

В наивный байесовский подход заложено предположение о статистической и позиционной независимости (англ. positional independence assumption) признаков объектов. В нем вероятность принадлежности к классу C классифицируемого объекта O, представленного набором из k признаков p_k, задается следующим образом.

$$P(C|O) \propto P(c) \prod_{1 \leqslant k \leqslant n_o}^{\square} P(p_k|C)$$

Здесь $P(p_k|C)$ – условная вероятность того, что признак p_k проявится в объекте из класса C, а P(C) – априорная вероятность принадлежности объекта классу C. $P(p_k|C)$ интерпретируется в качестве оценки вклада признака p_k в то, что объект принадлежит классу C. Самым подходящим для объекта классификации классом в наивном байесовском методе считается наиболее вероятный класс C_{map}, имеющий максимальную апостериорную вероятность.

$$C_{map} = arg \max_{c \in C} P(c|O) = arg \max_{c \in C} P(c) \prod_{1 \leqslant k \leqslant n_o}^{\square} P(p_k|C)$$

К преимуществам наивного байесовского метода можно отнести достаточную устойчивость к шумовым признакам и высокую производительность. Метод хорошо справляется с большим количеством одинаково важных признаков, оказывающих совместное влияние на окончательный выбор класса.

Деревья принятия решений представляют собой структуру данных дерево, где в вершинах лежат признаки, а листья соответствуют классам. Очередной объект классифицируется деревом принятия решений при

прохождении от корня к одному из листов, в каждой вершине выбирая одного из ее потомков – в соответствии со значением очередного признака у данного объекта. Существует большое число различных подходов к построению деревьев принятия решений. В общем случае, построение дерева на обучающей выборке происходит рекурсивно. Для очередной вершины выбирается один из признаков, после чего для него задается правило, в соответствии с которыми делится множество объектов обучающей выборки в текущей вершине. Ключевым является момент выбора очередного признака для разбиения – для этого обычно используются значения энтропии и прироста информации.

Если в какой-то момент все объекты в вершине оказываются одного класса, вершина становится листом, которому присваивается соответствующий класс. По сравнению с описанными ранее методами, деревья принятия решений обладают хорошей интерпретируемостью и наглядностью моделей построенных классификаторов.

В методе k ближайших соседей для нового объекта при классификации находятся k наиболее близких к нему объекта, вычисляется самый частый среди них класс и присваивается искомому объекту. Главным достоинством метода является простота его реализации и возможность онлайн-обучения, недостатком можно отметить необходимость хранить всю обучающую выборку.

В задаче политематической классификации существует два главных метода: метод трансформации проблемы и метод адаптации алгоритма. Метод трансформации переводит задачу политематической классификации в множество задач бинарной или мультиклассовой классификации, которые могут решать соответствующие классификаторы, называющиеся базисными. Метод адаптации преобразует классификаторы так, чтобы они могли решать задачу политематической классификации напрямую.

Самым простым алгоритмом метода трансформации является метод

бинарной релевантности (англ. binary relevance), который строит независимо по одному бинарному классификатору для каждого класса. Итоговое множество категорий для объекта состоит из всех классов, для которых соответствующие классификаторы дали положительный результат. Метод Label Powerset преобразует проблему в задачу мультиклассовой классификации, где категориями являются подмножества изначальных классов. Цепной классификатор (англ. Classifier chains) строит N бинарных классификаторов для каждого класса, которые объединены в цепь посредством пространства признаков. Иными словами, для каждого следующего классификатора ответы предыдущих являются признаками. Таким образом, в отличие от метода бинарной релевантности, Label Powerset и цепной классификатор учитывают зависимости между категориями.

Метод адаптации ML-knn основывается на традиционном методе k ближайших соседей. Для нового объекта находятся k наиболее близких объектов из обучающей выборки. Затем, на основе статистики принадлежности соседей к множествам классов вычисляется оценка апостериорного максимума и определяется искомое множество категорий.

1.2.2. Уменьшение размерности пространства признаков

Уменьшение размерности пространства признаков применяется по нескольким основным причинам:

- Избавление от шумовых признаков;
- Уменьшение требований к вычислительным мощностям;
- Улучшение интерпретируемости модели.

В задачах текстовой обработки достаточно часто используется статистическая мера TF-IDF, в частности для отражения релевантности документа поисковому запросу. Мера рассчитывается на основе двух составляющих, TF и IDF:

$$TF-IDF=TF(t,d)\times IDF(t)=\frac{n_t}{\sum n_k}\times\log\left(\frac{|D|}{|\{d_i\in D|t\in d_i\}|}\right) \quad ,$$

где n_t – частота слова в документе d, в знаменателе – число слов во всем документе. |D| – общее число документов, в знаменателе – число документов, в которых встречается термин t.

1.2.3. Оценка эффективности классификаторов

Качество текстовых классификаторов может быть измерено различными методами. Базовые меры оценки эффективности рассчитываются на основе частот true positive (TP), false positive (FP), true negative (TN) и false negative (FN). TP – количество текстов, верно отнесенных к данному классу; FP – количество текстов ошибочно не отнесенных к данному классу; FN – количество текстов, ошибочно не отнесенных к данному классу. TN – количество текстов, верно не отнесенных к данному классу. Эти значения наглядно представлены в Табл. 1.

		Истинная оценка	
		положительная	отрицательная
Оценка классификатора	положительная	TP	FP
	отрицательная	FN	TN

Табл. 1. Значения TP, TN, FP, FN.

На основе этих четырех значений рассчитываются две широко использующиеся меры оценки эффективности классификации: точность (англ., precision) и полнота (англ., recal). Точность и полнота вычисляются по формулам

$$precision=\frac{TP}{TP+FP} \qquad recall=\frac{TP}{TP+FN}$$

Таким образом, точность отражает долю истинно положительных ответов среди всех положительных ответов классификатора. Полнота же оценивает

долю положительных ответов классификатора относительно общего числа документов, истинно относящихся к данному классу.

При этом необходимо учитывать баланс между точностью и полнотой, чтобы не допускать ложных срабатываний и пропусков истинных ответов. Для решения этой задачи существуют F_β -мера. Она задается формулой

$$F_\beta = (\beta^2 + 1)\frac{precision \times recall}{(\beta^2\, precision) + recall},$$

Если предпочтение отдается точности, то $0 < \beta < 1$, и $\beta > 1$, если предпочтение отдается полноте. Если $\beta = 1$, то мера называется сбалансированной и обозначается F_1.

Выделяют два основных подхода в вычислении F_β-меры для мультиклассовых классификаторов. В подходе MicroF_β все отдельные значения TP, FP и FN суммируются, после чего производится расчет. При MacroF_β усреднение производится уже после расчета значений F_β-меры для каждого класса. Таким образом, оценка эффективности MicroF_β равнозначно интерпретирует все документы, то есть отражает эффективность для наиболее популярных классов. Оценка MacroF_β наоборот равновесно учитывает каждую категорию вне зависимости от того, каким числом документов она представлена в выборке.

1.3. Семантическая близость слов

Определение семантической близости между словами естественного языка является задачей дистрибутивной лингвистики. Одним из основных подходов к ее решению является построение векторных представлений слов на базе огромного корпуса текстов. В таком случае семантическая близость двух слов считается как косинусная мера между соответствующими векторами. Эффективным инструментом, строящим такое представление, является Word2vec. Word2vec представляет собой нейронные сети со

скрытым уровнем. Существует два вида архитектуры этих нейронных сетей: continuous bag-of-words (CBOW) и skip-gram. В архитектуре continuous bag-of-words сеть предсказывает слово по некоторому числу окружающих его контекстных слов. Порядок этих слов не влияет на предсказание. В архитектуре continuous skip-gram сеть наоборот предсказывает контекстные слова по заданному слову.

У word2vec много параметров, от которых зависят результаты ее обучения.

1. Размер окна. Он определяет, как много слов до и после заданного слова будут участвовать в обучении.
2. Размерность векторного пространства. Качество векторного представления увеличивается с большей размерностью. Однако после некоторой точки прирост качества уменьшается. Обычно размерность пространство указывается между 100 и 1000.
3. Порог допустимой частоты слов. Слова с высокой частотой обычно несут мало информации, поэтому слова с частотой выше заданного порога не участвуют в обучении. За счет этого также улучшается производительность модели.

Глава 2. Практическая реализация системы

2.1. Обучение классификаторов

В процессе эксперимента сравнивались следующие классификаторы: Label Powerset, цепной классификатор (Classifier Chains) и ML-knn. Поскольку методы Label Powerset и цепной классификатор не решают задачу политематической классификации напрямую, а переводят ее к виду, в котором ее могут решать мультиклассовые и бинарные классификаторы, называемые базисными, то проводилась серия экспериментов на установление лучшего базисного классификатора для каждого метода. Базисными классификаторами выступали метод опорных векторов, дерево принятия решений, метод k ближайших соседей и наивный байесовский классификатор. Методы политематической классификации с лучшими базисными классификаторами затем сравнивались между собой и с методом ML-knn.

Для проведения эксперимента был составлен набор из 3922 тегов и 407008 постов, написанные 15773 пользователями. Для того, чтобы отсеять авторские и малоупотребительные теги, в датасет вошли теги, которые использовались более чем 50 уникальными пользователями.

Тексты в датасете включают 118990 уникальных слов. Поскольку такой массив требует много памяти, и не все слова одинаково полезны для классификации, то в качестве методов оценки признаков использовалась мера TF-IDF. Для каждого заданного порога значения TF-IDF собирался датасет, на котором обучались классификаторы. Значения TF-IDF варьировались от 0.003 до 0.0046. Верхняя граница обусловлена тем, что при ее увеличении количество признаков в датасете уменьшалось настолько, что некоторые классы оставались без признаков вовсе и исключались из датасета, а нижняя граница обусловлена вычислительными мощностями,

доступными при исследовании.

2.2. Извлечение временных интересов

Интерес людей к той или иной тематике изменяется во времени. Зная о том, что интересно людям в определенный момент времени, можно сильно улучшить качество рекомендаций. Даже ничего не зная об интересах человека, можно уже в качестве рекомендаций предложить ему самое актуальное на данный момент. Однако показывать просто самые популярные интересы нецелесообразно, потому что они, как правило, довольно общие. Например, самыми распространенными интересами в базе данных являются «фото», «россия» и «видео». Поэтому при анализе интересов важно смотреть не только на частоту их употребления в данный момент, но и изменение этой частоты во времени.

Для каждого поста из набора данных известна дата, когда он был написан с точностью до дня. Самая ранняя дата, сохраненная в базе данных – 2001-06-04 00:12:58, самая поздняя – 2016-03-29 11:56:21. Таким образом, данные охватывают промежуток времени в 15 лет. Поскольку обрабатывать такой массив по часам, дням и даже неделям не удобно, для каждого интереса строился временной ряд по месяцам. В качестве меры частоты интереса на промежутке было выбрано количество уникальных постов, которые были написаны с данным тегом за месяц. Учитывать именно количество постов, а не пользователей, было осознанным шагом, поскольку именно посты характеризуют деятельность пользователей.

Набор данных включает информацию о 522243 тегах, которые считаются интересами пользователей. Однако далеко не все из них полезны, так как отличаются низкой частотой. Для того, чтобы обрезать подобные теги, был установлен порог фильтрации на количество пользователей, использующих этот тег, таким образом определяя минимальный размер

сообщества людей, которым этот тег интересен. Порог задавался вручную и был установлен на значении 10, что позволило убрать большую часть авторских и малоупотребительных тегов. После фильтрации осталось 7503 тега. При этом многие временные ряды оказались разреженными, то есть для большого числа дат посты с заданными тегами отсутствовали. Они никак не фильтровались, потому что разреженность свойственна большинству временных интересов. Частота интереса в случае отсутствия данных полагалась 0. Производилась попытка применить медианное сглаживание и сглаживание простым скользящим средним, однако из-за того, что в качестве временного промежутка выступал месяц, а для некоторых тегов время интереса к ним исчисляется месяцем, например «14 февраля», «хэллоуин» и другие, то сглаживания приводят к тому, что их пики становятся нечеткими и слезают на соседние месяцы, что мешает анализу.

Величиной интереса за некоторый промежуток времени будем называть частоту постов с этим интересом за данный промежуток. Вспышкой интереса будем называть непрерывный промежуток времени, на протяжении которого величина интереса остается выше среднего.

Введем классификацию интересов. Будем называть интересы с ярко выраженными пиками направленными. Они характеризуются малым количеством вспышек интереса. При этом очевидно, что интерес, являющийся направленным на одном интервале времени, может быть постоянным на более узком интервале. Например, интерес «2013» на протяжении года выглядит как постоянный, в то время как является направленным на всем интервале. Интересы, в которых прослеживается периодичность, будем называть циклическими. Они характеризуются наличием большего числа вспышек и периодом между ними, однако в общем случае не имеют постоянной амплитуды. Остальные интересы, в которых нельзя выделить структуру, назовем постоянными интересами.

Итак, в качестве признаков временного ряда интереса использовались

следующие его характеристики: количество вспышек, разница между максимальной и минимальной шириной вспышек, разница между минимальным и максимальным расстоянием межу вспышками. Средняя ширина вспышки в расчет не бралась, так как при большом их количестве пропорциональна этому количеству, а при малом количестве вспышек не играет роли в выделении интереса как временного. Алгоритм определения временных интересов основывался на задаваемых правилах, так как задача заключалась в выделении временных рядов с заранее известной четкой структурой. Направленными интересами признавались те, у которых число вспышек было меньше 4, циклическими те, у которых число вспышек находилось в диапазоне от 4 до 15 включительно и разница ширины вспышек не превосходит 3. Выбор 15 как верхней границы числа вспышек обусловлен тем, что ряды охватывают интервал времени в 15 лет, и 15 вспышек за это время соответствует циклу в год. Остальные интересы считались постоянными. В итоге было выделено 400 направленных интересов, 461 циклический и остальные были помечены как постоянные интересы. Теги, не прошедшие фильтрацию, то есть с количеством использующих их людей ниже порога, также считались постоянными.

Оценка качества выделения временных интересов проводилась на основе оценки всей системы в целом.

2.3. Определение семантической близости интересов с помощью модели word2vec

Для того, чтобы можно было расширять список интересов пользователя похожими и расширять рекомендации, нужно научиться определять семантически близкие теги. Для этого строилось векторное представление тегов с помощью модели wprd2vec, и семантическая близость между тегами

представляла собой косинусное расстояние между соответствующими векторами.

Датасет для обучения модели был составлен на основе тегов, проставленных к одному посту. То есть для каждого текста из набора данных формировалось предложение из всех его тегов, которое записывалось в датасет. Таким образом, контекстным окружением для тега в данном случае выступали те теги, которые были проставлены к посту вместе с ним. Порядок слов в предложении моделью не учитывался. Всего собранный датасет насчитывал 447127 предложений с 2034756 словами, в которых было 522243 уникальных слова.

При обучении модели были выставлены следующие параметры:

1. метод обучения – cbow, поскольку не учитывает порядок контекстных слов и обучается быстрее skip-gram ;
2. размерность векторного пространства – 200;
3. размер окна – 37, поскольку это максимальное количество тегов к одному посту;

Для увеличения производительности использовалось 10 тредов, в итоге модели потребовалось 16 минут на обучение.

Оценка качества обученной модели проводилась на основе оценки всей системы в целом.

2.4. Построение рекомендательной системы

Рекомендательная система строилась методом фильтрации на основе содержания, поэтому требовала составления профилей интересов

пользователей и постов. Пусть N – множество всех известных системе интересов. Тогда профили представляют собой вектора длины |N|, где каждому элементу вектора соответствует единственный тег-интерес из этого множества. Очевидно, профили пользователя и поста имеют одинаковую размерность.

Профиль интересов пользователей строился на основе написанных им постов и тегов к ним. Если пользователь не указывал к ним теги, то они определялись автоматически с помощью построенного ранее классификатора. Затем для каждого тега считалась частота использований этого тега в постах пользователя и определялся конечный вектор, который будем называть профилем интересов пользователя.

Профиль поста определялся сходным образом. Если у него были указаны теги, то его профиль представлял собой бинарный вектор, в котором 1 означала наличие соответствующего интереса, 0 означал его отсутствие. Будем называть такой профиль поста базовым.

Затем базовый профиль расширялся с помощью семантически близких интересов и временных интересов. Сначала для каждого интереса находились 10 наиболее близких к нему тега и мера близости между ними прибавлялась к соответствующему близкому тегу элементу вектора. После этого значение каждого тега i в профиле поста умножалось на величину A_i, которая вычислялась по следующей формуле:

$$A_i = 1 + \ln(1 + C_i * (k + t_i)) \quad ,$$

где C_i – частота тега i в рассматриваемый момент времени, t_i – индикаторная функция, равная 1, если интерес является направленным или циклическим и в рассматриваемый момент времени наблюдается вспышка интереса к данному тегу, k – коэффициент предпочтения временных интересов.

Как видно из формулы A_i, она будет равняться 1, если частота тега i в

рассматриваемый момент времени равна 0. Тогда никакого увеличения веса тега не происходит. Если же частота тега C_i больше нуля, но интерес не является временным, то его вес увеличится пропорционально частоте. Если же он является временным, то его вес увеличится пропорционально частоте и коэффициенту предпочтения временных интересов k. Таким образом, при наличии интересов с одинаковой частотой больший вес будет у временного интереса. Функция логарифма используется для сглаживания величин частот тегов.

Алгоритм преобразования базового профиля в расширенный можно наглядно продемонстрировать следующим кодом на языке python:

```
new_post_profile = copy(post_profile)
for tag in post_profile:
    for similar_tag, similarity_measure in get_most_similar_tags(tag):
        new_post_profile[similar_tag] +=  similarity_measure
post_profile =  new_post_profile
for tag in post_profile:
    A =  1 + log(1 + tag_count[tag][date]*(k + is_temporal_interest[tag][date]))
    post_profile[tag] =  post_profile[tag] * A
```

После построения профилей пользователя u и профиля поста p мера заинтересованности I пользователя в посте вычисляется как скалярное произведение их профилей по формуле

$$I = p * u = \sum_{i=1}^{|N|} p_i u_i$$

При составлении списка рекомендаций для пользователя посты сортировались по убыванию меры.

2.5. Программная реализация системы

Программа была написана на языке программирования python3 с использованием библиотек gensim для работы с моделью word2vec, numpy для векторных вычислений, scikit-learn для построения классификаторов, scipy.stats для проведения статистических тестов и matplotlib для построения графиков. Листинг некоторых классов программы приведен в приложении.

Глава 3. Оценка результатов

3.1. Оценка классификации

Были построены обучающие выборки с варьированием значения меры TF-IDF для оценки признаков. На Рис.1-Рис.2 представлены графики МикроF$_1$ (Рис.1) и МакроF$_1$ (Рис.2) меры эффективности классификации методом Label Powerset с использованием различных базисных классификаторов: метода опорных векторов, наивного байесовского классификатора, деревьев принятия решений и k ближайших соседей.

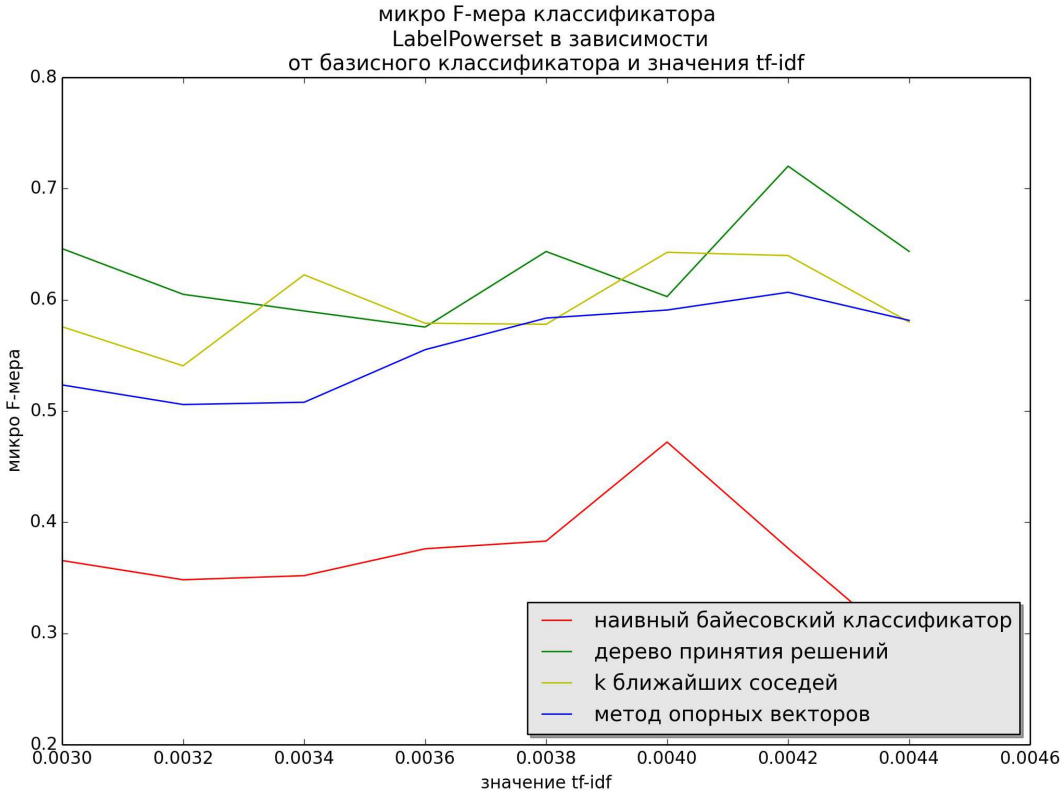

Рис. 1. микро F-мера классификатора LabelPowerset в зависимости от базисного классификатора и значения tf-idf

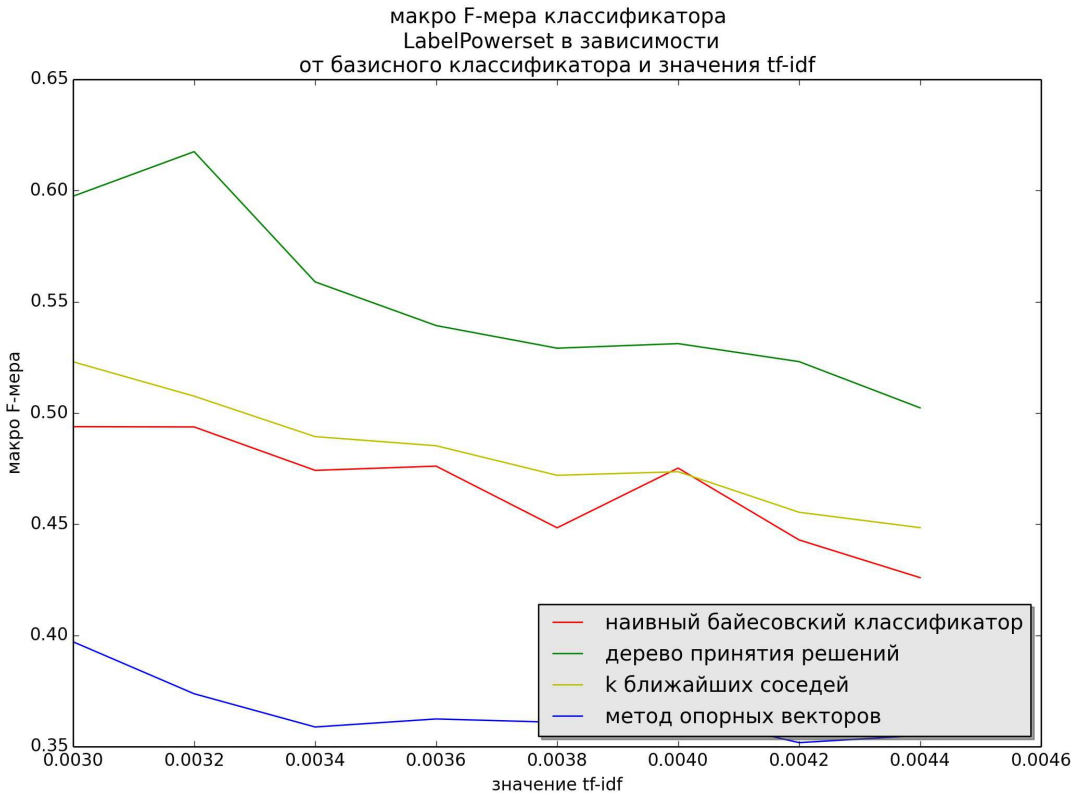

Рис. 2. макро F-мера классификатора LabelPowerset в зависимости от базисного классификатора и значения tf-idf

В среднем значения МикроF$_1$-меры больше, чем МакроF$_1$-меры. Это говорит, о том, что классификаторы лучше различают популярные классы, чем редкие, потому что МакроF$_1$-мера более чувствительна к менее представленным классам. Метод Label Powerset с деревом принятия решений в качестве базисного классификатора демонстрирует лучшие результаты по обеим мерам по сравнению с остальными.

На Рис.3-Рис.4 представлены графики МикроF$_1$ (Рис.3) и МакроF$_1$ (Рис.4) меры эффективности классификации методом Classifier Chains с использованием различных базисных классификаторов: метода опорных векторов, наивного байесовского классификатора, деревьев принятия решений и k ближайших соседей .

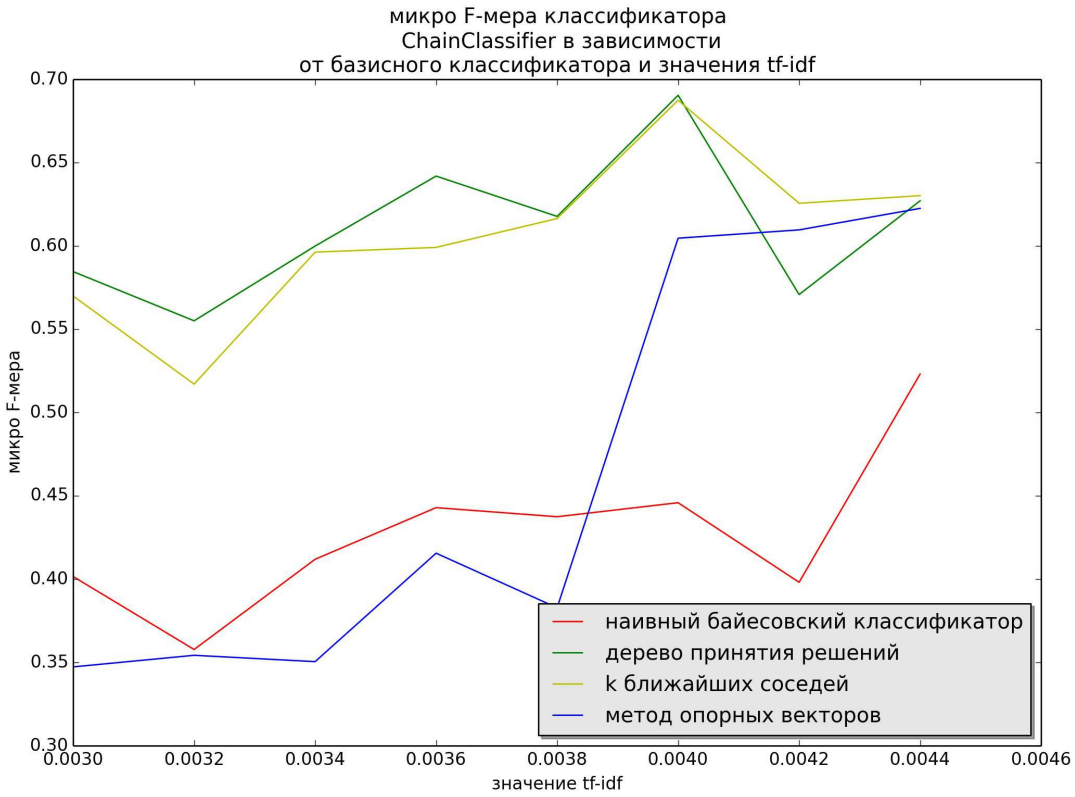

Рис. 3. микро F-мера цепного классифкатора в зависимости от базисного классификатора и значений tf-idf

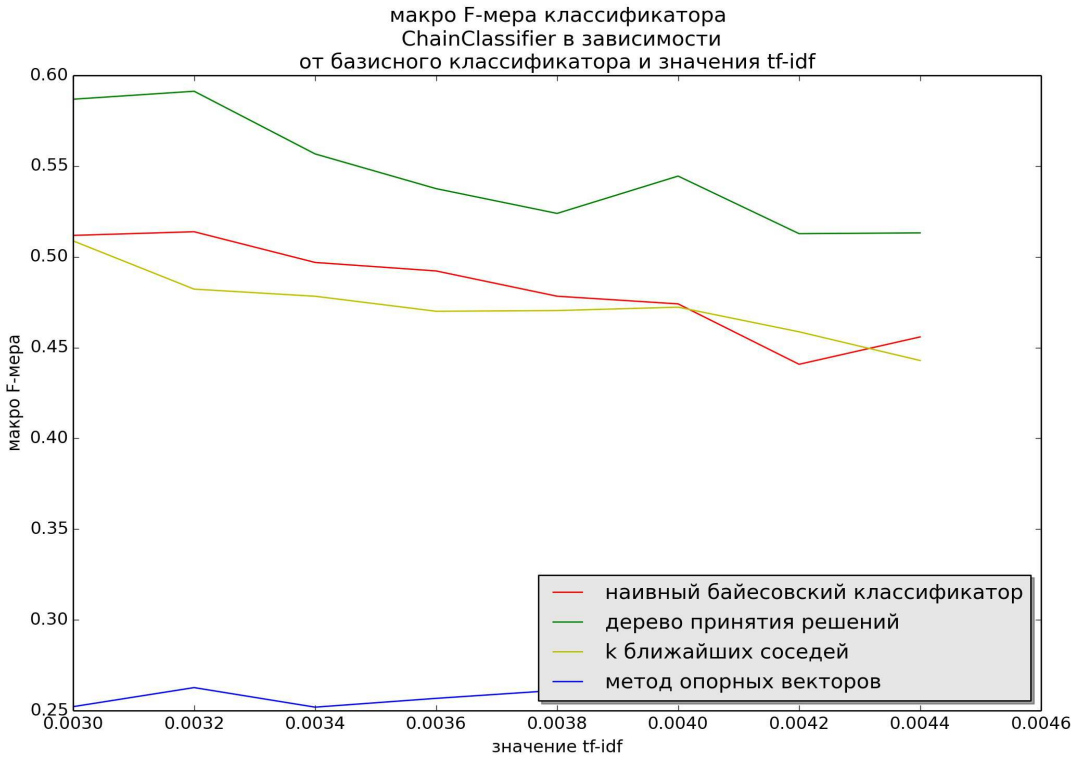

Рис. 4. макро F-мера цепного кассификатора в зависимости от базисного классификатора и значений tf-id.

Стоит отметить, что при увеличении порога TF-IDF оценки МакроF$_1$-меры убывают. Это может быть связано с тем, что при увеличении порога количество признаков уменьшается, а редкие классы ввиду изначально малой представленности в датасете и соответственно малого количества признаков острее реагируют на их отсечение. При этом значение МикроF$_1$-меры увеличивается, так как популярные классы при увеличении порога избавляются от шума.

Лучшие результаты в качестве базисного классификатора для метода цепного классификатора показывает дерево принятия решений.

В качестве лучшего базисного классификатора для методов Label Powerset и цепного классификатора было выбрано дерево принятия решений. На Рис.5-Рис.6 представлены графики МикроF$_1$ (Рис.5) и МакроF$_1$ (Рис.6) меры эффективности классификации методом ML-knn и методов Label Powerset и цепного классификатора с лучшим базисным классификатором в зависимости от порога значений меры TF-IDF.

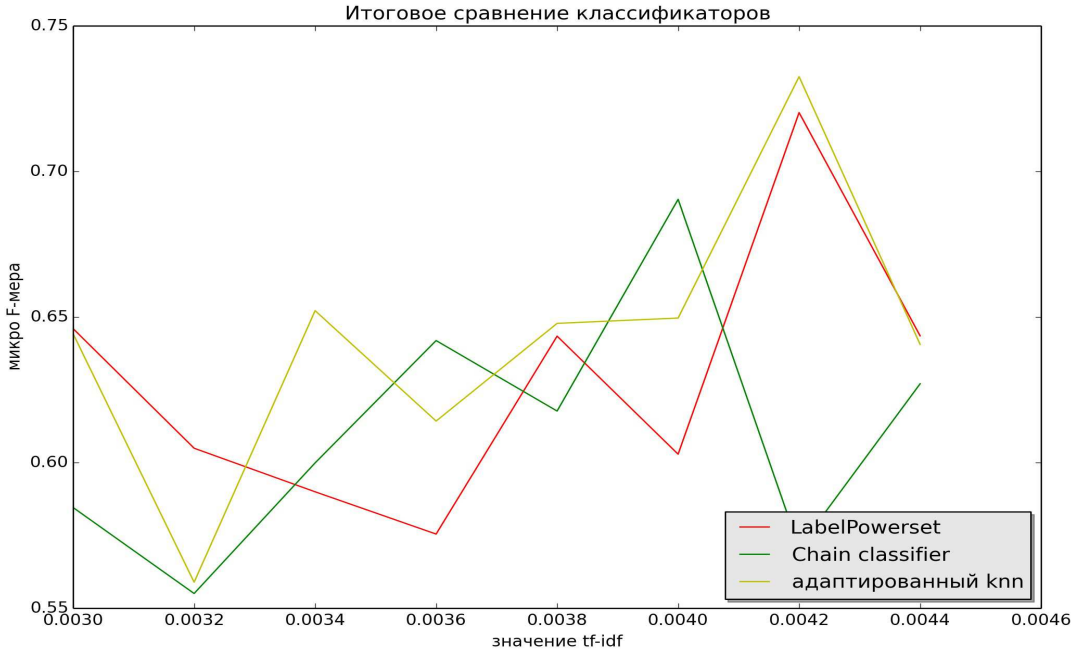

Рис. 5. микро F-мера политематических классификаторов.

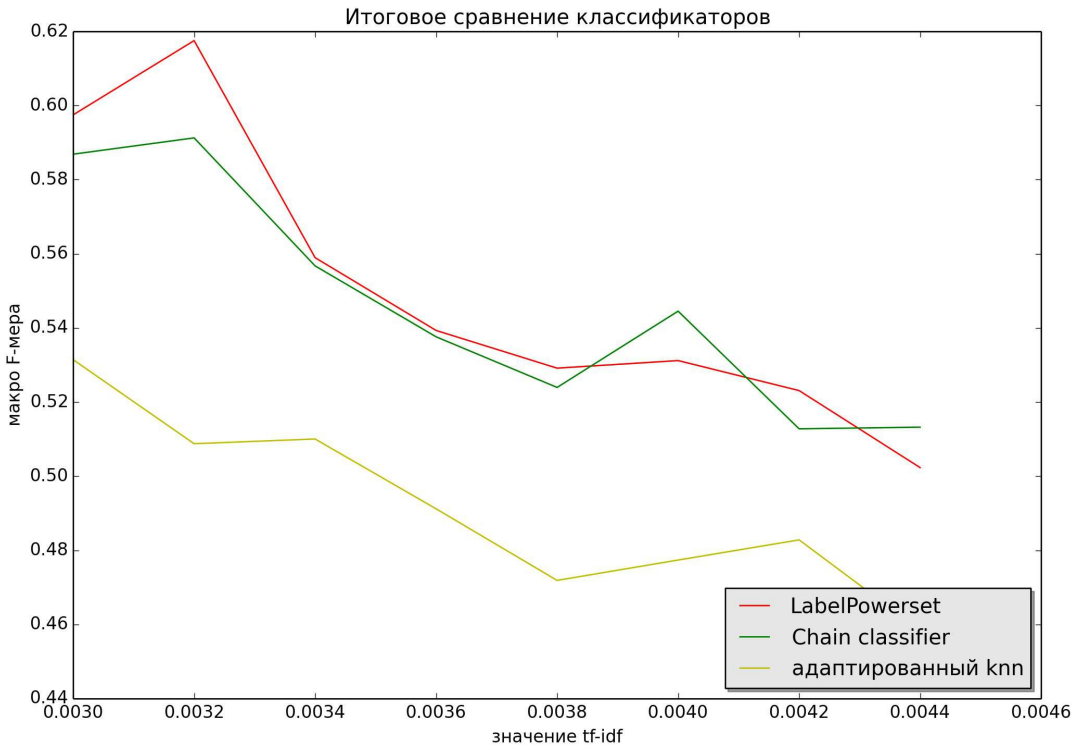

Рис. 6. макро F-мера политематических классификаторов.

По МакроF$_1$-мере адаптированный knn уступает методам Label Powerset и цепному классификатору. Это происходит потому, что он хранит всю обучающую выборку, никак не обрабатывая ее, а редкие классы в ней хуже представлены. В то же время Label Powerset и цепной классификатор показывают сравнимые результаты по МакроF$_1$-мере. По МикроF$_1$-мере все классификаторы показывают примерно одинаковую эффективность. Однако у Label Powerset по сравнению с остальными наивысшая скоростью обучения и вычисления ответа, обусловленная тем, что для работы метода необходимо обучать всего один мультиклассовый классификатор, поэтому в качестве рабочего классификатора для рекомендательной системы был выбран Label Powerset с базисным классификатором – деревом принятия решений.

3.2. Примеры выделения временных интересов

Среди 7503 тегов 400 оказались помечены как направленные, 461 оказались циклическими, а оставшиеся 6642 не имеют четкой структуры.

Примеры направленных интересов представлены на Рис. 7-Рис. 10.

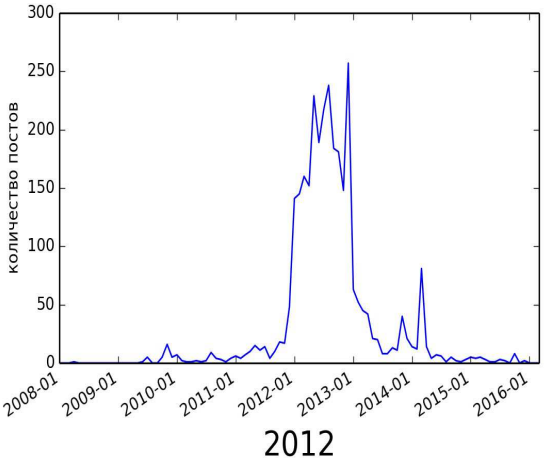

Рис. 7. Частота тега “2012” по месяцам

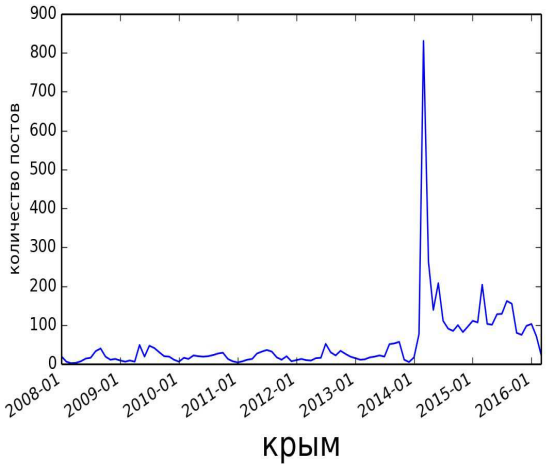

Рис. 8. Частота тега “крым” по месяцам

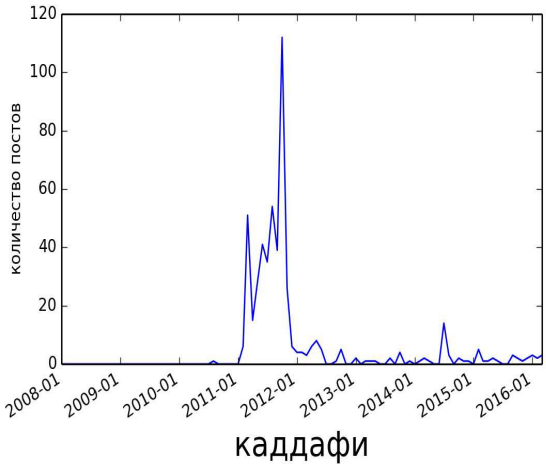

Рис. 9. Частота тега “каддафи” по месяцам

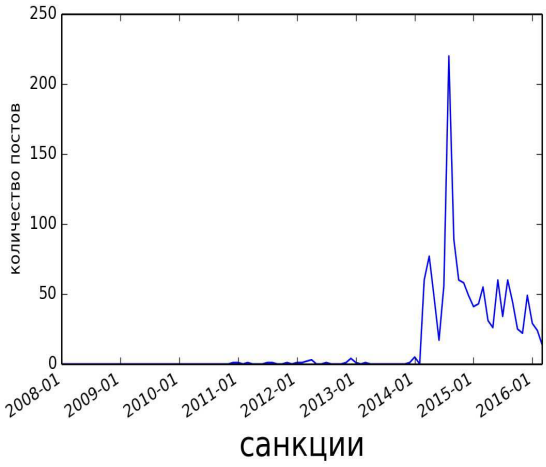

Рис. 10. Частота тега “санкции” по месяцам

Примеры циклических интересов представлены на Рис. 11-Рис. 16.

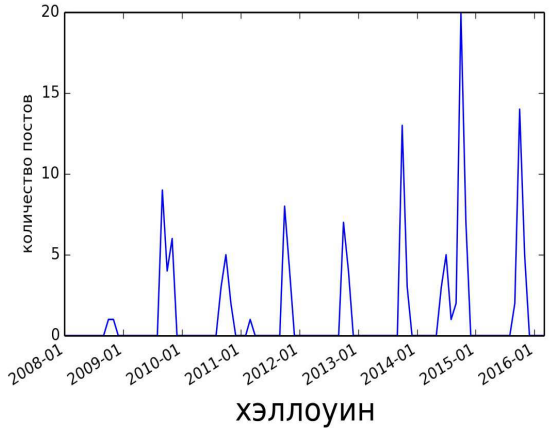

Рис.11.Частота тега “хэллоуин” по месяцам

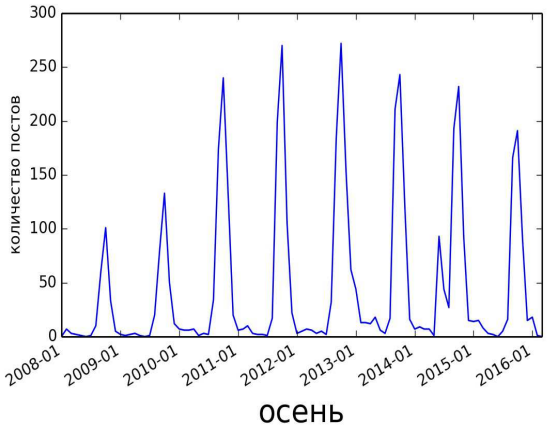

Рис. 12. Частота тега “осень” по месяцам

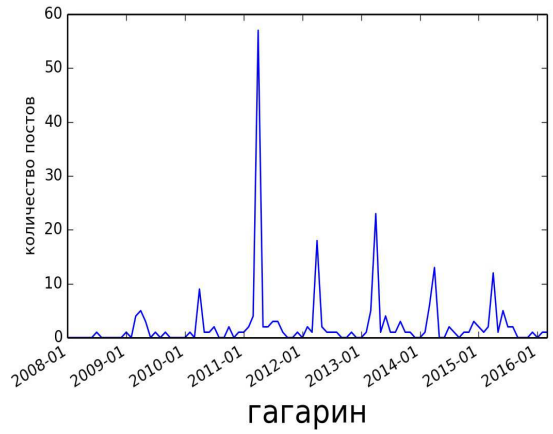

Рис. 13. Частота тега “гагарин” по месяцам

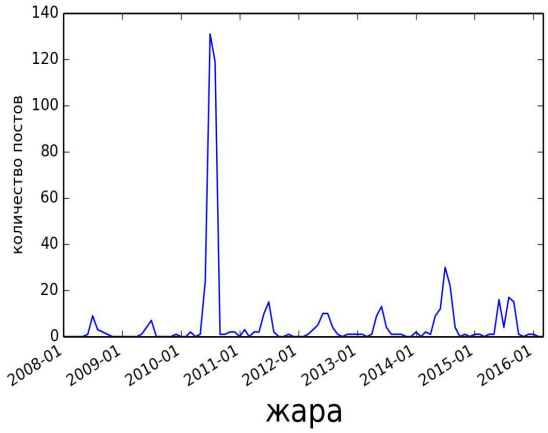

Рис. 14. Частота тега “жара” по месяцам

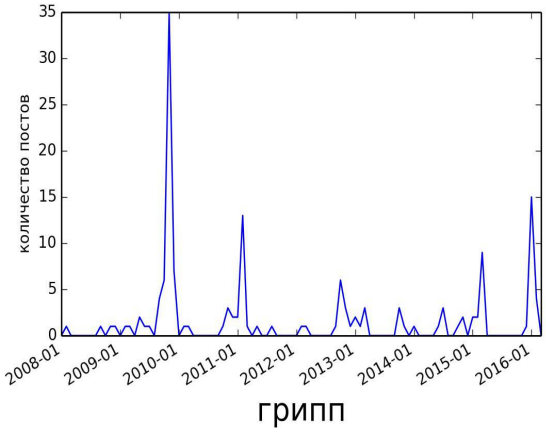

Рис. 15. Частота тега “грипп” по месяцам

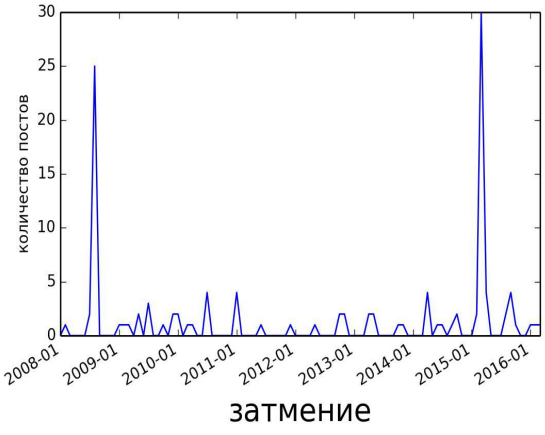

Рис. 16. Частота тега “затмение” по месяцам

Примеры постоянных интересов, то есть интересов, не имеющих четкой структуры, представлены на Рис. 17-Рис. 20.

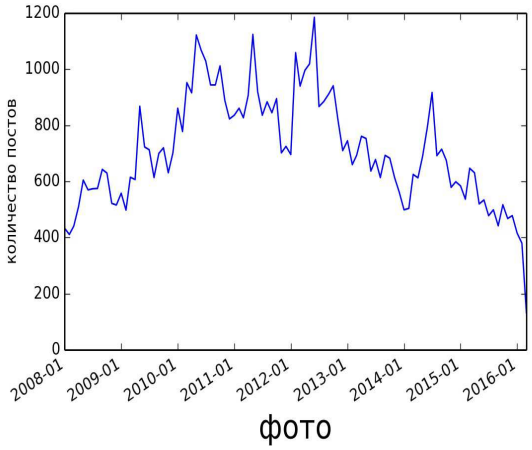

Рис. 17. Частота тега “фото” по месяцам

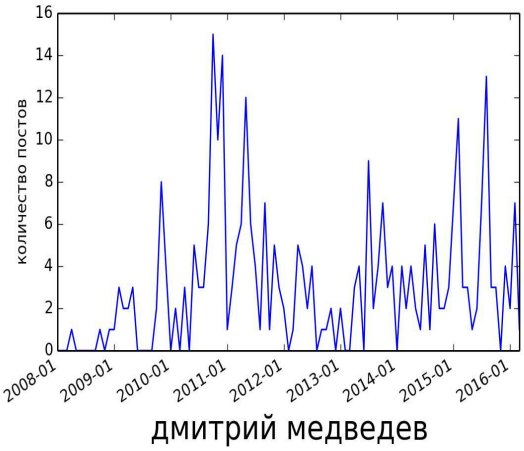

Рис. 18. Частота тега “дмитрий медведев” по месяцам

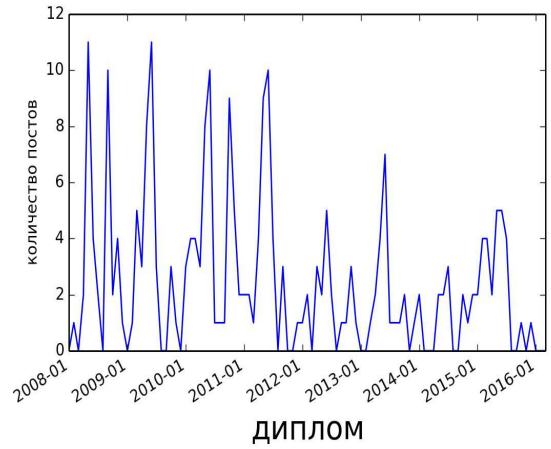

Рис. 19. Частота тега “диплом” по месяцам

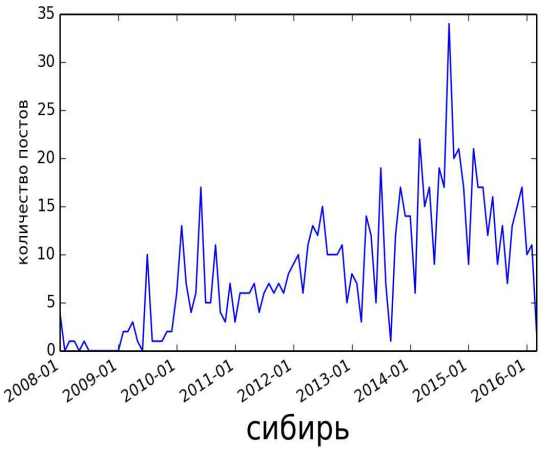

Рис. 20. Частота тега “сибирь” по месяцам

3.3. Примеры результатов определения близости интересов

Примеры близости некоторых случайно выбранных тегов приведены в таблицах 2, 3, 4 и 5.

ягоды	
Тег	Мера близости
варенье	0.6367260813713074
виноград	0.5696146488189697
вишня	0.5660034418106079
десерт	0.5546150207519531
яблоки	0.554611325263977
пирог	0.5437015295028687
заготовки	0.5365356206893921
творог	0.5351770520210266
соус	0.526886522769928
выпечка	0.5156711339950562

Табл. 2. семантически близкие теги к тегу “ягоды”

путешествия	
Тег	Мера близости
путешествие	0.7461340427398682
поездки	0.6544085144996643
отдых	0.6396958827972412
города	0.572009265422821
отпуск	0.5576168298721313
водопад	0.553404688835144
фото	0.5419069528579712
автостоп	0.532516598701477
крит	0.5258797407150269
визы	0.5174714922904968

Табл. 3. семантически близкие теги к тегу “путешествия”

наса	
Тег	Мера близости
nasa	0.659703254699707
марс	0.6349472999572754
спутник	0.6346179246902466
космос	0.578291118144989
мкс	0.5712102055549622
затмение	0.5088368654251099
роскосмос	0.5086774826049805
планета	0.493272066116333
исследования	0.4884713590145111

Табл. 4. семантически близкие теги к тегу “наса”

врач	
Тег	Мера близости
поликлиника	0.6188291907310486
больница	0.5746082067489624
лечение	0.5164672136306763
грипп	0.5030218362808228
врачи	0.4989854693412781
аптека	0.4774535894393921
нервы	0.4535110294818878
медицина	0.4523693919181824
прививки	0.4496365189552307

Табл. 5. семантически близкие теги к тегу “врач”

3.4 Оценка работы всей системы.

Для оценки работы всей системы использовался метод экспертной оценки. Было собрано 25 различных наборов данных по 200 пользователей в каждом. Для каждого пользователя из тестового набора автоматически составлялись два ранжированных списка рекомендаций из 10 постов: с использованием временных интересов и семантической близости интересов и без их использования, причем в случае использования временных интересов каждому набору присваивалась отдельная дата, относительно которой составлялись рекомендации. Каждый список рекомендаций был рассмотрен вручную, и каждому документу в списке исходя из профиля пользователя и соответствующей тестовому набору даты была присвоена мера от 0 (не релевантный) до 3 (очень релевантный). Затем по каждому списку рекомендаций считалась мера nDCG и усреднялась по каждому датасету отдельно для двух способов составления рекомендаций. Таким образом, для каждого датасета из тестового набора определились два значения: средняя мера nDCG по списку рекомендаций, составленных с учетом временных интересов и семантически близких интересов, и без их учета. Средние меры nDCG по каждому датасету из 200 пользователей для каждого метода составления списка рекомендаций представлены в Таблице 6.

Для того, чтобы проверить, улучшилось ли качество рекомендаций, был проведен парный t-тест с нулевой гипотезой о равенстве средних в выборках и альтернативной гипотезой о том, что средние первой выборки, то есть выборки рекомендаций без учета временных интересов и их семантики, меньше, чем второй. Значение статистики t = -4.0145382460134424 p-value = 0.00025394974. Так как p-value меньше 0.05, то нулевая гипотеза отвергаеься в пользу альтернативной. Отсюда можно сделать вывод, что использование временных интересов и семантической близости между ними улучшает

качество рекомендаций.

Номер датасета	Среднее значение nDCG рекомендаций без учета временных интересов и семантической близости	Среднее значение nDCG рекомендаций с учетом временных интересов и семантической близости
1	0.8569262947863503	0.89434903373152663
2	0.79722260523227928	0.97128609909116526
3	0.86033924700337572	0.90903623793892763
4	0.86317859143316833	0.91757423766043278
5	0.95673658532224704	0.87166194428748667
6	0.81765806602555391	0.90937012950297713
7	0.89259994494508699	0.97252635301728318
8	0.7701662648236044	0.89412416787640525
9	0.95379915922076952	0.89497705317656573
10	0.8944189855245902	0.85697483030394561
11	0.90084425371820098	0.93001336182694072
12	0.89886513845491645	0.94565172922560781
13	0.93829362280633755	0.89479150853611999
14	0.85954407645886532	0.94758647781938488
15	0.83171326627124931	0.86174094910649945
16	0.91803848267108146	1.0
17	0.80475869882141193	0.90021257930100074
18	0.89966608676783411	0.94608046125105061
19	0.80158805137053235	0.89290840434453744
20	0.84273411521274066	0.94630763998681355
21	0.90172667188832445	0.94520668711097078
22	0.94410043403728905	0.91533780651697383
23	0.80165714366242913	0.94658545672239702
24	0.90104537728256473	0.94529947532791936
25	0.8663959261318307	0.93145150163728119

Таблица 6. Средние значения nDCG по датасетам для каждого метода составления списка рекомендаций.

Выводы

В процессе подготовки дипломной работы был исследован набор данных с сайта livejournal.com для создания рекомендательной системы, которая находит пользователям наиболее интересные им документы. Набор данных включал 87597 пользователей с 2042011 постами и 522243 уникальными ключевыми словами к ним. Необходимо было обучить классификатор для автоматического извлечения интересов пользователя из постов и провести общий анализ интересов, а именно поведение их во времени и семантические связи между ними, а также оценить целесообразность использования результатов анализа при составлении рекомендаций.

В ходе выполнения работы были изучены такие политематические классификаторы как Label Powerset, Classifier Chains и ML-knn, и такие мультиклассовые и бинарные классификаторы как метод опорных векторов, деревья принятий решений, метод k ближайших соседей и наивный байесовский классификатор. Были изучены методы уменьшения размерности и оценки классификаторов, проведено сравнение качества политематической классификации текста в зависимости от задаваемого порога значений TF-IDF. Результаты показали, что при увеличении задаваемого порога МакроF$_1$-мера у всех классификаторов уменьшается, а МикроF$_1$-мера возрастает.

Было проведено сравнение эффективности работы классификаторов Label Powerset и Classifier Chains в зависимости от базисного классификатора. По результатам исследования дерево принятия решения показало лучшие результаты для обоих классификаторов.

Было проведено сравнение политематических классификаторов Label Powerset, Classifier Chains и ML-knn, которые показали примерно одинаковы оценки МикроF$_1$-меры, но Label Powerset и Classifier Chains превзошли ML-knn по МакроF$_1$-мере.

Все эксперименты проводились на наборе данных из 407008 документов и 3922 классов.

В работе было исследовано поведение интересов пользователей за отрезок времени длиной 15 лет с 2001 года по 2016 год. Была введена классификация интересов на постоянные, временные и направленные, а также заданы правила классификации временного ряда каждого интереса. Исследование проводилось на временных рядах 7503 интересов, и в результате проведенного анализа было выделено 400 направленных интересов, 461 циклический и остальные были помечены как постоянные интересы.

Для определения семантической близости между интересами пользователей были изучены алгоритмы построения векторного пространства слов, в частности модель word2vec. Был построен алгоритм создания датасета, и определены параметры для эффективного обучения модели.

Была построена рекомендательная система и разработан алгоритм расчета рекомендаций для пользователя, учитывающего временность интересов и их семантику. Было проведено тестирование полученной системы на 5000 пользователей и с помощью парного t-теста Стьюдента статистически доказано улучшение качества рекомендаций при использовании информации о временных свойствах интересов и семантической близости между ними.

Заключение

В работе произведен обзор предметной области рекомендательных систем. Рассмотрены главные подходы к их построению: метод коллаборативной фильтрации и метод фильтрации на основе содержания. Предложен новый подход в разработке рекомендательных систем методом фильтрации по содержанию, основанный на автоматическом извлечении интересов пользователей, определении семантически похожих интересов и выявлении временных интересов.

Рассмотрены популярные методы машинного обучения для бинарной и мультиклассовой классификации: метод опорных векторов, байесовские методы, деревья принятия решений и метод ближайших соседей. Изучены главные подходы к задаче политематической классификации текстов: методы на основе сведения проблемы к задаче бинарной или мультиклассовой классификации, такие как Label Powerset и цепной классификатор, и адаптивные методы как адаптивный метод ближайших соседей. Рассмотрены ключевые методы оценки эффективности политематических классификаторов. В результате проведенного исследования на наборе данных лучшим классификатором для задачи классификации текста на естественном русском языке оказался Label Powerset с деревьями принятия решений в качестве базисных классификаторов. Рассмотрены методы построения векторного представления слов на основе нейронных сетей и разработан алгоритм построения набора данных и определены параметры для эффективного обучения модели word2vec. Изучено поведение интересов во времени и предложен алгоритм распознавания циклических, направленных и постоянных интересов. Построена рекомендательная система, учитывающая эти свойства интересов на коллекции документов блог-платформы LiveJournal. Произведена оценка целесообразности использования общей статистики по интересам при составлении рекомендаций и показана статистическая значимость наблюдаемого улучшения работы системы.

Список литературы

1. Краюшкин О., Смирнов М., Чернобай Ю. Сборка, хранение и предобработка коллекции документов для обучения multi-label классификатора текстов на естественном русском языке // Программная инженерия и организация информации, SEIM-2016: труды 1-й научно-практической конференции / Под ред.В. М. Ицыксона, А. С. Ярыгиной. Спб.: Computer Science Center, С. 49-53.

2. Pollock, S. A rule-based message filtering system // ACM Transactions on Office Information Systems, 1988. Т. 6, Вып. 3. С. 232-254.

3. Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. Using collaborative filtering to weave an information tapestry. // Communications of the ACM, 1992. Т. 35, Вып. 12. С. 61-70.

4. Resnick, P., Varian, H.R. Recommender systems // Communications of the ACM, 1997. Т. 40, Вып. 3. С. 56- 58.

5. Brusilovsky, P. Methods and techniques of adaptive hypermedia // User Modeling and UserAdapted Interaction, 1996. Т. 6, Вып. 2. С. 87-129.

6. Pazzani, M.J. A framework for collaborative, content-based and demographic filtering // Artificial Intelligence Review, 1999. Т. 13. С. 393-408.

7. Ahn, H., Kim, K.J., Han, I. Mobile advertisement recommender system using collaborative filtering // Proceedings of the 2006 Conference of the Korea Society of Management Information Systems, 2006. С. 709-715.

8. A¨ımeur, E., Brassard, G., Fernandez, J.M., Onana, F.S.M. Alambic: a privacy-preserving recommender system for electronic commerce // International Journal of Infectious Diseases, 2008. Т. 7. Вып. 5. С. 307–334.

9. Golbeck, J. Generating predictive movie recommendations from trust in social networks // Trust Management, 4th International Conference, iTrust 2006, Pisa, Italy, May 16-19, 2006, Proceedings, 2006 . pp. 93–104.

10. Cortes C., Vapnik V. Support-vector networks // Machine Learning,

1995. T. 20. C. 273-297.

11. Joachims T. Text categorization with support vector machines: Learning with many relevant features // Springer Berlin Heidelberg, 1998. C. 137-142.

12. Russell, S., Norvig, P. Artificial Intelligence: A Modern Approach (2nd ed.). Prentice Hall, 2003. 946 c.

13. Rokach, L., Maimon, O. Data mining with decision trees: theory and applications. World Scientific Pub Co, 2008. 305 c.

14. Tsoumakas, G.; Katakis, I. Multi-label classification: an overview. // International Journal of Data Warehousing & Mining, 2007. Т. 3. Вып. 3. С. 1-13.

15. Read, J.; Bernhard P.; Geoff H.; Eibe F. Classifier Chains for Multi-label Classification. // Machine Learning, 2011. Т. 85. Вып. 3. С. 333-359.

16. Tsoumakas, G., Vlahavas, I. Random k-labelsets: An ensemble method for multilabel classification // Machine Learning: ECML 2007, 2007. С. 406-417

17. Sinclair, J. The automatic analysis of corpora // Directions in Corpus Linguistics (Proceedings of Nobel Symposium 82), 1992. Berlin: Mouton de Gruyter, С. 379-397

18. Митрофанова О.А. Измерение семантических расстояний как проблема прикладной лингвистики // Структурная и прикладная лингвистика. Межвузовский сборник: журнал. Издательство СПбГУ, 2008. Вып.7. С. 92-101.

19. Laurene V. F., Fundamentals of neural networks. Prentice-Hall, Inc. Upper Saddle River, NJ, USA,1994. 476 c.

20. Mikolov T., Chen K., Corrado G., Dean J. Efficient Estimation of Word Representations in Vector Space, Researchgate, 2014. 12 c.

21. Levy, O., Goldberg, Y., Dagan, I. Improving Distributional Similarity with Lessons Learned from Word Embeddings // Transactions of the Association for Computational Linguistics, 2015. Т. 3. С. 211-225.

Приложение

```
from gensim.models import Word2Vec


class Similarity_measurer:

    PATH = 'w2v_data2.bin'

    def __init__(self):
                self.model = Word2Vec.load_word2vec_format(self.PATH,
binary=True)
        self.cache = {}

    def get_similarity_measure(self, t1, t2):
        try:
            return self.model.similarity(str(t1), str(t2))
        except KeyError:
            return 0

    def most_similar(self, t1):
        if t1 not in self.cache:
            try:
                self.cache[t1] = self.model.most_similar(str(t1))
            except KeyError:
                self.cache[t1] = []
        return self.cache[t1]
```

```
'''Класс для извлечения посчитанных временных интересов'''


class Actuality_rater:

    FILENAME = 'actuality_rates_2015-06.txt'
    FILENAME = 'actuality_rates_2014-08.txt'

    def __init__(self, actuality_rates_filename):
        self.actuality_rates = {}
        with open(actuality_rates_filename) as handle:
            for line in handle:
                tid, rate = line.strip().split()
                self.actuality_rates[int(tid)] = float(rate)

    def get_actuality_rate(self, tid):
        return self.actuality_rates.get(tid, 0)



import numpy as np
from post_profile_transformer import Post_profile_transformer
'''Класс для составления рекомендаций пользователю'''


class Recommender_engine:

    def __init__(self, act_rater, interest_similarities, good_tags):
            self.ppt = Post_profile_transformer(act_rater, interest_similarities,
```

```
good_tags)
            self.posts = []

        def add_posts(self, post_profiles):
            i = 0
            for post_profile in post_profiles:
                transformed_post_profile = self.transform_post_profile(post_profile)
                self.posts.append(transformed_post_profile)

        def transform_post_profile(self, post_profile):
            return self.ppt.transform(post_profile)

        def get_interest_rate(self, user_profile, post_profile):
                return sum([post_profile.dct[tid] * user_profile.dct[tid] for tid in
user_profile.dct.keys() & post_profile.dct.keys()])

        def recommend(self, user_profile):
            interest_rates = {}
            for post_profile in self.posts:
                 interest_rates[post_profile.id] = self.get_interest_rate(user_profile,
post_profile)
            max_interest_rate = max(interest_rates.values())
            return sorted(interest_rates.items(), key=lambda x: -x[1])
                return [(prid, rate) for prid, rate in interest_rates.items() if rate ==
max_interest_rate]

    import numpy as np
    '''Класс для составления расширенного профиля документа'''
```

```
class Post_profile_transformer:
    POPULARITY_WEIGHT = 0.1

    def __init__(self, act_rater, interest_similarities, good_tags):
        self.actuality_rates = {tid: 1 + np.log(act_rater.get_actuality_rate(tid) +
1) for tid in good_tags}
        self.sm = interest_similarities

    def spread_similarities(self, post_profile):
        new_p_dct = dict(post_profile.dct)
        for tid in post_profile.dct:
            val = post_profile.dct[tid]
            for a, b in self.sm.most_similar(tid):
                try:
                    if int(a) in new_p_dct:
                        new_p_dct[int(a)] += self.actuality_rates[b] * b
                    else:
                        new_p_dct[int(a)] = self.actuality_rates[b] * b
                except Exception:
                    pass
                post_profile.dct = new_p_dct
        return post_profile

    def actualize(self, post_profile):
        post_profile.dct = {a: post_profile.dct[a] * self.actuality_rates[a] for a
in post_profile.dct}
        return post_profile

    def transform(self, post_profile):
```

```
        return post_profile
    new_post_profile = self.actualize(post_profile)
    return self.spread_similarities(new_post_profile)
```