Правительство Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
**«Московский институт электроники и математики
национального исследовательского университета
«Высшая школа экономики»**

Факультет информационных технологий и вычислительной техники

Вычислительные машины комплексы системы и сети
(Специализация)
Кафедра Информационно-коммуникационных технологий
(кафедра)

ДИПЛОМНАЯ РАБОТА

Разработка сервиса извлечения мнений
(Название темы)

Выполнил
Студент группы № С-94
Левчик Анатолий Васильевич
(Ф.И.О.)

Научный руководитель
ассистент кафедры, Игнатьев Иван Сергеевич
(должность, степень, звание, Ф.И.О.)

Консультант
_____
(должность, степень, звание, Ф.И.О.)

Москва, 2013

«Разработка сервиса извлечения мнений»

ОГЛАВЛЕНИЕ

1. ВВЕДЕНИЕ И ЦЕЛЬ РАБОТЫ

Социальные сети и блоги в настоящее время стали очень популярными средствами общения, в которых миллионы пользователей помещают свои суждения о различных аспектах повседневной жизни. Именно поэтому такие Интернет-сервисы являются ценными источниками данных для извлечения и анализа мнений. Появившись и бурно развившись сравнительно недавно (прежде всего - микроблоги Twitter), эти системы вызывают к жизни все больше инструментов анализа своего содержимого.

Текстовая информация может быть разделена на две основные категории: факты и мнения. Факты являются объективными высказываниями о некоторых сущностях или событиях. Мнения – это субъективные высказывания, отражающие отношение человека или восприятия им какого-нибудь события или сущности. Большая часть существующих исследований в области обработки естественного языка сосредоточены на сборе и извлечении фактологической информации. Примером является традиционный информационный поиск в целом, и веб-поиск – в частности, равно как и другие задачи анализа текста и обработки естественного языка. Исследований же в области обработкой мнений (субъективной текстовой информации) было сравнительно немного, хотя при принятии решений очевидна полезность учета других мнений не только для пользователей – частных лиц, но также и для организаций.

С возникновением и развитием Интернета, и, как следствие, со взрывным ростом контента, создаваемого его пользователями, открылись новые возможности распространения и потребления информации. Появилась возможность публиковать отзывы о продуктах в интернет-магазинах и выражать свою точку зрения в отношении практически любых вещей во

всевозможных интернет-форумах, блогах и социальных сетях. Результатом всех вышеперечисленных видов коммуникации, является объемный контент, созданный пользователями. Теперь потребителям, намеревающимся сделать покупку, можно не спрашивать совета друзей или родственников – в сети можно найти огромное количество обзоров и отзывов, оценивающих практически любой товар.

Когда организации требуется понять отношение потребителей относительно ее продуктов или услуг, то, традиционно, нанимаются консультанты, проводятся опросы или организуется фокус-группы. Этого можно было бы избежать при наличии развитых инструментов анализа мнений сетевого сообщества.

Однако мониторинг источников мнений (в некоторых случаях также применяется термин *анализ тональности текстов*) все еще является трудной задачей. В сети довольно много разнообразных форумов, блогов и т.п., каждый из которых сам по себе содержит огромный объем информации, в то время как количество собственно мнений в них по какому-то конкретном вопросу может быть относительно невелико. Вручную практически невозможно обработать такие массивы данных, отыскать и извлечь оттуда необходимые мнения, распознать их тональность, обобщить и привести результат к удобной форме. Таким образом, необходима система автоматического сбора и анализа мнений, причем для нас важным является то, чтобы она обрабатывала и русскоязычные источники.

Таким образом, целью данной работы стала разработка сервиса извлечения мнений из сообщений социальных сетей.

2. ПОСТАНОВКА ЗАДАЧИ

Рассматривается корпус документов (набор сообщений), содержащих мнения их авторов о каком-то событии, явлении, объекте в заданной предметной области.

Наиболее показательными являются корпуса документов таких рекомендательных Интернет-сервисов как yandex.market, imdb.com, imhonet (мнения о товарах, кинофильмах, книгах, компьютерных играх). В этих системах кроме собственно текста сообщения их авторы также проставляют описываемому объекту оценку, например, от 0 до 5, от 0 до 10 и т. п.

Однако имеются гораздо большие по объему корпуса документов, отражающих эмоционально окрашенное мнение их авторов, но не снабженных выставленной этими авторами оценкой. Сюда относятся многочисленные сообщения участников социальных сетей Facebook, Twitter, ВКонтакте и др..

Ставится задача: разработать сервис извлечения мнений из корпуса документов, представляющих собой эмоционально окрашенные суждения/сообщения, дающий в результате как полярность содержащейся в них эмоциональной оценки (положительная или отрицательная), так и степень такой эмоциональности (от безразличия – до крайних степеней положительной или отрицательной эмоции).

Представляется, что такой сервис потребует предварительного обучения на вручную выставленных оценках, поэтому в качестве обучающих предлагается использовать корпуса документов из перечисленных выше рекомендательных (оценочных) Интернет-систем. Особенностью задачи является тот факт, что такие Интернет-системы позволяют авторам сообщений оставлять оценку по 5-ти или 10-ти балльной шкале, не имеющей явно указанной нейтральной эмоциональной середины, а в результате обработки целевого корпуса документов требуется определить прежде всего положительный или отрицательный характер оценок.

3. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Здесь будут рассмотрены следующие темы: 1) абстрактная модель анализа мнений, 2) классификация текстов по тональности, 3) анализ мнений, основанный на выделении ключевых характеристик, черт и признаков (*features*) объектов, обобщение этих данных, 4) извлечение мнений из предложений, содержащих сравнение.

Для начала необходимо научиться выявлять оценочные слова, как например, *отличный*, *замечательный*, *удивительный*, *плохой*, *нехороший*. Появились исследования, целью которых является выделение таких слов и определение их семантической ориентации (позитивной или негативной направленности). Например, в [5] выведены несколько лингвистических правил, которые могут быть использованы для идентификации оценочных слов и их ориентации при помощи большого текстового корпуса. Данный метод был применен, расширен и усовершенствован в [1, 9, 12]. В [6, 3], был предложен подход бутстрепинга (bootstrapping), использующий небольшое множество исходных оценочных слов, для которых могут быть далее найдены синонимы и антонимы в системе WordNet (http://wordnet.princeton.edu/).

Следует отметить методы классификации на документном уровне в обзорах товаров, фильмов и т. п. Их целью является разбиение сообщений на положительные или отрицательные, отражая отношение их авторов к объекту обзора [11, 2, 7]. Темой исследований [3, 14] стало определение того, являются ли негативными или позитивными отдельные предложения. В [6, 10] предложена модель выделения характеристик, атрибутов объектов и их обобщений, определяющая ключевые информационные элементы текста и позволяющая структурировать мнения из неструктурированной текстовой информации. Проблема извлечения мнений из сравнительных предложений обсуждается в [4, 10].

3.1. АБСТРАКТНАЯ МОДЕЛЬ АНАЛИЗА МНЕНИЙ

В целом, мнение может быть выражено по отношению к чему угодно: к товару, сервису, человеку, организации или событию. Понятие *объект* используется для обозначения сущности, о которой было высказано мнение. Объект имеет множество *компонентов* (или *частей*), атрибутов(или *характеристик*). Каждый компонент, в свою очередь, может также иметь, свои компоненты и атрибуты и т. д. Таким образом, объект может быть разложен на иерархическую структуру в отношениях "*является частью*" (*part-of*) [15].

Объекту O ставится в соответствие пара *(T, A)*, где *T* является *иерархией* или *таксономией компонентов и подкомпонентов O* со *множеством атрибутов A*. В свою очередь, каждый из подкомпонентов имеет свое собственно множество подкомпонентов и атрибутов. В этой иерархии (дереве) корнем является сам объект. Каждый не-корневой узел является компонентом или подкомпонентом. Каждая связь выражает отношение "является частью". Каждый узел связан со множеством атрибутов. Мнение может быть выражено по отношению к любому узлу и атрибуту узла.

Однако, для рядового пользователя такое иерархическое представление, вероятно, будет являться слишком сложным. Для упрощения дерево можно сделать плоским. Термины *характеристики* или *черты* ("*features*") будут использоваться для обозначения и компонентов, и атрибутов. Использование подобных терминов по отношению к объектам (к товарам, в особенности), довольно часто встречается на практике. Также, следует обратить внимание на то, что объект сам по себе есть характеристика (черта), которая в свою очередь, является корнем дерева.

Пусть оценочный документ *d*, представляющий обзор некоторого продукта, является сообщением или записью в блоге, которые содержат оценочное

суждение относительно объекта *O*. В самом общем случае, *d* состоит из последовательности предложений $d = \{s_1, s_2, \ldots, s_m\}$.

Оценочным пассажем для характеристики (черты) *f* объекта *O*, оцениваемого в *d*, является группа последовательных предложений в *d*, которые выражают позитивное или негативное отношение к *f*.

То есть, вполне возможно, что последовательность предложений (по крайней мере, одно) вместе выражают мнение об объекте или компоненте объекта. Также, возможно, что единственное предложение выражает сразу несколько мнений по отношению более чем к одному компоненту объекта, например *"Качество изображения камеры хорошее, но время жизни батареи коротко"*.

Автором (*источником мнения*) может быть человек или организация. В случае обзоров товаров, сообщений на форумах и блогах, источником мнения обычно является автор самого сообщения или блога. Определение источника мнения играет важную роль в новостных статьях, так как в них зачастую явно указывается человек или организация высказывающие определенное мнение. Например, источником мнения в новости *"Иванов выразил свое несогласие"* является Иванов.

Семантическая ориентация мнения для характеристики (черты) *f*, показывает, является ли мнение позитивным, негативным или нейтральным, относительно данной характеристики.

Суммируя вышесказанное, модель анализа мнений, ориентированная на характеристики (*feature-based)*, может быть описана таким образом: объект *O* представлен конечным множеством характеристик (или компонентов), $F = \{f_1, f_2, \ldots, f_n\}$, в которые входят и сам объект. Каждая характеристика $f_i \in F$ может быть выражена в виде конечного множества слов или фраз W_i, являющимися синонимами. То есть существует множество соответствующих

множеств $W = \{W_1, W_2, ..., W_n\}$ для n характеристик. В оценочном документе d, который оценивает объект O, источник мнения j излагает комментарии по отношению к подмножеству характеристики $S_j \subseteq F$. Для каждой характеристики $f_k \in S_j$, в отношении которых автор (источник мнений) j выражает свое мнение, он выбирает слово или фразу из W_k для того, чтобы описать характеристику, и выразить позитивное, негативное или нейтральное мнение по отношению к f_k. Задача анализа мнений состоит в том, чтобы извлечь все эти характеристики, компоненты и отношение пользователя к ним из данного оценочного документа d.

Для данного оценочного документа d результатом анализа является множество *четверок*. Каждая *четверка* обозначается (H, O, f, SO), где H – источник мнения, O – объект, f – характеристика или компонент объекта, и SO – семантическая ориентация мнения, выраженного по отношению к характеристике или компоненту f в предложении документа d. Нейтральные мнения не учитываются в силу того, что, как правило, они являются бесполезными.

В рамках конкретного набора оценочных документов D, содержащих мнения по отношению к объекту, могут быть выделены три основные задачи:

Задача 1: Извлечение характеристик и компонентов объекта, которые были прокомментированы в каждом документе $d \in D$.

Задача 2: Определение того, являются ли эти мнения позитивными, негативными или нейтральными.

Задача 3: Группировка синонимов для характеристик (это необходимо потому, что разные источники мнений могут использовать разные слова, чтобы описать одни и те же особенности).

Эти задачи, а также форма представления результатов *feature-based* модели, детализированы ниже.

3.2. АНАЛИЗ ТОНАЛЬНОСТИ ТЕКСТА

Анализ тональности текста является относительно хорошо изученной задачей в области обработки естественного языка [11, 2, 7]. Она формулируется следующим образом: для документа *d* из множества оценочных документов *D* необходимо определить, положительное или отрицательное мнение по отношению к объекту описания он содержит. Например, на некотором множестве обзоров фильмов система должна классифицировать обзоры на положительные и отрицательные.

Задача классификации текстов по тональности похожа на более традиционную задачу классификации по темам – таким, например, как политика, наука или спорт. При классификации по темам важны слова, возможно, взаимосвязанные, отражающие тему класса. Напротив, в задаче определения тональности текстов, тематические слова не важны. Вместо этого важны оценочные слова, выражающие позитивное или негативное отношение. Такие слова, как, например, *замечательный*, *превосходный*, *ужасный*, *отвратительный*, *плохой*, *наихудший* и т. д. Существует множество подходов для решения данной задачи. Большинство из них применяют для классификации методы машинного обучения [2]. Для определения тональности также существуют и специально разработанные алгоритмы, которые используют оценочные слова и фразы совместно с некоторой функцией расчета веса этих слов и фраз.

Классификация по тональности производится на документном уровне, рассматривая каждый оценочный документ как смысловую единицу информации в предположении, что он фокусируется на единственном объекте *O* и содержит мнения от единственного источника. Так как в приведенной выше модели анализа мнений объект *O* также сам по себе является своим компонентом (корневым узлом иерархии объекта), то

классификация по тональности, по существу, определяет семантическую ориентацию мнения выраженную по отношению к *O* в каждом оценочном документе при условии выполнения описанного выше предположения.

Кроме классификации на уровне документа, применяется и классификация на уровне предложений: на объективные или субъективные, на положительные и отрицательные [3, 14]. Проблемой здесь являются сложносочиненные предложения, зачастую выражающие более одного мнения, например *“Качество картинки камеры потрясающее, как впрочем, и время жизни батареи, но видоискатель слишком маленький ”*.

3.3. АНАЛИЗ ХАРАКТЕРИСТИК И КОМПОНЕНТОВ ОБЪЕКТА

Классификация оценочных текстов на документном уровне или на уровне предложений не отвечает на вопрос, что конкретно нравится или не нравится источнику мнения. Если, к примеру, документ является в целом положительным, это не означает, что источник мнения имеет положительное мнение относительно всех частей или характеристик этого объекта. Подобно этому, и в случае, если документ в целом несет негативное мнение, это не означает, что автору не нравится абсолютно все относительно данного объекта. В оценочном документе (например, в обзоре товара) источник мнения затрагивает, как положительные, так и отрицательные аспекты объекта, хотя общая тональность документа может быть в целом положительной или отрицательной. Для извлечения таких аспектов, необходим анализ на уровне характеристик и компонентов объекта (*future-based*). На основе модели, представленной выше, можно выделить три ключевые задачи.

1) Выявление характеристик и компонентов объекта. Например, в предложении *“Качество изображения у этой камеры просто потрясающее”*, характеристикой объекта будет является *“качество изображения”*. В [10] предложен метод нахождения фраз по заданному

шаблону с учителем. В [6, 12], используется метод без учителя. Основным подходом, является нахождение часто встречающихся существительных и конструкций с существительными в качестве характеристик объекта, которые, как правило, таковыми и являются. Несомненно, также применимо и множество других техник извлечения информации, как например, conditional random fields (CRF), скрытая марковская модель (HMM) и др.

2) Определение ориентации мнения: задача состоит в определении того, является ли мнение относительно характеристики или компонента объекта, позитивным, негативным или нейтральным. В предложении-примере, представленном выше, мнение относительно *“качества изображения”* является положительным. Опять же, для решения этой задачи существует множество подходов. Достаточно хорошо показал себя подход на основе словарей [9, 6]. Словарный подход использует оценочные слова и фразы в предложении, для определения тональности мнения относительно характеристики или компонента [12]. Также для этой задачи применимы различные методы машинного обучения с учителем.

3) Группировка синонимов. В связи с тем, что отношение к одним и тем же характеристикам объекта может быть выражено посредством разных слов и фраз, представляется разумным группировать эти синонимы [8].

Существует множество способов представления результатов анализа. Одним из них является создание ориентированного на характеристики (feature-based) резюме мнений объекта, например, такого вида [10]:

Цифровая_камера_1:
КАМЕРА:
Положительных: 125 <предложений>
Отрицательных: 7 <предложений>
Характеристика: качество картинки
Положительных: 123 <предложения>
Отрицательных: 6 <предложений>
Характеристика: размер
Положительных: 82 <предложения>
Отрицательных: 10 <предложений>

В [10] эта сводка наглядно визуализируется гистограммой:

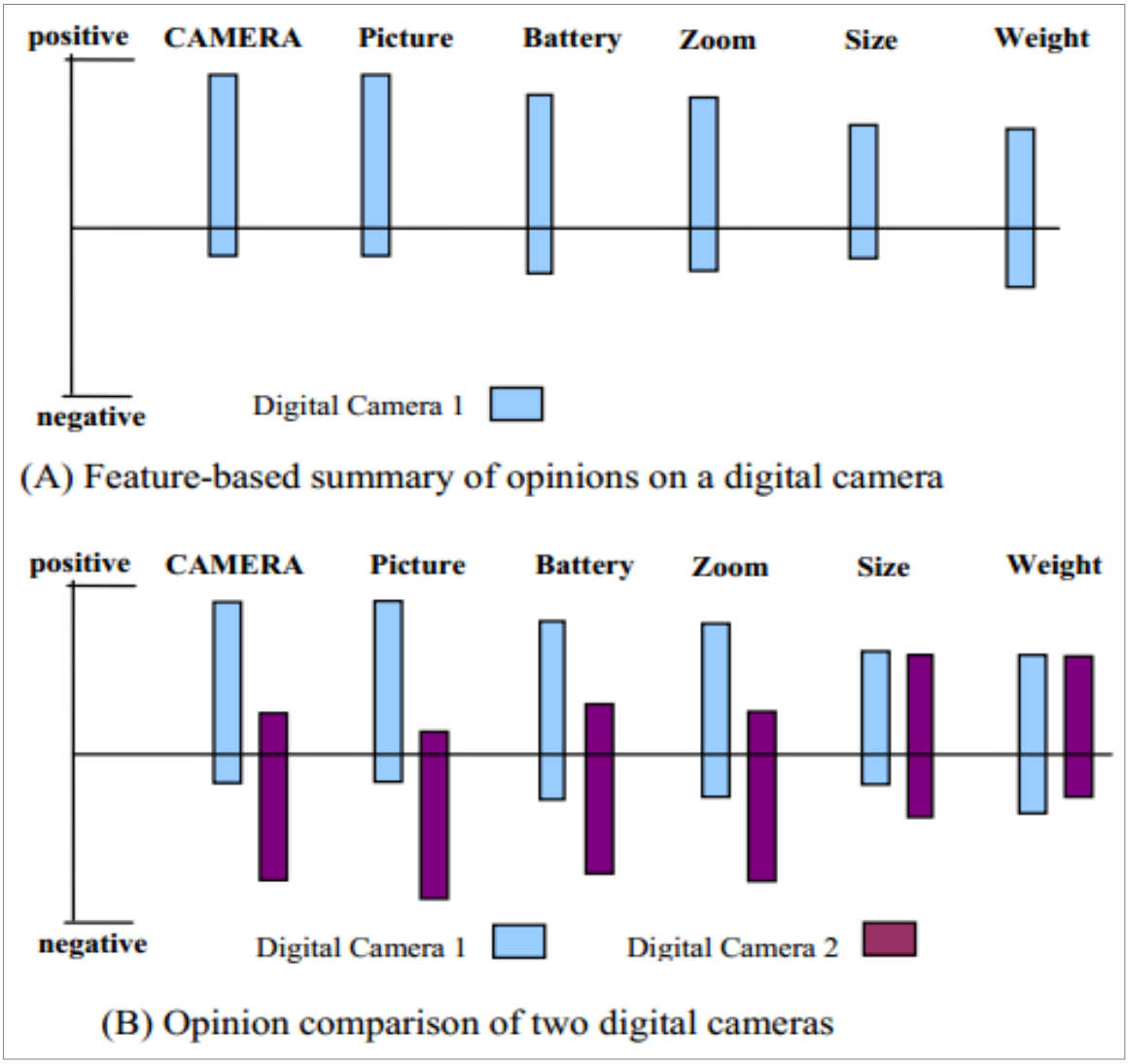


Рис.3.1 Визуализация результатов из сводки featured-based мнений

На диаграммах рис.3.1 часть прямоугольника, расположенная выше оси абсцисс, отображает количество положительных мнений в отношении характеристики (которые, в свою очередь, перечислены вверху), а та часть, которая ниже, отображает количество негативных мнений для той же характеристики. Очевидно, возможны и другие варианты визуализации. К примеру, можно показывать только процент позитивных (или же негативных) мнений для каждой характеристики. Однако представление в виде гистограммы может оказаться более наглядным в случае, когда необходимо сравнение характеристик нескольких конкурирующих продуктов – и рис.3.1 демонстрирует удобный визуальный способ сравнения мнений потребителей для двух цифровых камер. Данное представление наглядно

показывает, насколько различно потребители смотрят на характеристики обеих камер.

3.4. АНАЛИЗ ПРЕДЛОЖЕНИЙ СО СРАВНИТЕЛЬНЫМИ И ПРЕВОСХОДНЫМИ СТЕПЕНЯМИ

Кроме выражения положительного или отрицательного мнения существует и такой способом оценки объекта, как сравнение его с другими подобными. Сравнения одновременно и похожи, и отличаются от прямых, однозначно выраженных, мнений. Например, типичным оценочным предложением является предложение *“Качество изображения камеры х – отличное”*. Пример типичного предложения, содержащего сравнение: *“Качество изображения камеры х лучше, чем у камеры у”*. В целом, сравнительные предложения выражают отношения, основанные на сходстве или различии более чем одного объектов. В русском языке сравнения обычно выражаются сравнительными или превосходными степенями прилагательных и наречий или различными слов-модификаторов, например, *“очень”*, *“менее”* перед прилагательным или наречием. Извлечение сравнительных предложений требует определения того, какие части и характеристики объекта подвергаются сравнению, и того, какой из объектов предпочтительней автору мнения [5, 10].

Вывод:

Проблеме извлечения и анализа мнений из интернет-источников уже уделяется внимание, однако его нельзя признать достаточным – при том, что такие инструменты/сервисы имели бы широкую сферу применения. Они были бы востребованы и частными лицами – при совершении покупок, и организациям – для понимания того, как потребители воспринимают их продукты и продукты их конкурентов. Получение интегрированных оценок удобнее, чем изучение большого количества обзоров для формирования представления о сильных и слабых сторонах продукта.

4. ОБЗОР СУЩЕСТВУЮЩИХ АЛГОРИТМОВ КЛАССИФИКАЦИЙ МНЕНИЙ

Задачей классификации текстов является автоматическое присвоение документу d одной из предопределенных категорий $C=\{c_1,...,c_n\}$, исходя из анализа содержания d.

Кроме фактологической информации (описание ситуации, изложение событий, перечисление фактов) документы могут содержать еще и эмоционально окрашенное оценочное суждение, мнение автора. То есть, согласно [26] и [27], тексты могут быть разбиты на два класса: "*объективные'"* или "*субъективные"* .
В свою очередь, *"субъективные*" документы тоже могут быть разбиты на два класса, в зависимости от полярности в целом содержащейся в каждом из них оценки: *"положительные"* или *"отрицательные"*.

Таким образом, простейшей, но важной задачей классификации в рамках анализа мнений является разбиение документов на два класса с учетом эмоциональности (субъективности) их содержания.

4.1. МЕТОДЫ МАШИННОГО ОБУЧЕНИЯ (MACHINE LEARNING)

Для облегчения обзора алгоритмов машинного обучения введем следующие обозначения согласно [18, 21]. Пусть $\{f_1 , ..., f_m\}$ – предопределенное множество из m возможных свойств (признаков) документа, в качестве которых могут выступать содержащиеся в нем слова или биграммы (пары соседних слов). Пусть в документе d свойство f_i встречается $n_i(d)$ раз. Тогда каждый документ d будет представлен вектором документа $d=(n_1(d),n_2(d), ...,n_m(d))$.

4.1.1. НАИВНЫЙ БАЙЕСОВСКИЙ КЛАССИФИКАТОР

Одним из подходов к классификации текстов является присвоение данному документу *d* наиболее вероятного для него класса $c^* = argmax_C\ P(c|d)$ исходя из теоремы Байеса:

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

где *P(d)* не влияет на *c**, а при вычислении *P(d | c)*, предполагается, что все f_i независимы друг от друга:

$$P_{NB}(c|d) := \frac{P(c)(\prod_{i=i}^{m} P(f_i|c)^{n_i(d)})}{P(d)}$$

Несмотря на простоту и предположении об условной независимости свойств, наивный байесовский классификатор хорошо работает на реальных задачах [23]. В [24] показано, что он оптимален для определенных задач в которых классы содержат сильно зависимые свойства. С другой стороны, более сложные алгоритмы могут давать и дают на практике лучшие результаты.

4.1.2. МЕТОД ОПОРНЫХ ВЕКТОРОВ (SVM)

Метод опорных векторов является очень эффективным для традиционной классификации текстов и обычно превосходит наивный байесовский классификатор [25]. В случае задачи с двумя классами основная идея обучения состоит в переводе исходных *векторов документов* в пространство более высокой размерности и, далее – в нахождении гиперплоскости, задаваемой вектором ω, которая разделяет вектора документов с максимальным зазором в этом пространстве. Поиск такой гиперплоскости является задачей оптимизации. Допустим, $c_j \in \{1, -1\}$ (corresponding to positive and negative) является правильным классом для документа d_j, тогда решение может быть задано следующим образом:

$$\varpi := \sum_j a_j c_j \vec{d}_j, a_j \geq 0,$$

где a_j получены путем решения задачи дуальной оптимизации. Вектора d_j , для которых $a_j > 0$, называются *опорными векторами*. Классификация тестовых объектов состоит в определении, того на какую из сторон гиперплоскости они попадают.

4.2. ЭМОЦИОНАЛЬНО ОКРАШЕННАЯ ЛЕКСИКА

Мнения выражаются, и их выражение усиливается использованием эмоционально окрашенных слов, извлечение которых является важной подзадачей, помогающей полярной классификации. Анализируемыми термами здесь являются прилагательные, наречия и их сочетания, а также существительные.

4.2.1. АНАЛИЗ ПАР ПРИЛАГАТЕЛЬНЫХ

В работе Hatzivassiloglou and McKeown [16] решалась задача определения тональности (полярности) прилагательных путем анализа их пар, соединенных с помощью *и, или, но, или...или, ни...ни,* из большого набора неразмеченных документов. Интуитивно понятно, что факт объединения прилагательных, с помощью конструкций, перечисленных выше, задает лингвистические ограничения тональности для данных прилагательных (союзом *и* обычно объединяют два прилагательных одинаковой тональности, в то время, как союз *но* объединяет два прилагательных противоположной тональности). Например:

1) *Мне подарили красивую и интересную книгу;*

2) *Мне подарили красивую, но неинтересную книгу;*

однако возможно и такое:

3) *Мне подарили красивую и бесполезную книгу.*

Данный метод вывода тональности прилагательных из анализа их сочетаний включает в себя трехступенчатый метод обучения с учителем:

1) Из множества документов извлекаются все сочетания прилагательных.

2) Множество извлеченных сочетаний разделяются на тренированные и тестовые множества. Сочетания прилагательных из тренированного множества используются для тренировки классификатора, основанного на математической логарифмической регрессии, который классифицирует пары прилагательных как имеющих одинаковую, либо же различную тональность. Затем классификатор применяется к тестовому множеству, производя граф, в котором узлами являются термы, а ребра обозначают, что соединенные прилагательные имеют “одинаковую ориентацию”, либо “противоположную ориентацию”, в зависимости от сочетаний извлеченных из множества документов.

3) Алгоритм кластеризации использует граф созданный на шаге 2, для разделения прилагательных на два кластера. Следуя интуиции о том, что положительные прилагательные, как правило, используются чаще чем отрицательные, кластер содержащий термы, имеющие большую среднюю частоту появление во множестве документов, считается содержащим положительные термы.

Для экспериментов в [16] использовано множество термов, состоящее из 657/679 прилагательных, помеченных как *положительные* и *отрицательные* соответственно. Коллекцией документов, используемой для извлечения сочетания прилагательных, является неразмеченное множество документов 1987 Wall Street Journal. В [16] вышеописанный алгоритм определяет тональность прилагательных с точностью 78.08%.

4.2.2. УЧЕТ СЕМАНТИЧЕСКОЙ БЛИЗОСТИ ТЕРМОВ ПО ВЗАИМНОЙ ИНФОРМАЦИИ

Turney and Littman [17] подошли к проблеме определения тональности термов используя метод *самораспространения* (bootstrapping) термов из двух начальных полярных множеств термов:

Sp = {хороший, отличный, позитивный, счастливый,правильный, превосходный} *[good, nice, excellent, positive, fortunate,correct, superior]*

Sn = {плохой, неприятный, скудный, негативный, несчастный, неправильный, худший} *[bad, nasty, poor, negative, unfortunate, wrong, inferior]*

Метод основан на вычислении *поточечной взаимной информации* (pointwise mutual information, PMI) для целевого терма *t* и каждого терма t_i из начального множества как меры их семантической близости. Для терма *t* значение его тональности *O(t)*, задаются следующим образом:

$$O(t) = \sum_{t_i \in S_p} PMI(t,t_i) - \sum_{t_i \in S_n} PMI(t,t_i) \quad (1)$$

Положительный знак *O*(t) означает положительную тональность, а большее абсолютное значение означает более выраженную тональность.

Авторы протестировали свой метод на наборе термов из [16], а также для категорий, определенных в General Inquirer Lexicon [21]. Данная система анализа текстов может оперировать, в зависимости от потребностей задачи, различными категориями термов. Двумя основными категориями являются *Положительная*/*Отрицательная* (содержащие 1,915/2,291 положительных и отрицательных термов соответственно). Примерами положительных термов являются *польза, добросовестность, достоинство,* в качестве отрицательных можно привести термы *плохо, рак, вялый.* В своих экспериментах, авторы сократили список термов до 1,614/1,982 вхождений (здесь и далее – *множество термов TL*). Для этого авторы удалили термы, находящиеся одновременно в обоих категориях (всего таких термов 17,

например, терм *бороться*), и удалили те термы, которые повторялись вследствие их многозначности.

Поточечная взаимная информация вычислялась с использование двух методов, один из которых основан на методах извлечения информации (PMI-IR), а другой – на латентно-семантическом анализе (PMI-LSA). В методе PMI-IR частоты термов и частоты совместного появления термов, измерялись путем выполнения следующих запросов к поисковому движку по набору документов: *"t", "t_i", "t NEAR t_i"*. Далее, полученные значения количества документов, найденных поисковым движком, использовались для расчета вероятностей, необходимых для вычисления PMI. В поисковом движке AltaVista, который использовался в ходе эксперимента, оператор NEAR считает документ релевантным если операнды встречаются в любом порядке в документе с максимальной дистанцией в 10 термов между ними. Это более сильное ограничение, нежели чем те, которые устанавливает оператор AND. Оператор AND просто требует, чтобы его операнды были в любом месте в одном документе.

Для экспериментов были использованы три набора документов на английском языке:

1) AV-Eng, состоящий из документов, проиндексированных движком AltaVista, и содержащий 350 миллионов страниц со ста миллиардами термов (приблизительно);
2) AV-CA, состоящий из документов домена '.ca' и насчитывающий 7 миллионов страниц с приблизительно 2 миллиардами термов;
3) TASA, состоящий из документов коллекции Touchstone Applied Science Associates, создаваемой для "The Educator's Word Frequency Guide", из 61000 документов с 10 миллионов слов.

Результаты в [17] показывают, что точность увеличивается с ростом используемой коллекции документов. Это довольно очевидно на

интуитивном уровне: чем большее количество документов обрабатывается, тем достоверней получается информация о совместном вхождении слов.

4.2.3. УЧЕТ СЕМАНТИЧЕСКОЙ БЛИЗОСТИ СИНОНИМОВ

В [19] Kamps et al. использовали лексические отношения, извлеченные из словаря WordNet (WN, http://wordnet.princeton.edu/). Они создали граф, содержащий прилагательные из пересечения множеств TL и WN, и добавляли связь между двумя прилагательными всякий раз, когда в WN была синонимичная связь между этими словами. На этом графе, авторы определили расстояние $d(t_1,t_2)$ между термами t_1 и t_2 , выражающее кратчайший путь, который соединяет t_1 и t_2.
Если t_1 и t_2 не связаны, то $d(t_1,t_2) = +\infty$,). В результате, ориентация терма определяется относительным расстоянием до двух начальных термов *хороший* и *плохой*:

$$SO(t) = \frac{d(t,bad) - d(t,good)}{d(good,bad)}$$

Прилагательное t считается положительным, если $SO(t) > 0$. Абсолютное значение $SO(t)$ определяется численным значением этой величины, а постоянный знаменатель d(*хороший*, *плохой*) является нормализующим фактором, ограничивающий значения SO в пределах [-1, 1].

С помощью этого метода могут быть обработаны только прилагательные, для которых существовал путь в графе до начальных множеств термов. Это является причиной того, почему авторы ограничили свои эксперименты 663 прилагательными из множества TL (что является 18.43% всех 3596 термов). Для данных термов существовал путь до темов *хороший*/ *плохой* через синонимичные связи WN. Авторы получили 67.32% точности.

5. РАНЖИРОВАНИЕ В ИНФОРМАЦИОННОМ ПОИСКЕ

5.1. ЦЕЛИ И СИСТЕМЫ ИНФОРМАЦИОННОГО ПОИСКА

Термин *информационный поиск* (*information retrieval)* имеет очень широкое значение. В частности, его можно определить таким образом: информационный поиск – это процесс нахождения неструктурированных данных (документов, сообщений), удовлетворяющий информационные потребности. В связи с бурным ростом интернета в настоящее время в процесс веб-поиска оказались вовлечены миллионы людей по всему миру. Информационный поиск все больше преобладает над более традиционным реляционным подходом к извлечению данных, затрагивая и другие проблемы информационного характера, помимо тех, которые указаны в определении выше.

Термин *неструктурированные данные* относится к данным, не имеющим той структуры, с которой компьютер мог бы работать напрямую. В противоположность этому, структурированные данные, каноническим примером которых является реляционная база данных, обычно используются компаниями для поддержки своих продуктов и учета кадров. В реальности, практически нет данных, которые являлись бы полностью неструктурированными. Например, это является справедливым для текстовой информации, если принимать во внимание скрытую лингвистическую структуру человеческого языка. Помимо этого, большинство текстов содержат такие структурные элементы как оглавление, параграфы и сноски, в большинстве документов являющиеся явной разметкой. Информационный поиск использует такие элементы разметки для улучшения качества поиска, например, путем задания ограничений области поиска до названий документов. Информационный поиск также применяется к уже извлеченным документам.

Системы информационного поиска могут предназначаться для работы с совершенно различными по объемам массивами данных. В *веб-поиске* система должна предоставить возможность поиска по миллиардам документов, расположенных на миллионах компьютеров, то есть сбора и последующей индексации огромного объема данных при поддержке системой особенностей строения веба. В частности, здесь возникают проблемы поискового спама и так называемой “накрутки” рейтинга сайта.

С другой стороны, в операционных системах уже обычны встроенные поисковые сервисы (например, Spotlight в Apple OS X или Instant Search в Windows VIsta), применяющиеся к разнообразным типам документов, обычно хранящихся на персональном компьютере. Современные почтовые клиенты предоставляют кроме поиска по содержимому писем и адресным спискам еще и спам-фильтры, а также ручную, либо автоматическую классификацию почты. Эти задачи должны решаться быстро и эффективно в условиях ограниченных ресурсов персонального компьютера пользователя.

Помимо перечисленных областей информационного поиска существует промышленный и корпоративные сектора, где поиск может проводиться, например, во внутренней коллекции документов компании, базе данных патентов или научных статей. В этом случае документы обычно хранятся в централизованной файловой системе, где поиск по коллекции производит одна или небольшое количество машин.

5.2. ПРИМЕРЫ ЗАДАЧ ИНФОРМАЦИОННОГО ПОИСКА

5.2.1. БУЛЕВА МОДЕЛЬ

Предположим, необходимо определить, какая пьеса Шекспира содержит слова *Брут* и *Цезарь*, и не содержит слово *Кэлпурния*. Можно начать

просматривать весь текст, отмечая каждую пьесу со словами *Брут* и *Цезарь*, исключая при этом пьесы со словом *Кэлпурния* – это простейшей линейный поиск по всем документам. Он может оказаться эффективным главным образом благодаря производительности современных компьютеров и обычно является хорошим инструментом для поиска по текстовым шаблонам. С помощью современных компьютеров данная модель может неплохо работать для простых запросов в небольшой коллекции документов.

Но для перечисленных выше областей поиска данных возможностей может не хватить. В веб-поиске еще требуется:

1) Быстро обрабатывать большую коллекцию документов. Объем данных онлайн растет вместе с производительностью компьютеров, и сегодня востребован поиск по миллионам документов с триллионами слов.
2) Необходимо иметь более гибкие поисковые операции. Например, непрактичным является выполнение запроса "*Romans* NEAR *countrymen*" с помощью простого линейного поиска, где NEAR может быть определено как "*в пределах 5 слов*" или "*в пределах одного предложения*".
3) Иметь возможность поиска с ранжированием: в большинстве случаев необходимо среди множества документов найти ответ, который наиболее полно удовлетворяет информационным потребностям.

Одним из способов, который позволяет избежать подобного рода линейного поиска для каждого запроса, является индексация документов. На примере уже упоминавшегося Шекспира, рассмотрим булеву модель информационного поиска. Предположим, мы делаем запись для каждого документа (в данном случае пьесы Шекспира), которая показывает какие слова, из всех используемых в произведении, содержатся в нем. В результате получается бинарная матрица инцидентности:

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	…
Antony	1	1	0	0	0	1	
Brutus	1	1	0	1	0	0	
Caesar	1	1	0	1	1	1	
Calpurnia	0	1	0	0	0	0	
Cleopatra	1	0	0	0	0	0	
mercy	1	0	1	1	1	1	
worser	1	0	1	1	1	0	
…							

► **Figure 1.1** A term-document incidence matrix. Matrix element (t, d) is 1 if the play in column d contains the word in row t, and is 0 otherwise.

Рис. 5.1 Бинарная матрица инцидентности.

Термы являются единицами индексации; обычно это просто слова, но в литературе на тему информационного поиска устоялся термин *терм*, т. к. некоторые понятия, например "*К-9*" или "*Санкт-Петербург*" не выражаются в виде единственного слова. Теперь, в зависимости от того выбираем мы строки или столбцы матрицы, мы можем создать либо вектор для каждого терма, который показывает в каких документах он содержится, либо же вектор для каждого документа, показывающий, какие из термов в нем встречаются.

Для того, чтобы ответить на запрос "*Брут* AND *Цезарь* AND NOT *Кэлпурния*", необходимо с векторами для термов *Брут*, *Цезарь* и *Кэлпурния* выполнить побитовую операцию AND, инвертировав, при этом, последний вектор:

110100 AND 110111 AND 101111 = 100100

Для данного поискового запроса получаются следующие ответы:

Antony and Cleopatra, Act III, Scene ii
Agrippa [Aside to Domitius Enobarbus]: Why, Enobarbus,
When Antony found Julius Caesar dead,
He cried almost to roaring; and he wept
When at Philippi he found Brutus slain.

Hamlet, Act III, Scene ii
Lord Polonius: I did enact Julius Caesar: I was killed i' the
Capitol; Brutus killed me.

Рис. 5.2 Ответ на запрос "*Брут* AND *Цезарь* AND NOT *Кэлпурния*".

Более реалистичной является ситуация с коллекцией из 1 миллиона документов, содержащий каждый, к примеру, 1000 слов. Тогда во всей коллекции будет около 500'000 уникальных термов, для которых потребуется огромная бинарная матрица "терм-документ" из 500'000 * 1'000'000 ячеек, подавляющее большинство из которым будет нулевыми. Тогда более рационально будет запоминать только те записи, в ячейках которых содержится единица. Данная идея является ключевой для такого понятия информационного поиска, как *инвертированный индекс*, при котором в поддерживаемом *словаре* термов каждому терму сопоставляется список идентификаторов тех документов, в которых встречается данный терм:

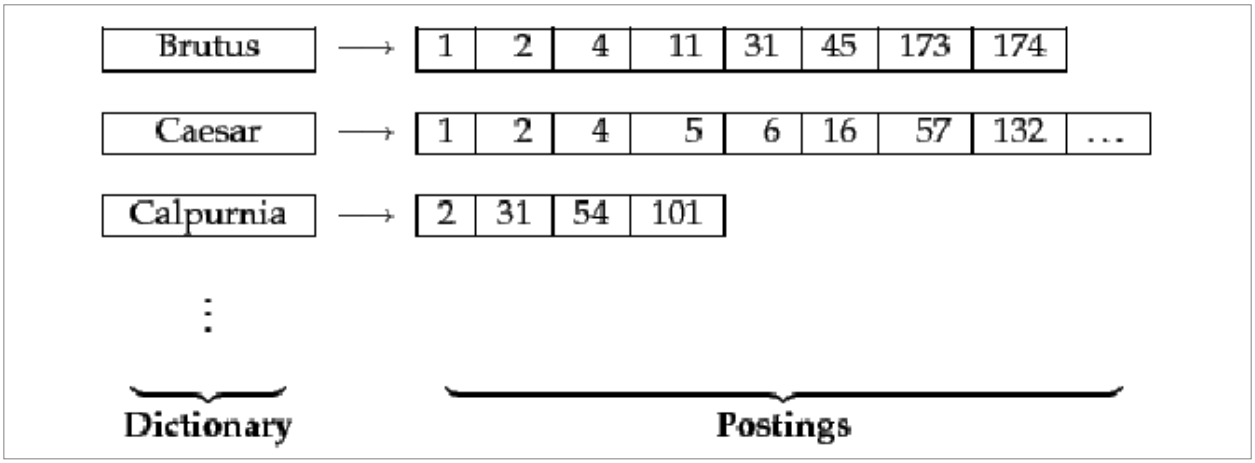


Рис.5.3 Инвертирование индекса.

5.2.2. ВЕКТОРНАЯ МОДЕЛЬ ПОИСКА С РАНЖИРОВАНИЕМ

Для построения запросов, отвечающих различным информационным потребностям, может оказаться недостаточно возможностей строгих логических выражений вместе с неотсортированными результатами выдачи. Для решения этой проблемы булевой модели поиска противопоставляются *модели поиска с ранжированием*, одной из которых является *векторная модель*, в рамках которой пользователь выполняет *свободные поисковые запросы*, просто набирая одно или несколько слов, а не следуя точному формату поисковых операторов. В итоге система решает, какие из документов наилучшим образом удовлетворяют запросу.

В векторной модели, документ представляется в виде вектора, в котором учитывается различная важность термов в документе. Представления набора документов векторами в векторном пространстве называется векторной моделью.

Определим $V(d)$ как вектор, полученный из документа d, где в качестве компонентов вектора – термы. Набор документов в коллекции может быть представлен как множество векторов в векторном пространстве, в котором каждому терму соответствует своя ось. Данное представление не учитывает порядок слов в каждом документе.

В качестве подобия двух документов d_1 и d_2 в векторном пространстве используют *косинусную меру близости* их векторов $V(d_1)$ и $V(d_2)$ соответственно.

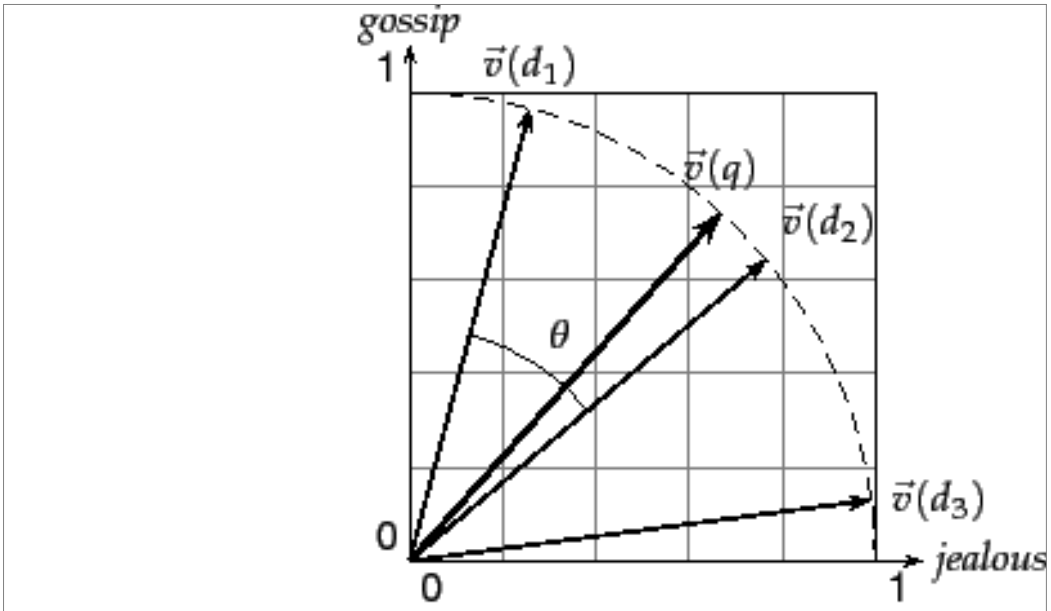


Figure 6.4 Cosine similarity illustrated. $\text{sim}(d_1, d_2) = \cos\theta$.

$$\text{sim}(d_1, d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{|\vec{V}(d_1)||\vec{V}(d_2)|},$$

Рис.5.4 Косинусная мера близости векторов.

Числитель представляет собой скалярное произведение векторов $V(d_1)$ и $V(d_2)$, а знаменателем является произведением их Евклидовых расстояний.

В рамках данной модели поисковый запрос можно трактовать как очень короткий документ, и как следствие, представить его в векторном виде. Таким образом, мы можем вычислить косинусную меру между вектором запроса и вектором документа:

$$\text{score}(q, d) = \frac{\vec{V}(q) \cdot \vec{V}(d)}{|\vec{V}(q)||\vec{V}(d)|}.$$

Документ может получить большое значение косинусной мера для запроса, даже не содержа всех термов из запроса.

5.2.2.1. ВЗВЕШИВАНИЕ ТЕРМОВ

В качестве веса терма в векторе документа можно использовать рассмотренную выше булеву модель. Однако можно развить идею и предположить, что те документы, в которых терм из запроса встречается чаще, имеют больше общего с запросом, и, следовательно, должны получить более высокий ранг. Но простой подсчет частоты термов (*частота терма t в документе d* обозначается как df_t) имеет существенную проблему: при расчета релевантности документа запросу, предполагается, что все термы являются одинаково важными. В действительности, некоторые термы слабо или вообще не влияют на релевантность документа запросу. Например, в коллекции документов на тему автоиндустрии, терм *автомобиль,* скорее всего, присутствует практически в каждом документе. Для ослабления веса термов, появляющихся слишком часто, можно использовать *обратную документную частоту*:

$$\mathrm{idf}_t = \log \frac{N}{\mathrm{df}_t}.$$

где
N – количество документов в коллекции;
$\mathrm{df_t}$ – документная частота терма.

Таким образом, idf_t у редких термов имеет высокое значение, тогда как значение idf_t у высокочастотных термов является низким.

Для получения итоговой формулы взвешивания термов объединим частоту терма с документной частотой. *Tf-idf мера* для терма *t* в документе *d*, рассчитывается по формуле:

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} * \text{idf}_t$$

Значение данной меры ведет себя следующим образом:

1) Является высоким, если *t* встречается часто в малом количестве документов.
2) Является низким, когда терм встречается малое количество раз в документе или содержится во многих документах.
3) Является низким, когда терм встречается практически во всех документах.

5.2.3. АЛГОРИТМ PageRank

В рассмотренной выше векторной модели ранжирование зависит от содержания поискового запроса. Но существуют модели, которые используют другие факторы для осуществления ранжирования. Например, алгоритм PageRank использует ссылочную структуру сети Интернет [28, 29]. Обычно ссылки на страницы создаются пользователями. Подразумевается, что пользователь создают ссылки на те страницы, которые он считает качественными. Таким образом можно воспользоваться данными связями, для того, чтобы упорядочить веб-страницы в зависимости от их качества.

Принцип работы алгоритма:

Представим пользователя веба, который переходит от веб-страницы на страницу, выбирая каждую следующую ссылку для перехода случайно с равной вероятностью. Для случая, когда на текущей страницы нет ссылок для дальнейшего перехода, пользователь с некоторой малой вероятностью выбирает случайную страницу из всего набора страниц веба. Для некоторого достаточного числа итераций алгоритма, вероятность того, что пользователь окажется на странице *j* в какой-то момент времени, задается следующей формулой:

$$P(j)=\frac{(1-\alpha)}{N}+\alpha\sum_{i\in B_j}\frac{P(i)}{|F_i|} \qquad (1)$$

где

F – множество страниц на которые ссылается страница *i;*

B_j – множество страниц, которые ссылаются на страницу *j*.

Значение PageRank для страницы *j* является данной вероятностью: *PR(j)=P(j)*. Поскольку выражение (1) рекурсивно, итерации вычислений должны продолжаться, пока *P(j)* не сойдется (как правило, начальное распределение *P(j)* является равномерным). Смысл в том, что страница тем лучше, чем чаще на нее будет попадать пользователь вследствие случайных переходов. Или, другими словами, каждая страница получает значение ее качества в виде значения *P(j)*.

Для применение алгоритма PageRank для коллекции страниц, сопоставимой со всем объемом web-пространства Интернета, требуются большие распределенные вычислительные мощности.

5.2.4. АЛГОРИТМ HITS

Алгоритм оперирует следующими понятиями:

Авторитетный документ (авторитетная страница, автор) – это документ, соответствующий запросу пользователя, имеющий больший удельный вес среди документов данной тематики, то есть большее число документов ссылаются на данный документ [33].

Хаб-документ (хаб-страница, посредник) – это документ, содержащий много ссылок на авторитетные документы [30, 31].

Страница, на которую ссылаются многие другие должна быть хорошим "автором". В свою очередь страница, которая указывает на многие другие, должна быть хорошим "посредником". Основываясь на этом, в алгоритме HITS для каждой веб-страницы рассчитываются две оценки: оценка авторитетности и посредническая оценка. То есть для каждой страницы рекурсивно вычисляется ее значимость как "автора" и как "посредника".

В алгоритме HITS первым этапом является выделение наиболее релевантных страниц в поисковом запросе. Данный набор страниц, называемый *корневым*

набором, получается путем отбора *n* лучших страниц, возвращаемых текстовым алгоритмом поиска. *Базовый набор* генерируется путем расширения корневого набора множеством тех страниц, на которые ссылаются страницы из корневого набора, и некоторыми страницами, ссылающимися на страницы самого корневого набора. Страницы из базового набора и все гиперссылки между ними формируют сосредоточенный подграф. Расчеты алгоритма HITS производятся только для этого подграфа.

Значение авторитетности и посредническая оценка определены в терминах друг друга и находятся во взаимной рекурсии. Оценка авторитетности страницы вычисляется как сумма значений оценок посреднических страниц, которые указывают на эту страницу. Значение оценки посредника вычисляется как сумма оценок авторитетных страниц, на которые он указывает. Некоторые реализации, также, учитывают значимость связных страниц.

Алгоритм выполняет ряд итераций, каждая из которых содержит два основных шага:

1) Обновление авторитетной оценки каждого узла подграфа, эквивалентное сумме посреднических оценок каждого из вершин, указывающих на них.

2) Хаб-обновление: обновление посреднической оценки каждой вершины подграфа, путем суммирования авторитетных оценок каждой из вершин, на которые они указывают.

Вычисление посреднической оценки и значения авторитетности для вершины происходит следующим образом:

1) Начать с вершин, оценка авторитетности и посредническая оценка которых равна 1.

2) Выполнение правила обновления авторитетности.

3) Выполнение правила хаб-обновления.

4) Нормализация значений путем деления каждой посреднической оценки на корень квадратный из суммы квадратов всех посреднических оценок, и деления каждой оценки авторитетности на корень квадратный из суммы квадратов всех оценок авторитетности.

5) Повторение со второго шага по мере необходимости.

Псевдокод алгоритма:

```
 1 G := набор страниц
 2 for each page p in G do
 3   p.auth = 1 // p.auth значение авторитетности для страницы p
 4   p.hub = 1 // p.hub посредническая оценка страницы p
 5 function HubsAndAuthorities(G)
 6   for step from 1 to k do
 7     norm = 0
 8     for each page p in G do  // обновление посреднических оценок
 9       p.auth = 0
10       for each page q in p.incomingNeighbors do // p.incomingNeighbors
множество страниц, ссылающихся на p
11         p.auth += q.hub
12       norm += square(p.auth) // вычисление суммы квадратов значений
авторитетности
13     norm = sqrt(norm)
14     for each page p in G do  // обновление значений авторитности
15       p.auth = p.auth / norm  // нормализация значений авторитетности
16     norm = 0
17     for each page p in G do  // обновление всех посреднических оценок
18       p.hub = 0
19       for each page r in p.outgoingNeighbors do // p.outgoingNeighbors
множество страниц на которые ссылается p
20         p.hub += r.auth
21       norm += square(p.hub) // вычисление суммы квадратов посреднических
оценок
22     norm = sqrt(norm)
23     for each page p in G do  // обновление всех посреднических оценок
24       p.hub = p.hub / norm   // нормализация посреднических оценок
```

Алгоритм HITS, как и алгоритм PageRank, является итеративным алгоритмом, основанном ссылочной структуре веба. Тем не менее, существует ряд важных отличий [32]:

- в отличии от PageRank, HITS является запросозависимым.
- как результат, вычисления для алгоритма производятся непосредственно в момент запроса, в то время как, PageRank может рассчитываться во

время индексации. Данная особенность прямым образом влияет на время обработки запроса.

- HITS вычисляет два значения для документа (авторитетность и посредническая оценка), против одной у PageRank.
- данный алгоритм обрабатывает только некоторое подмножество “релевантных” документов (базовый набор), в то время как PageRank оперирует всем веб-графом.

6. ВЫБОР ИСТОЧНИКОВ МНЕНИЙ

Мнения пользователей являются основным критерием для улучшения качества услуг и продуктов. Сообщения в блогах, на сайтах, содержащих обзоры, и на платформах микроблоггинга могут помочь дать представление о том, как пользователи воспринимают предоставляемые услуги или продукты.

В последние годы блоги стали весьма популярным средством для выражения пользователями их мнений. Блоггеры зачастую ежедневно описывают события своей жизни и выражают свои мнения, чувства и эмоции [34]. Блоги, содержащие оценочные суждения, используются в качестве источников во многих исследованиях, посвященных анализу мнений. На данный момент Твиттер является крупнейшей микроблоггинговой платформой. Созданный в июле 2006 года, сервис быстро завоевал популярность во всем мире и на данный момент насчитывает более 500 миллионов зарегистрированных пользователей.

Твиттер позволяет пользователям отправлять короткие текстовые заметки (до 140 символов), используя веб-интерфейс, SMS, средства мгновенного обмена сообщениями или сторонние программы-клиенты [36]. Сервис предоставляет доступ к своему API [38]. Для этого необходимо выполнить GET-запрос с параметрам (по необходимости) по адресу нужной функции и обработать полученные данные. Одной из функций является функция поиска, с помощью которой можно производить поисковые запросы по сообщениям пользователей.

Пример запроса к поисковому API:

http://search.twitter.com/search.json?q=blue%20angels&rpp=5&include_entities=true&result_type=mixed

Фрагмент ответа:


```
{
  "completed_in":0.031,
  "max_id":122078461840982016,
  "max_id_str":"122078461840982016",
  "next_page":"?page=2&max_id=122078461840982016&q=blue%20angels&rpp=5",
```

```
  "page":1,
  "query":"blue+angels",
  "refresh_url":"?since_id=122078461840982016&q=blue%20angels",
  "results":[
    {
      "created_at":"Thu, 06 Oct 2011 19:36:17 +0000",
      "entities":{
        "urls":[
          {
            "url":"http://t.co/L9JXJ2ee",
            "expanded_url":"http://bit.ly/q9fyz9",
            "display_url":"bit.ly/q9fyz9",
            "indices":[
              37,
              57
            ]
          }
        ]
      },
      "from_user":"SFist",
      "from_user_id":14093707,
      "from_user_id_str":"14093707",
      "geo":null,
      "id":122032448266698752,
      "id_str":"122032448266698752",
      "iso_language_code":"en",
      "metadata":{
        "recent_retweets":3,
        "result_type":"popular"
      },
      "profile_image_url":"http://a3.twimg.com/profile_images/51584619/SFist07_normal.jpg",
      "source":"&lt;a href=&quot;http://twitter.com/tweetbutton&quot;
rel=&quot;nofollow&quot;&gt;Tweet Button&lt;/a&gt;",
      "text":"Reminder: Blue Angels practice today http://t.co/L9JXJ2ee",
      "to_user_id":null,
      "to_user_id_str":null
    },
    {
      "created_at":"Thu, 06 Oct 2011 19:41:12 +0000",
      "entities":{

      },
...
```


Подавляющее большинство перечисленных выше исследований и сервисов анализа мнений использовали именно Твиттер в качестве источника мнений. Вследствие огромной аудитории сервиса и удобного API именно Твиттер был выбран в качестве одного из источников мнений в настоящей работе.

На сегодня множество социальных сетей предоставляют элементы функционала блогов. Российская социальная сеть «ВКонтакте» содержит такие элементы. По данным на февраль 2013г. ежедневная аудитория «ВКонтакте» – более 43 миллионов человек [37]. Взаимодействие приложения с API происходит путем создания HTTP-запроса по адресу: http://api.vkontakte.ru/api.php.

Как и Твиттер, «ВКонтакте» также имеет API для поиска по записям пользователей, вследствие чего и выбирается в качестве еще одного источника мнений.

Параметры, передаваемые в запросе [39]:

Параметр	Описание
q	Поисковой запрос, по которому необходимо получить результаты.
count	указывает, какое максимальное число записей следует возвращать, но не более 100.
offset	смещение, необходимое для выборки определенного подмножества результатов поиска.
start_time	время, в формате **unixtime**, начиная с которого следует получить новости для текущего пользователя. Если параметр не задан, то он считается равным значению времени, которое было сутки назад.
end_time	время, в формате **unixtime**, до которого следует получить новости для текущего пользователя. Если параметр не задан, то он считается равным текущему времени.
start_id	Строковый id последней полученной записи. (Возвращается в результатах запроса, для того, чтобы исключить из выборки нового запроса уже полученные записи)
extended	**1** если нужно получить информацию о пользователе или группе, разместившей запись. По умолчанию **0.**

Пример ответа в формате XML:

```
<?xml version="1.0" encoding="utf-8"?>
<response>
 <count>1</count>
 <item>
  <id>2151</id>
  <owner_id>66748</owner_id>
  <date>1307134338</date>
  <text>API</text>
  <geo>
   <type>place</type>
   <coordinates>59.9357328591 30.3258929702</coordinates>
   <place>
    <place_id>10404</place_id>
    <title>Зингер</title>
    <type>3</type>
```

```
    <country_id>1</country_id>
    <city_id>2</city_id>
    <address>Невский просп. 28</address>
   </place>
  </geo>
  <comments>
    <count>0</count>
    <can_post>1</can_post>
  </comments>
  <likes>
     <count>0</count>
     <user_likes>0</user_likes>
     <can_like>1</can_like>
     <can_publish>0</can_publish>
  </likes>
 </item>
</response>
```

7. ПРИМЕРЫ РЕШЕНИЯ СХОДНЫХ ЗАДАЧ ИЗВЛЕЧЕНИЯ МНЕНИЙ

7.1. ПРИМЕР РЕАЛИЗАЦИИ БАЙЕСОВСКОГО КЛАССИФИКАТОРА ДЛЯ TWITTER

В [22] для обучения байесовского классификатора были собраны положительные и отрицательные твитты (сообщения из Твиттера), а источником объективных сообщений стали Твиттер-аккаунты популярных газет и журналов

Их сбор осуществлялся путем подачи запроса API Твиттера, где в качестве текста запроса использовались два типа значков-смайлов, часто встречающихся в твиттах:

- положительные эмоции: “:-)”, “:)”, “=)”, “:D” и т.д.;
- отрицательные: “:-(”, “:(”, “=(”, “;(” и т. д.

Вследствие ограничений сервиса на длину сообщений (140 символов), те, как правило, представляют из себя одно предложение. Поэтому предполагалось, что смайл напрямую определяет эмоциональную окраску всего сообщения. Собранные данные использовались для обучения классификатора.

Для получения n-грамм (n соседних слов) из сообщений удалялись ссылки, имена пользователей Твиттера и другие специальные обозначения сервиса (например, “RT”). Производилась токенизация. Удалялись знаки стоп-слова и знаки препинания. При выделеннии n-грамм отрицания (как например, “no” и “not”) присоединялись также и к следующими за ними словам. Таким образом, например, из предложения “I do not like fish” получались следующие биграммы: “I do+not”, “do+not like”, “not+like fish”. Данная процедура позволяет улучшить точность, т. к. отрицания играют особую роль в оценочных текстах [20].

В качестве классификатора был выбран полиномиальный наивный байесовский классификатор.

Наивный байесовский классификатор основывается на теореме Байеса:

$$P(s|M) = \frac{P(s) \cdot P(M|s)}{P(M)}$$

где

s – полярность сообщения (один из двух классов),

M – сообщение.

Вследствие того, что было собрано одинаковое количество как положительных, так и отрицательных отзывов, выражение было упрощено:

$$P(s|M) = \frac{P(M|s)}{P(M)}$$

$$P(s|M) \sim P(M|s)$$

Было обучено два байесовских классификатора, которые использовали в качестве характеристик наличие n-граммы и информацию об распределении частей речи соответственно. Классификатор, основанный на n-граммах, в качестве бинарной характеристики использовал факт присутствия n-граммы. Классификатор, использующий информацию о распределении частей речи, оценивал вероятность наличия тегов частей речи в других множествах текстов и использовал ее для вычисления апостериорной вероятности.

Для повышения точности классификатора удалялись распространенные n-граммы, которые не характеризовали ни оценочные, ни объективные тексты. Подобные n-граммы, равномерно распределены по всем собранным наборам данных. Для выявления таких n-грамм, были предложены две подхода.

Первый подход, основан на вычислении энтропии распределения n-грамм по классам. Согласно формуле Шеннона [49]:

$$entropy(g) = H(p(S|g)) = -\sum_{i=1}^{N} p(S_i|g) \log p(S_i|g)$$

где N – число классов (в данном случае = 3).

Высокое значение энтропии означает, что распределение n-грамма по классам близко к равномерному. Следовательно, данная n-грамма вносит малый вклад в качество классификации. Низкое значение энтропии, напротив, означает, что n-грамма встречается значительно чаще в одних классах, чем в других, и, таким образом, имеет более выраженную объективную или субъективную направленность. Поэтому, для увеличения точности классификации использовались n-граммы с низким значением энтропии. Выбиралось некоторое минимальное пороговое значение θ, и по нему производилось отсечение. Стоит отметить, что данная мера понижала полноту.

Во втором подходе авторы вводят понятие "*значимости*"(*salience*), которое вычисляется для каждой n-граммы:

$$salience(g) = \frac{1}{N} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} 1 - \frac{min(P(g|s_i), P(g|s_j))}{max(P(g|s_i), P(g|s_j))}$$

В ходе эксперимента было показано, что отсечение списка n-грамм по *значимости* увеличивает точность и превосходит по этому параметру отсечении по значению энтропии.

7.2. ON-LINE АНАЛИЗАТОРЫ ТОНАЛЬНОСТИ ДЛЯ TWITTER

Известны коммерческие и некоммерческие реализации поисковых инструментов для анализа тональности твитов. Как правило, пользователь вводит поисковый запрос и получает все положительные и отрицательные (а иногда и нейтральные) твиты вместе с некоторыми графиками и круговыми диаграммами.

7.2.1. АНАЛИЗАТОР Setiment140

Сервис создан тремя выпускниками Стэнфордского университета Alec Go, Richa Bhayani и Lei Huang [40]. Функциональность сервиса полностью включает в себя вышеописанные пункты.

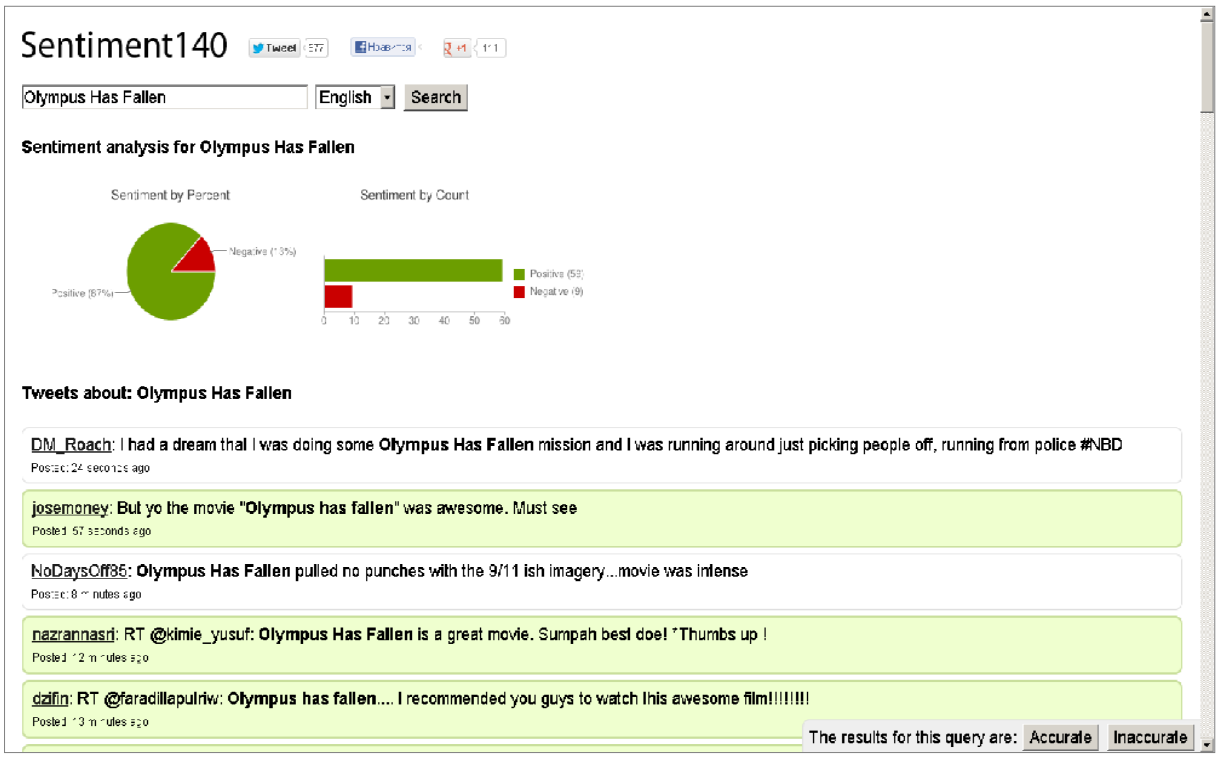


Рис.7.1 Пример выдачи результата поиска для фильма.

На данный момент поддерживаются английский и испанский языки. По замыслу разработчиков, в выдаче присутствует только 10% нейтральных твитов. Исходной код сервиса закрыт, однако суть алгоритма описана в [41].

В качестве бэкэнда используется Amazon EC2. Google Closure для Javascript библиотек. Google Visualization API и Google Closure для графиков. Google Sites для размещения документации. В качестве форм для отзывов используется Google Spreadsheets.

Сервис предлагает API для классификации твитов. Для запроса API подается JSON объекта с текстом сообщения посредством HTTP POST запроса на http://www.sentiment140.com/api/bulkClassifyJson.

Алгоритм работы.

Как и в [22], для получения необходимого количества сообщений для обучения классификатора и его тестирования, подавались запросы к API твиттера, содержащие смайлы: ‘:)’, ‘:(‘. Делалось предположения о том, что если, к примеру, сообщение содержит какой-либо положительный смайл (к примеру ':)'), то и все сообщение несет соответствующую окраску. Данный подход удобен тем, что позволяет извлечь большое количество эмоционально окрашенных твитов, с хорошей точность, и при этом не используя эксперта для разметки этих сообщений. В ходе исследование тестировались следующие алгоритмы машинного обучения: наивный байесовский классификатор, классификатор, основанный на принципе максимальной энтропии (MaxEnt), и метод опорных векторов (SVM). В качестве признаков использовались юниграммы, биграммы, юниграммы и биграммы и юниграммы вместе с тегами частей речи. Также, тестировался метод на основе ключевых слов. В данном методе используется публично доступный список слов [42]. Данный список состоит из 174 положительных и 185 отрицательных слов. Для каждого сообщения считается количество слов содержащихся в данном списке.

Классификатор выбирает тот класс, для которого количество слов, в результате подсчета, оказалось наибольшим. Если количество слов одинаково для обоих классов, то выбирается положительный класс.

В следствии того, что термы запроса могут сами по себе обладать эмоциональной окраской, они заменялись на вспомогательный нейтральный терм QUERY_TERM. В ходе экспериментов, было принято решение удалять смайлы из сообщений, т. к наличие первых негативно влияло на точность классификаторов MaxEnt и SVM, при этом лишь немного увеличивая точность наивного байесовского классификатора. Для сокращения

пространства признаков использовались некоторые свойства специфичного формата сообщений Твиттера.

Имя пользователя.

Часто пользователи твиттера для верной адресации своих сообщений используют имена других пользователей, которым адресовано сообщение. Для этого перед упоминаемым именем пользователя добавляют специальный знак @ (например, @alecmgo). В результате, все слова, начинающиеся со знака @, заменялись на специальный токен USERNAME.

Использование ссылок.

Очень часто пользователи включают в свои сообщения ссылки. Все ссылки в сообщениях заменялись на токен URL.

Повторяющиеся символы.

Обычно сообщения в Твиттере написаны очень неформальным языком. Можно найти самые разнообразные описания одного слова. Например слово “круто”, может быть написано, как “крууууууто” или “крууууууууууто” и т.д. Подобные слова предварительно обрабатываются таким образом, что те символы, которые повторяются более двух раз подряд, заменяются на два вхождения данного символа.

Перечисленные выше признаки помогли достичь сокращение пространства признаков на 45.85%.
Из особенностей настройки классификаторов стоит отметить следующее:

При использовании наивного байесовского классификатора, для тех признаков (юниграмм, биграмм) которые не встречались в тренировочном корпусе и в тестовом корпусе встретились впервые, использовалось простое add-1 сглаживание.

Для метода опорных векторов использовалась библиотека SVM Light с линейным ядром. В качестве значений элементов векторов, использовалась бинарная характеристика в виде присутствия или отсутствия признака.

Результаты экспериментов авторов Setiment140:

Хуже всех показал себя метод с использованием ключевых слов (65.2% точности). Для юниграмм, алгоритмы машинного обучения показали следующие результаты: 81.3%, 80.5% и 82.2% точности для наивного байесовского классификатора, MaxEnt и SVM соответственно. Использование одних лишь биграмм не дало какого-либо заметного прироста точности, в то время как совместное использование биграмм и юниграм увеличило точность для наивного байесовского классификатора и MaxEnt до 82.7%; для SVM, точность понизилась до 81.6%. Использование информации о частей речи не принесло пользы и только ухудшило значение точности.

7.2.2. АНАЛИЗАТОР Twitrratr

Приложение позволяет ввести название некоторой сущности и получить в результате перечни твитов по теме, расклассифицированные на положительные, нейтральные и отрицательные, вместе с соответствующей статистикой. При этом в твитах двух крайних классов цветом выделены слова, имеющие, соответственно, позитивную или негативную окраску.

Для классификации сообщений, сервис использует алгоритм, основанный на ключевых словах, описанный выше.

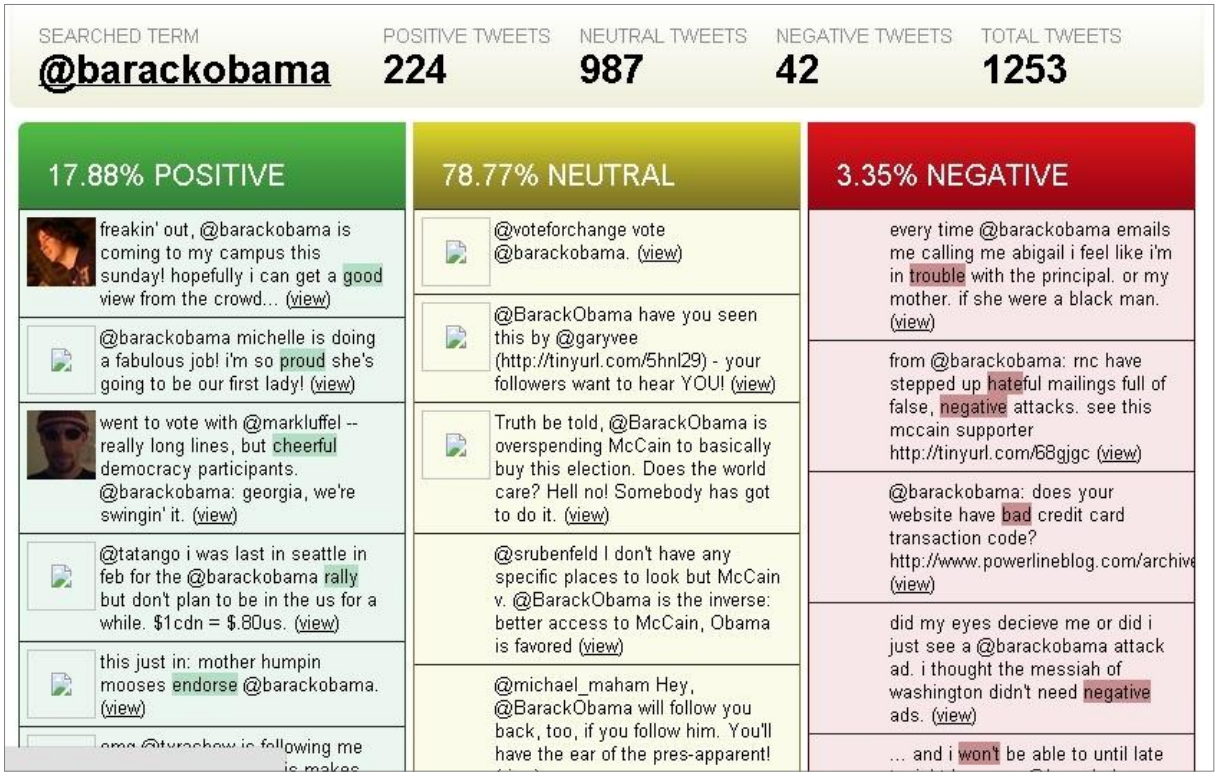


Рис.7.2 Пример выдачи результата поиска по запросу @barackobama .

8. АЛГОРИТМЫ, РАЗРАБОТАННЫЕ ДЛЯ РЕШЕНИЯ ЗАДАЧИ

Для решения задачи в настоящей работе были разработаны алгоритмы:

- алгоритм разбиения документов на два класса (с положительным и отрицательным авторским отношением к объекту высказывания).
- алгоритм вычисления степени субъективности документов.

Разработанные алгоритмы проверялись на предметной области “Кинофильмы”, которой, однако, их применимость не ограничивается – они являются универсальными, настраиваемыми на любую предметную область, для которой имеется исходный обучающий корпус документов (отзывов, суждений) с уже проставленными вручную оценками.

8.1. РАЗБИЕНИЕ КОРПУСА ДОКУМЕНТОВ НА ДВА КЛАССА

Алгоритм классификации включает в себя два обучающих этапа и один результирующий:

- составление словаря лемм (значимых слов в основной грамматической форме) по обширному обучающему корпусу документов;
- использование этого словаря для автоматического подбора границы между “положительными” и “отрицательными” классами путем сравнения с разбиением на эти классы, однократно выполненном человеком-экспертом вручную на небольшом корпусе документов (например, на образце из целевого корпуса);
- собственно разбиение на два класса целевого корпуса документов.

8.1.1. СЛОВАРЬ ЛЕММ

В качестве обучающего корпуса документов были собраны 300’000 отзывов о фильмах с сайта www.imhonet.ru, имеющие каждый оценку фильма автором

по десятибалльной шкале (от 1 до 10). Корпус был обработан морфологическим анализатором *pymorphy* [43] для получения упорядоченного перечня (словаря) лемм – всех встретившихся в тексте морфологических единиц (слов в основной грамматической форме), за вычетом стоп-слов (предлогов, союзов, частиц).

Далее в словаре для каждой леммы вычислялась ее "оценка" – арифметическое среднее значение оценок тех документов (отзывов) в которых она встречалась:

$$mean_l(l) = \frac{\sum_{d \in D_l} score(d)}{|D_l|} \quad (2)$$

где

l – лемма, для которой вычисляется оценка,

D_l – набор документов в которых содержится лемма l,

$|D_l|$ – количество документов в наборе D_l

d – очередной документ из набора D_l

$score(d)$ – исходная авторская оценка документа d (для imhonet – от 1 до 10)

"Оценки" лемм, полученные таким образом из авторских оценок документов (и находящиеся в том же диапазоне от 1 до 10) носят вспомогательный характер – по ним вычисляются оценки целевого корпуса документов.

В качестве оценки документа из целевого корпуса принимается арифметическое среднее оценок лемм этого документа:

$$mean_d(d) = \frac{\sum_{l \in d} mean_l(l)}{|d|} \quad (3)$$

где

d – документ, для которого вычисляется оценка,

$|d|$ – количество лемм в документе d.

Для того, чтобы определить “положительный” или “отрицательный” класс документа, полученное значение $mean_d$ сравнивается с заданной величиной m. В случае, если $mean_d > m$, документ считается положительным, иначе – несущем негативную окраску.

8.1.2. ГРАНИЦА МЕЖДУ ДВУМЯ КЛАССАМИ

Далее проводится настройка алгоритма с привлечением эксперта, который оценивает небольшую часть (несколько сотен) отзывов из обучающего корпуса, но уже выставляя им оценки “+” и “-”, исходя из собственного восприятия их содержания и никак не принимая во внимание уже имеющуюся там авторскую оценку от 1 до 10 (и даже не зная ее).

Для определения граничного значения m, наиболее подходящего для данной предметной области, вышеописанный алгоритм повторяется с различными значениями m до максимизации своей точности – до наилучшего совпадения с выбором человека (простым сравнением суммарного количества совпадений).

Настройка алгоритма путем автоматического подбора границы m по экспертным оценкам дает лучшую точность, чем вычисление m как среднего арифметического всех оценок документов.

8.1.3. ДОСТИГНУТЫЙ РЕЗУЛЬТАТ

После вышеописанного обучения и настройки на предметную область алгоритм применяется к любому целевому корпусу документов из этой предметной области, разделяя их на “положительные” и “отрицательные”.

Этот простой алгоритм показал лучшую точность 0,881 и 0,875 на соревновании РОМИП в 2011 г. в разделе “классификация на два класса для предметной области “фильмы” (см. в [44] и ниже результаты участника zzz-23 для “film” в колонке Object):

Table 4. Two-class classification results (OR)

Run_ID	*Object*	*Macro_P*	*Macro_R*	*Macro_F1*	*Accuracy*
xxx-40	book	0.714	0.804	**0.747**	0.895
xxx-0	book	0.751	0.721	0.735	0.924
xxx-24 (46)	book	0.968	0.630	0.690	**0.938***
xxx-19	book	0.790	0.651	0.694	0.931
Baseline	book	0.460	0.500	0.479	0.920
yyy-24	camera	0.918	0.940	**0.929***	**0.959***
yyy-16	camera	0.944	0.898	0.919	0.956
Baseline	camera	0.426	0.500	0.460	0.852
zzz-23	film	0.776	0.797	**0.786**	**0.881**
zzz-9	film	0.706	0.794	0.730	0.812
zzz-14	film	0.743	0.597	0.623	0.860
Baseline	film	0.427	0.500	0.461	0.854

Table 5. Two-class classification results (AND)

Run_ID	*Object*	*Macro_P*	*Macro_R*	*Macro_F1*	*Accuracy*
xxx-34	book	0.698	0.761	**0.723**	0.902
xxx-0	book	0.739	0.709	0.723	0.921
xxx-24 (46)	book	0.967	0.614	0.668	**0.936***
xxx-19	book	0.789	0.651	0.693	0.929
Baseline	book	0.459	0.500	0.478	0.917
yyy-24	camera	0.909	0.934	**0.921***	**0.957***
yyy-16	camera	0.936	0.881	0.905	0.953
yyy-9	camera	0.890	0.929	0.908	0.949
Baseline	camera	0.422	0.500	0.457	0.843
zzz-23	film	0.760	0.781	**0.770**	**0.875**
zzz-9	film	0.680	0.772	0.702	0.801
zzz-14	film	0.715	0.580	0.600	0.853
Baseline	film	0.423	0.500	0.458	0.846

Рис.8.1. Результаты участников РОМИП’2011.

Вероятно, точность алгоритма можно еще более повысить при учете в нем не только лемм, но и пар взаимосвязанных соседних слов (т.н. биграмм). В данной работе это не реализовано, однако учет биграмм применен для решения второй части задачи.

8.2. РАНЖИРОВАНИЕ ПО СТЕПЕНИ СУБЪЕКТИВНОСТИ

После того, как выяснена положительная или отрицательная эмоциональная полярность документов целевого корпуса, содержащих субъективные оценки/мнения, решается вторая часть задачи по определению степени их субъективности.

8.2.1. ИНФОРМАЦИОННАЯ ЭНТРОПИЯ N-ГРАММ

Аналогично предыдущему алгоритму, тот же обучающий корпус документов обрабатывается морфологическим анализатором pymorphy [43] для добавления к словарю лемм еще и встретившихся в корпусе их биграмм (в общем случае – n-грамм), а также ранее не включавшихся в него стоп-слов (предлогов, союзов, частиц).

Для независимого случайного события с n возможными исходами и функцией распределения вероятности p по исходам, информационная энтропия рассчитывается по формуле Шеннона [49]:

$$H_n = -\sum_{i=1}^{n} p(i) \cdot log_2 p(i) \quad (4)$$

В нашем случае информационная энтропия n-граммы может быть вычислена по вероятностям появления ее в сообщениях с различными авторскими оценками.

То есть, рассматривая документы обучающего корпуса, содержащие n-грамму g, исходом случайного события полагаем обнаружение ее в документе с определенной авторской оценкой s.

$$H_n(g) = -\sum_{s=1}^{n} p(g|s) \cdot log_2 p(g|s) \quad (5)$$

где

n – верхняя граница авторских оценок (в случае imhonet n=10)

$p(g|s)$ – вероятность обнаружить n-грамму g в сообщении с s-ой оценкой.

Вероятности *p(g|s)* рассчитываются по формуле:

$$p(g|s) = \frac{D_l(s)}{D(s)} \quad (6)$$

где

D_l(s) – количество документов с оценкой *s*, содержащих *g* ;

D(s) – количество документов с оценкой *s* .

Согласно Шеннону, энтропия максимальна при равномерном распределении вероятностей. Значит, она будет большей для тех n-грамм, которые встречаются достаточно равномерно во всех сообщениях, и поэтому не являются характерными для сообщения с любой *i*-ой оценки – это справедливо, например, для предлогов.

Соответственно, чем сильнее распределение вероятности n-граммы отличается от равномерного, тем меньше ее значение энтропии. Например, энтропия низка у лемм "*отличный"* и "*хорошо"*, которые гораздо вероятнее встретить в отзывах с оценками 9 или 10, чем в отзывах с оценками 1 или 2.

Аналогично, низкоэнтропийные леммы *"поганый"* и *"плохо"* более вероятны в отзывах с оценками 1 или 2, чем в отзывах с оценками 9 или 10.

Полагаем, что именно неравномерность распределения вероятности и связанная с этим низкая энтропия, свидетельствуют о том, что n-грамма используются при более эмоциональных оценках.

Безусловно, в разных предметных областях энтропия одних и тех же n-грамм будет отличаться. Кроме того, точность вычисления значения энтропии выше на б*о*льших объемах обучающего корпуса.

Примеры для предметной области "кинофильмы" из обучающего корпуса imhonet:

20 слов с минимальной энтропией и частотой употребления более 100:

редкостный, средне, дерьмо, отстой, отвратительно, гадость, обалденный, дыхание, бездарный, уг, преданность, средненький, хрень, лажа, супер, отвратительный, бредятина, шлак, потрясать, суперский

20 слов с максимальной энтропией:

что, быть, мочь, в, на, с, как, конец, потому, стать, тот, она, может, а, чтобы, того, показать, при, и, они

Интересно сравнить предлагаемое “энтропийное” выделение субъективных слов с другими методами, рассмотренными в [45], где корпус документов той же предметной области “кинофильмы” и из того же источника imhonet анализируется шестью алгоритмами для разбиения лемм на “оценочные” и “неоценочные” (“субъективные” и “объективные”):

1) метод k ближайших соседей (kNN);
2) наивный байесовский классификатор (Naive Bayes);
3) персептрон (Perceptron);
4) нейронная сеть (двух- и трехслойная);
5) логистическая регрессия (Logistic regression);
6) метод опорных веторов SVM (стандартный и с радиальной ядровой функцией).

При этом раздельно учитывались прилагательные и остальные части речи.

В [45] полученные разбиения сравнивались с оценками этих же лемм экспертами, при этом лучшими результатами стали: для прилагательных 68,1% успеха показал алгоритм логистической регрессии 5), для не-прилагательных 50,9% показал алгоритм нейронных сетей 4).

В нашем же случае два эксперта оценивали 1000 извлеченных из документов лемм с наименьшей энтропией и подтвердили их субъективность: для прилагательных: **71%** и **92%**, для не-прилагательных: **36%** и **61%**. (Первый эксперт сравнивал строже, трактуя свои сомнения в субъективности леммы как ее объективность, а второй - все-таки как ее субъективность).

8.2.2. СТЕПЕНЬ СУБЪЕКТИВНОСТИ ДОКУМЕНТА

Степень субъективности n-граммы вычисляется по формуле:

$$subj\ (l) = 1 / H\ , \qquad (7)$$

где

H – энтропия n-граммы;

$subj(l)$ – степень субъективности n-граммы l.

Полагаем, что степень субъективности документа d находится по формуле:

$$subj_d(d) = \frac{\sum_{l \in d} subj_l(l)}{|d|} \qquad (8)$$

где

$subj_l(l)$ – степень субъективности l-ой n-граммы

d – документ, для которого вычисляется оценка,

$|d|$ – количество n-грамм в документе d.

8.2.3. ФУНКЦИЯ РАНЖИРОВАНИЯ

Для вычисления ранга документов, найденных по поисковому запросу, используем функцию ранжирования, применимую к различным источникам сообщений s (целевым корпусам документов):

$$range_S(q,d) = w1 * r_{q,d} + w2 * subj_d(d) + w3 * k_S * log(L_{S,d}) \qquad (9)$$

где

$r_{q,\,d}$ – значение косинусной меры между запросом q и документом d.

$subj_d$ (d) – степень субъективности документа d,

$w1, w2, w3$ – веса,

k_S – коэффициент нормализации для источника s,

$L_{s,d}$ – число “лайков” (или аналогичной меры условного одобрения документа d в источнике s.

Функция ранжирования из одного только слагаемого $subj_d(d)$ при внеконкурсных испытаниях на данных РОМИП’2012 в категории *“извлечение мнений из поисковой выдачи”* показала значение метрики NDCG равное **0,2176**, вполне сравнимую с результатами участников соревнования:

System_ID	Object	Clas-ses	Precision@1	Precision@3	Precision@5	Precision@10	**NDCG@10**
xxx-3	film	3	0.493506493506	0.430735930736	0.448701298701	0.437708719852	0.338309302671
xxx-8	film	3	0.493506493506	0.441558441558	0.447402597403	0.443532261389	0.332481298569
xxx-1	film	3	0.467532467532	0.411255411255	0.422077922078	0.421258503401	0.324505855424
xxx-6	film	3	0.493506493506	0.428571428571	0.438528138528	0.431771799629	0.324207945653
xxx-9	film	3	0.25974025974	0.244588744589	0.27316017316	0.277138734282	0.209403526001
xxx-2	film	3	0.246753246753	0.229437229437	0.229437229437	0.227952999382	0.168468593863
xxx-10	film	3	0.246753246753	0.229437229437	0.229437229437	0.227952999382	0.168468593863
xxx-0	film	3	0.25974025974	0.242424242424	0.238528138528	0.234838177695	0.167340869992
xxx-7	film	3	0.25974025974	0.242424242424	0.238528138528	0.234838177695	0.167340869992
xxx-4	film	3	0.311688311688	0.264069264069	0.266666666667	0.265800865801	0.155715460969
xxx-5	film	3	0.298701298701	0.279220779221	0.275324675325	0.275324675325	0.1470939938

(Метрика NDCG описана в [46])

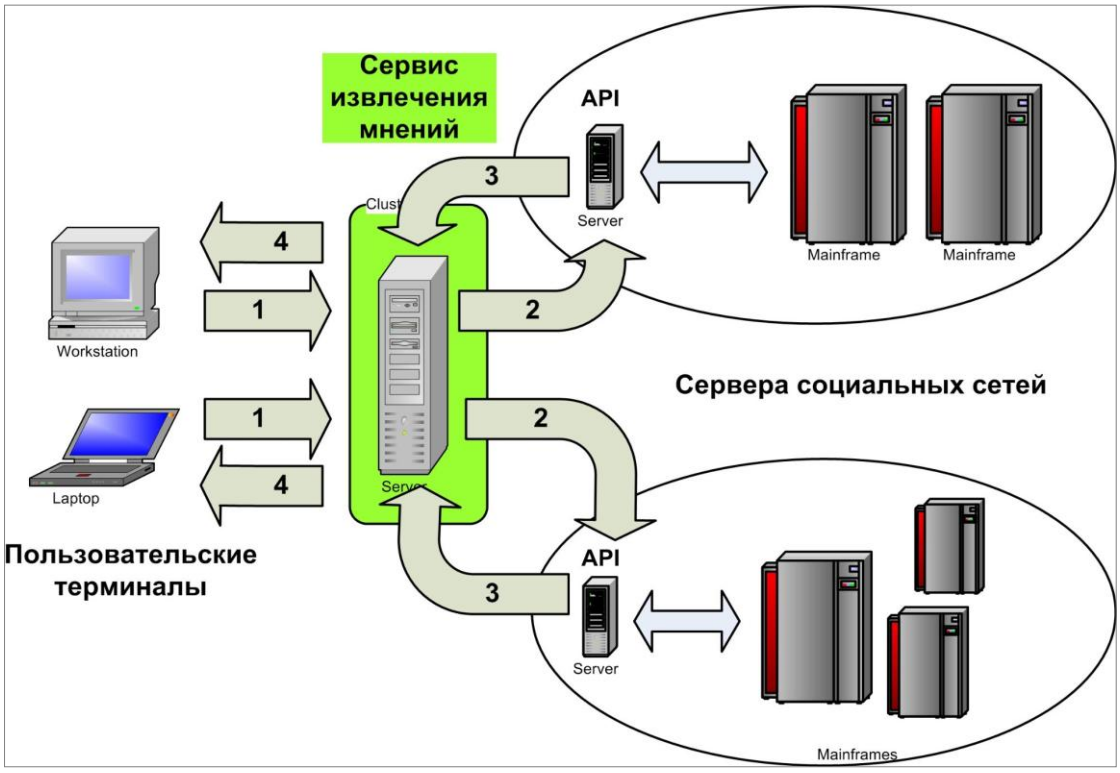
Таким образом, учет только степени субъективности документа, его эмоциональности может служить весомой добавкой к любому методу ранжирования.

Еще одним путем улучшения ранжирования является использование “лайков” (like) – условных одобрений, широко распространенных в социальных сервисах. В [47] продемонстрировано, что качество поиска по сравнению с базовой векторной моделью поиска составляет при этом 7%. (Здесь следует заметить, что в результат поискового запроса отбираются документы мне зависимости от их полярности, то есть и “положительные”, и “отрицательные”, поэтому с помощью “лайков”, фактически, учитывается популярность документа).

9. ОПИСАНИЕ СЕРВИСА ИЗВЛЕЧЕНИЯ МНЕНИЙ

Разработанный on-line сервис предназначен для получения результатов поисковых запросов, направленных в социальные сети, в виде перечней документов, разбитых на два класса (с “положительным” и “отрицательным” отношением их авторов к теме) и отранжированных по степени субъективности (эмоциональности).

Ниже представлена схема внешнего обмена информацией между пользовательскими терминалами, издающими поисковые запросы, и сервисом – с одной стороны, и между сервисом и социальными сетями (их API-интерфейсами) – с другой:



1 - поисковые get-запросы
2 - запросы к API социальных сетей
3 - результаты поиска - целевые корпуса документов
4 - ответы на поисковые заппросы: документы, разбитые на два класса (“положительные” и “отрицательные”) и отранжированные

Рис.9.1 Схема внешнего информационного обмена сервиса.

Разработанный сервис состоит из шести компонентов.

9.1. МОДУЛЬ ОБУЧЕНИЯ

Модуль обучения запрашивает у внешнего Интернет-сервиса imhonet обучающий корпус документов и обрабатывает его для составления словаря n-грамм (лемм и биграмм) данной предметной области. При обработке документов используется модуль нормализации текстов (см. ниже). Для каждой n-граммы словаря по описанным выше оригинальным алгоритмам вычисляется ее оценка (для последующей классификации) и степень субъективности (для последующего ранжирования, а сам словарь будет использоваться далее соответствующими модулями.

Фрагмент кода, получающего сообщения для обучающего корпуса:

```
def download_opinion(url, opener, id):
    try:
        soup = get_soup(url, opener)
        larr = soup.find('a', {'class': 'larr'})
        pages = 1
        opinions = []
        if larr:
            href = larr['href']
            eq = href.find('=')
            pages = int(href[eq + 1:]) + 1
            urls = [url + ('?page=%s' % i) for i in xrange(1, pages + 1)]
            for page, url_ in enumerate(urls):
                subs = get_soup(url_, opener)
                parsed = parse_page(subs, '%s_%s' % (id, page))
                for p in parsed:
                    opinions.append(p)
                print '    page: ', url_
        else:
            opinions = parse_page(soup, id)
            print 'url',url
        return opinions
    except BaseException, e:
        print url, e


def parse_page(soup, id):

    all = soup.findAll('div', {'class': 'opinion-item  post-root'})

    opinions = []
    for i, a in enumerate(all):
        rating = a.find('span', {'class': re.compile(r'rate-mark (good|middle|bad) rate')})
        if not rating:
```

```
            continue
        opinion = a.find('p', {'class': 'opinion-text'})
        if not opinion:
            continue
        opinions.append({'text': opinion.text, 'rating': int(rating.text), 'id': '%s_%s' % (id, i)})

    return opinions
```

В рамках данной работы модуль обучения на конкретной предметной области “кинофильмы” запускается однократно на старте описываемого сервиса, до начала приема пользовательских поисковых запросов. Поэтому он не показан на приведенной ниже структурной схеме сервиса, отражающей его run-time функционирование уже при обработке поисковых запросов.

9.2. МОДУЛЬ WEB-ИНТЕРФЕЙСА

Модуль web-интерфейса (набор хэндлеров - обработчиков URL-ов), предлагает пользователю форму для поискового запроса и получает затем через нее get-запрос для передачи в следующий модуль сбора сообщений (см. ниже). Кроме того, модуль web-интерфейса получает результаты работы модулей классификации и ранжирования - списки найденных по запросу документов, разбитых на два класса “положительные” и “отрицательные” и отранжированных внутри каждого из них. Эти результаты модуль объединяет и направляет пользователю в качестве ответа на его запрос.
Для построения html-страниц запросов и ответов используется шаблонизатор jinja2.

Фрагмент кода с классом, предназначенным для обработки запроса и генерации веб-страницы результата:

```
class SearchHandler(BaseHandler):
    def get(self):
        query_string = self.request.GET.get('query')
        query_info = QueryInfo(lang='ru', query_string=query_string)

        if not query_info:
            return self.render_response('output.html', **{'empty_query_string': True})

        return self.process_query(query_info)
```

```
def process_query(self, query_info):
    query_info.put()
    collector = Collector()
    rank_model = RankModel(model)
    entries = list(collector.collect(query_info))

    feauture_extractor = TestFeatureExtractor(model)
    features = feauture_extractor.extract(entries, 10)
    features = [{'text': unicode(f[0]).lower(), 'score': f[1]} for f in features]

    ranks = (rank_model.get_rank(entry) for entry in entries)
    ids = [(i, r) for i, r in enumerate(ranks)]
    ids = sorted(ids, key=lambda x: x[1], reverse=True)

    classifier = BinaryClassifier(model)
    entries = [{'text': entries[i].get_snippet(),
                'polarity': classifier.classify(entries[i]),
                'url': entries[i].get_url(),
                'icon': entries[i].get_icon()}
                    for i, r in ids]

    pos_n, neg_n = 0, 0
    for e in entries[:100]:
        if e['polarity']:
            pos_n += 1
        else:
            neg_n += 1

    context = {'entries': entries, 'features': features,
               'positive_n': pos_n,
               'negative_n': neg_n}
    return self.render_response('output.html', **context)
```

9.3. МОДУЛЬ СБОРА СООБЩЕНИЙ

Модуль сбора сообщений анализирует поступившие от модуля web-интерфейса поисковые запросы, обращается к социальным сетям через их API-интерфейсы и получает оттуда целевые корпуса документов для их объединения и последующей нормализации модулем нормализации (см. далее).

Набор опрашиваемых настоящим сервисом социальных заранее предопределен. Сейчас в него включены русскоязычные Twitter, ‘Вконтакте’ и Facebook. Таким образом, пользователь указывает в своем запросе только интересующую его тему (например, в нашем случае - название фильма),

полагаясь на то, что поиск будет выполнен сразу во всех наиболее популярных социальных сетях.

Модуль написан таким образом, чтобы добавление нового источника сообщений (целевого корпуса документов) осуществлялось максимально легко.

Фрагмент кода с классом, отвечающим за получения сообщений из социальной сети Вконтакте:

```
class Vk(BaseSource):
    def __init__(self, settings):
        self.__settings = settings
        self.__token, _user_id = vk_auth.auth(self.__settings['email'], self.__settings['password'],
                                              self.__settings['api_id'],"newsfeed")

    def make_document(self, news):
        return Post(text=strip_tags(news['text']), url="http://vk.com/wall%s_%s" % (news['owner_id'],
news['id']))

    def download(self, query, n, lang):
        news_feed = api.search_newsfeed(query, n, self.__token)
        return [self.make_document(news) for news in news_feed]

    def get(self, query_info):
        return self.download(query_info.query_string, self.__settings['n'], 'ru')
```

9.4. МОДУЛЬ НОРМАЛИЗАЦИИ

Модуль нормализации, получив объединенный целевой корпус документов (сообщений), удаляет из них него некоторые малозначимые стоп-слова (их перечень легко корректируется), оставшиеся слова проходят процесс лемматизации и сбора из лемм биграмм. В качестве лемматизатора используется морфологический анализатор pymorhy2.

Результатом работы модуля являются нормализованные тексты документов, готовые для дальнейшего анализа в модулях классификации и ранжирования.

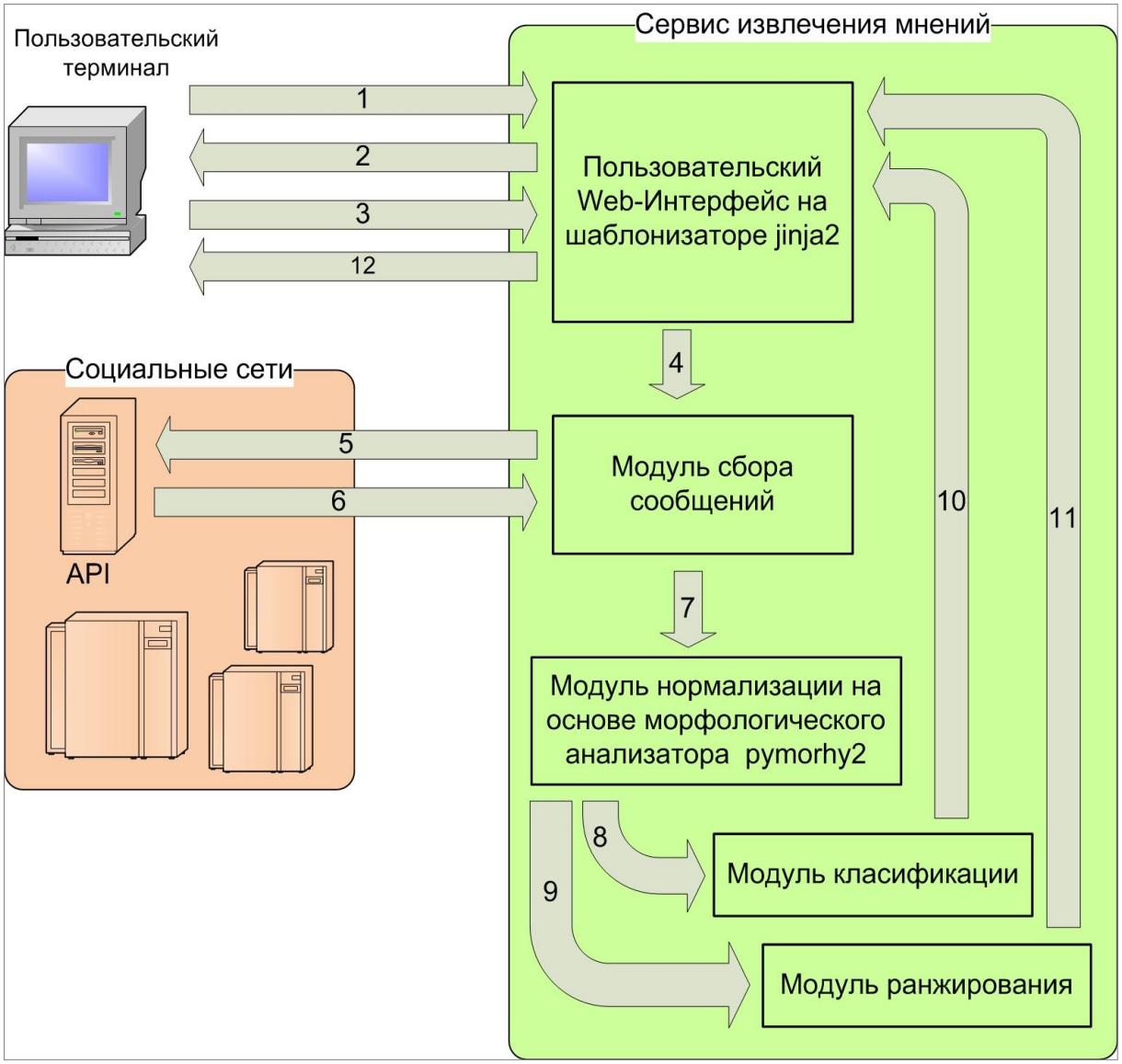
9.5. МОДУЛЬ КЛАССИФИКАЦИИ

Модуль классификации разработан на основе описанного выше оригинального алгоритма классификации и при своей работе использует составленный на этапе обучения сервиса словарь лемм предметной области с их оценками.

При разработке сервиса выбор был сделан в пользу оригинального алгоритма (показавшего, как было указано выше, точность 0,881 и 0,875) по сравнению с наивным байесовским классификатором. Последний был реализован для тестирования с помощью модуля библиотеки NLTK [48] на юниграммах и показал на данной предметной области меньшую точность 0,74 (для его обучения использовался размеченный экспертом корпус из 300 документов и аналогичная нормализация документов).

9.6. МОДУЛЬ РАНЖИРОВАНИЯ

Модуль ранжирования реализует функцию ранжирования (9).
Модули классификации и ранжирования передают результаты своей работы (перечень документов, отранжированных и разбитых на “положительные” и “отрицательные”) модулю web-интерфейса для отправки запрашивающему пользователю.



1 - вызов сервиса
2 - html-форма запроса
3 - поисковый get-запрос
4 - поисковый запрос
5 - запросы к API социальных сетей;
6 - результаты поиска в социальных сетях (целевые корпуса документов)
7 - объединенный корпус документов
8, 9 - нормализованный корпус документов
10, 11 - результаты классификации и ранжирования
12 - ответ на поисковый запрос

Рис.9.2 Схема внутреннего информационного обмена сервиса.

10. ВЫБОР ЯЗЫКА И СРЕДЫ РАЗРАБОТКИ

10.1. ЯЗЫК РАЗРАБОТКИ PYTHON

Для реализации сервиса был выбран высокоуровневый язык программирования Python, реализация интерпретатора 2.7. Одним из его преимуществ является его удобочитаемость и ясность. Программный код на языке Python читается легче, а значит многократное его использование и обслуживание выполняется гораздо проще, чем использование программного кода на других языках. Кроме того, Python поддерживает современные механизмы многократного использования программного кода из арсенала объектно-ориентированного и функционального программирования. По сравнению с компилирующими или строго типизированными языками, такими как C, C++ и Java, Python во много раз повышает производительность труда разработчика. Большая часть программ на этом языке выполняется без изменений на всех основных платформах. В составе Python поставляется большое число собранных и переносимых функциональных возможностей, составляющих стандартную библиотеку.

10.2. СРЕДА РАЗРАБОТКИ ECLIPSE

В качестве среды разработки сервиса была выбрана IDE Eclipse – достаточно развитая и широко известная интегрированная среда разработки модульных кросс-платформенных приложений, являющаяся свободным программным обеспечением. В пользу этого выбора сыграл и тот факт, что у автора работы уже имелся опыт работы с Eclipse. Являясь кросс-платформенной, система поддерживает много языков программирования, а также позволяет добавлять поддержку новых посредством гибкой системы плагинов. Используя этот фунцкионал, автором разработки в Eclipse была добавлены поддержка языка Python.

Система развивается и поддерживается консорциумом Eclipse Foundation.

Цель ее создания была сформулирована следующим образом: "разработать богатую, полнофункциональную индустриальную платформу коммерческого качества для разработки сильно-интегрированных инструментов".

Для достижения этой цели консорциум нацелен на три главных проекта:

1) The_Eclipse (http://www.eclipse.org/eclipse/index.html) непосредственно Eclipse IDE ("платформы", содержащей и исполняющей инструменты Eclipse), инструментов разработки для Java (Java Development Tools, далее JDT) и среды разработки Plug-In (Plug-In Development Environment, далее PDE), позволяющих расширять платформу.
2) Eclipse_Tools (http://www.eclipse.org/tools/index.html) имеет своей целью создание инструментов для платформы Eclipse (в текущей разработке находятся подпроекты создания IDE для Cobol, IDE для C/C++, а также инструмента для построения EMF моделей).
3) The_Eclipse_Technology (http://www.eclipse.org/technology/index.html) ответственен за технологические разработки, эволюцию и обучение использованию платформы Eclipse.

Платформа Eclipse в сочетании с JDT включает многие из возможностей, которые включаются в коммерческие IDE: редактор с подсветкой синтаксиса, инкрементальная компиляция кода, потокобезопасный отладчик, навигатор по классам, менеджеры файлов и проектов, а также интерфейсы к стандартным системам контроля исходных текстов, таким как CVS и ClearCase.

Проект Eclipse представляет собой первую столь мощно поддержанную мировым IT-сообществом попытку создания единой открытой интегрированной платформы разработки приложений, обладающей надежностью, функциональностью и уровнем качества коммерческого продукта. Фактически, эта платформа предназначена для всего и ни для чего конкретно, представляя собой основу, имеющую блочную структуру и интегрирующую инструменты разработки ПО различных производителей для создания приложений на любом языке, с использованием любых

технологий и для любой программной платформы. Вокруг проекта Eclipse в настоящее время сформировано сообщество крупнейших IT-компаний, среди которых Borland, IBM, SAP AG, RedHat и другие.

Также Eclipse предлагает множество уникальных возможностей, например рефакторинг кода (http://www.refactoring.com), автоматические обновление/установка кода (с помощью Менеджера Обновлений), список текущих задач, отладку модулей с помощью JUnit (http://www.junit.org) и интеграцию с инструментом компоновки Jakarta Ant (http://jakarta.apache.org/ant/index.html).

Несмотря на большое число стандартных возможностей, Eclipse отличается от традиционных IDE по ряду особенностей. Наверное самое интересное в Eclipse то, что она полностью независима от платформы и языка. Помимо языков, поддерживаемых консорциумом в настоящий момент (Java, Cobol, C/C++), ведутся разработки по добавлению в Eclipse поддержки таких языков, как Python, Eiffel, PHP, Ruby, и C#.

10.2.1. СТРУКТУРА И СОСТАВ ECLIPSE

IDE Eclipse является результатом коллективного труда как компаний-вендоров так и некоммерческих объединений компаний и частных лиц. Будучи универсальной средой разработки приложений, Eclipse использует и включает в себя большое количество open-source технологий и программных продуктов, хорошо зарекомендовавших себя на рынке в соответствующих областях. Ниже приведен перечень этих разработок:

Ant - Apache Ant Apache Ant; http://ant.apache.org/index.html

Blowfish Encryption Algorithm - плагин, реализующий 64-битный алгоритм шифрования;

GTK+ - многоплатформенный инструментарий создания GUI, часть проекта GNU Project; http://www.gnu.org

JSch - Java-реализация SSH2; http://www.jcraft.com/jsch

JUnit - инструментарий для тестирования и контроля работы приложений; http://www.junit.org/index.htm

Lucene - поисковый "движок", часть проекта Apache Jakarta, http://jakarta.apache.org;

Open Motif for Linux - часть проекта opengroup.org;

The Java Ssh Applet - плагин, http://www.cl.cam.ac.uk/~fapp2/software/java-ssh;

Tomcat - веб-контейнер, часть проекта Apache Jakarta;

XML Parser for Java - XML-парсер для Java, основанный на Apache Xerces (http://www.alphaworks.ibm.com/tech/xml4j).

В целом, платформа Eclipse предоставляет базис, состоящий из общих строительных блоков и API для рабочих областей и рабочую среду, а также различные точки расширения, через которые может интегрироваться новая функциональность. Через эти точки расширения утилиты, реализованные в виде отдельных плагинов, могут расширять платформу Eclipse. Пользователь видит интегрированную среду разработки (IDE), специализированную набором доступных плагинов. Утилиты могут также задавать новые точки расширения и API, служа тем самым строительными блоками и точками интеграции с другими утилитами.

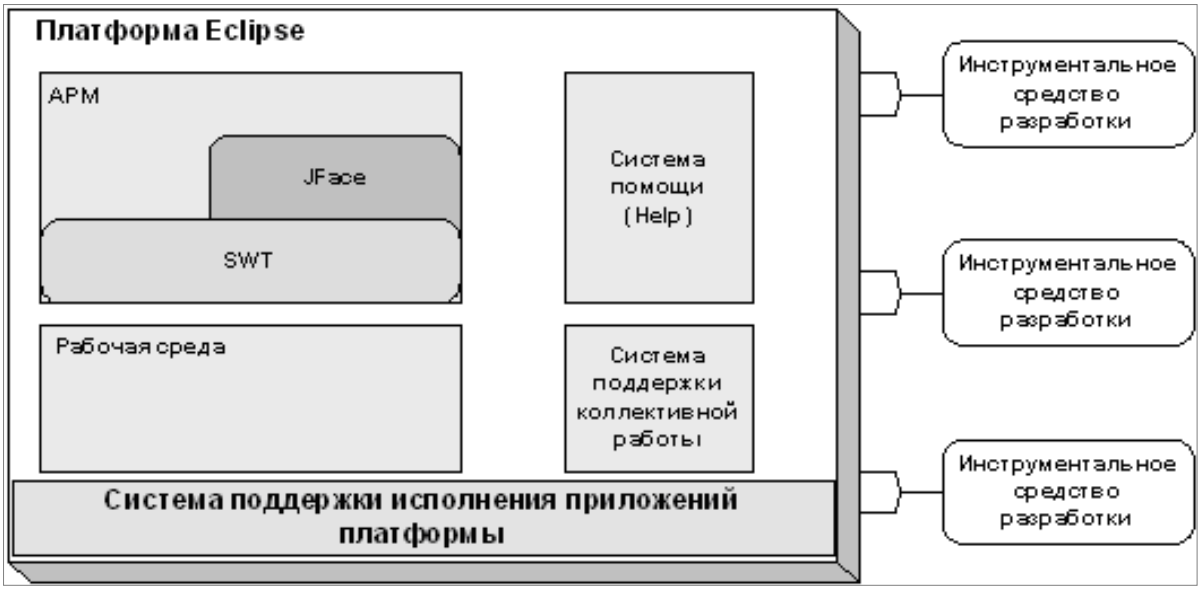


Рис.10.1 Архитектура платформы Eclipse.

Каждое инструментальное средство разработки оформляется в виде плагина (plug-in), реализующего определенный набор функций, присоединяемого к платформе Eclipse посредством своего API и написанного на Java. Как правило, плагин представляет собой Java-код, оформленный в виде архива JAR, несколько файлов для чтения и набор других ресурсов, необходимых для работы плагина, например, библиотеки, графические изображения, шаблоны и т. д.

Консорциум предоставляет готовые исполняемые файлы для Windows, Linux, Solaris, HP-UX, AIX, QNX и Mac OS X. Большой интерес в Eclipse представляет plug-in архитектура, а также богатый API, предоставляемый PDE, позволяющий расширять Eclipse. Добавление поддержки для нового редактора, представления или языка программирования является достаточно простым, благодаря грамотно разработанным API и большим строительным блокам, предоставляемым Eclipse.

Для функционирования необходимы библиотеки поддержки Java. После установки Java VM, система готова к установке Eclipse. Для получения ide, необходимо посетить страницу скачивания файлов Eclipse (http://www.eclipse.org/downloads/) и выбрать последнюю готовую (release) версию для необходимой платформы.

10.2.2. НАЧАЛО РАБОТЫ И ИНТЕРФЕЙС ECLIPSE

Если JVM установлена верно и правильно распакован архив, то программа готова к первому запуску Eclipse. Во всех распространяемых версиях имеется запускающее приложение. Название загрузчика eclipse.exe. Для запуска Eclipse нужно написать файл .bat с двумя строчками, в первой из которых указывается путь к установленной Java VM, а во второй вызывается сам загрузчик eclipse.exe :

```
set PATH=C:\j2sdk1.4.2_04\bin
```

eclipse.exe.

При первом запуске загрузчика Eclipse перед появлением самой среды выполняется ряд завершающих установочных шагов (например, создание директории workspace для хранения файлов проектов).

Рис.10.2 Создание директории workspace для хранения файлов проектов.

После того, как установка завершена, Eclipse готов к использованию – появляется окно со стандартными меню и панелью инструментов:

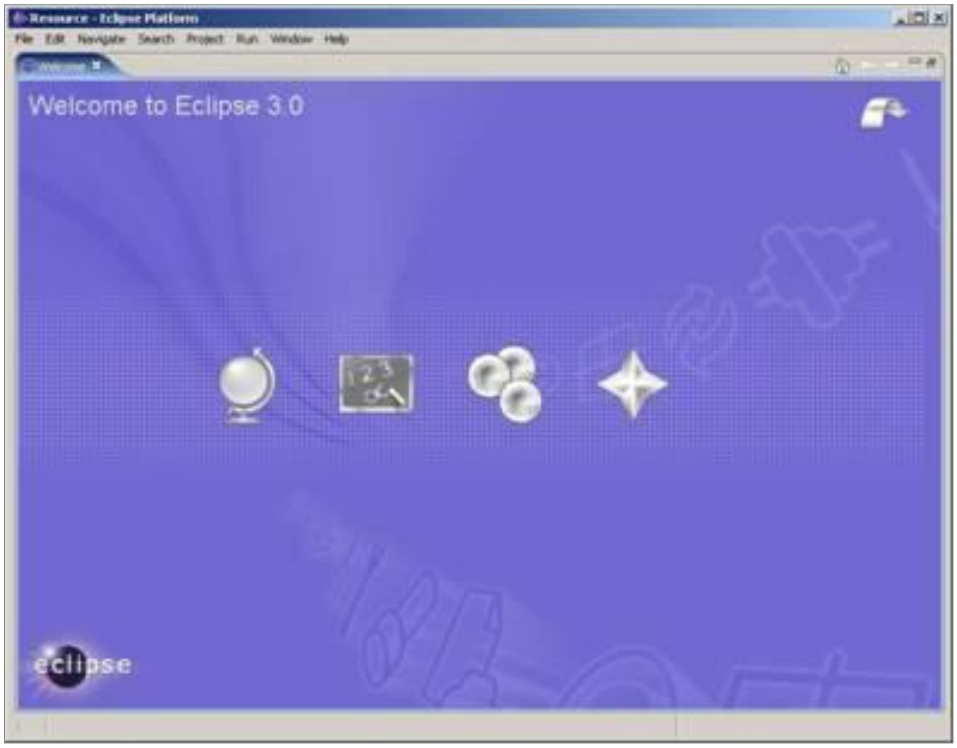


Рис.10.3 Первоначальный вид окна Eclipse.

Также имеется многостраничный Редактор с закладками, Файловый Навигатор (Navigator), Список Текущих Задач (Tasks) и Группировщик Кода (Outline).

После создания проекта, можно обратить внимание на то, что окно Eclipse выглядит не совсем так, как в начале: представление Outline переместилось в правую часть окна, Navigator был заменен на Package Explorer и так далее. Новый внешний вид носит название “Перспектива Java”.

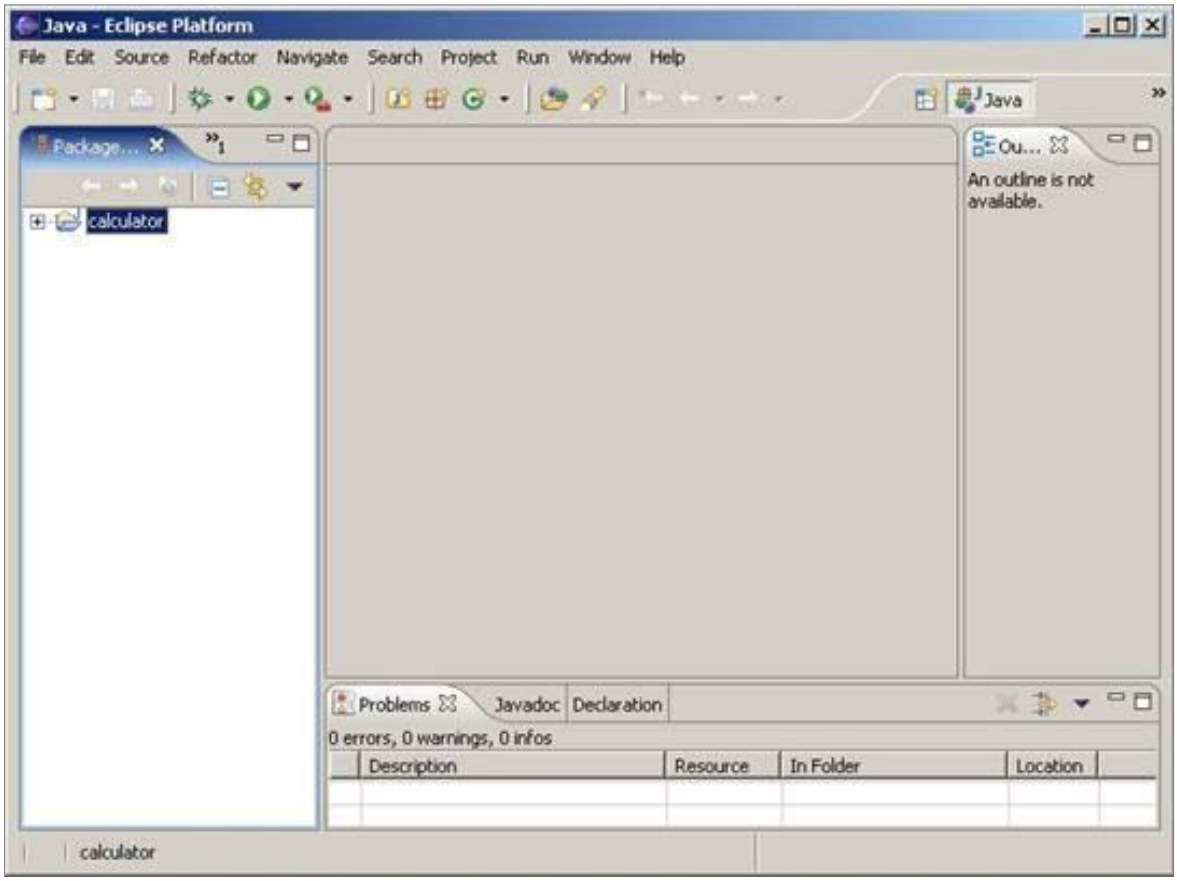


Рис.10.4 Перспектива Java.

“Перспектива” в терминах Eclipse - это сохраняемый внешний вид окна, включающий любое число редакторов (editors) и представлений (views). В поставку Eclipse уже входит несколько готовых настраиваемых перспектив (Java, Debug, Resource и т.д.), но можно также самостоятельно создавать

новые. Перспективы управляются с помощью элементов меню Window или панели инструментов, расположенной обычно вдоль левой границы окна Eclipse.

Eclipse также предоставляет отладчик для разрабатываемого приложения:

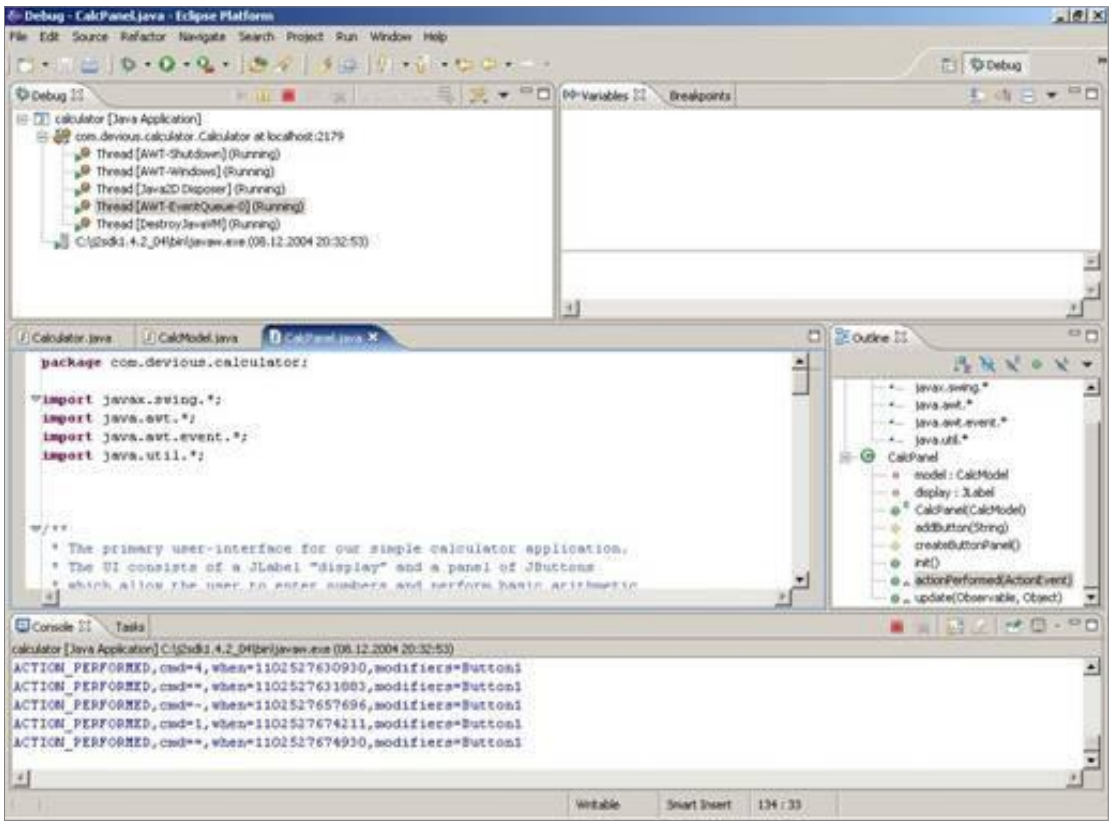


Рис.10.5 Отладчик Eclipse.

11. РАЗМЕЩЕНИЕ РАЗРАБОТАННОГО СЕРВИСА

Для онлайн размещения разработанного сервиса была выбрана платформ Google App Engine. Выбор был продиктован перечисленными ниже ее достоинствами, оказавшимися достаточными для данной задачи. В частности, она поддерживает интерпретатор Python 2.7.

Google App Engine является облачным сервисом построенным по модели «платформа как услуга» (PaaS), предоставляющий инструменты для разработки и хостинга веб-приложений в дата центрах Google. App Engine предоставляет автоматическое масштабирование для приложений, автоматически выделяя все больше вычислительных ресурсов по мере повышения их потребления. Использование Google App Engine является бесплатным вплоть до некоторого уровня потребления ресурсов. Плата взимается за дополнительное дисковое пространство, интернет-трафик и процессорное время, требуемые приложению.
На данный момент уже поддерживаются такие языки программирования как Python, Java и Go. В дальнейшем планируется поддержка большего количества языков.
Веб-фреймворк для Python включает в себя поддержку GAE фреймворка, Django, CherryPy, Pyramid, Flask, web2py и webapp2. Любой Python фреймворк, поддерживающий WSGI, используя CGI адаптер, может быть использован для создания приложения. Фреймворк может быть загружен вместе с разрабатываемым приложением.

Веб-фреймворк Django и приложения, его использующие, могут быть задействованы в App Engine с небольшими модификациями. Модуль Django-nonrel, целью которого является возможность работы Django с не-реляционными базами данных, также поддерживается App Engine.

Приложения разработанные для веб-фреймворка Grails могут быть развернуты на Google App Engine с помощью App Engine Plugin.

Все платные Hight-Replication Datastore App Engine приложения имеют 99.95% аптайма [16]. AppEngine спроектирован таким образом, чтобы выдержать отключение нескольких дата-центров без простоев. Показательно, что хранилище Hight Replication работало без простоев в течении года [17]. Платная поддержка инженеров Google предлагается, как часть Premier аккаунта. Бесплатная поддержка представлена в виде форумов App Engine Groups и сервиса Stackoverflow.

11.1. ОГРАНИЧЕНИЯ ПЛАТФОРМЫ

- Файловая система App Engine доступна разработчикам только на чтение.
- Приложения могут использовать только виртуальные файловые системы, например gae-filestor.
- App Engine может выполнять код только вызванный путем HTTP запроса (запланированные фоновые задачи выполняются с помощью выполнения HTTP запросов самим приложением).
- Пользователи могут загружать любые Python модули, но только написанные на “чистом” Python; C и Pyrex модули не поддерживаются.
- Java приложения могут использовать только подмножество классов из JRE/
- Нельзя использовать домены без ‘www’, например http://example.com .
- Файловое хранилище не позволяет применять более одного фильтры более чем на одно свойство сущности за запрос.
- Процесс, запущенный на сервере для ответа на запрос, не может выполняться более 60 секунд.

11.2. ОТЛИЧИЯ ОТ ДРУГИХ ХОСТИНГОВ

По сравнению с другими масштабируемыми хостингами, таким например, как Amazon EC2, App Engine предоставляет более развитую инфраструктуру для написания масштабируемых приложений.
Инфраструктура App Engine устраняет многие проблемы администрирования и разработки приложения, способного выдерживать сотни и более запросов в секунду. Google обеспечивает развертку кода в кластер, мониторинг, отказоустойчивость и запуск большего количества инстансов приложения при необходимости.

В том время как другие сервисы позволяют пользователям устанавливать и конфигурировать практически любое *NIX совместимое ПО, App Engine требует от разработчиков использовать только поддерживаемые языки, API и фреймворки. Текущие API позволяют хранить и извлекать данные из BigTable – не-реляционной базы данных; совершать HTTP-запросы; отправлять электронную почту; работать с изображениями; поддерживается механизм кэширования. Существующие веб-приложения, требующие реляционной базы данных, не будут работать на App Engine без модификации.

Существует квоты на использование интернет трафика, процессора, количества обработанных запросов, количества одновременных запросов, количсетва вызовов различных API и для отдельных запросов если они выполняются более 60 секунд и возвращают более 32 Мб данных.

11.3. РАЗЛИЧИЯ МЕЖДУ SQL И GQL

Хранилище Google App Engine поддерживают SQL-подобный синтаксис запросов GQL. GQL намеренно не поддерживает оператор JOIN, ввиду того, что данный оператор является неэффективным, когда в обработку запроса

вовлечены более чем одна машина [28]. Вместо этого, связи “один-ко-многим” и “много-ко-многим” могут быть выполнены с использованием ReferenceProperty() [29]. Данный подход позволяет системе оставаться работоспособной при отказе жестких дисков. Уход от реляционной базы данных к Datastore, требует от разработчиков иного подхода к моделированию баз данных.

В отличие от реляционной базы данных, Datastor API не является реляционным в том смысле, в котором является SQL.

Java-версия поддерживает асинхронные неблокирующие запросы при помощи интерфейса Twig Object Datastore. Это альтернатива использованию потоков для параллельной обработки данных.

11.4. ПЕРЕНОСИМОСТЬ

Разработчики обеспокоены тем, что их приложения, при необходимости, не смогут быть портированы с App Engine [32]. Тем не менее, существует множество проектов с открытым исходным кодом, целью которых является создание бэкендов к различным проприетарным API движкам, в особенности к API хранилища. Эти проекты находятся в разных стадиях завершенности и на данный момент ни один из них не находится в том состоянии, в котором развертывание приложения App Engine было бы таким же простым, как и для оригинального сервиса Google. AppScale и TyphoonAE являются представителями таких проектов с открытым исходным кодом.

AppScale может запускать Python, Java и Go Googe App Engine приложения в облаках Amazon EC2 и облаках других компаний.

TyphoonAE может запускать Python App Engine приложения в любых облаках, поддерживающие Linux-машины.

Web2py веб-фреймворк предлагает миграцию между SQL Databases и Google App Engine, хотя и не поддерживает некоторые специфические возможности Google App Engine, такие как транзакции и пространства имен.

11.5. НАДЕЖНОСТЬ

В 2011 году, Google анонсировал App Engine Backends, которые могут работать бесперебойно и потреблять больше памяти.

11.6. Google Cloud SQL

В октябре 2011 года, Google представил SQL базу данных, не требующую обслуживания, которая поддерживает JDDC и DB-API [38]. Данный сервис позволяет создавать, конфигурировать и использовать реляционные базы данных с App Engine приложениями. Движком базы данных является MySql версии 5.1.59 и устанавливается ограничение размера базы данных в 10 Гб.

11.7. КВОТЫ

Для использования Google App Engine требуется аккаунт Google. Разработчик может создать до 10 приложений на одном аккаунте, однако это число может быть увеличено сотрудниками Google.

Google App Engine определяет базовые квоты для бесплатного приложения, которые за дополнительную плату могут быть увеличены:

Неизменяемые ограничения:

Quota	Limit
Время для обработки запроса	60 секунд для обычного запроса, 10 минут для задач, не ограничено для бэкендов
Размер HTTP ответа	32 Мб
Размер объекта в хранилице	1 Мб

Бесплатные квоты:

Quota	Limit (per day)
Процессорное время	28 hours
Количество почтовых сообщений	100 (5000 admin emails)
Входящий трафик	Unlimited
Исходящий трафик	1 GB
Хранилище	1 GB
Операции с хранилищем	50k
Blob хранилище	5 GB
XMPP API	10k stanzas
Channel API	100 channels opened
Conversion API	100 conversions
URLFetch API calls per day	657,000

12. ТЕСТИРОВАНИЕ РАЗРАБОТАННОГО СЕРВИСА

Сервис доступен по адресу www.opinionminer0.appspot.com.

12.1. ПРИМЕРЫ ВЫДАЧИ РЕЗУЛЬТАТОВ СЕРВИСА

Рис.12.1 Наиболее эмоциональные отзывы на фильм “Железный человек” – все оказались положительными

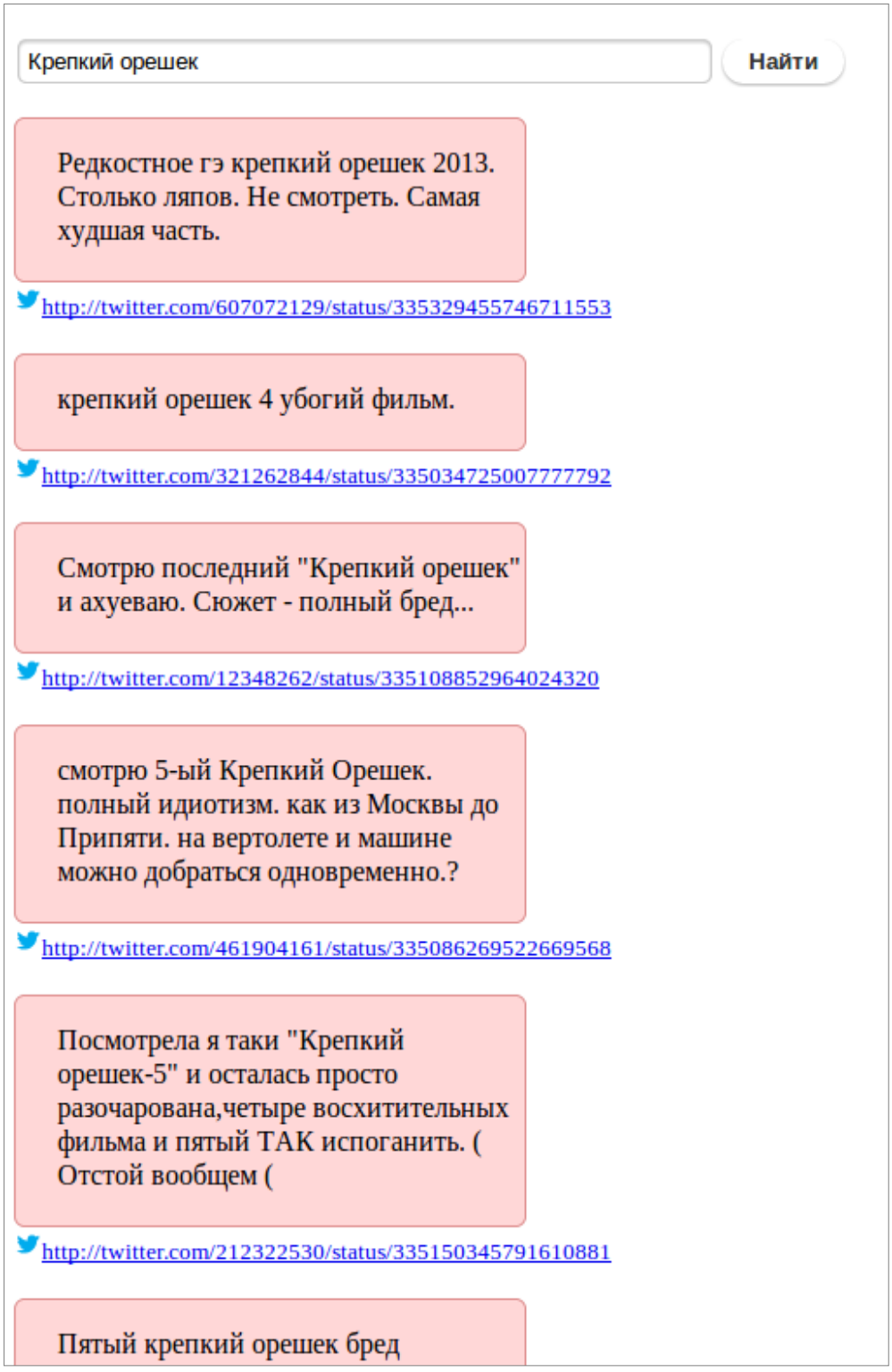


Рис.12.2 Наиболее эмоциональные отзывы на фильм “Крепкий орешек - 5” – все оказались отрицательными.

Рис.12.3 Наиболее эмоциональные отзывы на фильм “Великий Гэтсби” – и положительные, и отрицательные.

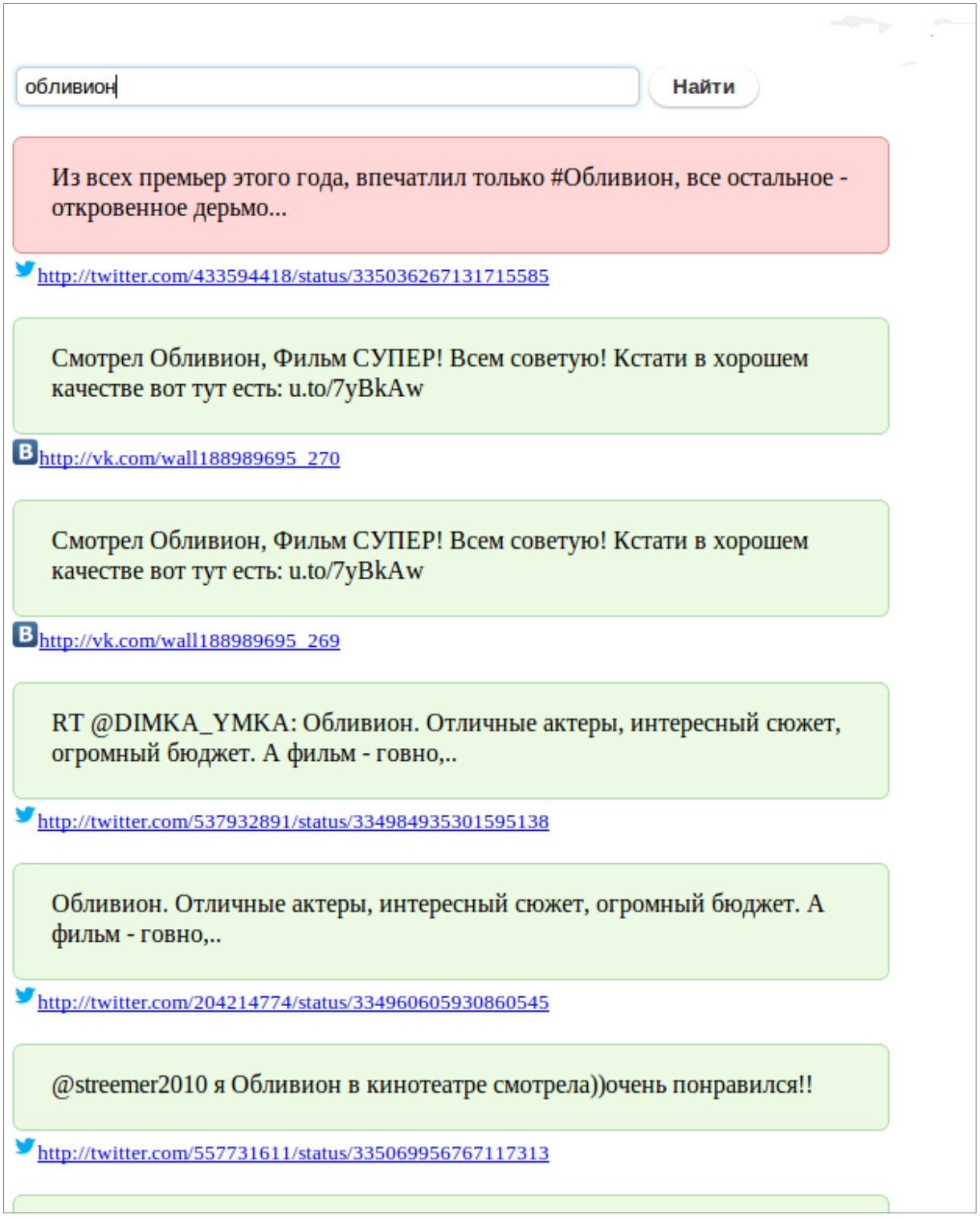


Рис.12.4 Наиболее эмоциональные отзывов на фильм “Обливион” – с двумя ошибками (1-й и 4-й отзывы), демонстрирующими ограниченность любой классификации на документном уровне и даже на уровне предложений (документы-твиты состоят, по-существу, из одного предложения).

12.2. НАГРУЗОЧНОЕ ТЕСТИРОВАНИЕ СЕРВИСА

Нагрузочное тестирование производилось с помощью утилиты siege. Ее запуск был произведен со следующими параметрами:

siege -d1 -c5 -r 40 http://opinionminer0.appspot.com/search?query=startrek

Результат тестирования:

Lifting the server siege... done.
Transactions: 184 hits
Availability: 92.00 %
Elapsed time: 748.25 secs
Data transferred: 4.61 MB
Response time: 18.21 secs
Transaction rate: 0.26 trans/sec
Throughput: 0.01 MB/sec
Concurrency: 4.14
Successful transactions: 170
Failed transactions: 16
Longest transaction: 29.86
Shortest transaction: 7.49

13. ОХРАНА ТРУДА

Согласно определению ФЗ "Об основах охраны труда в Российской Федерации" от 17.07.99 № 181-ФЗ (с изм.), охрана труда - система сохранения жизни и здоровья работников в процессе трудовой деятельности, включающая в себя правовые, социально-экономические, организационно-технические, санитарно-гигиенические, лечебно-профилактические, реабилитационные и иные мероприятия;

Условия труда персонала как совокупность факторов производственной среды и трудового процесса, оказывающих влияние на работоспособность и здоровье работника, должны исключать или минимизировать до нормативно допустимого влияние вредных (могущих вызвать недомогание или заболевание), и, тем более, опасных (могущих вызвать привести к травме) производственных факторов. Такие условия труда признаются безопасными.

Безопасность условий труда должна обеспечиваться как на рабочем месте (там, где работник должен находиться или в которое ему необходимо прибыть в связи с его работой и которое прямо или косвенно находится под контролем работодателя), а также в тех рабочих или транспортных зонах, где присутствует персонал.

Таким образом, мероприятия по обеспечению охране труда устраняют либо минимизируют риски травм, заболеваний или иных вредных воздействий на персонал, предоставляя тому возможность выполнять свои трудовые обязанности в условиях, соответствующих нормативным требованиям гигиены, механической, вибрационной, электрической, радиационной и т.п. видам безопасности.

При эксплуатации средств вычислительной техники имеются риски воздействия некоторых опасных и вредных факторов, однако соблюдение техники безопасности полностью исключает первые из них, и минимизирует влияние вторых, при том, что в случае персональных ЭВМ степень этого влияния находится полностью под контролем самого работника (в случае

использования исправного оборудования, прошедшего в установленном порядке подтверждение соответствия - далее имеем в виду только такое оборудование).

Меры электробезопасности

В настоящее время персональные ЭВМ работают от сети переменного тока напряжением 220В и частотой 50Гц, то есть отвечать требования Технического регламента Таможенного союза ТР ТС 004/2011 “О безопасности низковольного оборудования”, утв. Решением Комиссии Таможенного союза от 16.08.11 № 768. Этот регламет в своей ст.4 устанавливает для низковольтного оборудования следующие исчерпывающие требования безопасности:

Низковольтное оборудование должно быть разработано и изготовлено таким образом, чтобы при применении его по назначению и выполнении требований к монтажу, эксплуатации (использованию), хранению, перевозке (транспортированию) и техническому обслуживанию это оборудование обеспечивало:

необходимый уровень защиты от прямого или косвенного воздействия электрического тока;

отсутствие недопустимого риска возникновения повышенных температур, дуговых разрядов или излучений, которые могут привести к появлению опасностей;

необходимый уровень защиты от травм вращающимися и неподвижными частями низковольтного оборудования;

необходимый уровень защиты от опасностей неэлектрического происхождения, возникающих при применении низковольтного оборудования, в том числе вызванных физическими, химическими или биологическими факторами;

необходимый уровень изоляционной защиты;

необходимый уровень механической и коммутационной износостойкости;

необходимый уровень устойчивости к внешним воздействующим факторам, в том числе немеханического характера, при соответствующих климатических условиях внешней среды;

отсутствие недопустимого риска при перегрузках, аварийных режимах и отказах, вызываемых влиянием внешних и внутренних воздействующих факторов;

отсутствие недопустимого риска при подключении и (или) монтаже.

Низковольтное оборудование должно быть разработано и изготовлено таким образом, чтобы оно не являлось источником возникновения пожара в нормальных и аварийных условиях работы.

Потребителю (пользователю) должен быть предоставлен необходимый уровень информации для безопасного применения низковольтного оборудования по назначению.

В этих же целях указанный техрегламент устанавливает также требования к маркировке и эксплуатационным документам (ст.5) и к порядку подтверждения соответствия как в форме его декларирования, так и в форме сертификации.

Таким образом, необходимым условием соблюдения перечисленных выше видов безопасности, является наличие у персональных ЭВМ полученного установленным порядком документа, подтвердающий соответствие требованиям, установленных техническим регламентом.

Современные персональные ЭВМ, при использовании не для бытовых нужд требуют заземления при подключения к питающей электросети. Как в производственных, так и в бытовых условиях дополнительной защитной мерой является использование в питающей цепи автоматаческого устройства защитного отключения.

При обнаружении каких-либо признаков неисправности питающего провода или вилки с розеткой (механические повреждения, искрение, горелый запах)

следует прекратить работу на персональных ЭВМ и обратиться к работнку, ответственному за их ремонт или за соблюдение правил электробезопасности (или правил охраны труда в целом).

Эксплуатация персональных ЭВМ не подразумевает их сборку-разборку и ремонт работниками, поэтому любые такие действия должны проводиться другими лицами, в чьи трудовые обязанности они входят. Нахождении работника, не занятого ремонтом вблизи разобранной персональной ЭВМ допускается только при полном отклчении оборудования от питающей сети переменного тока.

Меры защиты от электромагнитного излучения

Электронно-лучевая трубка (ЭЛТ) дисплея персональной ЭВМ является источником рентегновского излучения (у персональных ЭВМ с LCD-экраном такой вредный фактор отсутствует). Экраны как ЭЛТ, так и LCD имеют частоту кадровой развертки до 100Гц и строчной развертки - десятки кГц, являясь, тем самым, источником электромагнитного излучения низкой частоты. Кроме того, светящаяся поверхность экрана может являться источником ультрафиолетового излучения. Интенсивность этих вредных факторов у сертифицированных моделей не превышает санитарно-гигиенических норм.

Меры защиты зрения

Программное обеспечение современных персональных ЭВМ позволяют пользователю установить любый комфортный для него рамер шрифтов (при работе с текстом) или масштаба изображения. Кроме того, в настоящее время выпускаются экраны и видео-контроллеры, поддерживающие достаточно высокие частоты разверток (см. выше), позволяющие работникам с нормальным зрением многочасовую работе за экраном. В случаях нарушения зрения индивидуальные рекомендации работникам выдаются по результатм медицинского обследования.

Общими рекомендациями при работе на персональных ЭВМ являются:

Время непрерывной работы - не более 2 часов, продолжительность перерыва - не менее 15 мин;

Суммарное время работы в течение рабочего дня - до 4 часов (пользователь ЭВМ) и 6 часов (технический персонал, обслуживающий ЭВМ);

Удаление глаз от излучающей поверхности экрана - не менее 0,5 м (оптимально 0,6-07 м, с учетом особенностей зрения работника);

Расстояние между экранами с ЭЛТ - более 1,5 м.

Направление освещения рабочего места с персональной ЭВМ - сверху или сбоку (предпочтительно - слева), экраны следует располагать так, чтобы на них не падал прямой дневной свет и не появлялись блики от источников света. Освещенность экрана должна быть в пределах 200-300лк, клавиатуры и стола - 400лк.

Организация рабочего места, эрогономика

Минимальная площадь помещения для одного рабочего места составляет 6 кв.м., объем - 20 куб.м. Производственное помещение должно иметь принудительную или регулярную естественную вентиляцию. Микроклимат должен иметь стандартные санитарно-гигиенические параметры (температура 18-22°С, влажность воздуха 55-62%). Уровень шума сертифицированных моделей персональных ЭВМ и периферийного оборудования (принтеров, сканеров, графопостроителей) не превышает санитарно-гигиенических норм, и при соблюдении указанного выше расстояния между ними от 1,5 м никаких мер шумозащиты не требуется.

Рабочие места должны располагаться от стен с оконными проемами на расстоянии не менее 1,5 м, от стен без оконных проемов на расстоянии не менее 1,0 м.

Несмотря на наличие статического электричества, особенно на поверхностях экранов с ЭЛТ, какие-либо специальные меры защиты от него не требуются -

достаточно выполнения рутинных санитарно-гигиеических процедур по влажной уборке в помещении и протирки пыли.

Особые категории рабтников

К работе с персональными ЭВМ не допускаются беременные и кормящи грудью женщины. Работникам с нарушением зрения или иными заболеваниями, затрудняющими работу с персональными ЭВМ, условия такой работы устанавливаются по результатм медицинского обследования.

Выводы

Использование сертифицированного оборудования и последовательно проводимые вышеперечисленные меры охраны труда обеспечивают защиту от известных вредных и опасных факторов при работе на персональных ЭВМ.

14. ЗАКЛЮЧЕНИЕ

В данной работы был проведен анализ предметной области извлечения мнений (opinion mining & sentimen analisys). Автором разработаны алгоритмы классификации и ранжирования, показавшие хорошую точность в сравнении с результатами РОМИП'2011 и РОМИП'2012. По результатам испытаний оригинальному алгоритму классификации было отдано предпочтение по сравнению с реализованным для этой цели наивным байесовским классификатором. С использованием этих методов был реализован онлайн-сервис извлечения мнений из социальных сетей в предметной области “кинофильмы”.

14.1. ДАЛЬНЕЙШЕЕ РАЗВИТИЕ СЕРВИСА

1. В перспективе целесообразным видится детализация выдачи путем идентификации конкретных объектов, относительно которых выражено мнение в сообщении (новые модуль выделения аспектов).

2. На сегодня интернет-магазины имеют, зачастую, уникальные функционалы добавления мнений к товарам – каждый из них предлагает свою шкалу оценок. Небольшие или недавно запущенные магазины испытывают недостаток мнений даже для самых популярных товаров. В то же время и крупные интернет-магазины имеют аналогичную проблему относительно слабо востребованных товаров. Но и те, и другие могли бы привлекать мнения из внешних источников.
Решением может стать добавление компонента (кнопки) на сайт интернет-магазина для получения мнений из социальных сетей на конкретный товар, используя возможности функционала сервиса, реализованного в данной работе.

ИСПОЛЬЗОВАННЫЕ ИСТОЧНИКИ

1. Kanayama, H. and Nasukawa, T. Fully Automatic Lexicon Expansion for Domain-Oriented Sentiment Analysis. // Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP'06), 2006.

2. Pang, B., Lee, L. and Vaithyanathan, S. Thumbs up? Sentiment Classification Using Machine Learning Techniques. // Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP'02), 2002.

3. Kim, S. and Hovy, E. Determining the Sentiment of Opinions. // Proceedings of the 20th International Conference on Computational Linguistics (COLING'04), 2004.

4. Hatzivassiloglou, V. and McKeown, K. Predicting the Semantic Orientation of Adjectives. // ACL-EACL'97, 1997.

5. Ganapathibhotla, G. and Liu, B. Identifying Preferred Entities in Comparative Sentences. // To appear in Proceedings of the 22nd International Conference on Computational Linguistics (COLING'08), 2008.

6. Hu, M and Liu, B. Mining and Summarizing Customer Reviews. // Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04), 2004.

7. Turney, P. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. // ACL'02, 2002.

8. Carenini, G., Ng, R. and Zwart, E. Extracting Knowledge from Evaluative Text. // Proceedings of the Third International Conference on Knowledge Capture (K-CAP'05), 2005.

9. Ding, X., Liu, B. and Yu, P. A Holistic Lexicon-Based Approach to Opinion Mining. // Proceedings of the first ACM International Conference on Web search and Data Mining (WSDM'08), 2008.

10. Jindal, N. and Liu, B. Mining Comparative Sentences and Relations. // Proceedings of National Conference on Artificial Intelligence (AAAI'06), 2006.

11. Dave, D., Lawrence, A., and Pennock, D. Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews. // Proceedings of International World Wide Web Conference (WWW'03), 2003.

12. Popescu, A.-M. and Etzioni, O. Extracting Product Features and Opinions from Reviews. // Proceedings of the 2005 Conference on Empirical Methods in Natural Language Processing (EMNLP'05), 2005.

13. Liu, B., Hu, M. and Cheng, J. Opinion Observer: Analyzing and Comparing Opinions on the Web. // Proceedings of International World Wide Web Conference (WWW'05), 2005.

14. Wiebe, J. and Riloff, E. Creating Subjective and Objective Sentence Classifiers fromUnannotated Texts. // Proceedings of International Conference on Intelligent Text Processing and Computational Linguistics (CICLing'05), 2005.

15. Bing Liu. Opinion mining. // www.cs.uic.edu/~liub/FBS/opinion-mining.pdf 23.08.2012

16. V. Hatzivassiloglou and K. R. McKeown. Predicting the semantic orientation of adjectives. In Proceedings of ACL-97, 35th Annual Meeting of the Association for Computational Linguistics, pages 174–181, Madrid, ES, 1997.

17. P. D. Turney and M. L. Littman. Measuring praise and criticism: Inference of semantic orientation from association. // ACM Transactions on Information Systems, 21(4):315–346, 2003.

18. Bo Pang and Lillian Lee. Thumbs up? Sentiment Classification using Machine Learning Techniques. // Proc. of the Conference on Empirical Methods in Natural Language processing (EMNLP). Philadelphia: ACL, 2002, pp.79-86.

19. J. Kamps, M. Marx, R. J. Mokken, and M. D. Rijke. Using WordNet to measure semantic orientation of adjectives. // Proceedings of LREC-04, 4th International Conference on Language Resources and Evaluation, volume IV, pages 1115–1118, Lisbon, PT, 2004.

20. Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. // HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing, pages 347–354, Morristown, NJ, USA. Association for Computational Linguistics

21. Stone, Dunphry, Smith and Ogilvie. Harvard General Inquirer lexicon, 1966 // http://www.wjh.harvard.edu/~inquirer/

22. Pak, A., and Paroubek, P. Twitter as a corpus for sentiment analysis and opinion mining. // Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10), Valletta, Malta, 2010

23. David D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. // Proc. of the European Conference on Machine Learning (ECML), p.4-15. Invited talk. 1998

24. Pedro Domingos and Michael J. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. // Machine Learning, 29(2-3):p.103-130. 997

25. Thorsten Joachims. Text categorization with support vector machines: Learning with

many relevant features. // Proc. of the European Conference on Machine Learning (ECML), pages 137-142, 1998

26. B. Pang and L. Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. // Proceedings of ACL-04, 42nd Meeting of the Association for Computational Linguistics, pages 271-278, Barcelona, ES, 2004.

27. H. Yu and V. Hatzivassiloglou. Towards answering opinion questions: Separating facts from opinions and identifying \the polarity of opinion sentences. // Proceedings of EMNLP-03, 8th Conference on Empirical Methods in Natural Language Processing, p.129-136, 2003.

28. S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. // Proceedings of the Seventh International Wide Web Conference, Brisbane, Australia, - Elsevier, 1998.

29. L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford University, Stanford, CA, 1998.

30. Kleinberg J. Authoritative sources in a hyperlinked environment, 1999

31. Ландэ Д.В.,Снарский А.А.,Безсуднов И.В. Интернетика. Навигация в сложных сетях: модели и алгоритмы. - Либроком, 2009. - 264 с. - ISBN 978-5-397-00497-8

32. Leo J.G., Jonathan R.P. Discrete Calculus. Applied Analysis on Graphs for Computational Science. - Springer, 2010. - 366 p. - ISBN 978-1-84996-289-6

33. Scime A. Web Mining: Applications and Techniques. - Idea Group Inc., 2005. - 433 p. - ISBN 1591404150

34. Chau, M., & Xu, J. (2007). Mining communities and their relationships in blogs: A study of online hate groups. // International Journal of Human – Computer Studies, 65(1), p.57–70.

35.Martin, J. Blogging for dollars. Fortune Small Business, 15(10), p. 88–92, 2005.

36. https://support.twitter.com/articles/20169341- 20.11.2012

37.http://www.liveinternet.ru/stat/vkontakte.ru/index.html?period=week&id=8&report=index.html%3Fperiod%3Dweek 17.01.2013

38. http://www.dev.twitter.com 22.02.2013

39. http://vk.com/developers.php?oid=-1&p=newsfeed.search 22.02.2013

40. Alec Go, Lei Huang, Richa Bhayani. Twitter sentiment analysis, CS224N – Final project report, June 6, 2009.

41.Alec Go, Richa Bhayani, Lei Huang. Twitter Sentiment Classification using Distant Supervision. 2009.

42. http://twittratr.com 16.12.2012

43. http://pythonhosted.org/pymorphy/ 18.01.2013

44. Chetviorkin I.I., Braslavski P.I., Loukachevitch N.V. Sentiment analysis track at ROMIP 2011 // http://www.cir.ru/SentiLexicon/83.pdf 11.03.2013

45. Лукашевич Н.В., Четверкин И.И. Извлечение и использование оценочных слов в задаче классификации отзывов на три класса. // “Вычислительные методы и программирование”, 2011, т.12, стр.73-81.

46. http://en.wikipedia.org/wiki/Discounted_cumulative_gain 18.02.2013

47. Кийко А.С. Ранжирование в информационно-поисковых системах на основе социальных сервисов. - М.: МГУ, 2011, - 41с.

48.http://nltk.googlecode.com/svn/trunk/doc/api/nltk.classify.naivebayes.NaiveBayesClassifier-class.html 30.03.2013

49. Shannon C. E. A Mathematical Theory of Communication // Bell System Technical Journal. — 1948. — V. 27. — P. 379—423, 623—656.