Московский государственный университет
имени М. В. Ломоносова

Факультет Вычислительной Математики и Кибернетики

Кафедра Математических Методов Прогнозирования

ДИПЛОМНАЯ РАБОТА СТУДЕНТА 517 ГРУППЫ

«Иерархическая классификация текстов»

Выполнила студентка 5 курса
517 группы:

Токарева Евгения Игоревна

Научный руководитель:
доцент кафедры ММП, д.ф.-м.н.

Дьяконов Александр Геннадьевич

Заведующий кафедрой
Математических Методов
Прогнозирования
академик РАН _____ *Ю. И. Журавлёв*

К защите допускаю
«____» _____ 2010 г.

К защите рекомендую
«____» _____ 2010 г.

Москва, 2010

Содержание

Введение

С ростом текстовой информации в электронном виде задача автоматической классификации текстов приобретает все большую актуальность. Например, такая задача возникает при автоматической обработке новостного потока и распределении текстов-новостей по каталогам. Для удобства пользователей каталоги организованы в иерархическую структуру: каталог состоит из нескольких подкаталогов и т.д.

После преобразования документов в векторный вид, данная задача может решаться методами машинного обучения. Сейчас TextMining активно развивается: ведутся исследования, запускаются проекты и конкурсы на выявление лучших по точности алгоритмов. Тем не менее, качество алгоритмов автоматической классификации оставляют желать лучшего. Так, в международном проекте Large Scale Hierarchical Text classification (LSHTC) Pascal Challenge [1], проводившемся осенью 2009 года, результаты наилучшего алгоритма по доле правильно классифицированных документов (14) – 0.467632. Наилучшее значение макро F-меры (15) 0.35494.

Причина таких результатов в специфике задачи: классификация проводится по большому числу классов (порядка 1000), в среднем на класс приходится 5.5 документов обучающей выборки, к некоторым классам отнесены только 2 документа выборки.

В проекте LSHTC были предложены реальные данные, которые являются частью Открытого Каталога (ODP, Open Directory Project) [2]. Этот факт позволяет расценивать закономерности, выделенные при исследовании алгоритма, как общие закономерности при автоматической классификации текстов, а не только как решение конкретной задачи. Данные проекта положены в основу настоящей дипломной работы.

Алгоритм иерархической классификации текстов состоит из нескольких блоков: предобработка входных данных, метод классификации, способ учета иерархии. По отдельности все блоки многократно описаны и являются довольно простыми. Успешный, качественный алгоритм – это удачно подобранные все части алгоритма. Конечный результат зависит от предобработки входных данных и учета иерархии не в меньшей степени, чем от метода машинного обучения.

Целью данной дипломной работы является исследование методов предобработки данных и способов модификации стандартных алгоритмов для задачи иерархической классификации текстов.

В итоге в MATLAB был разработан пакет прикладных программ, в котором реализованы функции загрузки и выгрузки данных, алгоритмы классификации, функции предобработки признакового пространства, модуль для проведения экспериментов.

В ходе экспериментов был выявлен ряд закономерностей, позволяющих решить задачу с большей точностью.

В конце работы будет предложен простой метод решения, который улучшает качество классификации по сравнению с лучшими алгоритмами проекта LSHTC. Доля правильно классифицированных документов (14): 0,51346, макро F-мера (15): 0,37669.

Ниже дается более конкретная постановка задачи и описание исходных данных. Также подробно рассматриваются особенности задачи. Далее приведен обзор существующих методов решения задачи классификации текстов и метрики оценки качества алгоритмов. В конце обзора описаны наилучшие алгоритмы-участники проекта LSHTC.

Затем описываются проведенные эксперименты по поиску новых эффективных алгоритмов. Дается обзор закономерностей, выявленных в ходе решения задачи. В конце приводится метод, позволяющий решить задачу наиболее эффективно.

Постановка задачи

Имеется множество документов D (обучающая выборка) и множество классов C, организованных в иерархию, каждому документу из D приписан один класс из C. Требуется на основе этих данных построить процедуру автоматической классификации текстов.

Структуру классов можно представить в виде дерева, причем классификация проходит только по листьям этого дерева. В литературе по TextMining вместо терминов «класс» и «классификация» иногда используют «рубрика» и «рубрикация» [10, 13].

Качество полученной процедуры оценивается на основе тестовой выборки, в которую так же входят документы, для которых заранее известны соответствующие им рубрики. Документы из тестовой выборки подаются на вход классификатору, результаты сравниваются с заранее известными значениями. Функции оценки качества классификации будем называть метриками [10]. Метрики качества классификации подробнее описаны ниже.

Векторное представление документа

Документ, как последовательность символов, является плохим объектом для классификации. Обычно документы преобразуются в векторное представление в пространстве $\mathbb{R}^n$ (называемое также пространством признаков), так как большинство алгоритмов машинного обучения работает именно в нем.

Отображение документа в пространство признаков – задача сложная и неоднозначная, ее подробное рассмотрение выходит за рамки данной работы. Тем не менее, для лучшего понимания данных, над которыми будет производиться дальнейшая работа, рассмотрим классический подход к преобразованию документов.

Метод основан на предположении, что попадание документа в тот или иной класс зависит от слов, входящих в данный текст. Преобразование заключается в том, что каждому слову, которое встречается в каком-либо документе, соответствует определенная координата в пространстве признаков. Значение этой координаты конкретного документа – это количество слов, встречающихся в документе. Для слова, которое не встречается в документе, значение соответствующей координаты равно нулю.

Количество всех слов, встречающихся в документах огромно, при этом многие из них не несут смысловой нагрузки, другие имеют синонимы.

Для уменьшения размерности и увеличения информативности используются различные техники еще на этапе приведения документа к векторному виду. Во-первых, составляется список, так называемых «стоп-слов» (предлоги, союзы и т.п.), эти слова не включаются в признаковое пространство. Если решается задача рубрикации web-страниц, то текст также нужно очистить от элементов управления, оформления и прочего «мусора».

Во-вторых, используется нормализация слов: все слова в документе приводятся к начальной форме, а иногда все синонимы заменяются на одно слово. Для английского языка задача нормализации решается проще, чем для русского: обычной процедурой является отсечение окончания слова (stemming). В

русском языке возникают проблемы многозначности слов или неоднозначности начальной формы. В виду особенностей используемых в данной работе данных, эти вопросы рассматриваться не будут.

Отдельно стоит отметить, что при приведении к векторному виду, признаком может быть не отдельное слово, а группа слов-синонимов. Другие варианты – это устойчивые словосочетания, имена собственные и т.п. Все такие признаки являются некоторой элементарной, неделимой смысловой единицей текста, по аналогии с [11] будем называть их **термами.**

При работе алгоритмов машинного обучения уже не так важно, что конкретно прячется под тем или иным термом, он становится одним из признаков.

Обозначения.

$C = \{c_1, c_2, \dots, c_{|C|}\}$ – набор классов, по которым производится классификация, может иметь иерархическую структуру;

$D = \{d_1, d_2, \dots, d_{|D|}\}$ - обучающая выборка;

$d_j = (t_1, t_2, \dots t_f, \dots, t_T)$ – векторное представление документа;

T – количество слов

V – словарь, множество слов.

Описание решаемой задачи и данных

Данные, используемые в работе, являются данными международного проекта по исследованию методов иерархической классификации текстов Large Scale Hierarchical Text classification (LSHTC) Pascal Challenge[1]. Проект проводился осенью 2009 года.

Особенности задачи: большой объем данных; признаки могут быть зависимыми и распределенными по-разному; иерархия классов сложная и разветвленная.

Данные проекта являются частью Открытого Каталога (ODP, Open Directory Project) [2]. Задание состоит из четырех блоков данных, данные в блоках частично повторяются. Первый блок является наибольшим и базовым, остальные три содержат дополнительную информацию о классах и требуют другого подхода к решению, нежели базовый блок. В данной работе рассматривается только базовый вариант задания.

В каждом блоке все данные разбиты на три множества: обучение (train), валидация (validation), контроль (test). Первые два множества используются для обучения, третий – тестовый. Так как организаторы проекта заранее сделали такое разбиение, то оно и используется без изменений в данной работе в различных экспериментах.

Документы – это web-страницы в векторном представлении. Преобразование к векторному представлению происходит по стандартной цепочке: предобработка, обрезание окончаний, избавление от мусорных элементов и стоп-слов.

Отдельным файлом прилагается иерархическая структура классов, которая представляет собой дерево. Документы могут принадлежать только «листовому» классу в иерархическом дереве. Каждый документ может принадлежать только одному классу.

Признаковое пространство состоит из 55766 термов. Классификация проходит по 1139 классам. Всего в иерархии классов 2387, глубина дерева – 5.

Объем выборок: train – 4463, validation – 1860, test – 1858.

Для проведения экспериментов множества train и validation были объединены в одну обучающую выборку, множество test составило контрольную выборку.

Особенности задачи

Задача иерархической классификации текстов может быть решена любым алгоритмом машинного обучения в силу векторного представления объектов. Обучающие алгоритмы не являются специфичными для задачи классификации текстов.

К сожалению, такой общий подход, без предварительной обработки входных данных, дает очень плохие результаты. Дело в том, что задача обладает рядом особенностей, которые нельзя проигнорировать.

1. *Разреженность пространства.*

Поскольку координаты вектора – это все слова, которые хот раз встречались хоть в одном документе, то в каждом конкретном векторе большинство признаков остаются нулевыми

2. *Высокая размерность пространства.*

Чем больше документов входит в выборку, тем больше будет объем словаря и, соответственно, размерность пространства. Даже предварительное выбрасывание заведомо «шумовых» слов и группировка синонимов не сильно сокращает размерность. При этом многие слова слабо влияют на результаты рубрицирования (либо вообще не влияют). Высокая размерность пространства признаков может приводить к высокой вычислительной погрешности и низкой скорости работы алгоритмов. В связи с этим, следует стремиться уменьшить пространство признаков. Методы отбора признаков будут рассмотрены ниже.

3. *Нестатистический характер данных.*

Значения координат векторов в «сыром», начальном виде: сколько раз слово встречается в документе – являются довольно неинформативными. Ведь документы могут быть разного размера, с разным количеством слов, входящих в него. Тогда значения одного признака для разных документов будет несопоставимы.

Различные методы «взвешивания» признаков позволяют выявить важность того или иного признака для класса. Методы преобразования векторного пространства приведены ниже.

4. *Большой объем данных и сложная структура классов.*

Эта особенность может привести к высоким вычислительным затратам, а также затратам по объему памяти. При выборе алгоритма следует учитывать скорость его работы на большом объеме данных. Часто в алгоритмах используются дополнительные техники и модификации для увеличения скорости работы алгоритма.

Обзор методов классификации

В данном разделе будут описаны основные блоки алгоритма классификации текстов.

Предобработка векторного пространства

Предобработка входных векторов-документов позволяет учесть описанные выше особенности пространства и сделать признаки более информативными.

Веса признаков

«Взвешивать» признаки можно по-разному. Простейшим вариант является приведение к бинарному виду: 1, если терм в документе встречается и 0 иначе. На таких признаках, например, можно построить алгоритм Байеса [13]. Широкого распространения такой подход все же не получил, ибо бинарной информации зачастую оказывается недостаточно для качественной классификации.

Существует ряд более сложных преобразований, которые учитывают значимость признаков как для конкретного документа, так и для пространства в целом. После преобразования значения каждого признака показывает, какой вклад терм привносит в документ. Наиболее распространенным является преобразование $TF * IDF$ и различные его модификации. Остановимся на нем подробнее.

$TF * IDF$

Название преобразования уже является формулой, но требует пояснений. Как видно она состоит из двух множителей, рассмотрим каждый из них по отдельности.

TF (term frequency — частота слова) — отношение числа вхождения некоторого слова к общему количеству слов документа. Таким образом, оценивается важность терма t_i в пределах отдельного документа d_j.

$$tf_{ij} = \frac{t_i}{\sum_{l=1}^{T} t_l} \qquad (1)$$

Недостатком такого представления является то, что недооцениваются длинные документы, так как в них больше слов и средняя частота слов в тексте реже. Для борьбы с этим эффектом можно применить дополнительную нормализованную частоту:

$TF_{norm} = 0.5 + 0.5 * \frac{tf_{ij}}{atf}$, где atf – средняя частота терма в документе.

Также в литературе встречаются другие модификации TF [15]:

1. $1 + \log(tf_{ij})$;
2. $(1 + \log(tf_{ij}))/(1 + \log(mtf))$,
   где mtf – максимальная частота слова в документе;
3. $\log(tf_{ij} + 1)$. (2)

IDF (inverse document frequency — обратная частота документа) — инверсия частоты, с которой некоторый терм встречается в документах коллекции. IDF учитывает тот факт, что если терм встречается во многих документах множества, то он не может являться существенным критерием принадлежности документа рубрике и наоборот.

$$idf_i = \log\frac{|D|}{n_i} \qquad (3)$$

Здесь $|D|$– количество документов в обучающей выборке, n_i — количество документов, в которых встречается i-е слово.

Иногда n_i – это сумма числа вхождений i-го слова во все документы выборки.

Классическим вариантом $TF * IDF$ считается произведение (1) и (3).

Другой вариант представления IDF, используемый, например, в [8]:

$$idf_i = \left(\frac{|D|}{n_i}\right)^{\alpha}, \qquad \alpha \in [0,1] \tag{4}$$

Обозначения те же, что и в формуле (3).

$\boldsymbol{Maxstr}$ (maximum strength — максимальная сила) – альтернатива IDF. Менее распространенный, но неплохо себя зарекомендовавший способ посчитать значимость признаков. С вычислением этой величины автоматически происходит неявный отбор признаков, так как часто встречающимся во всех классах словам присваивается минимальный вес.

Формула для подсчета $Maxstr$ несколько сложнее формулы IDF, рассмотрим ее:

1. Регуляризованной долей документов, содержащих терм в некоторой коллекции документов, будем называть функцию:

$$\grave{p}(x,n) = \frac{x+\tau}{n+2\tau}$$

где $\tau = 0.5z^2_{\alpha/2}$, такое что $\Phi\left(z_{\alpha/2}\right) = \alpha/2$, Φ – распределение Стьюдента.

Авторы метода рекомендуют $\propto = 0.95$, что означает, что $\tau = 1.96$

2. Для каждого признака f и каждого класса c вычисляются 2 величины:

$\grave{p}_+ = \grave{p}(x,n)$ — x – это количество векторов, которые принадлежат классу c и имеют ненулевой признак f; n – количество документов, принадлежащих классу c.

$\grave{p}_- = \grave{p}(x,n)$ — x – это количество векторов, обладающие признаком f, но не входящие в класс c; n – количество векторов, не принадлежащие классу c.

3. Доверительный интервал имеет вид $(\underline{\grave{p}}, \overline{\grave{p}})$, где

$$\underline{\grave{p}} = \grave{p} - \tau\sqrt{\frac{\grave{p}(1-\grave{p})}{n+2\tau}}, \overline{\grave{p}} = \grave{p} + \tau\sqrt{\frac{\grave{p}(1-\grave{p})}{n+2\tau}}.$$

4. Назовем силой признака в классе следующую функцию:

$$str_{f,c} = \begin{cases} \log_2\left(2\frac{\underline{\grave{p}_+}}{\underline{\grave{p}_+} + \overline{\grave{p}_-}}\right), \text{если } \underline{\grave{p}_+} > \overline{\grave{p}_-} \\ 0, \text{иначе} \end{cases}$$

Получается, что признак f имеет ненулевой вес в классе c, если частота появления f в документах класса c больше, чем частота появления этого признака в документах всех остальных классов.

5. Максимальная сила вычисляется по формуле:

$$Maxstr(f) = \left(\max_{c \in C} str_{f,c}\right)^2$$

P.Soucy и G.W.Mineau [14] применили данные веса для создания статистической меры, конкурирующей с $TF * IDF$:

$$ConfWeight_{ij} = \log(tf_{ij} + 1) * Maxstr(i) \tag{5}$$

Результаты экспериментов для алгоритмов SVM, k-NN [14][9] показывают, что $ConfWeight$ дает более точные результаты, чем $TF * IDF$. В настоящей работе исследуется эффективность данной функции для алгоритмов, основанных на центроидных.

Как видно, $ConfWeight$ – это произведение одной из модификаций TF (2) и $Maxstr$. Поскольку обе формулы $TF * IDF$ и $ConfWeight$ похожи по своей сути, будем предполагать, что над ними можно

проводить одинаковые преобразования. Поэтому далее, описывая преобразования $TF * IDF$, будем подразумевать, что аналогичные преобразования можно проводить и с весами $ConfWeight$.

Преобразования, основанные на $TF * IDF$

Возможны различные модификации основной формулы $TF * IDF$. Будет ли лучше классический вариант, или одна из модификаций, зависит от конкретной задачи и может быть проверено только экспериментальным путем.

Взвешивание

$TF^p * IDF^{1-p}$
$p \in (0,1]$

Вводится дополнительный параметр p, который позволяет регулировать вклад множителей в конечный вес признака. Как вариант, возможно возведение в степень только одного из признаков. Степень варьируется от 0 до 1;

Нормировка

$$L_2: \frac{tf_{ij} idf_i}{\sqrt{\sum_{s=1}^{|T|} (tf_{sj} idf_i)^2}} \quad (6)$$

$$L_1: \frac{tf_{ij} idf_i}{\sum_{s=1}^{|T|} |tf_{sj} idf_i|} \quad (7)$$

Очевидно, что нормирование векторов можно проводить по различным нормам. Выше приведены только L_1 и L_2, как наиболее распространенные.

В статье [16] приводится подробное исследование различных подходов к выбору весов признаков. Результаты экспериментов, описанных в этой статье, показывают, что одной из лучших формул вычисления весов является (6) с классическими весами $TF * IDF$.

Преобразование признаков с помощью сглаживающих функций

Иногда важно, что каких-либо слов в документе много, но не важно, насколько больше, чем других часто встречающихся слов в том же документе.

Для учета этого факта иногда полезно применять возрастающие на положительной полуоси функции, которые асимптотически сходятся к какой-либо константе. В данной работе такие функции применялись к начальным данным, после чего полученные вектора подавались на преобразование $TF * IDF$ и т.п.

В работе применялись наиболее типичные функции:

$$f_1(x) = \frac{x}{x+1} \quad (8)$$

$$f_2(x) = \max(\ln x, 0) \quad (9)$$

Результаты экспериментов по сглаживанию больших значений приведены ниже.

Уменьшение размерности признакового пространства

Уменьшение размерности позволяет снизить вычислительную сложность алгоритма. При этом важно не ухудшить качество классификации. Предполагается, что ряд термов практически неинформативны. Тогда их удаление не ухудшит качества алгоритма. Неинформативные термы вычисляются по определенным правилам, которые будут рассмотрены ниже.

Иногда предполагается, что ряд термов зависит друг от друга. То есть они с большой вероятностью попадут в один класс или в один документ. Очевидно, что за счет таких признаков можно уменьшить размерность пространства без ухудшения точности алгоритма.

Например, одним из способов выделения зависимых признаков является кластеризация. Близкие термы объединяются в кластеры. Близость может определяться либо совместной встречаемостью в документе, либо совместной встречаемостью в классе. В итоге либо из каждого кластера выбирается один эталонный терм, а остальные убираются из признакового пространства, либо термы из одного кластера заменяются на один признак, представляющий из себя некоторую комбинацию термов кластера.

Уменьшение размерности может проходить локально или глобально. Локальность означает, что процедура проводится для каждой категории отдельно, затем оставшиеся слова объединяются в единое пространство. Глобальность подразумевает, что участие принимают сразу все документы обучающей выборки, без учета их принадлежности к той или иной категории.

Отбор неинформативных признаков

Существуют различные способы определения признаков, не влияющих на качество классификации. Разумно предположить, что если какой-либо терм практически не встречается в документах обучающей выборки, он не несет специфической информации, определяющей класс объекта. Когда терм, наоборот, встречается во всех документах и много раз, то он также будет нести мало полезной информации о принадлежности документа к тому или иному классу. Если все классы статистически независимы от какого-либо терма, то он также неинформативен. Такие простые правила, конечно, нуждаются в формализации. Приведем список некоторых возможных правил, которые позволяют определить наиболее ненужные признаки.

Не будем описывать стандартные процедуры отбора признаков, которые подойдут и в данном случае: поиск в глубину или в ширину, метод добавления и удаления, случайный поиск, генетический алгоритм и прочие [18]. Остановимся на методах, ориентированных на текст.

Итак, терм объявляется неинформативным, если он:

1. встречается во всех документах выборки меньше n раз;
2. имеет большое математическое ожидание $\mathcal{M}tf_i > a$ и маленькую дисперсию $\mathcal{D}tf_i < b$, то есть терм встречается одинаково часто во всех документах всех классов
3. имеет маленький информационный вес терма $MI(d,c)$ в рубрике:

$$MI(t_i,c) = \sum_{t_i\in\{0,1\}}\sum_{c\in\{0,1\}} P(t_i,c)\log\frac{P(t_i,c)}{P(t_i)P(c)}, \text{где}$$

$$P(t_i = 1) = 1 - P(t_i = 0) = \frac{n_i}{|D|} = \frac{кол-во\ документов, содержащих\ терм\ t_i}{кол-во\ всех\ документов};$$

$$P(c = 1) = 1 - P(c = 0) = \frac{|d_i \in c|}{|D|} = \frac{кол-во\ документов\ в\ классе\ c}{кол-во\ всех\ документов};$$

$P(t_i,c)$ – вероятность совместного распределения терма и класса

Если распределения терма t_i и класса c статистически независимы, то $MI(t_i,c) = 0$. Если же между встречаемостью терма t_i и класса c имеется строгая логическая зависимость, то $MI(t_i,c)$ – максимально. Метод сокращения размерности на основе выделения наиболее информационно-значимых слов применяется, например, в работе [17].

Классификация документов

Способы классификации

Одной из особенностей задачи является иерархическая структура классов. Это дополнительная информация, которую можно использовать, но можно и проигнорировать. Если алгоритм учитывает иерархию классов, он называется иерархическим (hierarchical), если не учитывает – плоским (flat).

Рассмотрим иерархический и плоский подходы подробнее. Возьмем некоторый метод машинного обучения Alg, который работает в векторном пространстве.

Если мы используем данный алгоритм как плоский, то все конечные классы, к которым могут быть приписаны документы, рассматриваются как равноправные. В простейшем случае иерархической классификации алгоритм Alg настраивается и потом работает на каждом уровне, выбирая наиболее вероятный путь до листа. Классификация выбирает только среди детей узла, в котором он работает.

Каждый из подходов обладает плюсами и минусами. Плоский классификатор, как правило, дает более точные результаты. Вероятность же ошибки иерархического классификатора выше из-за накапливания ошибок классификаторов узлов. С другой стороны, плоскому классификатору приходится работать со всем объемом обучающих данных и классифицировать сразу по огромному числу классов, что увеличивает вычислительные нагрузки. Плоский классификатор, как правило, работает гораздо медленнее иерархического.

Об этих особенностях пишут практически все исследователи, именно поэтому возникает необходимость придумывать какие-либо промежуточные варианты классификаторов, сочетающие в себе высокое качество и скорость работы. Подтверждение особенностей иерархического и плоского классификаторов можно будет наблюдать в результатах экспериментов, проведенных в рамках данной работы.

Вариантов модификации или совмещения иерархического и плоского классификаторов множество. Это может быть упрощение иерархической структуры или подсчет дополнительных очков по родительским узлам. Некоторые из них можно наблюдать в конкретных алгоритмах, представленных в проекте LSHTC. Эти алгоритмы описаны в отдельном разделе. Также некоторые варианты модификации иерархического алгоритма были реализованы в рамках данной работы, и их описание будет ниже.

Алгоритмы классификации

Практически все методы машинного обучения применимы к задаче классификации текстов. Часто используются и хорошо себя зарекомендовали SVM, Метод Байеса, Метод k-ближайших соседей, Нейронные сети, Деревья решений, Классификатор Роше. Об этих алгоритмах, в свете задачи классификации текстов, можно, например, прочитать в [11]. На алгоритме Роше остановимся подробнее.

Классификатор Роше

Классификатор Роше (Rocchio) — один из самых простых методов классификации. Для каждой категории вычисляется взвешенный центроид по формуле:

$$g_c = \frac{1}{|v_c|}\sum_{d\in v_c} d - \gamma\frac{1}{|\overline{v_{c,k}}|}\sum_{d\in \overline{v_{c,k}}} d \qquad (10)$$

v_c – множество документов, принадлежащих классу, $\overline{v_{c,k}}$ – k документов, не принадлежащих классу и наиболее близкий к центроиду $\frac{1}{|v_c|}\sum_{d\in v_c} d$, γ – параметр, указывающий относительную важность учета отрицательных прецедентов.

Обрабатываемый документ приписывается к тому классу, к центроиду которого он ближе всего расположен.

Данный метод обладает полезной особенностью: взвешенные центроиды можно быстро пересчитать при добавлении новых отрубрицированных примеров. Эта особенность полезна, например, в задаче адаптивной фильтрации, когда пользователь постепенно указывает системе, какие документы выбраны правильно, а какие нет. В ответ система может уточнить результаты, учитывая новые отрубрицированные документы.

Центроидный алгоритм

В данной работе подробно исследуется частный случай классификатора Роше при $\gamma = 0$ – Центроидный алгоритм. Центроидный алгоритм прост в реализации, дает заведомо неплохие результаты и позволяет сконцентрировать внимание не предобработке признакового пространства и различных способах классификации.

Функции близости

Для оценки близости документов к центрам классов возможны различные метрики в данной работе использовались наиболее популярные:

1. Метрика Городских кварталов

$$Abs(x,y) = \sum_{i=1}^{T} |x_i - y_i|; \quad (11)$$

2. Евклидова метрика

$$Sqr(x,y) = \sqrt[2]{\sum_{i=1}^{T} (x_i - y_i)^2}; \quad (12)$$

3. Косинус угла между векторами

$$Cossim(x,y) = 1 - cos(x,y) = 1 - \frac{\sum_i x_i * y_i}{\sqrt{\sum_i x_i^2 \sum_i y_i^2}}. \quad (13)$$

Метрики качества классификации

Для оценки качества алгоритмов, построенных в результате экспериментов, будем применять метрики, по которым оценивались алгоритмы проекта LSHTC [1]. А именно, доля правильных ответов, макро-точность, макро-полнота и макро-F-мера.

Пусть контрольная выборка состоит из M объектов. Из них m объектов классифицируются правильно. Тогда **доля правильных ответов** (accuracy) вычисляется по формуле:

$$A = \frac{m}{M} \quad (14)$$

Данная оценка является одной из простейших и не всегда отражает реальное качество. Далее будут описаны более показательные оценки качества.

Рассмотрим один класс и определим для него понятия точность и полнота. Пусть v – множество документов, принадлежащих классу и u – множество документов, приписанных классу алгоритмом.

Полнота (recall) классификации документов по классу вычисляется как отношение количества документов, правильно приписанных к классу к общему количеству документов, относящихся к данному классу:

$$r(u) = \frac{|u \cap v|}{|v|}.$$

Точность (precision) классификации документов по классу вычисляется как отношение количества документов, правильно приписанных к классу к общему количеству документов, приписанных к данному классу:

$$p(u) = \frac{|u \cap v|}{|u|}.$$

F-мера (F-measure) объединяет оценки точности и полноты в одну:

$$F(u) = \frac{2 * p(u) * r(u)}{p(u) + r(u)}$$

Если $p(u) = 0$ или $r(u) = 0$, то $F(u) = 0$.

Для получения сводных оценок качества классификации в целом, по всем классам вводится макро усреднение характеристик по всем рубрикам.

$$\text{Макро} - p = \frac{1}{|C|}\sum_{i=1}^{|C|} p(u_i);$$

$$\text{Макро} - r = \frac{1}{|C|}\sum_{i=1}^{|C|} r(u_i);$$

$$\text{Макро} - F = \frac{1}{|C|}\sum_{i=1}^{|C|} F(u_i). \quad (15)$$

Лучшие алгоритмы проекта LSHTC

Проект «Large-scale Hierarchical Text Classification» проходил осенью 2009 года. В данном разделе представлен обзор лучших алгоритмов проекта.

Улучшенный иерархический SVM (Improved Hierarchical SVMs for Large-scale Hierarchical Text Classification Challenge) [3]

Суть алгоритма: решающее дерево из SVM. Используется готовая реализация алгоритма, пакет SVM^{stuct} [19]. В SVM используется линейное ядро. При настройке для каждого узла подбирается оптимальный порог ошибки.

Подготовка

1. Преобразование признаков $TF * IDF$ (1, 3);
2. Упрощение иерархии классов;

Проходим по дереву сверху вниз. Если выявляется узел, количество потомков которого меньше порога и эти потомки не являются листьями, то «дети» потомков напрямую связываются с рассматриваемым узлом, а потомки убираются из иерархии совсем.

Цели: уменьшить глубину дерева и тем самым уменьшить накапливание ошибки; увеличить количество прецедентов во внутренних узлах для более точной классификации.

Результатом упрощения является увеличение точности и глобальное уменьшение сложности настройки алгоритма. С другой стороны, увеличивается локальная сложность настройки (для каждого конкретного узла) и при этом возможно превышение возможностей компьютера.

В итоге упрощения исследователи получили 2 уровня: 311 внуков корня становятся его детьми, а все остальные – их дети.

Обучение

1. Поиск оптимальных параметров для SVM в каждом узле.

Для каждого алгоритма находился оптимальный уровень ошибки алгоритма с помощью скользящего среднего на подмножествах множеств train и validation.

2. Настройка решающего дерева.

Настройка на объединении множеств train и validation.

При настройке решающего дерева найденные уровни ошибок умножаются на 1,2.

Результаты

A	Макро-r	Макро-p	Макро-F
0,44329	0,33772	0,30341	0,31965

Двухуровневая классификация (Deep Classifier for Large Scale Hierarchical Text Classification)[4]

Суть алгоритма: Классификация проходит в 2 этапа: первый – этап поиска (Search Stage), на котором с помощью метода ближайшего соседа отбирается подмножество наиболее вероятных классов. На втором этапе, классификационном (Classification Stage), работает Наивный Байесовский алгоритм, проводя классификацию только среди отобранных на предыдущем шаге классов. На каждом этапе классам приписываются некоторые очки. Комбинация этих очков указывает на наиболее вероятный класс.

Подготовка

1. Составление векторов категорий. Каждой категории приписывается 1 вектор – сумма векторов, относящихся к данной категории
2. Составление векторов родителей. Каждый узел, предшествующий листу получает вектор как сумму векторов своих детей.
3. Преобразование векторов. Каждый элемент вектора заменяется на $tf^{0,625} * idf$

Первый уровень (Search Stage)

Вычисляется косинус угла между классифицируемым вектором и векторами категорий. Значение косинуса – это очко категории. По ним отбираются N (N= 150) наиболее близких категорий. Далее вычисляем близость объекта к векторам родительских категорий объектов, вошедших в N лучших. Отбираем K (K = 10) ближайших.

По итогу первого шага остается K классов с очками вида:

$rs_c = s_c * s_{p(c)}^{0,52}$, где s_c - очко от близости с вектором категории, $s_{p(c)}^{0,52}$ - очко близости к родительскому вектору данной категории.

Второй уровень (Classification Stage)

Классификация проводится только по выбранному на предыдущем этапе подмножеству классов. Применяется алгоритм Наивного Байеса.

Значения априорной вероятности появления слова в категории вычисляется по формуле:

$p(t_i|c) = \frac{N_c^i + \alpha}{N_c + V\alpha}$, где N_c^i означает, сколько раз слово *i*, встречается в категории *c*, N_c – сумма всех слов в категории *c*. *V* – размер текущего словаря, α – параметр.

Размер словаря меняется от теста к тесту. Для каждого лучше подбирать свой параметр α. В таблице приведены наилучшие значения параметров для различных по длине словарей:

Размер словаря	Уровень разброса (distribution rate)	Оптимальное значение α
1- 2000	2.95%	0.36
2001-3500	19.1%	0.27
3501-6000	38.0%	0.16
6001-10000	18.4%	0.14
10001-∞	1 21.5%	0.08

Решающее правило

Итоговые очки s_c классов вычисляются как комбинация очков, полученных на двух предыдущих шагах.

$s_c = rs_c^{0,12} * ns_c$, где ns_c - очки с классификационной стадии

Результаты

A	Макро-r	Макро-p	Макро-F
0,43168	0,34431	0,30455	0,32321

Плоские техники классификации (Large-Scale Many-Class Prediction via Flat Techniques) [6]

Суть алгоритма: используется плоский классификатор. W – весовая матрица, в которой элемент w_{ij} – вес i-го признака j-го класса. d относится к классу c_d, если $c_d = \text{argmax}_c\, d^T W$. То есть c_d – класс, соответствующий максимальному элементу в векторе, получаемом при умножении.

Для подбора оптимальных весов – элементов матрицы W предлагаются 2 алгоритма настройки OOZ (Online Optimization of Z variables) – он-лайн оптимизация слабых переменных и PA (Passive – Aggressive) – Пассивно – Агрессивный. Алгоритмы будут описаны ниже.

Подготовка

1. Преобразование векторов. Каждый элемент вектора заменяется на $TF * IDF$, нормируются по L^{12}-норме.

Метод настройки OOZ

Предполагается, что все элементы матрицы W неотрицательны.

Из обучающей выборки выбирается произвольный вектор и умножается на матрицу W, в результате получаются очки по каждому классу. На основе этих очков происходит модификация весовой матрицы.

Инициализируем W нулевой матрицей. Пусть s_{c_d} – очки класса c_d, которому принадлежит вектор. Матрица будет обновлена, если $\exists c' \neq c, s_{c'} \geq s_{c_d} - \delta$, где δ – величина отступа (здесь $\delta = 0.005$). Все классы c' будем называть негативными.

Далее по заданному алгоритму, у части негативных классов убавляются положительные веса с признаков и перераспределяются по позитивному классу. Если у негативных классов веса всех признаков нулевые, то позитивному классу добавляются инициализаторы (например, 1).

Чтобы оставить матрицу разреженной по строкам, алгоритм всегда оставляет ненулевыми только *m* наибольших значений признаков ($m = 100, 500, 1000, 2000$). Пи обновлении признаков w_{ij}, общая величина одного признака увеличивается не более чем на $w_{ij}\beta$. В экспериментах полагали $\beta = 0.01$. Уменьшение признаков происходит у первых *k* негативных классов ($k = 15$). Чем меньше *k*, тем быстрее алгоритм, но качество немного ухудшается.

Псевдокод обучения OOZ

Вход:
d- документ из обучающей выборки;
c_d- класс документа x;
$\tilde{C}_d$- множество всех классов и очки по ним;
δ- отступ;
β – параметр масштабирования весов признаков

$\boldsymbol{OOZ\_Update}(d, c_d, \tilde{C}_d, \delta, \beta)$
{
$\beta = \min\left(\delta/_2, \beta\right)$; /*по возможности уменьшаем обучающую интенсивность*/
$S = \boldsymbol{Compute\_Deductions}(k, s_{c_d} - \delta, \beta, \tilde{C}_d)$;
$R = S$; /*R – это остатки*/
$\forall t_f \in d, \boldsymbol{Shift\_Connection\_Weights}(f, t_f, t_f * \beta, S, R, c_d)$;
}

$\boldsymbol{Compute\_Deductions}(k, s_{min}, \beta, \tilde{C}_d)$
{
$Y = \emptyset, S = \emptyset$; /* S – ассоциативный массив */
$\forall c \in \tilde{C}_d, if\ s_c > s_{min}, Y = Y \cup \{(c, s_c)\}, S(c) = 0$;
$s = sort(очки\ Y\ в\ порядке\ убывания)$;
$cum = 0; n = 1; i = 0$;
$loop\ while\ (cum < \beta)$ /* обновление вычитаемых значений */
{
$\Delta = \beta - cum$;
$if\ (i < k) \Delta = \min(\Delta, s[i] - s[i+1])$;
$\forall c, s_c > s[i+1], S(c) = S(c) + \frac{\Delta}{n}$;
$cum = cum + \Delta; n = n + 1; i = i + 1$;
}
return S;
}
/ Сдвиг весов признаков с негативных классов */*
$\boldsymbol{Shift\_Connection\_Weights}(f, t_f, r, S, R, c_d)$
{
$boost = 0$;
/ уменьшаем веса признаков в проблемных классах */*
$loop\ over\ c \in C, where\ w_{f,c} > 0\ and\ R(c) > 0$

{

$if\ (r == 0) break\ the\ loop;$

*/*Пока нет других условий выхода*/*

$h = \min\left(\frac{R(c)}{t_f}, w_{f,c}, t_f S(c), r\right);$

$boost = bost + h;\ w_{c,f} = w_{c,f} - h;$

$R(c) = R(c) - ht_f; r = r - h;$

}

/ поощрение истинного класса*/*

$boost = \min(w_{f,c_0}, r) + boost;$

$w_{f,c_d} = w_{f,c_d} + boost;$

}

Обучение PA

Прототипами классов будем называть столбцы матрицы W w^c.

Очки класса с могут быть вычислены по формуле: $s_c = d^T * w^c$.

Алгоритм PA обучает прототипы классов, решая следующую оптимизационную задачу:

$$w_{t+1} = arg\ \min_w \frac{1}{2}||w - w_t||^2,\ \mathcal{L}(w; (d, c_d)) = 0;$$

Алгоритм «пассивно» принимает решения, чьи потери нулевые, в это же время, он «агрессивно» создает новый прототип, стараясь его сделать как можно ближе к предыдущему.

Потери определяются как максимальное различие (вредом, violation) между негативным и позитивным классами. Потери будут положительной величиной до тех пор, пока исход классификации не превышает отступа в 1. Решение можно найти с помощью Лагранжиана.

Нужно заметить, что при обновлении данных, обновляются только 2 прототипа, что упрощает алгоритм.

Псевдокод обучение PA

Вход:

d- документ из обучающей выборки;

c_d- класс документа x;

$\tilde{C}_d$- множество всех классов и очки по ним;

C- параметр агрессивности.

$\boldsymbol{PA\_Update}(d, c_d, \tilde{C}_d, C)$

{

$if\ \mathcal{L} = 1 - s_{c_d} + s_{c'} \geq 0, where\ s_{c'} = \max_{c \neq c_d} s_c$, то производим обновление

{

$\tau = \frac{\mathcal{L}}{||d||^2 + \frac{1}{2C}};$

$w_{t+1}^{c_d} = w_t^{c_d} + \tau d, w_{t+1}^{c'} = w_t^{c'} - \tau d$

}

}

Результаты экспериментов

PA на Данных 1 за 10 итераций достигает точности 0.477

OOZ с различными параметрами, в лучшем случае достигает точности 0.488 за 60 итераций

Оба алгоритма работают довольно долго и занимают достаточно много места. Чтобы улучшить ситуацию, производился отбор признаков (каким образом – не рассказывается).

В итоге на проект был выдвинут алгоритм PA, на рассматриваемых данных получены следующие результаты:

A	Макро-r	Макро-p	Макро-F
0,46322	0,38001	0,33298	0,35494

Иерархический классификатор, использующий центры классов (Hierarchical Centroid-based Classifier for Large Scale Text Classification) [8]

Суть алгоритма: вычисляется очки классов, основанные на близости объекта к центрам классов. При этом используется информации об иерархии классов.

$\hat{c} = arg\max_c(w_c + w_{p(c)})d$, где w_c – центр масс класса c, $w_{p(c)}$ – центр масс узла-родителя класса c.

Обучение

Центры масс w_c и $w_{p(c)}$ вычисляются аналогично. Во втором случае прецеденты – это документы, относящиеся ко всем классам-наследникам. Рассмотрим вычисление центра класса только для w_c.

$w_c = b * w_c^{TF} + (1 - b) * w_c^o$, где

$w_c^{TF} = \sum_{d \in c} d^{TF}$: все элементы векторов заменяются на TF. Среднее классов вычисляется как сумма преобразованных векторов, входящих в данный класс.

$w_c^o = IDF * \sum_{d \in c}[d > 0]$: все объекты приводятся к бинарному виду (1, если слово встречается в документе, 0 иначе), далее эти объекты суммируются и умножаются на IDF.

В данном алгоритме $IDF = \left({}^{|Tr|}/_{n_c}\right)^{\alpha}$ (4), $\alpha \in [0,1]$

В конкретной реализации алгоритма авторами были выбраны параметры: $b = 0.4$, $\alpha = 0.1$

Результаты

A	Макро-r	Макро-p	Макро-F
0,44306	0,36286	0,31413	0,33674

Адаптация весов Soucy&Mineau (An Adaptation of Soucy and Mineau weighting for Large Scale Text Classification) [9]

Суть алгоритма: взвешенный метод ближайшего соседа. Для предобработки признакового пространства используется преобразование $ConfWeight$ (5) вместо стандартного $TF * IDF$.

Подготовка

1. Каждый элемент вектора заменяется на $ConfWeight$;
2. Функция расстояния $Cossim$ (13);
3. Выбираются веса признаков. Например, возможны варианты:

${}^{1}/_{(1+m)}, 1 - m, {}^{1}/_{m}, (1-m)^2, (1-m)^3, (1-m)^4, e^{1-m}$.

Ускорение работы алгоритма

Поиск наиболее близких векторов среди всех векторов может проходить довольно долго. Для ускорения процесса используется следующий прием. В классифицируемом векторе выделяются υ основных (наибольших) признаков. Вычисление расстояния между ним и другим вектором происходит только тогда, когда у второго вектора хотя бы один из υ выделенных признаков ненулевой.

Детализация параметров алгоритма

Параметр	Значение
τ ($\propto = 99{,}1$)	2,625
Количество соседей k	6
Количество основных признаков υ	3
Весовая функция	$(1-d)^3$

Результаты

A	Макро-r	Макро-p	Макро-F
0,40232	0,30781	0,26803	0,28655

Классификация с помощью резонанса ассоциативной сети (Classification by resonance in an associative network) [5]

Суть алгоритма: Строится ассоциативная сеть – ориентированный граф, вершинами которого являются термы и классы. На этапе обучения вычисляются веса ребер. На этапе классификации документ рассматривается как активатор. Активация распространяется по сети и вычисляется резонанс между документом и классами.

Обучение

На этапе обучения строится ассоциативная сеть: все узлы термов соединяются с узлами классов и наоборот. Все связи ориентированные и взвешенные.

Для определения весов ребер необходимо понять частоту совместного появления класса и терма. Степенью принадлежности (degrees of membership) терма или класса A к документу d будем называть $\mu_d(A) \in [0,1]$.

Веса ребер определяются следующим образом:

$$W_{AB} = \frac{\sum_{d \in Tr} \mu_d(A)\mu_d(B)}{\sum_d \mu_d(A)}$$

Если A – это признак, а B – это класс, то W_{AB}- это вероятность появления документа из класса B среди всех документов, содержащих признак A.

Классификация

Класс документа определяется по формуле:

$\hat{c} = arg\max_c R(d,c)$, где

$$R(d,c) = \sum_{t \in d} a(t) * \frac{W_{ct} * \sum_{t' \in d} a(t')W_{tc}}{\sum_{t \in V} W_{ct}} -$$

степень резонанса между документом d и классом c после активации сети.

$a(t)$ – значение признака t в документе d, V – множество всех термов (словарь).

Формула $R(d,c)$ представляет собой скалярное произведение между вектором – документом и вектором, который получается в результате активации сети.

Модификация алгоритма

Можно переписать формулу в следующем виде:

$$R(d,c) = \left[\frac{\sum_{t\in d} a(t)W_{ct}}{\sum_{t\in V} W_{ct}}\right] * \left[\sum_{t\in d} a(t)W_{tc}\right]$$

В формуле можно выделить 2 фактора. Первая скобка отвечает за путь сверху вниз от класса до документа, определяет, насколько типичен документ для класса. Вторая скобка, отвечающая за путь снизу вверх от документа к классу, вычисляет степень точности попадания документа в класс.

Можно добавить возможность регулировать важность этих двух факторов относительно друг друга, вводя новый параметр *p:*

$$R(d,c) = \left[\frac{\sum_{t\in d} a(t)W_{ct}}{\sum_{t\in V} W_{ct}}\right]^{p} * \left[\sum_{t\in d} a(t)W_{tc}\right]^{2-p}$$

Детализация параметров алгоритма

Параметры	Значения
Степень принадлежности $\mu_d(c)$ и $\mu_d(t)$	$\mu_d(c) = \begin{cases} 1, если\ d \in c \\ 0\ иначе \end{cases}$ $\mu_d(t) = \frac{nb_{ooc(d,t)}}{nb_{ooc(d,t)} + 2}$ где $nb\_ooc(d,t)$ – количество вхождений терма t в документе d.
Функция предобработки $a(T)$	$TF * IDF$
Отбор признаков	Убираются признаки, у которых $idf < 0{,}3$
Параметр значимости пути p	1,4

Результаты

A	Макро-r	Макро-p	Макро-F
0,291714	0,185995	0,162741	0,217004

Упрощение иерархии, SVM и Ленивый классификатор(Improving Hierarchical SVMs by Hierarchy Flattening and Lazy Classification) [7]

Суть алгоритма: Используется стандартный SVM. Для улучшения классификации используется техника упрощения иерархии и техника ленивого классификатора (Lazy Classification).

Упрощение иерархии (Hierarchy Flattening)

Используем начальную иерархию классов до k-го уровня. Все классы, находящиеся ниже этого уровня, становятся непосредственными детьми узлов k-го уровня. То есть, после k-го уровня классификатор становится плоским. В каждом внутреннем узле классификатор настраивается отдельно.

Ленивый классификатор(Lazy Classification)

Данная техника представляет собой 2х уровневую классификацию: выбор подмножества наиболее вероятных классов, затем классификация среди этого подмножества.

При классификации на каждом уровне иерархии отбирается n наиболее вероятных узла (которые получили наибольше количество очков). На уровне ниже, внутри каждого из выбранных узлов, опять отбирается n наиболее вероятных узла. В итоге получается подмножество классов, среди которых работает уже плоский классификатор.

Детализация параметров алгоритма

Параметры	Значения
Уровень упрощения k	2
Количество наиболее вероятных классов n	8

Результаты

A	Макро-r	Макро-p	Макро-F
0,41522	0,36809	0,29675	0,32859

Описание экспериментов

Алгоритм классификации текстов состоит из нескольких блоков: предобработка векторов, метод классификации, способ учета иерархии. По отдельности все блоки многократно описаны и являются довольно простыми. Успешный, качественный алгоритм – это удачно подобранные все «кирпичики» алгоритма.

Анализируя результаты экспериментов, можно убедиться, что конечный результат зависит от предобработки входных данных и учета иерархии не в меньшей степени, чем от метода машинного обучения.

Интерес в данной работе представляют именно варианты предобработки данных и модификации стандартного алгоритма. Было исследовано, как те или иные комбинации процедур влияли на качество классификации.

Ниже будут приведены виды реализованных «блоков», из которых составляется конечный алгоритм классификации. Далее описаны эксперименты: конфигурации запуска алгоритма.

Для удобства эксперименты описаны в виде таблиц с указанием всех необходимых параметров. Подробнее таблица будет описана ниже. Параметры алгоритма в таблице представлены в зашифрованном виде, шифры будут даваться по ходу описания блоков.

Реализованные части алгоритма

Простой на первый взгляд центроидный классификатор дает широкий простор для исследования. Он обладает минимумом параметров для настройки (нужно только задать функцию близости) и изначально дает неплохие результаты. Такой алгоритм позволяет сфокусироваться на исследовании видов предобработки векторного пространства документов и выборе способа учета иерархической структуры классов.

Для построения конечного алгоритма следует ответить на ряд вопросов:

1. Какие использовать веса признаков?
2. Проводить ли сглаживание функцией и какой?
3. Отбирать ли неинформативные признаки и как?
4. Какой метрикой определять близость документов к центрам классов?
5. Используется ли иерархия классов и как?

6. Если используется иерархический алгоритм, то:

Когда проводить преобразование пространства?

Как вычисляется центр классов в узлах?

Дадим подробные ответы на поставленные вопросы. Следует заметить, что изначально вопросы независимы (кроме последнего) и ответ на один вопрос никак не ограничивает другой. При проведении экспериментов будем стремиться максимально увеличить показатели качества классификации. По ходу проведения экспериментов становилось понятно, какие комбинации «кирпичиков» заведомо нежизнеспособны. Тогда они убирались из дальнейшего рассмотрения. Такие случаи будут описаны отдельно. Когда будут определены лучшие алгоритмы, из них будет выбран тот, который работает наименьшее время.

Выбор веса признаков

Ниже приведен список способов взвешивания, используемых в экспериментах. В правой колонке будет указан шифр преобразования, используемый в таблицах.

1. $\boldsymbol{TF}$ (1) – частота появления терма в документе	tf
2. $\boldsymbol{TF} * \boldsymbol{IDF}$, где IDF может быть представлен по-разному: как логарифм инверсной частоты (3) или степень (4). Причем в каждом из вариантов в знаменателе может стоять как количество документов, содержащих признак, так и количество термов во всех документах.	tfidf

Для того, чтобы не возникло путаницы, пронумеруем различные варианты IDF:

IDF_1	$\log\frac{\lvert D\rvert}{n_i}$, где n_i— количество документов, в которых встречается i-е слово
IDF_2	$\log\frac{\lvert D\rvert}{n_i}$, где n_i— сумма числа вхождений i-го слова во все документы выборки
IDF_3	$\left(\frac{\lvert D\rvert}{n_i}\right)^{\alpha}$, где n_i— количество документов, в которых встречается i-е слово; $\alpha \in (0,1)$ — параметр.
IDF_4	$\left(\frac{\lvert D\rvert}{n_i}\right)^{\alpha}$, где n_i— сумма числа вхождений i-го слова во все документы выборки; $\alpha \in (0,1)$ — параметр.

Для IDF_3 и IDF_4 значения α - узлы равномерной сетки от 0 до 1 с шагом 0.1.

3. **Взвешивание** $\boldsymbol{TF}$: $TF^p * IDF, p \in (0,1)$ IDF возможны любые из приведенного выше списка. В качестве параметра p были выбраны 3 значения: $p \in \{0.25, 0.5, 0.75\}$.	w tfidf

4. $ConfWeight$ (5), параметр $\tau = 1.96$ и $\tau = 2.625$. Первое значение параметра считается оптимальным с точки зрения авторов весовой функции [14]. Второе значение выбрано как наилучшее в алгоритме, представленном в проекте LSHTC [9]	CW

Нормировка признаков. Каждый из выше перечисленных вариантов может быть нормирован по норме L_1 (7) или по L_2 (6). Обозначать нормировку будем прибавление к варианту весов «/abs» в первом случае и «/sqr» во втором	<Тип веса>/abs <Тип веса>/sqr

Сглаживание функцией

Сглаживание проводилось до взвешивания признаков. Были рассмотрены 3 варианта:

1. Сглаживание никакой функцией не происходит;	нет
2. Применение к каждому признаку каждого документа функции f_1 (8);	f1
3. Применение к каждому признаку каждого документа функции f_2 (9).	f1

Отбор неинформативных признаков

Был реализован элементарный отбор признаков: если терм встречается в выборке слишком часто или слишком редко, то он убирается. Эксперименты проводились как в полном признаковом пространстве, так и в уменьшенном, без выбранных «шумовых» признаков.

Для отбора применялись следующие правила:

1. Шумовым является признак, для которого математическое ожидание $\mathcal{M}tf_i > 0.95$ и дисперсию $\mathcal{D}tf_i \leq 0.05$.

Математическое ожидание и дисперсия вычисляются по обучающей выборке. Множество признаков заменяется на частоту признака в документе TF (1). Вначале признаки усредняются по классам, получаем множество векторов - центров классов. Затем по полученным центрам уже вычисляются мат.ожидание и дисперсия.

Такой подход предупреждает некоторые ошибки. Например, суммарно в документах терм встречается часто, но в некоторых классах значительно чаще, чем в других. Допустим, представителей такого класса много в обучающей выборке. Тогда будет определено, что терм имеет высокое математическое ожидание, и, возможно, достаточно маленькую дисперсию. Терм незаслуженно будет отнесен к «шумовым». Когда же мы в начале усредняем частоту по классам, то получаем разброс значений именно по классам, а не по отдельным документам.

Пороги для математического ожидания и дисперсии были выбраны по итогам предварительных экспериментов.

2. Убирается признак, который встречается во всех документах выборки не больше, чем m раз: $sum(t_i) < m$. В проводимых экспериментах $m = 1$ и $m = 3$. При $m = 5 \, или \, 7$ обычный плоский классификатор сразу дал ухудшение качества, поэтому в дальнейшем значения $m > 3$ были исключены из рассмотрения.

Для отбора признаков применялись 5 видов правил:

1. $\mathcal{M}tf_i > 0.95 \;\&\&\; \mathcal{D}tf_i \leq 0.05$;	П1

2. $\mathcal{M}tf_i > 0.95 \;\&\&\; \mathcal{D}tf_i \leq 0.05 \;||\; sum(t_i) = 1$;	П2
3. $\mathcal{M}tf_i > 0.95 \;\&\&\; \mathcal{D}tf_i \leq 0.05 \;||\; sum(t_i) \leq 3$;	П3
4. $\mathcal{M}tf_i > 0.95 \;\&\&\; \mathcal{D}tf_i \leq 0.05 \;||\; sum(t_i) \leq 5$;	П4
5. $\mathcal{M}tf_i > 0.95 \;\&\&\; \mathcal{D}tf_i \leq 0.05 \;||\; sum(t_i) \leq 7$;	П5

Метрика пространства

Для определения близости документов к центрам классов применяется одна из трех функций близости:

1. Метрика Городских кварталов $Abs(x, y)$ (11);	abs
2. Евклидова метрика $Sqr(x, y)$ (12);	str
3. Косинус угла между векторами $Cossim(x, y)$ (13)	cossim

Использование иерархии классов

Были реализованы несколько вариантов классификаторов, основанных на Центроидном алгоритме. Для удобства описания результатов экспериментов присвоим каждому классификатору свой шифр.

<table>
<tr><th colspan="3">Классификатор</th><th>Шифр</th></tr>
<tr><td colspan="3">1. Плоский классификатор.
Наиболее простой. Все конечные классы считаются равноправными.</td><td>F</td></tr>
<tr><td rowspan="3">2. Иерархический классификатор.
На каждом уровне вычисляются центры масс узлов и производится классификация по близости к центрам узлов. Далее переходим в выбранный узел и повторяем процедуру. Так, пока не дойдем до листа. Он и будет конечным классом документа.</td><td rowspan="2">• Веса признаков вычисляются на каждом уровне. TF считается в начале. IDF пересчитывается по векторам, входящим в классы, расположенные ниже по иерархии.</td><td>o Сначала вектора преобразуются в веса, потом вычисляется по ним центр.</td><td>HL1</td></tr>
<tr><td>o Сначала по векторам вычисляется цент, затем он нормируется</td><td>HL2</td></tr>
<tr><td rowspan="2">• Все веса признаков вычисляются до начала настройки и классификации.</td><td>o Центр узла – среднее арифметическое все документов, принадлежащих детям</td><td>HG Down</td></tr>
<tr><td>o Центр узла – среднее арифметическое центров узлов детей</td><td>HG Up</td></tr>
</table>

Если используется иерархический классификатор, необходимо ответить на вопрос: какого рода иерархия используется? В данной работе было использовано три вида иерархии:

0. Начальная иерархия: такая, какой она представлена в данных проекта. В начальном варианте она имеет 5 уровней;	0-1. orig

1. Упрощенная иерархия: из начальной иерархии убраны все узлы, имеющие одного наследника. Такое упрощение никак не сказывается на качестве классификатора, но увеличивает скорость его работы. В упрощенном варианте остается также как и в начальной иерархии 5 уровней, но узлов становится меньше в полтора раза;

Так как в обоих вариантах качество алгоритма одинаково, то они будут объединены в один параметр. На этапе подсчета времени выполнения алгоритмов разъединим эти варианты.

2. 2level

2. Двухуровневая классификация: остается верхний уровень классификации, все листья-классы становятся непосредственными наследниками первого уровня. Такая модификации структуры классов получила в литературе название «уплощение» (Hierarchy Flattening [7]).

Эксперименты

Для описания проведенных экспериментов приведем таблицы с указанием параметров алгоритмов. Таблицы выглядят следующим образом:

Алгоритм			F			HL1		HL2		HG Down	HG Up
						Тип иерархии					
						0-1. orig		2. 2level			
Метрика			cossim			abs			sqr		
Веса	tf	tfidf	tfidf/abs	tfidf/sqr	w	w	w	CW	CW/abs	CW/sqr	
		idf			p			Tao			
		1	2	3	4	0,25	0,5	0,75	1,96	2,625	
Очищение			нет			П1		П2		П3	
Сглаживание			нет			f1			f2		

В таблицах экспериментов ячейка с используемым параметром закрашена синим цветом. Если в одной строчке закрашено несколько параметров, это означает, что эксперименты проводились по всем комбинациям параметров.

Все шифры блоков соответствуют описанным выше. Интерпретировать таблицу нужно следующим образом: верхняя строчка («Алгоритм») – тип используемого классификатора и учет иерархии классов. Следующий раздел – «тип иерархии». Используется только в случае иерархической классификации. «Метрика» – тип метрики, используемой для вычисления близости объекта к центрам классов. Строчка «Веса» – способ взвешивания признаков. Например, «w tfidf/abs» означают, что преобразование идет как взвешенный $TF^p * IDF$, нормированный по норме L_1. Отдел «idf» означает тип реализации IDF. Раздел «p» заполняется только при использовании $TF^p * IDF$ и обозначает степень при TF. Раздел «tao» заполняется, соответственно, при использовании весов $ConfWeight$ и задает значение параметра τ. «Очищение» означает тип отбора информативных признаков. Если признаковое пространство не очищается, выбирается ячейка «нет». В строчке «Сглаживание» выбирается тип сглаживающей функции. Если преобразование сглаживающей функцией не производится, выбирается ячейка «нет».

Итак, перейдем к непосредственному описанию экспериментов и их результатам.

Эксперимент 1

Алгоритм			F			HL1		HL2		HG Down	HG Up
						Тип иерархии					
						0-1. orig		2. 2level			
Метрика			cossim			abs			sqr		
Веса	tf	tfidf	tfidf/abs	tfidf/sqr	w	w	w	CW	CW/abs	CW/sqr	
		idf			p			Tao			
		1	2	3	4	0,25	0,5	0,75	1,96	2,625	
Очищение			нет			П1		П2		П3	
Сглаживание			нет			f1			f2		

Эксперимент показал, что подстановка в $TF * IDF$ IDF_1 или IDF_2 не меняет качества классификации: результаты на разных классификаторах, нормировках и функциях близости оказались одинаковыми. Поэтому в дальнейшем из двух вариантов IDF рассматривается только IDF_2. При этом, IDF_1 является более стандартным представлением IDF, чем выбранный вариант, поэтому на лучших конфигурациях проведены дополнительные эксперименты на проверку изменения качества алгоритма в зависимости от выбора IDF.

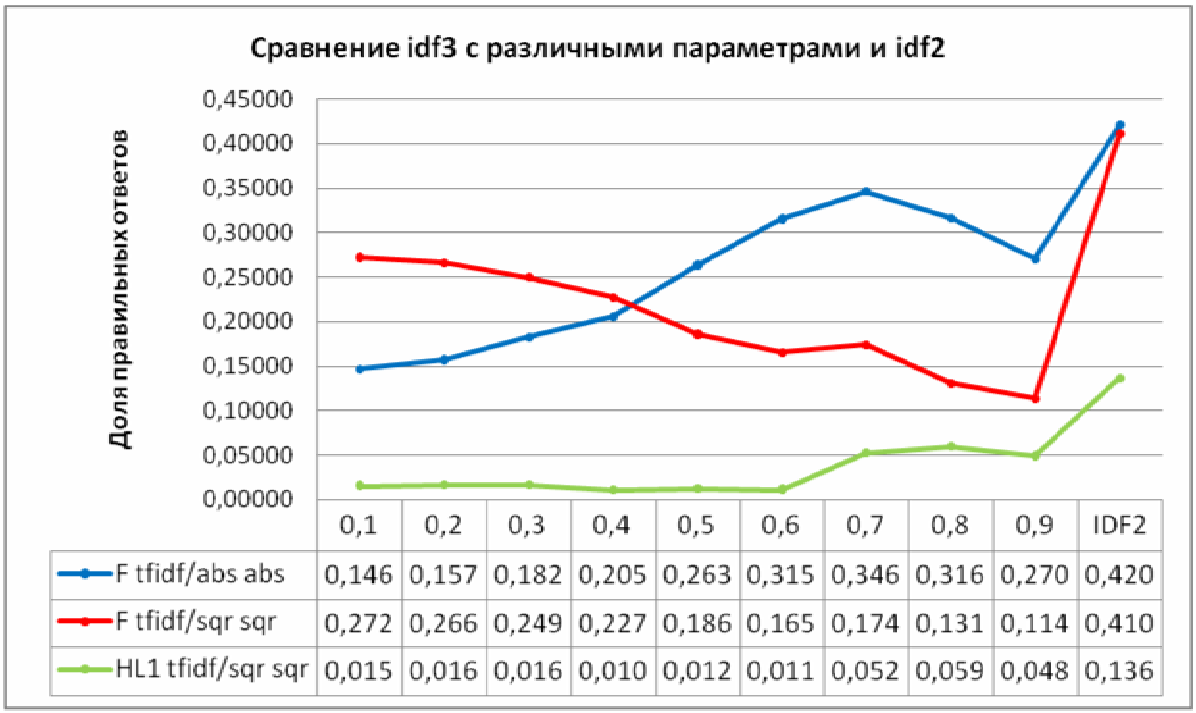
Далее, по эксперименту можно наблюдать качество работы двух видов иерархических классификаторов: HL1 и HL2. Их разница заключается в порядке вычисления центра узла относительно взвешивания признаков в векторах-прецедентах. Эксперимент показал, что результаты работы двух алгоритмов приблизительно одинаковые, HL1 в среднем по всем показателям работает на 0,2% лучше. Различия незначительные. В дальнейших исследованиях, при реализации метрики $Cossim$ алгоритм HL2 оказался неэффективным в работе (работает в 3-4 раза дольше алгоритма HL1). В силу малого отличия по качеству двух алгоритмов, в дальнейших экспериментах принимает участие только классификатор HL1.

Эксперимент 2

Алгоритм			F			HL1		HL2		HG Down	HG Up
						Тип иерархии					
						0-1. orig		2. 2level			
Метрика			cossim			abs			sqr		
Веса	tf	tfidf	tfidf/abs		tfidf/sqr	w		w		w	CW	CW/abs	CW/sqr
		idf				p				Tao	
		1	2	3	4	0,25		0,5	0,75	1,96	2,625
Очищение			нет			П1			П2	П3	
Сглаживание			нет			f1				f2	

Данный эксперимент был проведен для проверки, насколько удачно могут сработать IDF_3 и IDF_4. Задача была запущена при различных параметрах α: от 0.1 до 0.9 с шагом 0.1. Результаты показали, что на всех типах алгоритмов оба варианта IDF с возведением в степень работают гораздо хуже, чем IDF с логарифмированием.

Ниже приведены результаты работы алгоритмов. Выбраны те конфигурации, на которых алгоритм давал наилучшие показатели качества. Координаты по оси x – значения параметра α. Последняя точка – качество работы алгоритма с теми же конфигурациями, но с применением IDF_2. Сравнение алгоритмов идет по доли правильных ответов A (14).



	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	IDF2
F tfidf/abs abs	0,146	0,157	0,182	0,205	0,263	0,315	0,346	0,316	0,270	0,420
F tfidf/sqr sqr	0,272	0,266	0,249	0,227	0,186	0,165	0,174	0,131	0,114	0,410
HL1 tfidf/sqr sqr	0,015	0,016	0,016	0,010	0,012	0,011	0,052	0,059	0,048	0,136

Данный эксперимент был проведен потому, что IDF_3 и IDF_4 применяются в алгоритме – участнике проекта LSHTC [8]. В том варианте алгоритма получались неплохие результаты (доля правильных ответов – 0.44, правда, на другой выборке данных). В исследуемой конфигурации данные варианты оказались хуже. Вследствие этого они далее не рассматриваются в экспериментах.

Эксперименты 3-5 проводятся практически на одинаковых конфигурациях: это стандартные варианты преобразования признаков, все метрики, все методы очистки и сглаживания. Разница лишь в применяемых классификаторах. При этом в случае иерархических классификаторов используется стандартная иерархия. Данные конфигурации являются базовыми. В каждом эксперименте буду выведены лучшие и худшие результаты по каждому виду функций близости. Функции близости разделяются, так как являются неотъемлемой внутренней частью алгоритма, в то время как остальные параметры могут присутствовать в большей или меньшей степени. Вывод худших результатов помогает понять разброс качества различных конфигураций, и, с другой стороны, помогает отследить конфигурации, которые в последствии с большой вероятностью будут давать низкие показатели качества.

Эксперимент 3

Алгоритм	F	HL1	HL2	HG Down	HG Up
		Тип иерархии			
		0-1. orig	2. 2level		

Метрика	cossim	abs	sqr

Веса	tf	tfidf	tfidf/abs	tfidf/sqr	w	w	w	CW	CW/abs	CW/sqr
	idf				p			Tao		
	1	2	3	4	0,25	0,5	0,75	1,96	2,625	

Очищение	нет	П1	П2	П3
Сглаживание	нет	f1	f2	

Лучшие результаты					
Очищ	Сгл	Норм	A	Макро-r	Макро-p
cossim					
П1	нет	tfidf/sqr	0,4801	0,4317	0,3425
П2	нет	tfidf/sqr	0,4769	0,4276	0,3376
П3	нет	tfidf/sqr	0,4758	0,4257	0,3344
abs					
П2	нет	tfidf/abs	0,4241	0,4372	0,3489
П1	нет	tfidf/abs	0,4225	0,4308	0,3407
нет	нет	tfidf/abs	0,4209	0,4247	0,3459
sqr					
нет	нет	tfidf/sqr	0,4107	0,2971	0,2863
П3	нет	tfidf/sqr	0,4031	0,2873	0,2837
П2	нет	tfidf/sqr	0,3956	0,2773	0,2728

Худшие результаты					
Очищ	Сгл	Норм	A	Макро-r	Макро-p
cossim					
нет	f2	tfidf	0,0888	0,0982	0,0580
П1	f2	tfidf	0,0888	0,0982	0,0580
П2	f2	tfidf	0,0770	0,0833	0,0446
abs					
нет	f1	tfidf	0,0188	0,0214	0,0192
нет	f2	tfidf	0,0188	0,0214	0,0192
П1	f2	tfidf/sqr	0,0183	0,0192	0,0147
sqr					
нет	f1	tfidf/abs	0,0312	0,0299	0,0293
нет	f2	tfidf/abs	0,0312	0,0299	0,0293
нет	f1	tfidf	0,0118	0,0108	0,0081

Получаем наилучшую конфигурацию с плоским классификатором: функция близости $Cossim$, предобработка признаков по формуле $TF * IDF$ с нормировкой по L_2(6), без использования сглаживающей функции и с отбором признаков по правилу П1.

Эксперимент 4

Алгоритм		F			HL1			HL2			HG Down		HG Up
					Тип иерархии								
					0-1. orig			2. 2level					
Метрика					cossim			abs			sqr		
Веса	tf	tfidf		tfidf/abs	tfidf/sqr	w		w		w	CW	CW/abs	CW/sqr
		idf				p					Tao		
		1	2	3	4	0,25		0,5		0,75	1,96	2,625	
Очищение			нет			П1			П2		П3		
Сглаживание			нет			f1			f2				

Лучшие результаты					
Очищ	Сгл	Норм	A	Макро-r	Макро-p
cossim					
П1	нет	tfidf/sqr	0,3870	0,2973	0,2380
П2	нет	tfidf/sqr	0,3854	0,2959	0,2359
П3	нет	tfidf/sqr	0,3837	0,2942	0,2340
П2	нет	tfidf	0,3676	0,2793	0,2218
abs					
П3	нет	tf	0,1921	0,1433	0,1142
П2	нет	tf	0,1911	0,1409	0,1126
П1	нет	tf	0,1895	0,1391	0,1106
П3	f1	tf	0,1184	0,0845	0,0695
sqr					
П3	нет	tfidf/sqr	0,3030	0,1889	0,1609
П2	нет	tfidf/sqr	0,3019	0,1893	0,1604
П1	нет	tfidf/sqr	0,2987	0,1825	0,1561
П1	нет	tf	0,2885	0,1885	0,1543

Худшие результаты					
Очищ	Сгл	Норм	A	Макро-r	Макро-p
cossim					
нет	f2	tfidf	0,0005	0,0009	0,0000
П1	f2	tfidf	0,0005	0,0009	0,0000
П2	f2	tfidf	0,0005	0,0009	0,0000
П3	f2	tfidf	0,0005	0,0009	0,0000
abs					
П3	f2	tfidf/sqr	0,0048	0,0051	0,0030
нет	f2	tfidf/sqr	0,0043	0,0050	0,0025
П1	f2	tfidf/sqr	0,0043	0,0050	0,0025
нет	нет	tfidf/sqr	0,0011	0,0018	0,0001
sqr					
нет	нет	tfidf/sqr	0,1367	0,0831	0,0692
нет	нет	tf	0,0102	0,0046	0,0022
нет	нет	tfidf/abs	0,0043	0,0010	0,0010
нет	нет	tfidf	0,0038	0,0028	0,0008

Получаем наилучшую конфигурацию с иерархическим классификатором, пересчитывающим веса признаков на каждом уровне ту же, что и для плоского классификатора в Эксперименте 3. При этом, как и следовало ожидать, иерархический классификатор ошибается чаще. Разница в точности при лучших конфигурациях составляет 10%.

Эксперимент 5

Алгоритм		F			HL1			HL2			HG Down		HG Up
					Тип иерархии								
					0-1. orig			2. 2level					
Метрика					cossim			abs			sqr		
Веса	tf	tfidf		tfidf/abs	tfidf/sqr	w		w		w	CW	CW/abs	CW/sqr
		idf				p					Tao		
		1	2	3	4	0,25		0,5		0,75	1,96		2,625
Очищение			нет			П1			П2		П3		
Сглаживание			нет			f1			f2				

Результаты работы классификатора HG Down:

Лучшие результаты					
Очищ	Сгл	Норм	A	Макро-r	Макро-p
cossim					
нет	нет	tfidf/sqr	0,4279	0,3476	0,2691
П1	нет	tfidf/sqr	0,4279	0,3476	0,2691
П2	нет	tfidf/sqr	0,4257	0,3465	0,2678
нет	нет	tfidf/abs	0,4252	0,3447	0,2700
abs					
П2	нет	tfidf/abs	0,2164	0,1701	0,1312
П3	нет	tfidf/abs	0,2153	0,1714	0,1311
нет	нет	tfidf/abs	0,2083	0,1628	0,1261
П1	нет	tfidf/abs	0,2083	0,1628	0,1261

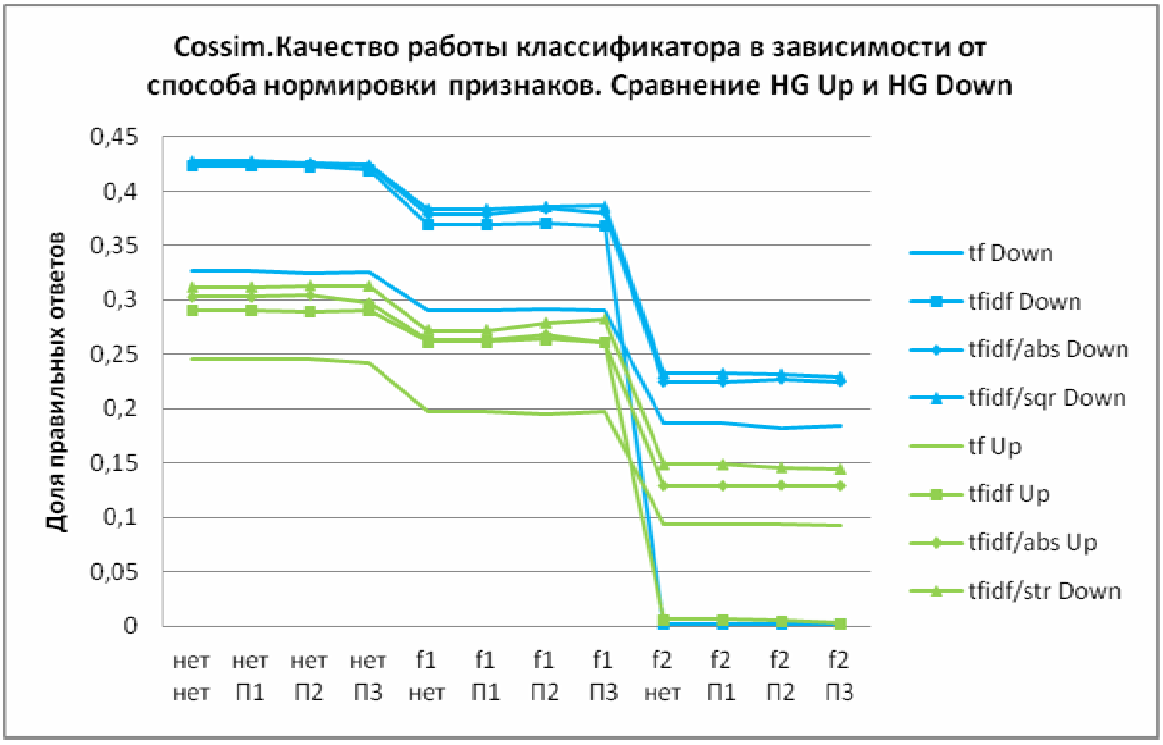
sqr					
П3	нет	tfidf/sqr	0,3767	0,2732	0,2359
П2	нет	tfidf/sqr	0,3714	0,2651	0,2296
нет	нет	tfidf/sqr	0,3671	0,2590	0,2263
П1	нет	tfidf/sqr	0,3671	0,2590	0,2263
Худшие результаты					
Очищ	Сгл	Норм	A	Макро-r	Макро-p
cossim					
нет	f2	tfidf	0,0022	0,0023	0,0009
П1	f2	tfidf	0,0022	0,0023	0,0009
П2	f2	tfidf	0,0016	0,0015	0,0009
П3	f2	tfidf	0,0016	0,0019	0,0009
abs					
нет	f2	tfidf/sqr	0,0054	0,0060	0,0035
П1	f2	tfidf/sqr	0,0054	0,0060	0,0035
П2	f2	tfidf/sqr	0,0054	0,0060	0,0029
П3	f2	tfidf/sqr	0,0054	0,0060	0,0031
sqr					
нет	f2	tf	0,1631	0,0585	0,0475
П1	f2	tf	0,1631	0,0585	0,0475
П2	f2	tf	0,1609	0,0567	0,0471
П3	f2	tf	0,1588	0,0561	0,0464

Результаты работы классификатора HG Up

Лучшие результаты					
Очищ	Сгл	Норм	A	Макро-r	Макро-p
cossim					
П2	нет	tfidf/sqr	0,3132	0,3196	0,2367
П3	нет	tfidf/sqr	0,3127	0,3188	0,2365
нет	нет	tfidf/sqr	0,3116	0,3183	0,2357
П1	нет	tfidf/sqr	0,3116	0,3183	0,2357
abs					
П3	нет	tfidf/abs	0,1722	0,1869	0,1319
П2	нет	tfidf/abs	0,1712	0,1867	0,1305
нет	нет	tfidf/abs	0,1695	0,1832	0,1308
П1	нет	tfidf/abs	0,1695	0,1832	0,1308
sqr					
П3	нет	tfidf/sqr	0,2357	0,2137	0,1938
П2	нет	tfidf/sqr	0,2287	0,2069	0,1889
нет	нет	tfidf/sqr	0,2234	0,2039	0,1857
П1	нет	tfidf/sqr	0,2234	0,2039	0,1857

Худшие результаты					
Очищ	Сгл	Норм	A	Макро-r	Макро-p
cossim					
нет	f2	tfidf	0,0059	0,0066	0,0043
П1	f2	tfidf	0,0059	0,0066	0,0043
П2	f2	tfidf	0,0048	0,0048	0,0024
П3	f2	tfidf	0,0027	0,0025	0,0015
abs					
нет	f2	tfidf/sqr	0,0032	0,0033	0,0004
П1	f2	tfidf/sqr	0,0032	0,0033	0,0004
П2	f2	tfidf/sqr	0,0032	0,0033	0,0005
П3	f2	tfidf/sqr	0,0032	0,0033	0,0005
sqr					
П3	f2	tf	0,0662	0,0365	0,0275
нет	f2	tf	0,0651	0,0359	0,0269
П1	f2	tf	0,0651	0,0359	0,0269
П2	f2	tf	0,0646	0,0346	0,0260

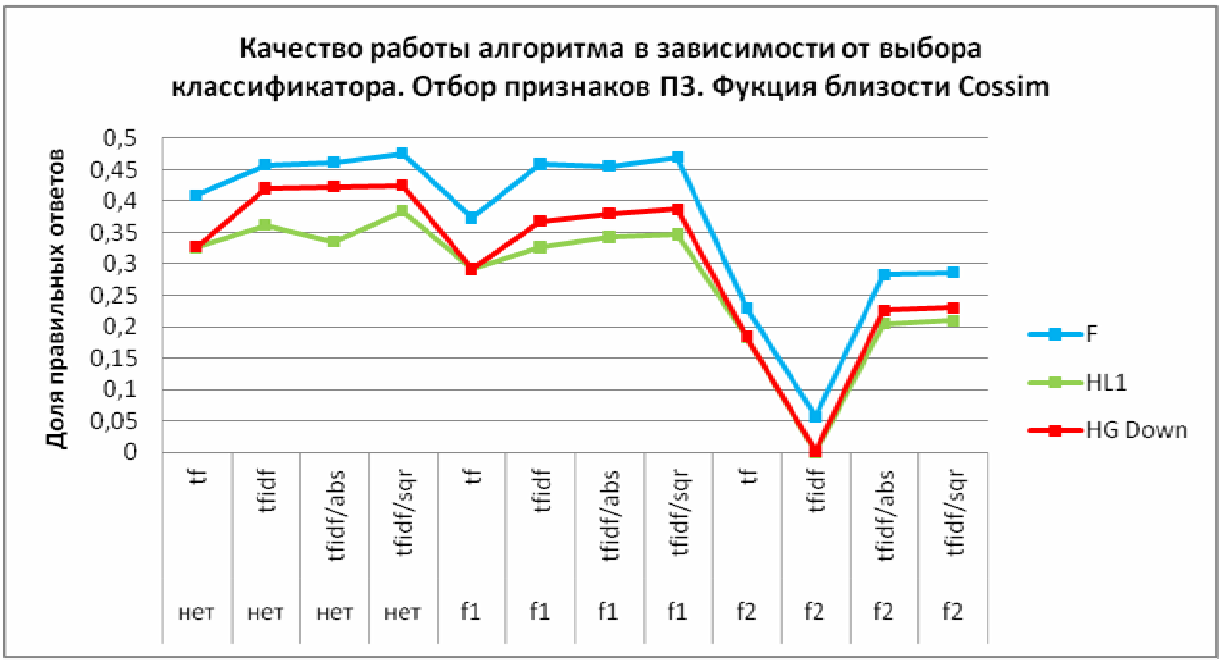
Сравнивая два подвида алгоритма, различающиеся способом вычисления центров узлов, видно, что лучше работает HG Down. Этот алгоритм вычисляет центры узлов как среднее арифметическое всех документов, принадлежащих наследникам этих узлов. В подтверждение написанному, ниже приведены графики качества двух классификаторов, использующих функцию близости *Cossim*:



Для других метрик результаты аналогичны. В дальнейшем будем рассматривать только лучший вариант алгоритма: HG Down. Наилучшая конфигурация для классификатора HG Down совпадает с наилучшими конфигурациями плоского и иерархического HL1 классификаторов.

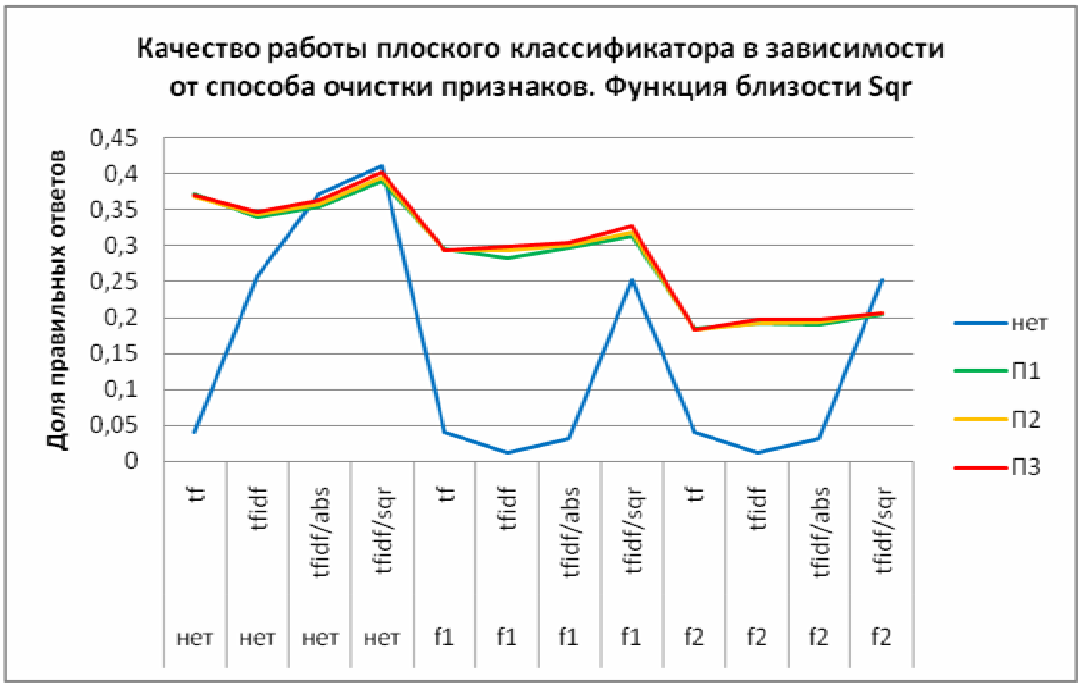
Общие наблюдения по итогам трех экспериментов

Результаты экспериментов показывают, что из трех рассмотренных классификаторов наиболее точные результаты выдает плоский, за ним следует HG Down, последний – HL1. Например, для функции близости $Cossim$ и правила отбора признаков ПЗ можно наблюдать следующую картину (для других правил отбора признаков и остальных функций близости порядок расположения графиков аналогичен):



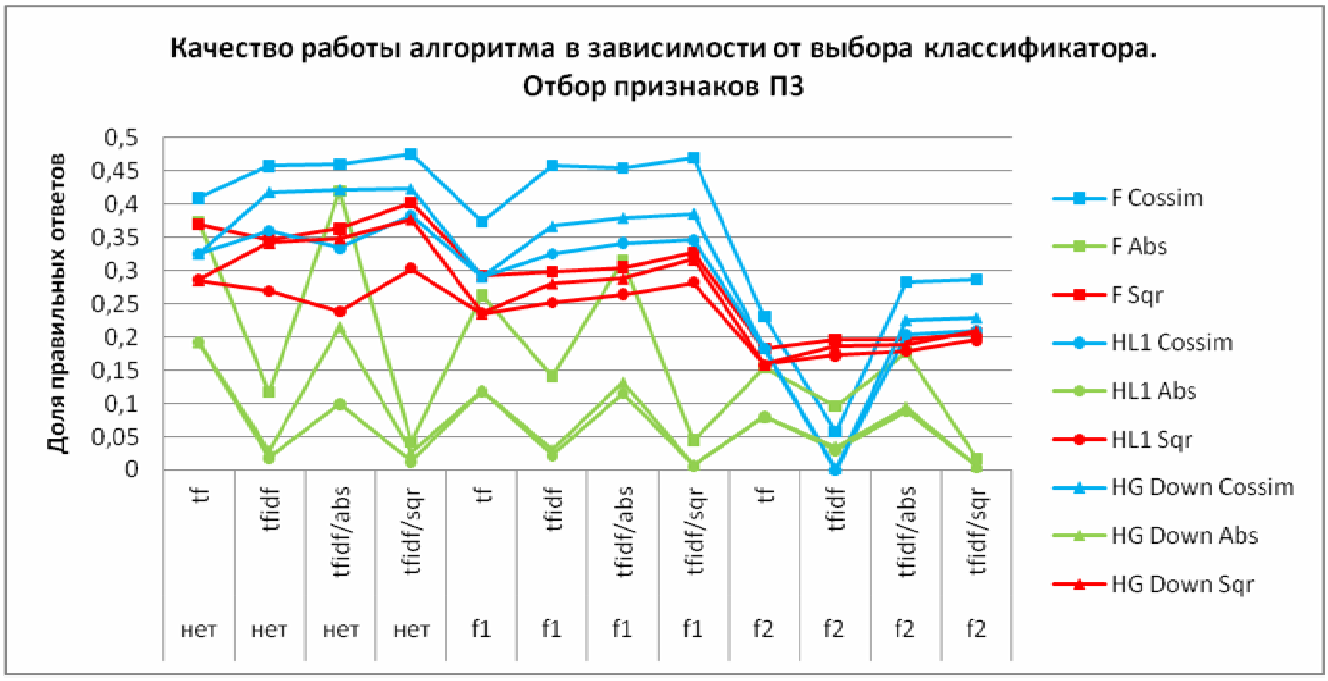
Большую точность алгоритма HG Down перед HL1 можно объяснить следующим. Когда IDF высчитывается отдельно в каждом узле, только по документам этого узла, возникает эффект переобучения, веса становятся слишком «специфичными», в то время как IDF должен учитывать информативность признака для классификации вообще, а не для отдельных подклассов. Кроме того в HG Down IDF вычисляется по большему числу документов, что делает значения весов точнее. Также подтверждаются утверждения о лучшем качестве плоского классификатора перед иерархическим: у иерархических классификаторов есть больше возможностей свернуть по неверной ветви в дереве классов. Дальнейшие эксперименты подтвердили преимущества классификатора HG Down перед HL1, поэтому в дальнейшем последний классификатор не будет рассматриваться.

Сравнение результатов работы алгоритмов при разных правилах отбора признаков показывает, что отбор признаков ведет к улучшению качества алгоритмов, а вот конкретные виды правил мало отличаются между собой. При иерархической классификации отбор признаков роли практически не играет (разница качества не более, чем на 0.5%). Плоский же классификатор при отборе шумовых признаков во многих случаях дает лучшее качество, что демонстрирует график ниже:



Поскольку виды отбора признаков мало отличаются между собой, графики сравнения функций близости и алгоритмов приводятся только для одного правила отбора признаков П3.

Использование функции близости $Cossim$ ведет к наилучшим результатам. Тогда как функция Abs в целом работает хуже. Для некоторых конфигураций алгоритма эти правила не выполняются, но в таком случае при любой функции близости результат работы очень низкий. Приведенные утверждения иллюстрирует график, на котором отображено качество классификации для различных алгоритмов и различных функциях близости:



Также по приведенному выше графику видно, что результаты работы алгоритма при функции сглаживания f_2 хуже при всех конфигурациях по сравнению с f_1 и отсутствием сглаживания.

По итогам Экспериментов 3-5 из дальнейшего рассмотрения была убрана сглаживающая функция f_2 (логарифм) и функция близости Abs, работающая явно хуже других двух метрик.

Следует заметить, что качество алгоритмов меряется по нескольким метрикам, но лучшие конфигурации алгоритма по разным метрикам совпадают. Поэтому результаты на графиках приводятся только для доли правильных ответов.

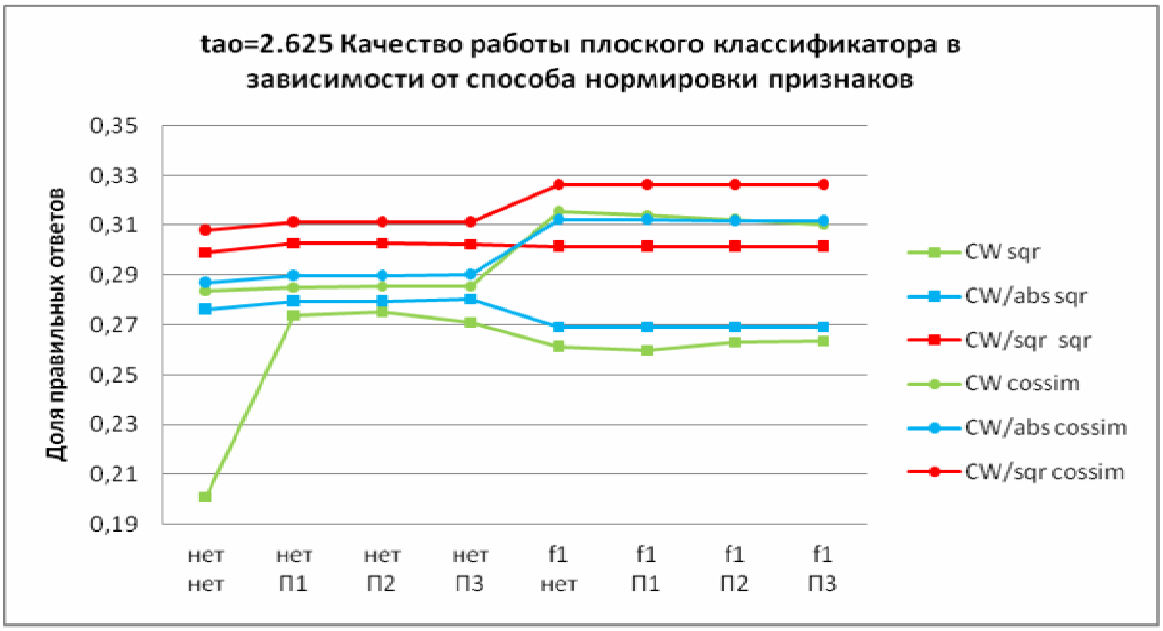
Эксперимент 6: проверка качества алгоритмов с ConfWeight

Алгоритм		F			HL1		HL2		HG Down		HG Up
					Тип иерархии						
					0-1. orig		2. 2level				
Метрика				cossim			abs			sqr	
Веса	tf	tfidf		tfidf/abs		tfidf/sqr	w	w	w	CW	CW/abs	CW/sqr
		idf					p			Tao		
		1	2	3		4	0,25	0,5	0,75	1,96		2,625
Очищение			нет				П1		П2		П3	
Сглаживание				нет			f1			f2		

Целью проведения эксперимента 6 было сравнение качества классификации при двух принципиально разных весах: $TF * IDF$ и $ConfWeight$. Как было написано выше, авторы весов $ConfWeight$ в своих экспериментах показали, что их веса улучшают качество классификации по сравнению с $TF * IDF$ при классификаторах, основанных на методе ближайшего соседа и SVM.

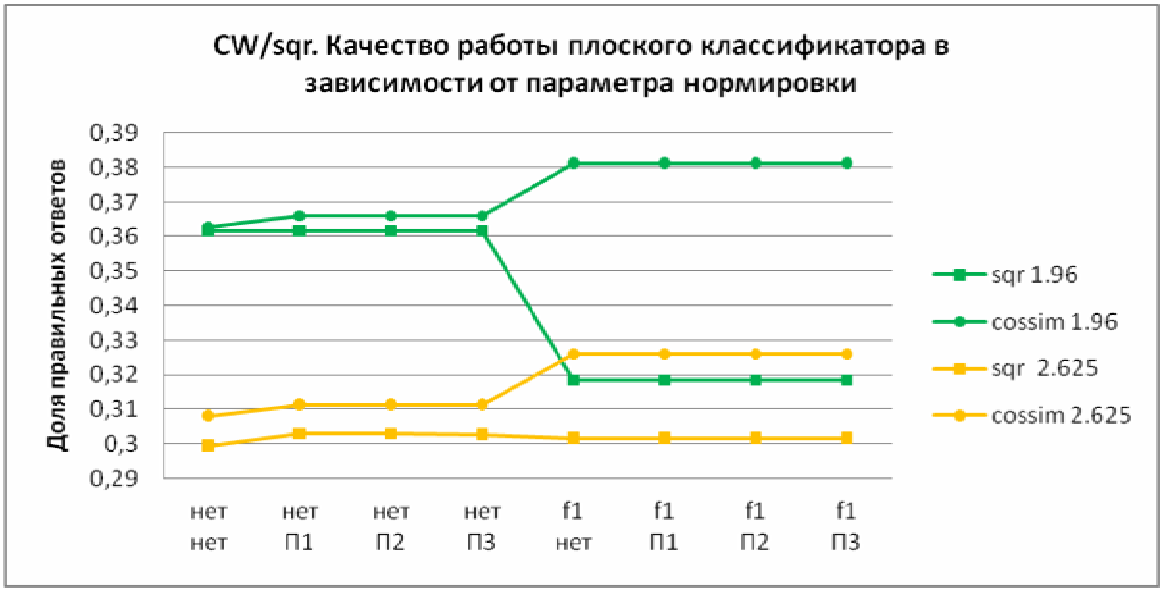
По результатам эксперимента сначала были выбраны те конфигурации алгоритма, которые дают наилучшее качество классификации. Далее было проведено сравнение качества выбранных конфигураций с аналогичными, но использующими $TF * IDF$.

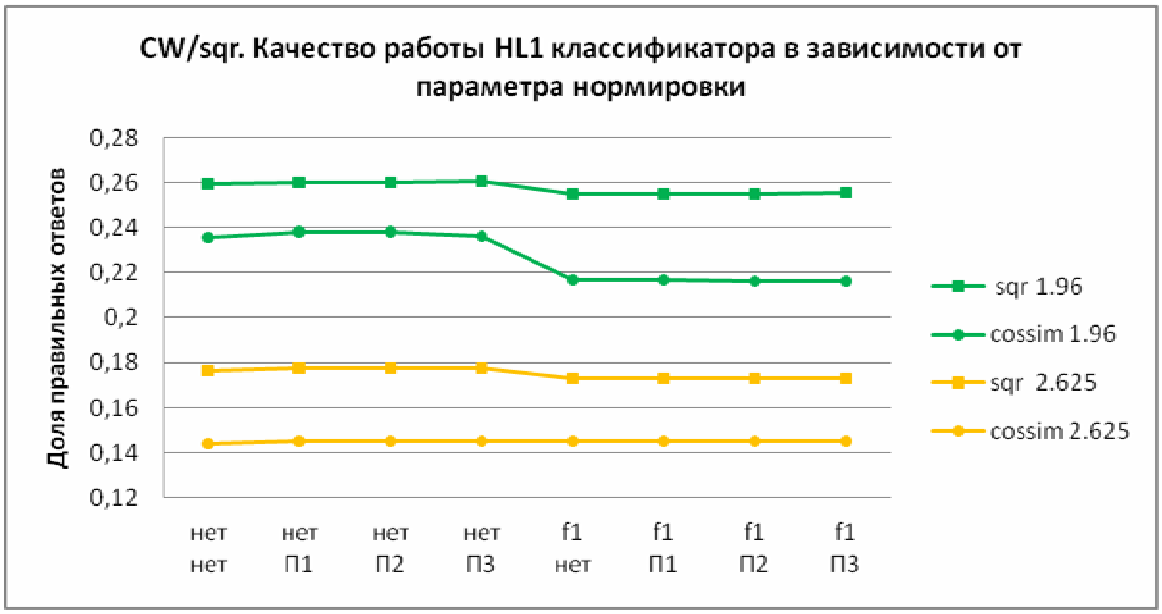
Результаты алгоритма с классификаторами F и HL1 при значениях параметра весов $ConfWeight$ $\tau = 2.625$ и $\tau = 1.96$ показали, что наилучшие результаты получаются при дополнительной нормировке признаков по L_2. В подтверждение сказанному ниже приведены графики качества алгоритма с плоским классификатором, значением параметра $\tau = 2.625$. Графика приведены для различных вариаций взвешивания признаков и двух типов функции близости: $Cossim$ и Sqr. Данные графики отражают тенденцию, которая прослеживается и для других комбинаций классификатор – параметр τ.



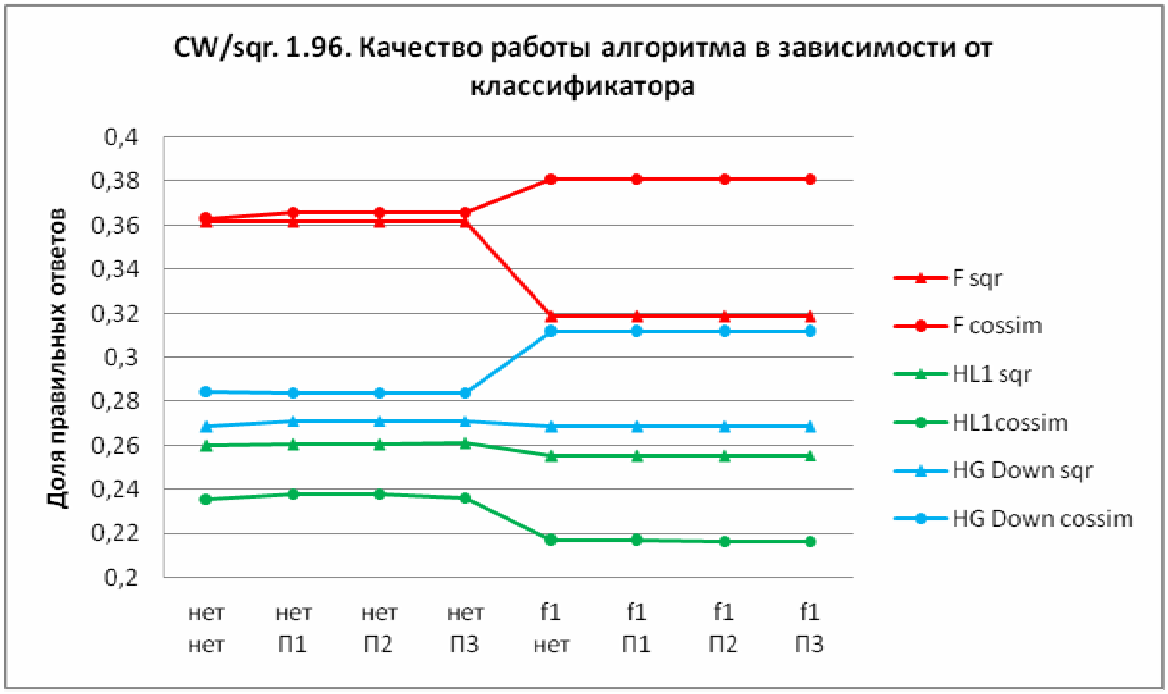
Далее зафиксируем веса признаков $ConfWeight/Sqr$ и выявим наилучшие значения других параметров.

Сравнение качества алгоритмов при различных значениях τ показали, что наилучшим вариантом является $\tau = 1.96$, что демонстрируют графики ниже.





Далее был зафиксирован параметр $\tau = 1.96$ и проведено сравнение по классификаторам.



Как и в экспериментах 3-5 алгоритм HL1 отстает по качеству от двух других. Наилучшее качество, опять же, получается при плоском классификатора и метрике $Cossim$. Как наилучшие алгоритмы, использующие веса $ConfWeight$, были выбраны следующие конфигурации: классификатор F, функция близости $Cossim$ или Sqr; классификатор HG Down, функция близости $Cossim$. Для обеих конфигураций выбирается дополнительная нормировка признаков по L_2 и параметр $\tau = 1.96$.

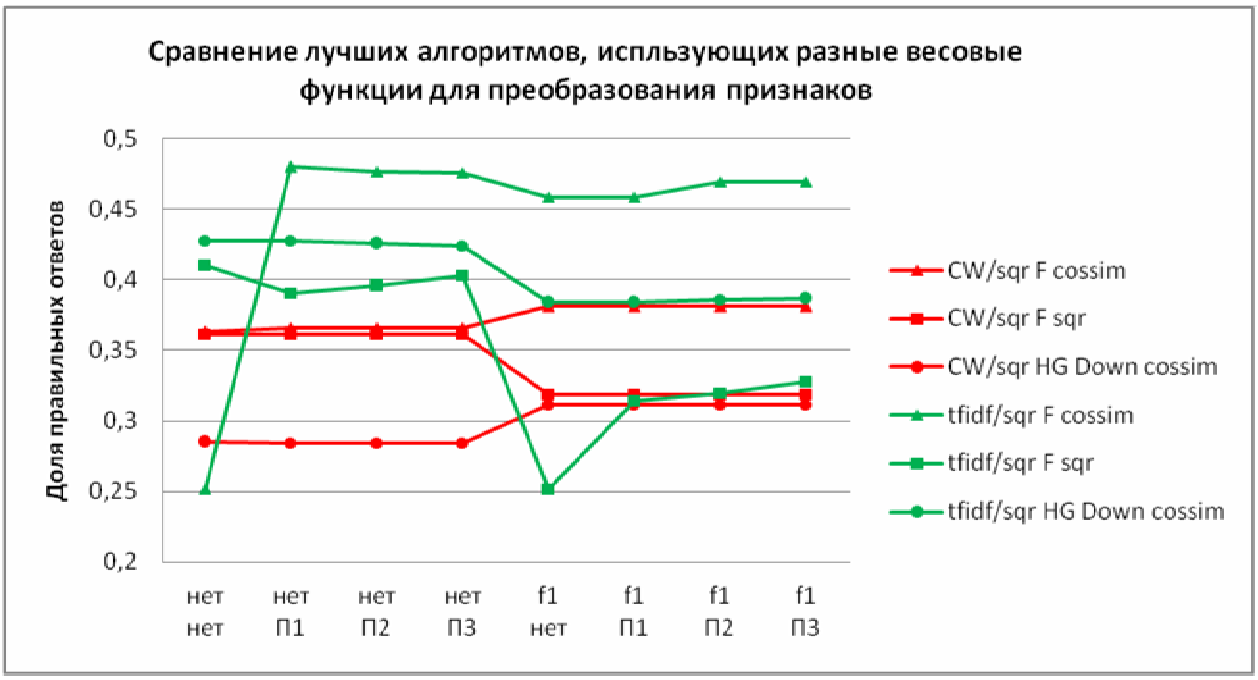
На данном этапе не фиксируется способ отбора признаков и использование функции сглаживания. Замечено, что способ отбора признаков никак не влияет на классификацию. Данное наблюдение можно объяснить тем, что веса $ConfWeight$ проводят неявный отбор признаков. Неинформативным признакам присваивается вес 0. Получается, что 0 значения получают признаки, отбираемые по правила П1-3, и они в любом случае не учитываются при классификации.

Использование функции сглаживания по-разному влияет на различные классификаторы.

В таблице представлены показатели качества наилучших конфигураций алгоритма.

Лучшие результаты								
Алг	Очищ	Норм	Сгл	Метр	A	Макро-r	Макро-p	Макро-F
F	все варианты	CW/sqr	f1	sqr	0,3811	0,2699	0,2163	0,2263
F			нет	cossim	0,3617	0,2722	0,2208	0,2305
HG Down			f1	cossim	0,3116	0,2114	0,1669	0,1728

Далее было произведено сравнение выбранных алгоритмов с аналогичными, но использующими вместо $ConfWeight$ $TF*IDF$. Стоит заметить, что в экспериментах с весовой функцией $TF*IDF$ выбранная конфигурация также давала лучшие результаты. Так что сравнение двух вариантов взвешивания признаков производится по лучшим конфигурациям обеих весовых функций.



Как видно, $TF*IDF$ в большинстве вариантов показывает лучшие результаты по сравнению с $ConfWeight$. В то же время, алгоритмы, использующие $ConfWeight$ менее чувствительны к отбору признаков. Это означает, что такие веса хороши, когда в словаре содержится много «шумовых» термов. Как видно по графику, $ConfWeight$ лучше некоторых алгоритмов с $TF*IDF$, когда не производится явный отбор признаков.

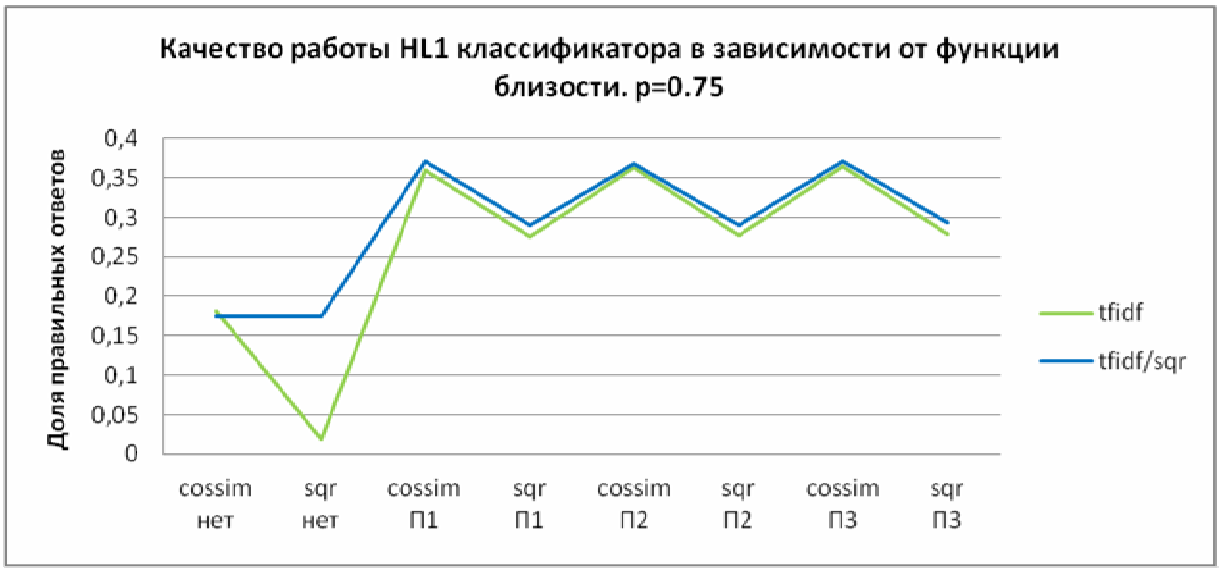
Эксперимент 7: проверка качества алгоритмов с взвешенным $TF*IDF$

Алгоритм			F			HL1		HL2		HG Down		HG Up
						Тип иерархии						
						0-1. orig		2. 2level				
Метрика					cossim				abs			sqr
Веса	tf	tfidf		tfidf/abs		tfidf/sqr	w		w		w	CW	CW/abs	CW/sqr
		idf					p					Tao	
		1	2	3		4	0,25	0,5		0,75		1,96	2,625
Очищение			нет				П1			П2		П3	
Сглаживание				нет				f1				f2	

В эксперименте производится сравнение алгоритмов, использующих взвешенный $TF * IDF$: $TF^p * IDF$. Параметр p принимает значения 0.25, 0.5, 0.75, 1. Последнее значение соответствует обычному $TF * IDF$. В начале все эксперименты проводятся для IDF_2, затем, когда проведен предварительный отбор лучших конфигураций, производится сравнение с IDF_1.

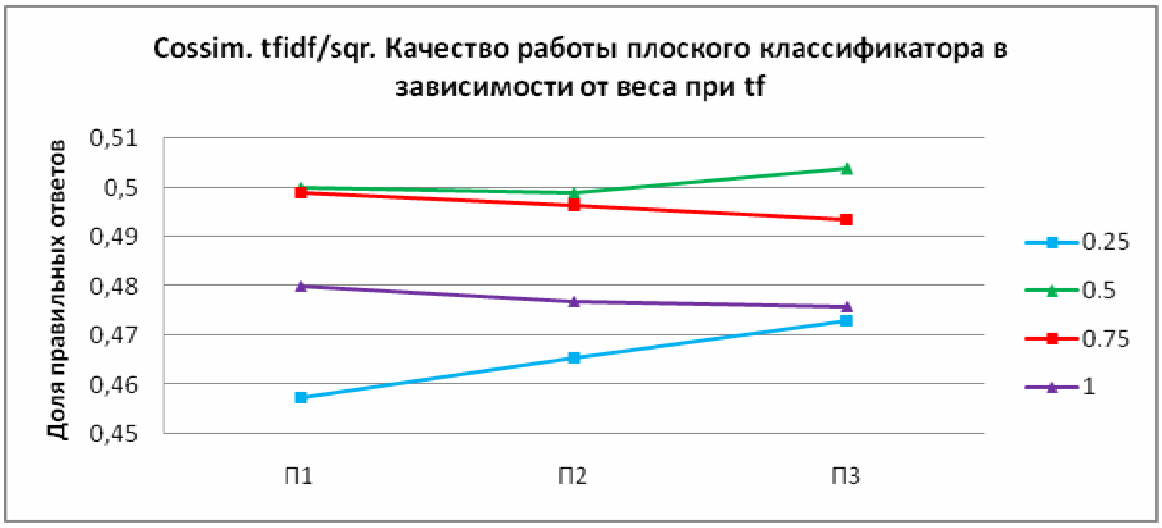
В эксперименте рассматривались те конфигурации алгоритма, которые во всех предыдущих экспериментах показали лучшие результаты. Так, используются только функции близости $Cossim$ и Sqr, не используется нормировка признаков по L_1, не используются сглаживающие функции.

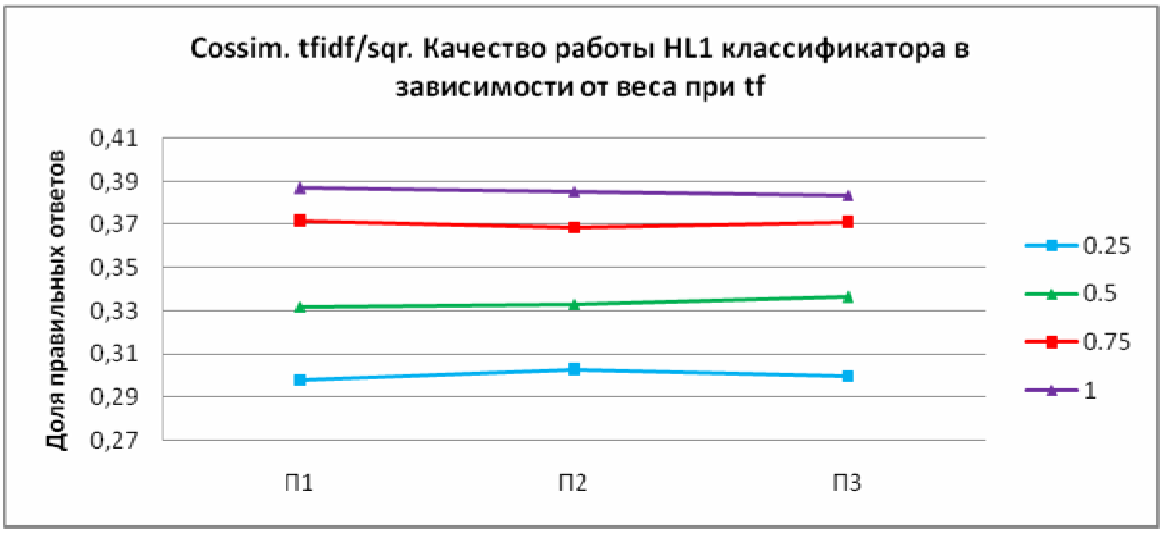
Первые эксперименты с классификаторами F и HL1 показали интересную тенденцию. Для обоих классификаторов и различных значений параметра p графики качества выглядят приблизительно одинаково: нормирование весов признаков по L_2 (tfidf/sqr) дает лучшие результаты, чем веса признаков без нормировки; функция близости $Cossim$ работает лучше, чем функция Sqr; отсутствие отбора признаков ухудшает качество классификации. Все вышеперечисленное демонстрируют графики качества классификатора HL1, значение параметра $p = 0.75$.

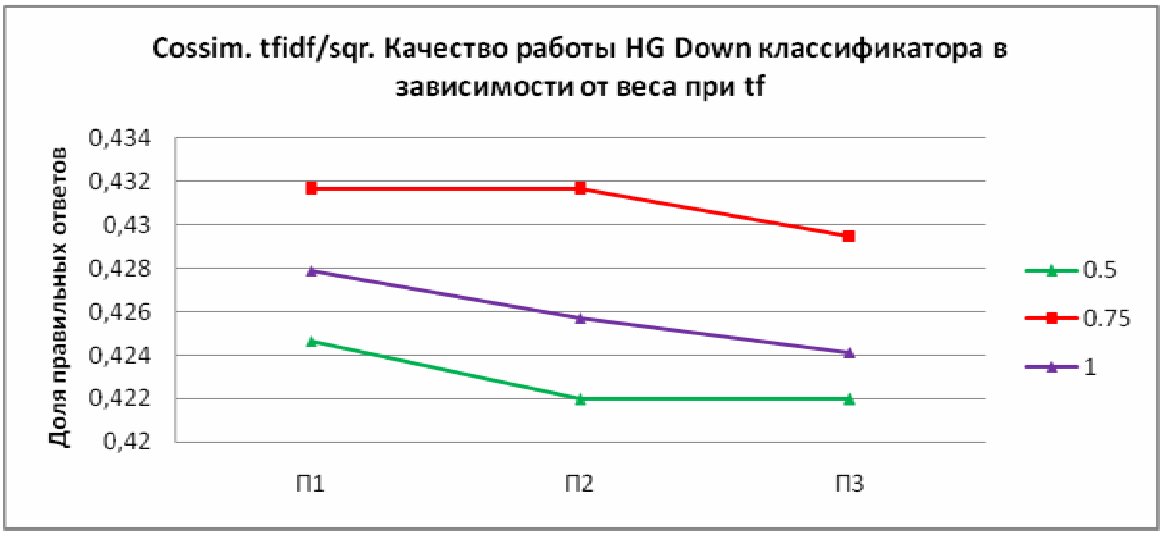


Таким образом, были зафиксированы лучшие параметры: функция близости $Cossim$, нормировка весов – tfidf/sqr, использование правил отбора признаков (пока оставляем все П1, П2, П3).

Далее были проведены сравнения алгоритмов при различных значениях параметра p.

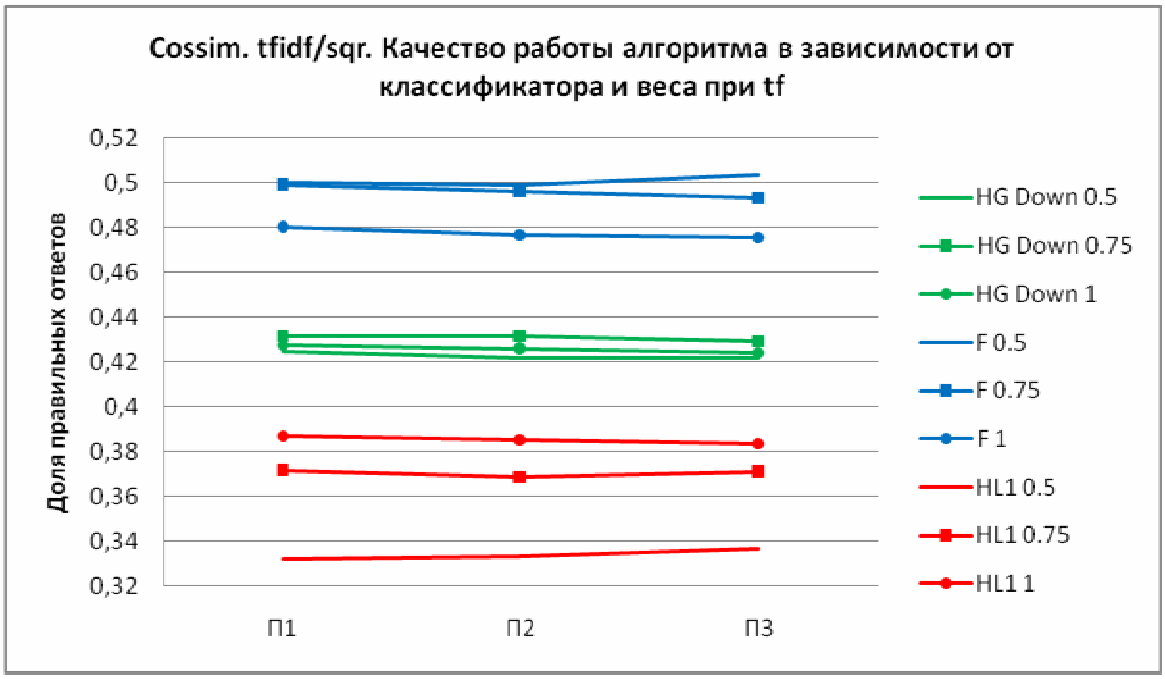






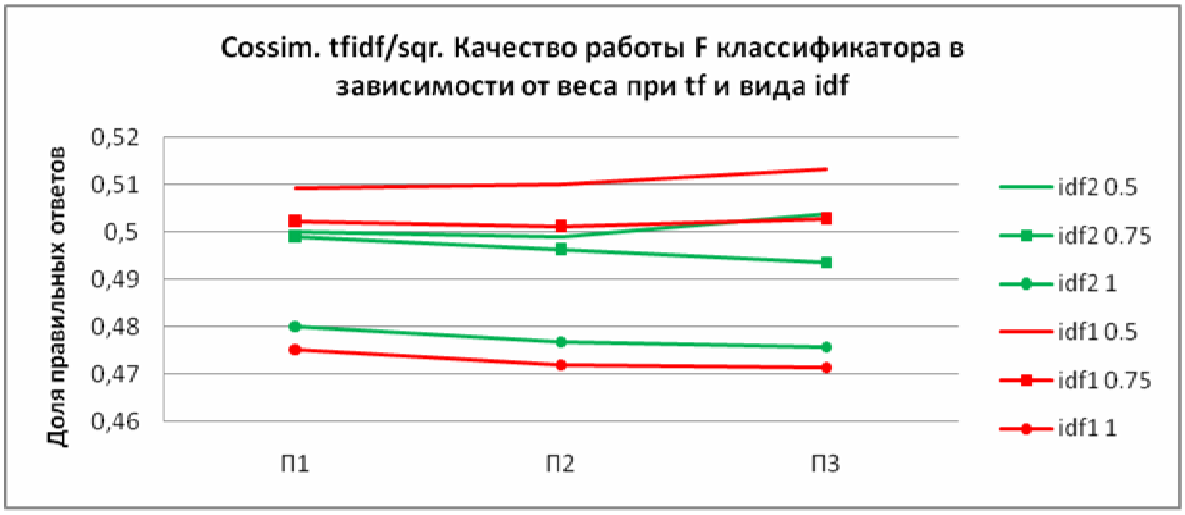
При классификаторах F и HL1 взвешивание с параметром $p = 0.25$ дают наихудшие результаты, поэтому классификатора HG Down с таким параметром уже не рассматривается.

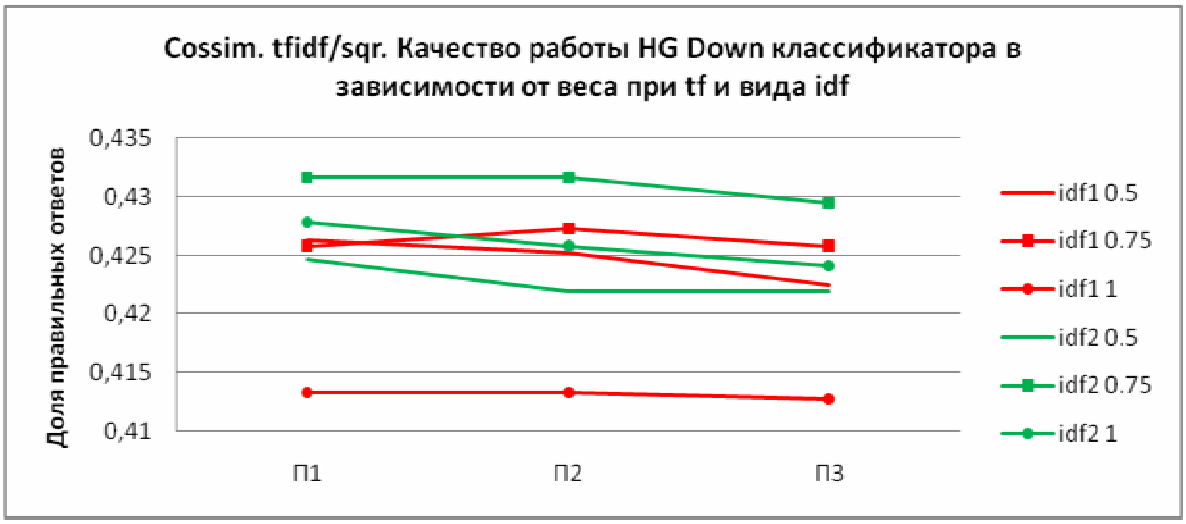
Сравнивая результаты по различным классификаторам, получаем:



Картинка по соотношению качества на разных классификаторах не меняется: плоский остается лучшим, HG Down работает немного хуже, но при этом лучше другого иерархического классификатора.

Далее были рассмотрены уже лучшие классификаторы: F и HG Down и эксперимент проводился с IDF_1.





Даже без изображения всех графиков на единой картинке видно, что алгоритм HG Down продолжает отставать от F. Но в данном случае больше интересно то, что для разных классификаторов лучшими оказались разные варианты IDF.

Способ отбора признаков влияет на классификацию на доли процентов во всех конфигурациях алгоритма. Поэтому сейчас правило отбора не фиксируется.

В таблице ниже приведены результаты работы наилучших алгоритмов.

Лучшие результаты по точности										
Норм	Сгл	Метр	Алг	idf	p	Очищ	A	Макро-r	Макро-p	Макро-F
tfidf/sqr	нет	cossim	HG Down	idf2	0,75	П2	0,43165	0,34020	0,27062	0,28571
						П1	0,43165	0,33891	0,27099	0,28570
						П3	0,42949	0,34058	0,26708	0,28417
			F	idf1	0,5	П3	0,51346	0,43152	0,36201	0,37669
						П2	0,51023	0,42508	0,35743	0,37140
						П1	0,50915	0,42175	0,35408	0,36812

Лучшие результаты по F-мере										
Норм	Сгл	Метр	Алг	idf	p	Очищ	A	Макро-r	Макро-p	Макро-F
tfidf/sqr	нет	cossim	HG Down	idf2	1	П1	0,42788	0,34765	0,26911	0,28772
						П3	0,42411	0,34636	0,26918	0,28766
						П2	0,42573	0,34652	0,26777	0,28662
			F	idf1	0,5	П3	0,51346	0,43152	0,36201	0,37669
					0,75	П3	0,50269	0,43781	0,35964	0,37496
						П2	0,50108	0,43615	0,35748	0,37318

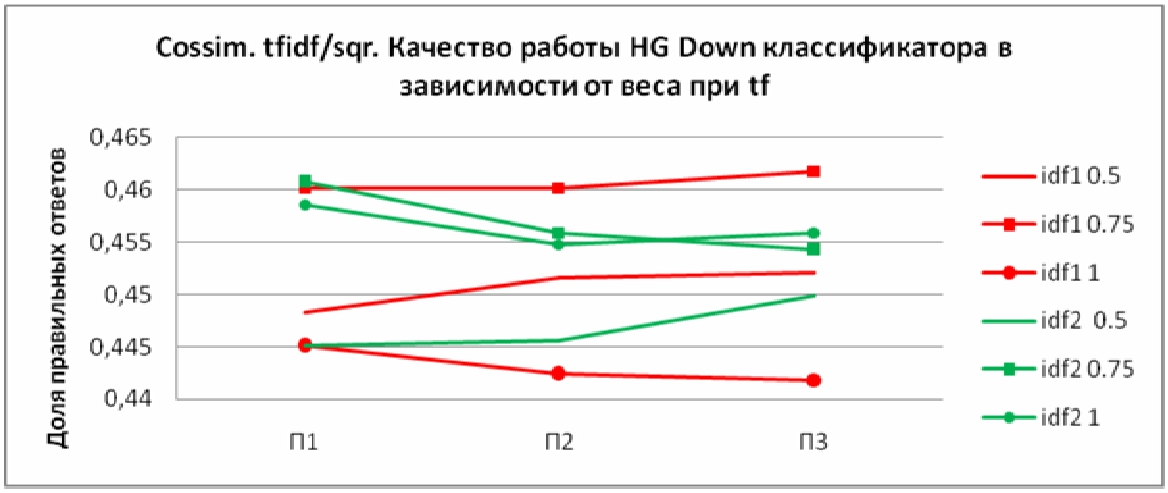
В таблицах цветом выделены ячейки с наилучшими показателями по точности и F-мере. Как видно, лучшие результаты на обеих метриках достигаются на одной конфигурации алгоритма. Наилучшая конфигурация оказалась следующей: плоский классификатор, веса признаков $TF^{0.5} * IDF_1 / Sqr$, очистка признаков по правилу П3, функция близости $Cossim$. Показатели качества алгоритма в такой конфигурации превосходят показатели алгоритмов, представленных в проекте LSHTC [1].

Эксперимент 8

Алгоритм		F		HL1		HL2		HG Down	HG Up
				Тип иерархии					
				0-1. orig		2. 2level			
Метрика				cossim		abs		sqr	
Веса	tf	tfidf	tfidf/abs	tfidf/sqr	w	w	w	CW	CW/abs	CW/sqr
		idf			p			Tao		
		1	2	3	4	0,25	0,5	0,75	1,96	2,625
Очищение			нет			П1		П2	П3	
Сглаживание			нет			f1			f2	

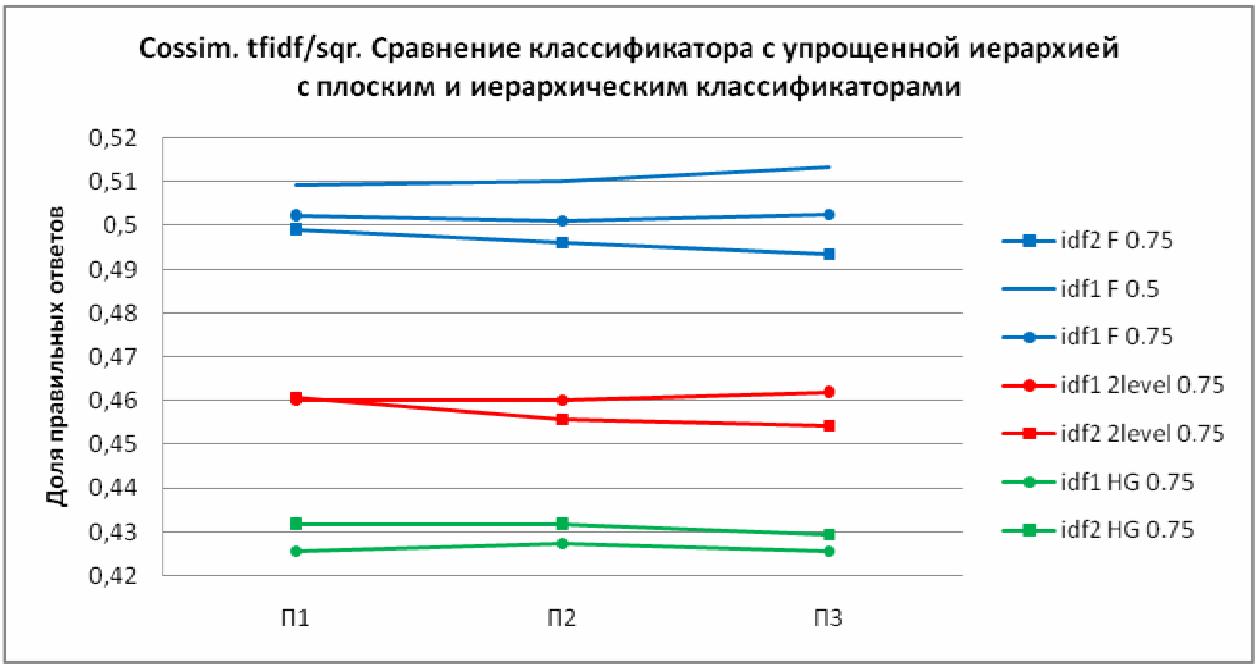
Эксперимент по упрощению иерархической структуры направлен на соединение положительных сторон плоского и иерархического принципов классификации. От упрощенной иерархии ожидается, что она будет работать быстрее плоского классификатора и точнее иерархического с начальной структурой. Поскольку выбор между IDF_1 и IDF_2 неоднозначен, результаты были получены для обоих видов IDF.

Время, затраченное на классификацию, будет рассмотрено ниже, здесь же приведены результаты по точности.



Лучшие результаты по точности получаются при параметре $p = 0.75$ в обоих вариантах IDF.

Далее приведены графики сравнения полученных лучших конфигураций с аналогичными, использующими плоский классификатор и HG Down с начальной иерархией.



Видно, что качество классификации с упрощением иерархии выросло, но все же остается ниже качества плоского классификатора.

Наилучшие результаты упрощенной иерархии приведены в таблице ниже:

Лучшие результаты HG Down с упрощенной иерархией по точности и F-мере									
Норм	Сгл	Метр	idf	p	Очищ	A	Макро-r	Макро-p	Макро-F
tfidf/sqr	нет	cossim	idf1	0,75	П3	0,461787	0,378164	0,310894	0,32377

Время работы алгоритмов: настройка и классификация

В разделе приведены данные по затратам времени на лучшие конфигурации алгоритмов, использующих классификаторы F, HG Down с изначальной и упрощенной иерархией.

Конфигурации по алгоритмам выглядят следующим образом:

Лучшие результаты по точности и F-мере										
Алг	Норм	Сгл	Метр	idf	p	Очищ	A	Макро-r	Макро-p	Макро-F
F	tfidf/sqr	нет	cossim	idf1	0,5	П3	0,51346	0,43152	0,36201	0,37669
HG ориг	tfidf/sqr	нет	cossim	idf2	0,75	П2	0,43165	0,34020	0,27062	0,28571
HG упрощ	tfidf/sqr	нет	cossim	idf1	0,75	П3	0,461787	0,378164	0,310894	0,32377

Отбор признаков и вид IDF дают несущественный прирост ко времени выполнения. Для чистоты эксперимента все вычисления проводились при отборе признаков П3 (сам отбор во времени не учитывается) и с использованием IDF_1.

Измерения проводились для трех видов иерархии: 0) оригинальная, 1) упрощенная (с упразднением всех узлов с одним наследником), 2) двухуровневая.

На вход подавались данные задачи. Описание задачи дано выше.

Алгоритм	Тип иерархии	Время(с)	На обучение	На классификацию
F		53,6302	87,30%	12,03%
HG Down	0	17,8587	31,65%	65,54%
HG Down	1	15,3598	25,83%	70,58%
HG Down	2	15,9929	47,03%	50,14%

Как и ожидалось, плоский классификатор работает в 3-3.5 раза медленнее иерархического. При упрощении иерархии до двух уровней время работы алгоритма увеличивается незначительно. При этом растет качество классификации.

По таблице видно, что плоский классификатор большую долю времени тратит на обучение, тогда как иерархический большую долю тратит на классификацию. Если рассматривать время на классификацию и обучение в абсолютных величинах, то можно заметить, что иерархический алгоритм работает быстрее за счет быстрого обучения, тогда как классификация проходит приблизительно за одно время (около 6 секунд для классификации плоским алгоритмом и 8-11 секунд на классификацию иерархическим алгоритмом).

Параметры компьютера

Эксперименты проводились на 32х битном персональном компьютере с процессором Intel Core2 Duo, частотой 2 ГГц, оперативной памятью 2Гб. Операционная система Windows Vista.

Все алгоритмы, загрузка и выгрузка данных, обработка результатов реализованы в MATLAB 2009b.

Заключение

В настоящей дипломной работе:

- Исследованы методы иерархической классификации текстов, а также возможные модификации стандартных алгоритмов машинного обучения для учета специфики решаемой задачи.
- Программно реализованы все необходимые алгоритмы и модуль для проведения экспериментов.
- Выявлены закономерности изменения качества работы при изменении конфигураций алгоритма, в том числе способа взвешивания признаков, вида сглаживающей функции, правил отбора неинформативных признаков, вида функции близости. Исследованы закономерности, уменьшающие время работы алгоритма.
- Описан алгоритм, решающий задачу проекта LSHTC точнее алгоритмов-победителей проекта.

Наилучший по рассмотренным метрикам качества алгоритм, основанный на центроидном, имеет следующую конфигурацию:

1. плоский классификатор (все конечные классы рассматриваются как равноправные, иерархическая структура не учитывается);
2. функция близости косинус ($Cossim$ (13));
3. очистка признакового пространства от всех термов, которые встречаются менее 3х раз или математическое ожидание частот которых больше 0.95, а дисперсия меньше 0.05;
4. преобразование каждого признака по формуле: $TF^{0.5} * IDF_1$ далее нормировка вектора документа по норме L_2;
5. без обработки функцией сглаживания.

Результат работы такого классификатора на данных задачи:

A	Макро-r	Макро-p	Макро-F
0,51346	0,43152	0,36201	0,37669

Наилучшее соотношение качество/время выполнения алгоритма имеет иерархический алгоритм с упрощенной двухуровневой структурой иерархии.

A	Макро-r	Макро-p	Макро-F	Время выполнения (с)
0,461787	0,378164	0,310894	0,32377	15,9929

Список литературы

[1] Large Scale Hierarchical Text classification (LSHTC) Pascal Challenge
http://lshtc.iit.demokritos.gr/

[2] Open Directory Project
http://www.dmoz.org/

[3] Xiao-Lin Wang, Bao-Liang Lu. Improved Hierarchical SVMs for Large-scale Hierarchical Text Classification Challenge [PDF]
http://lshtc.iit.demokritos.gr/system/files/LSHTC_wang-rev.pdf

[4] Dingquan Wang, Weinan Zhang, Gui-Rong Xue, and Yong Yu. Deep Classifier for Large Scale Hierarchical Text Classification [PDF]
http://lshtc.iit.demokritos.gr/system/files/Turing.pdf

[5] Christophe Brouard. Classification by resonance in an associative network [PDF]
http://lshtc.iit.demokritos.gr/system/files/brouard.pdf

[6] Omid Madani and Jian Huang. Large-Scale Many-Class Prediction via Flat Techniques [PDF]
http://lshtc.iit.demokritos.gr/system/files/Jhuang.pdf

[7] Hassan H. Malik. Improving Hierarchical SVMs by Hierarchy Flattering and Lazy Classification [PDF]
http://lshtc.iit.demokritos.gr/system/files/lshtc_malik.pdf

[8] Youdong Miao and Xipeng Qiu. Hierarchical Centroid-based Classifier for Large Scale Text Classification [PDF]
http://lshtc.iit.demokritos.gr/system/files/XipengQiu.pdf

[9] Cristobal Camarero Coterillo. An Adaptation of Soucy and Mineau weightin for Large Scale Text Classifcation [PDF]
http://lshtc.iit.demokritos.gr/system/files/Nakacristo.pdf

[10] Агеев М.С. Методы машинной рубрикации текстов, основанные на машинном обучении и знаниях экспертов. Диссертация на соискание ученой степени к.ф.-м.н, МГУ, 2004. [PDF]
http://www.cir.ru/docs/ips/publications/2005_diss_ageev.pdf

[11] Sebastiani, F.: Machine learning in automated text categorization, ACM Computing Surveys, vol. 34, pp. 1-47, 2002
http://nmis.isti.cnr.it/sebastiani/Publications/ACMCS02.pdf

[12] Ashwin Pulijala Susan Gauch. Hierarchical Text Classification, 2004
http://citeseer.uark.edu/publications/CITSA2004.pdf

[13] Дунаев Е. В. Автоматическая рубрикация web-страниц в интернет-каталоге с иерархической структурой / Е. В. Дунаев, А. А. Шелестов // Интернет-математика 2005. Автоматическая обработка веб-данных. - М., 2005. - С. 382-398
http://elar.usu.ru/bitstream/1234.56789/1419/1/IMAT_2005_20.pdf

[14] P. Soucy, G. W. Mineau. Beyond TFIDF Weighting for Text Categorization in the Vector Space Model // In Proceedings of the 19th International Joint Conference on Artifcial Intelligence (IJCAI 2005)
http://lvk.cs.msu.su/~bruzz/articles/classification/0304.pdf

[15] Губин, М.В. Модели и методы представления текстового документа в системах информационного поиска: дис. … к.ф-м.н./М.В. Губин. СПб., 2005. 95 с.
http://www.maxgubin.com/articles/thesis.pdf

[16] Salton G, Buckley C. Term-Weighting Approaches in Automatic Text Retrieval. / Information Processing and Management, —1988 — pp. 513-523

[17] Dumais S., Platt J,, Heckerman D., Sahami M. Inductive learning algorithms and representations for text categorization. // In Proc. Int. Conf. on Inform. and Knowledge Manage., 1998

[18] К. В. Воронцов Лекции по методам оценивания и выбора моделей, 2007
http://www.ccas.ru/voron/download/Modeling.pdf