САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Математико-механический факультет

Кафедра системного программирования

Классификация микроблогов с использованием Википедии

Дипломная работа студента 545 группы

Абишева Тимура Маратовича

Научный руководитель		к. ф.-м. н., доцент
	/подпись/	Барашев Д.В.
Рецензент		к. ф.-м. н., доцент
	/подпись/	Шалымов Д. С.
“Допустить к защите”		д.ф.-м.н., профессор
заведующий кафедрой,	/подпись/	Терехов А.Н.

Санкт-Петербург

2012

SAINT PETERSBURG STATE UNIVERSITY

Mathematics Mechanics Faculty

Software Engineering Chair

Micro blog posts classification using Wikipedia

Graduate paper

by

Timur Abishev

Supervisor		PhD, Associated Professor
	.....	D. V. Barashev
Reviewer		PhD, Associated Professor
	.....	D. S. Shalymov
“Approved by”		Professor
Head of Department,	.....	A. N. Terekhov

Saint Petersburg

2012

Содержание

Введение

1. Мотивация

Сервисы микроблогов, такие как, например, “Твиттер”[1], являются одной из тенденций последних лет. Эти сервисы позволяют своим пользователям публиковать записи различного рода, но при этом небольшой длины (в случае сервиса “Твиттер” — 140 символов). На момент апреля 2012 года сервисом пользовалось более 140 миллионов активных пользователей, которые публиковали до 1 миллиарда записей каждые 3 дня [1].

Во время президентских выборов в США в 2008 году “Твиттер” использовался, в частности, для анализа позиций кандидатов [2]. Были попытки прогнозирования кассовых сборов фильмов в США с помощью “Твиттера” [3].

Надо заметить, что активные пользователи сервиса могут получать сотни записей каждый день и далеко не всё могут прочитать сразу. Очень часто возникает желание сгруппировать полученные записи в зависимости от их тематики, чтобы правильно расставить приоритеты чтения, что приводит нас к проблеме фильтрации записей.

С учетом вышесказанного становится понятно, что задача классификации сообщений из “Твиттера” является актуальной и востребованной. При наличии такого механизма классификации мы смогли бы решать разнообразные задачи, такие как: нахождение спама, разделение сообщений по разным тематикам, отделение тематических сообщений от бессодержательных.

[1] https://twitter.com

2. Постановка задачи

Целью данной дипломной работы является создание классификатора записей из микроблогов (а конкретно “Твиттера”), который на основе обучающей выборки будет способен классифицировать записи. Для достижения данной цели были выделены следующие этапы:

1. изучение предметной области и существующих методов классификации записей из микроблогов;
2. реализация алгоритма классификации использующего контекст записей;
3. использование “Википедии” для получения неявных знаний о тексте;
4. тестирование созданного классификатора.

3. Структура работы

В главе 1 дается описание предметной области, вводятся понятия классификации, классификатора и кластеризации. Кроме того в данной главе рассказывается про основные метрики для оценки качества алгоритмов классификации и кластеризации, рассматриваются веб-сервисы “Твиттер” и “Википедия”. В завершении главы описывается одна из простых моделей текста для задач классификации и кластеризации.

В главе 2 рассказывается о нынешнем состоянии исследований в области классификации и кластеризации микроблогов, в том числе и с учетом “Википедии”.

В главе 3 рассказывается об алгоритме классификации, который предлагается в данной работе, приводится общая схема такого классификатора,

его конкретные реализации с использованием стандартной модели текста, а также модели с использованием “Википедии”.

В главе 4 приводится описание экспериментов, проделанных с разными вариантами классификаторов, и анализируется их результат.

В главе 5 дается краткое описание технических решений использованных при создании классификатора.

В части “Заключение” подытоживаются полученные результаты и приводятся направления для дальнейших исследований.

Глава 1

Обзор предметной области

1.1. Используемые веб-сервисы

1.1.1. Твиттер

“Твиттер” является популярной социальной сетью, появившейся в 2006 году, и набравшей с тех пор более 140 миллионов активных пользователей. Основой сервиса являются публикуемые пользователями короткие сообщения, длина которых ограничена 140 символами.

Стоит отметить, что “Твиттер” предоставляет прикладной программный интерфейс (API) для доступа к своим данным[1]. Благодаря этому было создано множество библиотек для работы с данными из “Твиттера” на различных языках программирования.

В силу популярности сервиса, актуальности анализа его данных, а также удобства работы с ним с точки зрения создания программного обеспечения этот сервис был выбран в качестве источника данных.

1.1.2. Википедия

“Википедия” представляет из себя свободную общедоступную энциклопедию. Главной особенностью Википедии является то, что каждый её участник может создавать, удалять и редактировать содержимое статей. Благодаря этой особенности англоязычная “Википедия” включает в себя порядка 4 миллионов статей (на момент мая 2012 года), а, к примеру, русскоязычная — 800 тысяч статей[2].

[1] https://dev.twitter.com/docs/api

[2] http://www.wikipedia.org/

Структурированность и энциклопедическая природа данных в “Википедии” позволяет использовать её при обработке естественных языков, к примеру, в задачах классификации и кластеризации. Данная идея не нова и существует множество исследований на данную тему. К примеру, в работе [4] сделана попытка разбить записи из “Твиттера” на кластеры с помощью “Википедии”.

1.2. Классификация

1.2.1. Формальное определение

Классификацией называется задача машинного обучения с учителем. Целью классификации является разделение некоторого множества объектов на конечное, заранее известное, число классов. Для конечного подмножества объектов, называемого обучающей выборкой, класс известен заранее.

Формально, пусть X — множество описаний объектов, Y — множество номеров (или наименований) классов. Существует неизвестная “целевая зависимость” — отображение $y^*\colon X \to Y$, значения которой известны только на объектах конечной обучающей выборки $X^m = \{(x_1, y_1), \ldots, (x_m, y_m)\}$. Требуется построить алгоритм $a\colon X \to Y$, аппроксимирующий целевую зависимость[5].

Большинство алгоритмов классификации для своей работы требуют описания классифицируемых объектов в виде набора некоторых признаков или характеристик. Эти признаки бывают бинарные, номинальные, порядковые и количественные. К примеру, для текста такими признаками могут быть: “наличие местоимений в тексте”, “наличие сочетания ’ия’ в тексте”, “средняя длина слова меньше 5”, и т.д.

1.2.2. Алгоритмы

Существует множество подходов для реализации алгоритмов классификации. Стоит выделить такие направления, как байесовские классификаторы, нейронные сети, линейные разделители, деревья решений. Далее более подробно рассматриваются алгоритмы, используемые в данной работе.

Наивный байесовский классификатор

Наивный байесовский классификатор (Naive Bayes classifier) основан на использовании теоремы Байеса[3] и предполагает независимость признаков объектов.

Пусть у нас есть объект $D \in X$, имеющий признаки $\{x_1, \ldots, x_n\}$, а также класс $C \in Y$. Умея оценивать $\mathsf{P}(C|D)$ мы бы могли построить классификатор по следующему правилу: $\text{classify}(D) = \text{argmax}_C \ \mathsf{P}(C|D)$.

Оценим $\mathsf{P}(C|D)$ по теореме Байеса:

$$\mathsf{P}(C|D) = \mathsf{P}(C|x_1, \ldots, x_n) = \frac{\mathsf{P}(C) \ \mathsf{P}(x_1, \ldots, x_n|C)}{\mathsf{P}(x_1, \ldots, x_n)}. \tag{1.1}$$

Знаменатель является константой относительно класса C, так как не зависит от него, соответственно в задаче максимизации $\mathsf{P}(C|D)$ мы можем его игнорировать:

$$Z = \frac{1}{\mathsf{P}(x_1, \ldots, x_n)} \tag{1.2}$$

$$\mathsf{P}(C|D) = Z\,\mathsf{P}(C)\,\mathsf{P}(x_1, \ldots, x_n|C). \tag{1.3}$$

Используя предположение о независимости признаков можно полу-

[3] http://ru.wikipedia.org/wiki/Теорема_Байеса

чить следующий результат:

$$\mathsf{P}(x_1, \ldots, x_n|C) = \prod_{i=1}^{n} \mathsf{P}(x_i|C) \tag{1.4}$$

$$\mathsf{P}(C|D) = Z\,\mathsf{P}(C) \prod_{i=1}^{n} \mathsf{P}(x_i|C). \tag{1.5}$$

Величины $\mathsf{P}(C)$ и $\mathsf{P}(x_i|C)$ можно оценить, подсчитав соответствующие относительные частоты в обучающей выборке. Таким образом мы получили алгоритм классификации.

Метод опорных векторов

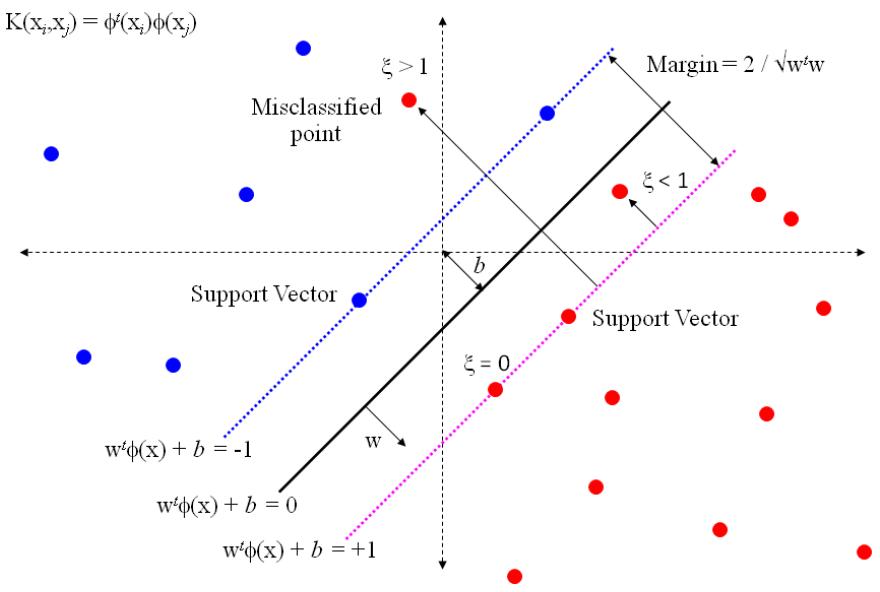

Рис. 1.1. Метод опорных векторов

Данный метод (support vector machines) применим для двух классов. Основной идеей метода является попытка разделения исходной выборки с помощью гиперплоскости. В случае линейно разделимой выборки, методом ищется гиперплоскость с максимальным “зазором” — суммой расстояний от плоскости до ближайших представителей обоих классов. В случае отсутствия линейной разделимости вводится понятие штрафа (C). Результатом работы алгоритма является линейный классификатор задаваемый двумя

параметрами: вектором $\mathbf{w}$ и константой $\mathbf{b}$. Сам алгоритм классификации задается следующим образом: $\text{classify}(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} - b)$, где $\cdot$ — скалярное произведение.

Стоит отметить, что в некоторых случаях применяют метод опорных векторов с каким-либо ядром. Главной идеей этого метода является замена скалярного произведения во всех операциях на какую-либо нелинейную функцию, называемую ядром. При использовании ядра, выборки, неразделимые линейным ядром, могут стать разделимыми. Однако в данной работе все случаи ядер, отличных от линейного, не рассматриваются. В частности в работе [6] сделан вывод, что для классификации текстов лучше всего подходит линейное ядро.

Для классификации более чем на два класса используется техника один-против-одного (one-against-one). Для каждой пары классов строится разделяющая гиперплоскость, в качестве итогового класса для документа выбирается тот, за который проголосовало большее число классификаторов.

Деревья решений

К данному классу алгоритмов классификации относятся алгоритмы строящие итоговый классификатор в виде дерева. Во внутренних вершинах построенного дерева записываются условия на классифицируемый объект, в листьях — результат классификации. Пример того, как может выглядеть дерево решения для задачи классификации людей на выживших и не выживших во время катастрофы "Титаника"[4], изображен на Рис. 1.2. Преимуществом деревьев решений является их интерпретируемость для человека — результат можно наглядно отобразить и проанализировать.

[4] http://ru.wikipedia.org/wiki/Титаник

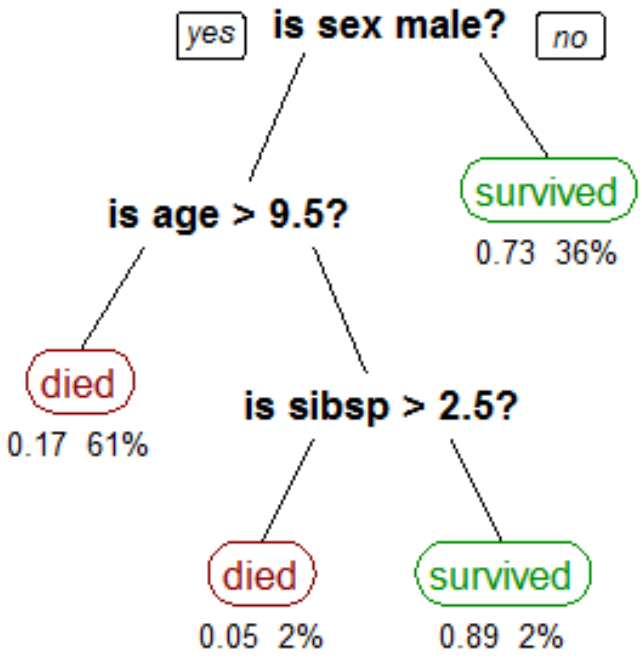

Рис. 1.2. Пример дерева принятия решений

1.3. Кластеризация

1.3.1. Формальное определение

Кластеризация — задача машинного обучения без учителя. В данной задаче имеется множество объектов. Целью кластеризации является выявление групп похожих объектов, разделения объектов на классы. Главным отличием кластеризации от классификации следует считать то, что никакие характеристики групп не задаются, группы определяются полностью автоматически на основе попарной схожести объектов.

Формально, пусть X — множество объектов, Y — множество номеров (имён, меток) кластеров. Задана функция расстояния между объектами $\rho(x, x')$. Имеется конечная выборка объектов $X^m = \{x_1, \ldots, x_m\} \subset X$. Требуется разбить выборку на непересекающиеся подмножества, называемые

“кластерами”, так, чтобы каждый кластер состоял из объектов, близких по метрике ρ , а объекты разных кластеров существенно отличались. При этом каждому объекту $x_i \in X^m$ приписывается номер кластера y_i .

“Алгоритм кластеризации” — это функция $a\colon X \to Y$, которая любому объекту $x \in X$ ставит в соответствие номер кластера $y \in Y$. Множество Y в некоторых случаях известно заранее, однако чаще ставится задача определить оптимальное число кластеров, с точки зрения того или иного “критерия качества” кластеризации [7].

Алгоритмы кластеризации, как и алгоритмы классификации, обычно имеют дело с некоторыми признаками объектов пространства X.

1.3.2. Алгоритмы

Входными данными для алгоритмов кластеризации выступают либо признаковые описания объектов, либо матрица попарных расстояний между объектами. Можно выделить следующие методы кластеризации: на основе графов, статистические, иерархические и на основе нейронных сетей. Стоит упомянуть, что существует задача нечеткой кластеризации, в которой, объект принадлежит кластерам с некоторой вероятностью. Далее подробней описаны алгоритмы использованные в этой работе.

Метод k-средних

В данном алгоритме количество кластеров фиксировано величиной k. Задачей данного алгоритма является минимизация величины

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2 , \tag{1.6}$$

где $\mathbf{S}$ — все возможные разбиения элементов X на k кластеров, а μ_i является центром масс кластера S_i.

Задача поиска глобального минимума является NP-трудной [8], однако для нее существует широко используемый алгоритм приближенного решения, находящий локальный оптимум.

Зафиксируем каким-либо образом (например случайным) k первоначальных центров кластеров $m_1^{(1)}, \ldots, m_k^{(1)}$. Далее алгоритм состоит из двух повторяющихся шагов:

Шаг присваивания Каждому объекту из выборки X сопоставляется кластер, центр которого находится к нему ближе всего:

$$S_i^{(t)} = \left\{ x_p : \left\| x_p - m_i^{(t)} \right\| \leq \left\| x_p - m_j^{(t)} \right\| \ \forall \ 1 \leq j \leq k \right\} \tag{1.7}$$

Шаг обновления На основе объектов каждого кластера считаются новые центры:

$$\mathbf{m}_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{\mathbf{x}_j \in S_i^{(t)}} \mathbf{x}_j \tag{1.8}$$

Процесс заканчивается в тот момент, когда центры кластеров перестают обновляться (с какой-либо точностью). Каждый шаг алгоритма делает $O(nkt)$ действий, где n — общее количество объектов, а t — время вычисления расстояния между одной парой объектов. Данный алгоритм имеет полиномиальное количество итераций [9]. На практике количество итераций обычно ограничивают.

XMeans

Данный алгоритм является некоторым обобщением алгоритма k-средних и использует его в своей реализации. Одним из главных отличий данного алгоритма можно назвать отсутствие требования точного количества искомых кластеров, задается лишь требуемый диапазон значений для количества кластеров. Более подробно можно прочитать в [10].

1.4. Модель текста для классификации и кластеризации

Вышеописанные методы классификации и кластеризации применимы, если у исходных объектов есть какие-либо признаки, как это описывалось ранее. Таким образом возникает задача выделения признаков у текста.

Для решения данной задачи существует множество подходов. Опишем здесь один из самых простых. Пусть признаками в результирующем пространстве будут слова, а значениями данных признаков — $\mathrm{tf} * \mathrm{idf}$ данного слова относительно данного документа, где $\mathrm{tf} * \mathrm{idf}$ определен следующим образом:

$$\mathrm{tf}(t, d) = \frac{n_i}{\sum_k n_k} \tag{1.9}$$

$$\mathrm{idf}(t, D) = \log \frac{|D|}{|d \in D : t_i \in d|} \tag{1.10}$$

$$\mathrm{tf} * \mathrm{idf}(t, d, D) = \mathrm{tf}(t, d) \times \mathrm{idf}(t, D), \tag{1.11}$$

где t текущее слово, n_i число вхождений текущего слова в текущий документ, $\sum_k n_k$ общее количество слов в текущем документе, d текущий документ, D — все документы в выборке. Величина tf характеризует частотность слова (терма) относительно документа, величина idf — долю документов содержащих данное слово, таким образом для популярных слов величина idf, и, соответственно $\mathrm{tf} * \mathrm{idf}$, уменьшается.

Стоит отметить, что имеет смысл преобразовывать входной документ; к примеру, приводить слова к нормальной форме. Для данных целей служит стемминг — процесс нахождения основы слова[5]. Не лишено смысла применения латентно-семантического анализа[6], для предобработки объектов. С помощью данного метода происходит выделение тематических при-

[5] http://ru.wikipedia.org/wiki/Стемминг

[6] http://en.wikipedia.org/wiki/Latent_semantic_analysis

знаков из текста. Похожий метод по выделению тематических признаков из текста с помощью метода LDA использовался в работе [11].

1.5. Методы оценки алгоритмов классификации

Пусть есть построенный алгоритм классификации a. Для оценки качества данного алгоритма нам понадобиться дополнительная, так называемая, тестовая выборка T. Для данной выборки мы будем знать верные значения классов элементов. Опишем некоторые величины характеристики качества нашего алгоритма классификации:

$$\text{precision}(c) = \frac{|d \in T\colon class(d) = a(d) = c|}{|d \in T\colon a(d) = c|} \tag{1.12}$$

$$\text{recall}(c) = \frac{|d \in T\colon class(d) = a(d) = c|}{|d \in T\colon class(d) = c|} \tag{1.13}$$

$$\text{F}(c) = \frac{2\text{precision}(c)\text{recall}(c)}{\text{precision}(c) + \text{recall}(c)}, \tag{1.14}$$

другими словами точность (precision) — это число элементов, квалифицированных правильно, по отношению к числу элементов, квалифицированных вообще, полнота (recall) — это число элементов, квалифицированных правильно, по отношению к общему размеру класса, а F-мера (F) — среднее гармоническое точности и полноты.

Недостатком этих величин является то, что они являются специфичными для классов. Введем обобщающие величины, характеризующие построенный алгоритм в целом:

$$\text{precision}_{\text{macro}} = \frac{1}{|C|}\sum_{c \in C}\text{precision}(c) \tag{1.15}$$

$$\text{recall}_{\text{macro}} = \frac{1}{|C|}\sum_{c \in C}\text{recall}(c) \tag{1.16}$$

$$\text{F}_{\text{macro}} = \frac{1}{|C|}\sum_{c \in C}\text{F}(c). \tag{1.17}$$

Данные величины можно использовать для оценки качества построенного классификатора.

1.6. Методы тестирования классификаторов

В простейшем случае нам дана как обучающая, так и тестовая выборка. Однако бывают случаи, когда тестовая выборка не дана. В этом случае разумно использовать часть обучающей выборки в качестве тестовой.

Одним из примеров использования такого подхода является "перекрёстная проверка" (cross-validation). В данном методе обучающая выборка разбивается на k частей, далее происходим обучение классификатора на $k-1$ части, и тестирование на оставшейся части. Процедура повторяется k раз, а полученные результаты оценки качества, например, усредняются.

Глава 2

Обзор литературы

На текущий момент существует множество исследований связанных с вопросами классификации и кластеризации, как с использованием “Википедии”, так и в контексте микроблогов. Существует работы классификации записей из ‘Твиттера‘ по их тональности (sentiment analysis). К примеру, в работе [12] сообщения классифицируются по своей эмоциональной тональности; в качестве алгоритмов классификации используются метод опорных векторов, наивный байесовский метод, метод максимальной энтропии. В качестве признаков текста используются n-граммы и части речи.

Стоит отметить работу [11]. В ней авторы сделали попытку кластеризовать записи из “Твиттера” с помощью алгоритма LDA[1]. Полученные кластеры были разделены на 5 категорий: substance, status, style, social и other.

Во многих работах сделаны попытки улучшения кластеризации с помощью “Википедии”. Так в работе [4] использован сравнительно простой подход: для каждого сообщения в выборке находится наиболее релевантная статья Википедии, далее вводится метрика схожести двух статей на основе расстояния между категориями этих статей.

В работах [13] и [14] соответственно с помощью “Википедии” решаются задачи определения смысловой схожести слов, выделения ключевых слов.

Конкретно вопрос классификации записей из микроблогов рассматривается в работе [15, 16]. В первой работе рассматривается задача классификации на классы новостей, событий, мнений, предложений и личных сообщений. Во второй работе делается попытка классифицировать записи

[1] http://en.wikipedia.org/wiki/Latent_Dirichlet_allocation

на два вида классов: на новостные, личные и от компаний, а также на мнения и факты. Также данная работа примечательна большим количеством алгоритмов участвующих в тестировании.

Глава 3

Описание алгоритма

3.1. Общая схема

Поставим задачу классификации. Пространством объектов для классификации будет являться пространство $Tweets$. Множеством классов — C. Также у нас есть обучающая и тестовая выборки.

Имея алгоритм классификации из пространства текстов ($Texts$) в множество C, можно получить и алгоритм классификации для пространства $Tweets$. Назовем данный алгоритм классификации (из текстов в классы) базовым алгоритмом base_classify: $Texts \to C$. Как было сказано на основе этого алгоритма можно построить алгоритм классификации записей из "Твиттера" simple_classify: $Tweets \to C$.

Пусть нам дан алгоритм кластеризации сообщений из "Твиттера" на какое-либо пространство Y clusterize: $Tweets \to Y$. Определим новый алгоритм классификации сообщений из Твиттера:

$$\text{context\_classify}(tweet) =$$
$$\underset{C}{\operatorname{argmax}} \ |\{neighbor \in neighbors\colon \text{clusterize}(neighbor) = \text{clusterize}(tweet),$$
$$\text{simple\_classify}(neighbor) = C\}|,$$

где $neighbors$ — это какие-либо сообщения автора исходного сообщения для классификации $tweet$. В данной работе такими сообщениями считаются последние n сообщений автора. Говоря неформальным языком, мы будем относить новую запись к тому же классу, к которому отнесли весь кластер схожих записей того же автора.

Итак, менять характеристики этого алгоритма можно меняя базовый

алгоритм класификации (simple_classify), алгоритм кластеризации (clusterize) и модель выделения признаков из текста, параметр n.

3.2. Модель текста на основе “Википедии”

Пример выделения признаков из текста был указан в секции 1.4. В данной секции мы хотели бы привести пример альтернативного метода, основанного на Википедии. Подход описанный в данной секции частично совпадает (в разделе нахождения релевантных статей) с подходом описанным в работе [17]. В частности, выбор констант в формулах обусловлен выводами в статье.

Хотелось бы обратить внимание на мотивацию использования “Википедии” для выделения признаков из текста. При использовании стандартной модели текста, к примеру, для задачи кластеризации, два текста становятся похожими только в случае использования общих слов. В случае коротких текстов (таких как в микроблогах), такие совпадения редки, и нам бы хотелось понимать, что записи содержащие слово java и слово python, являются похожими друг на друга.

Опишем алгоритм выделения признаков из текста. Данный алгоритм будет состоять из двух шагов: выделения релевантных данному тексту статей в “Википедии” и построению признаков по данным статьям.

3.2.1. Выделение релевантных статей

Найдем статьи на которые есть ссылки по словам участвующим в тексте. Для каждой статьи подсчитаем два значения: linking probability и overlapping rate. Пусть, к примеру, статья у нас про язык “Haskell”[1], а слово — “Haskell”. Первое значение — отношение количества ссылок веду-

[1] http://ru.wikipedia.org/wiki/Haskell

щих на статью к общему количеству ссылок состоящих из тех же слов (в нашем примере это отношение количества ссылок в “Википедии” с текстом “Haskell” на страницу про язык “Haskell” к общему количеству ссылок с текстом “Haskell”), второе значение — размер пересечения текста записи и текста статьи из “Википедии” по отношению к размеру текста сообщения с учетом стемминга. Итоговым показателем (confidence) соответствия страницы из “Википедии” данному тексту будет среднее арифметическое полученных linking probability и overlapping rate, таким образом $0 \leqslant \text{confidence} \leqslant 1$. Далее выбирается n страниц с наибольшим confidence, но большим 0.3 согласно статье [17]. Данные статьи и являются релевантными для данного текста.

3.2.2. Выделение признаков по статьям

Для каждой страницы в “Википедии” определены так называемые “категории” — страницы с помощью которых организуются данные в “Википедии”. К примеру, категориями страницы соответствующей языку программирования “Haskell”[2], являются “Языки программирования по алфавиту”, “Haskell” и “Языки программирования семейства Haskell”. В свою очередь для категорий определены над-категории — категории содержащие данную категорию в качестве подкатегории. Категория “Языки программирования семейства Haskell”[3] является под-категорией следующих категорий: “Семейства языков программирования”, “Функциональные языки программирования”. Основываясь на этих данных можно определить следующие

[2] http://ru.wikipedia.org/wiki/Haskell

[3] ru.wikipedia.org/wiki/Категория:Языки_программирования_семейства_Haskell

множества:

$$\mathrm{P}(category) = \text{все над-категории категории } category,$$

$$\mathrm{C}_0(page) = \text{все непосредственные категории страницы } page,$$

$$\mathrm{C}_{depth}(page) = \mathrm{C}_{depth-1}(page) \bigcup \Big\{ \bigcup_{c \in \mathrm{C}_{depth-1}(page)} \mathrm{P}(c) \Big\}.$$

Зафиксируем какое-либо значение max_depth. Теперь мы можем принять категории в “Википедии” в качестве бинарных признаков нашего текста (есть данная категория в множестве $\mathrm{C}_{max\_depth}(page)$ или её в нем нет). В данной работе max_depth было выбрано равным 4, так как такое значение по результатам работы [18] давало наилучший результат.

Глава 4

Эксперименты

4.1. Тестовая выборка

Для тестирования разработанной схемы для создания классификаторов необходимо создать тестовую выборку. Данная глава описывает четыре набора данных использованных для тестирования.

4.1.1. Тематические классы

Данный тестовый набор состоит из следующих классов:

- Физика
- Химия
- Биология
- Математика
- Программирование

Для каждого из классов были выбраны несколько пользователей “Твиттера”, которые пишут на данную тему. Они перечислены в Таблице A.1. Далее для каждого из пользователей в выборку вошли случайные 100 записей из последних 2000.

В качестве способа тестирования было выбрано два варианта: перекрёстная проверка с разделением на 4 части и тестовая выборка составленная из записей пользователей отличных от тех, что использовались в обучающей выборке.

4.1.2. Целевые классы

Данный тестовый набор частично взят из работы [16]. Он состоит из классов:

- Новостных сообщений
- Персональных сообщений
- Сообщений из официальных “Твиттер” аккаунтов компаний

Как и предыдущий, этот тестовый набор составлялся на основе пользователей “Твиттера”. В Таблице A.2 перечислены пользователи с классами к которым они были отнесены.

В качестве способа тестирования была выбрана перекрёстная проверка с разделением на 4 части и тестовая выборка составленная из записей пользователей отличных от тех, что использовались в обучающей выборке.

4.2. Алгоритмы

4.2.1. Простая классификация

К данной секции относятся следующие классифицирующие алгоритмы:

- наивный байесовский,
- метод опорных векторов,
- метод J48 для построения деревьев принятия решений.

Все эти алгоритмы на вход принимают объекты из пространства признаков, так что использовалось два метода для выделения признаков из текста:

- простая модель, описанная в секции 1.4;
- модель на основе “Википедии” (далее — энциклопедическая), описанная в секции 3.2.

В общем получается 6 вариантов алгоритма.

4.2.2. Классификация с учетом контекста

В данной секции использовался предложенный в работе алгоритм классификации. В качестве базового алгоритма были использованы те же алгоритмы, что и в предыдущей секции, аналогично для алгоритма выделения признаков из текста. В качестве кластеризующих были использованы следующие алгоритмы:

- k-means на 20 кластеров,
- k-means на 100 кластеров,
- xmeans с параметрами от 20 до 100 кластеров.

Во всех алгоритмах в качестве метрики для расстояния использовалась евклидова метрика[1]. Количество последних сообщений для кластеризации — 2000.

В общем получается 18 вариантов алгоритма.

4.3. Результаты экспериментов

Для начала хотелось бы отметить, что метод опорных векторов продемонстрировал наилучшие результаты на наших тестовых данных (рисунки

[1] http://ru.wikipedia.org/wiki/Евклидова_метрика

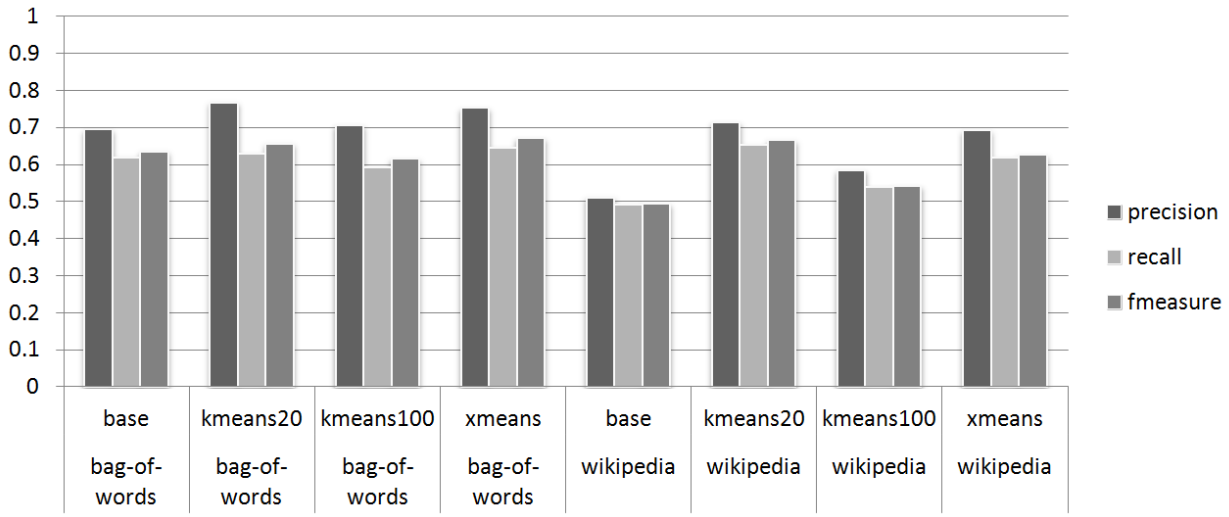

Рис. 4.1. Тематические классы: тестовая выборка; наивный байесовский классификатор

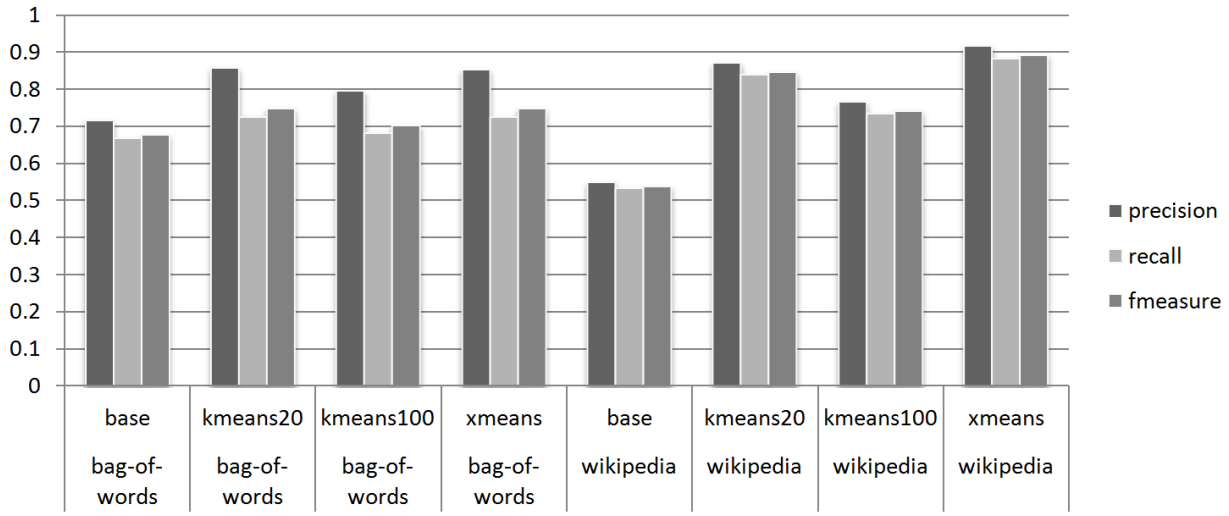

Рис. 4.2. Тематические классы: тестовая выборка; метод опорных векторов

4.2, 4.5, 4.8, 4.11). Кроме того, метод опорных векторов больше всех выигрывает от использования контекста в классификаторе (это лучше заметно на рисунках соответствующих тематической тестовой выборке: 4.2 и 4.5). Меньше всех выигрывает, а иногда даже проигрывает от использования контекста наивный байесовский классификатор. Также можно видеть не самый успешный результат алгоритма J48 в случае использования простой модели текста (Рис. 4.6).

При использовании энциклопедической модели теста, к сожалению,

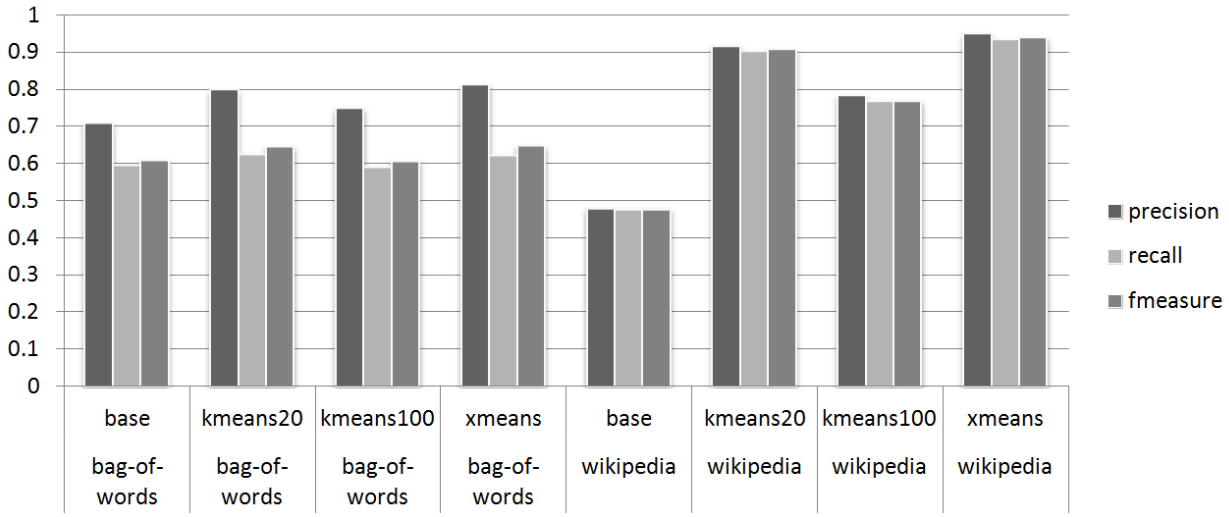

Рис. 4.3. Тематические классы: тестовая выборка; алгоритм построения деревьев решений J48

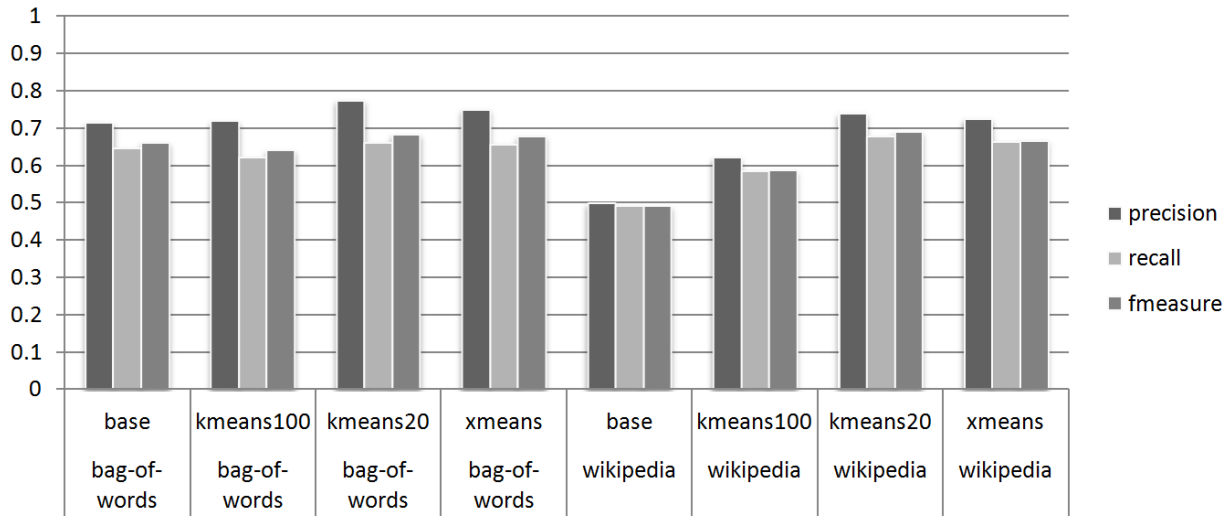

Рис. 4.4. Тематические классы: перекрёстная проверка; наивный байесовский классификатор

результат простых классификаторов значительно ухудшился. Этот результат показывает, что данный аспект, по всей видимости, нуждается в улучшении. Однако стоит отметить, что использование контекста в случае энциклопедической модели позволяет получить значительное (порядка 20% на Рис. 4.2) улучшение показателей классификатора. Связка признаков из “Википедии”, использования контекста и метода опорных векторов проде-

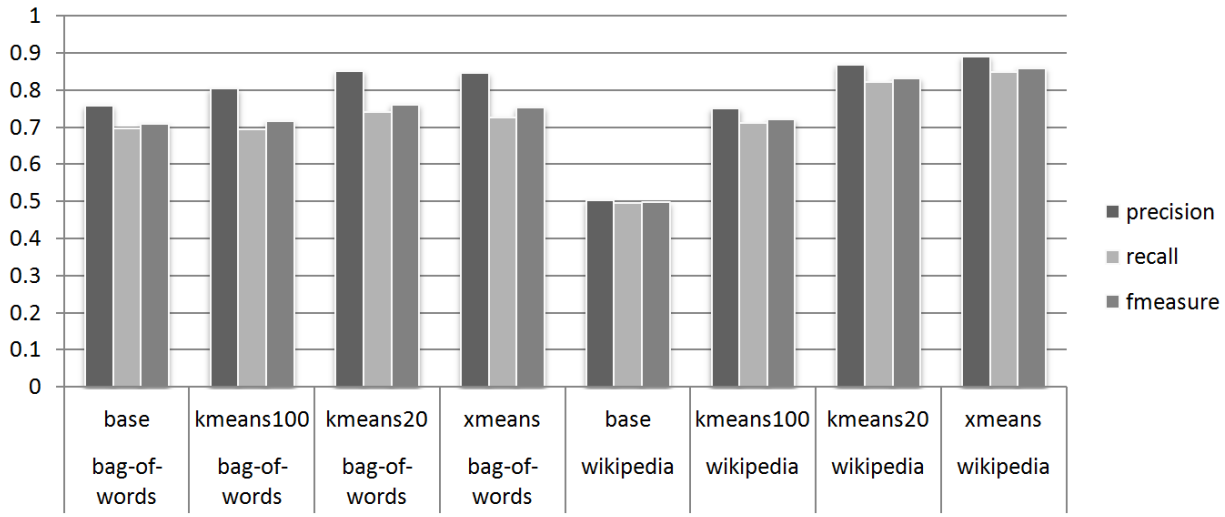

Рис. 4.5. Тематические классы: перекрёстная проверка; метод опорных векторов

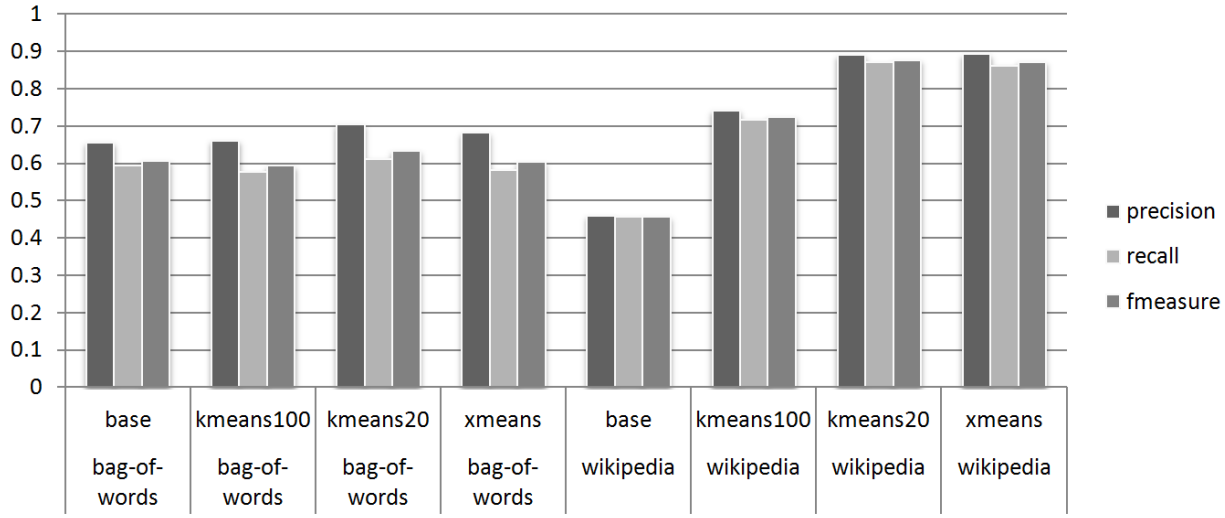

Рис. 4.6. Тематические классы: перекрёстная проверка; алгоритм построения деревьев решений J48

монстрировала наилучший результат в данной работе.

Стоит отметить, что при использовании контекста, улучшение на первом наборе данных было больше чем на втором. Это выглядит разумным в свете того, что кластеризация более вероятно объединяет сообщения похожие тематически, чем по каким-либо других критериям.

Из алгоритмов кластеризации наиболее предпочтительно выглядит XMeans. Именно этот алгоритм рекомендуется использовать при исполь-

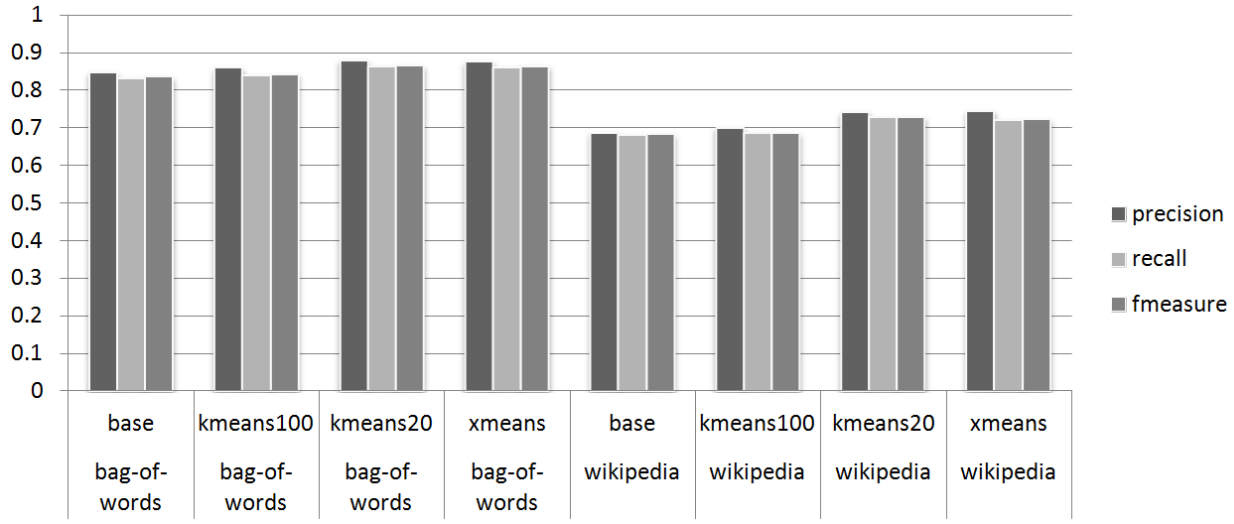

Рис. 4.7. Целевые классы: тестовая выборка; наивный байесовский классификатор

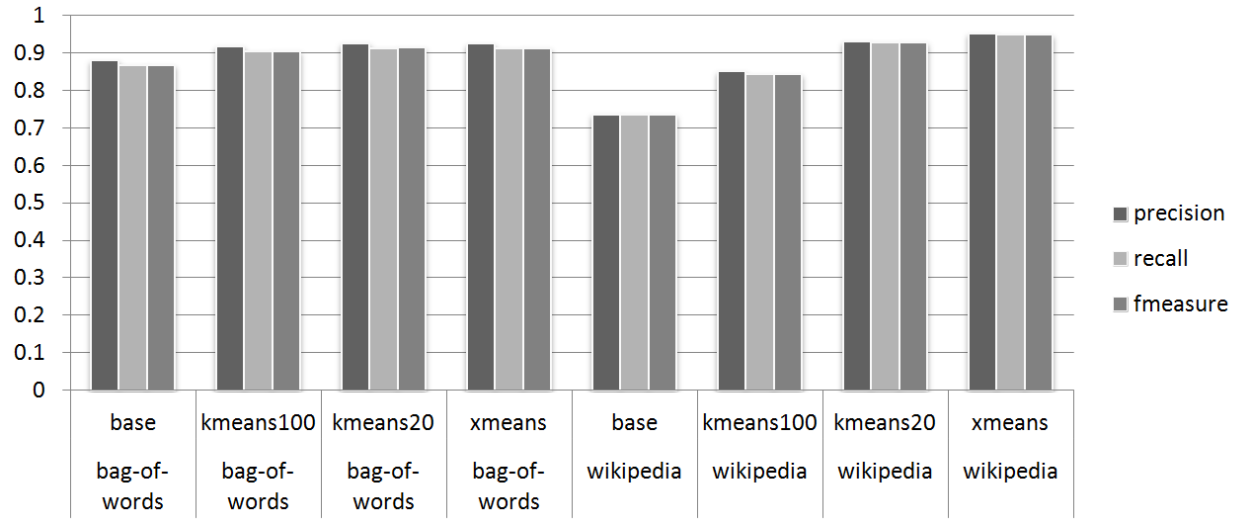

Рис. 4.8. Целевые классы: тестовая выборка; метод опорных векторов

зовании контекстной классификации.

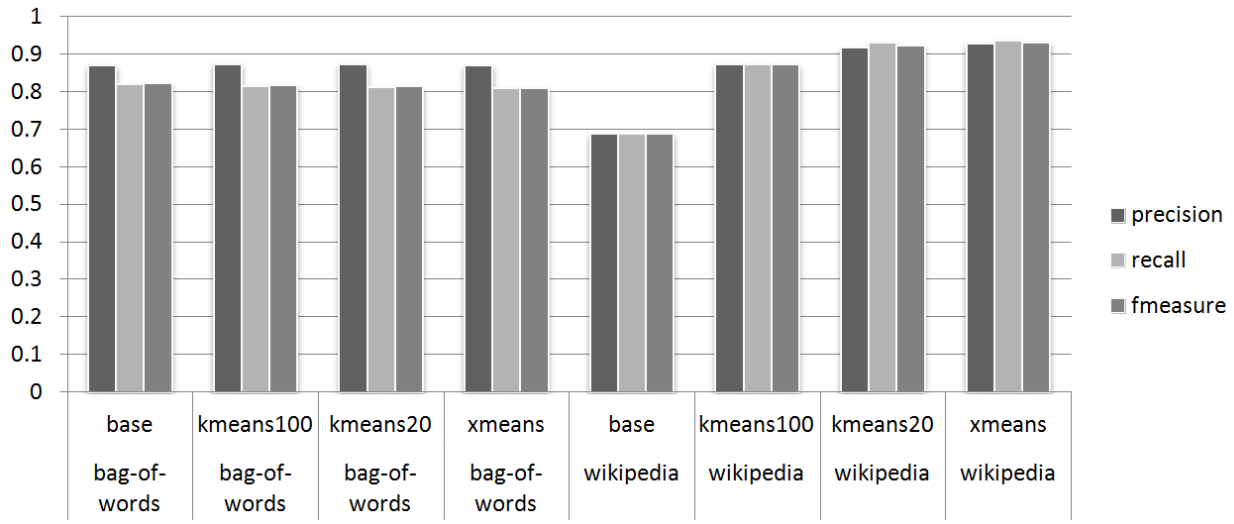

Рис. 4.9. Целевые классы: тестовая выборка; алгоритм построения деревьев решений J48

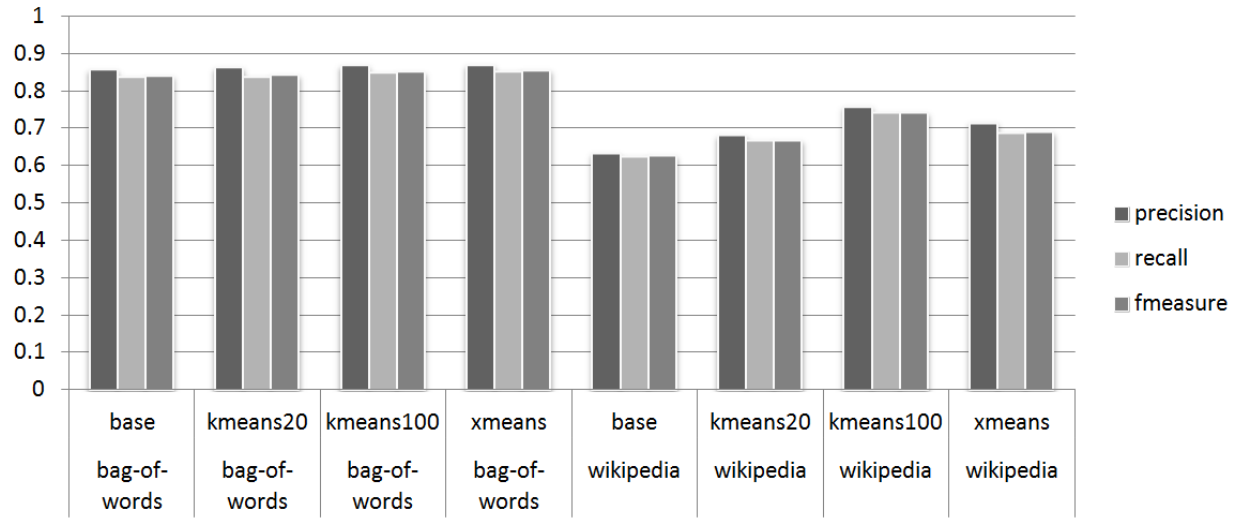

Рис. 4.10. Целевые классы: перекрёстная проверка; наивный байесовский классификатор

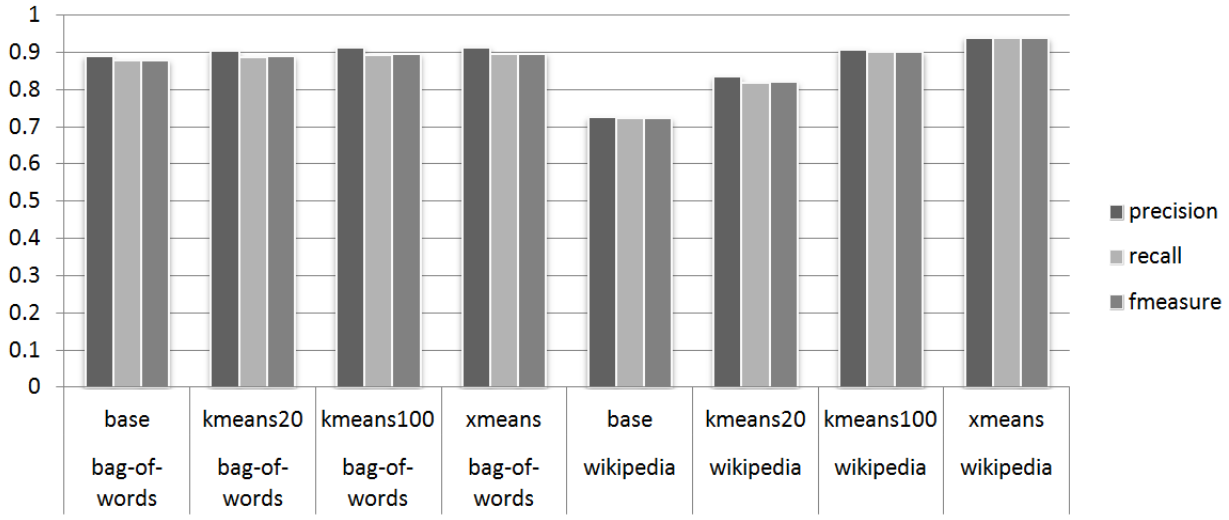

Рис. 4.11. Целевые классы: перекрёстная проверка; метод опорных векторов

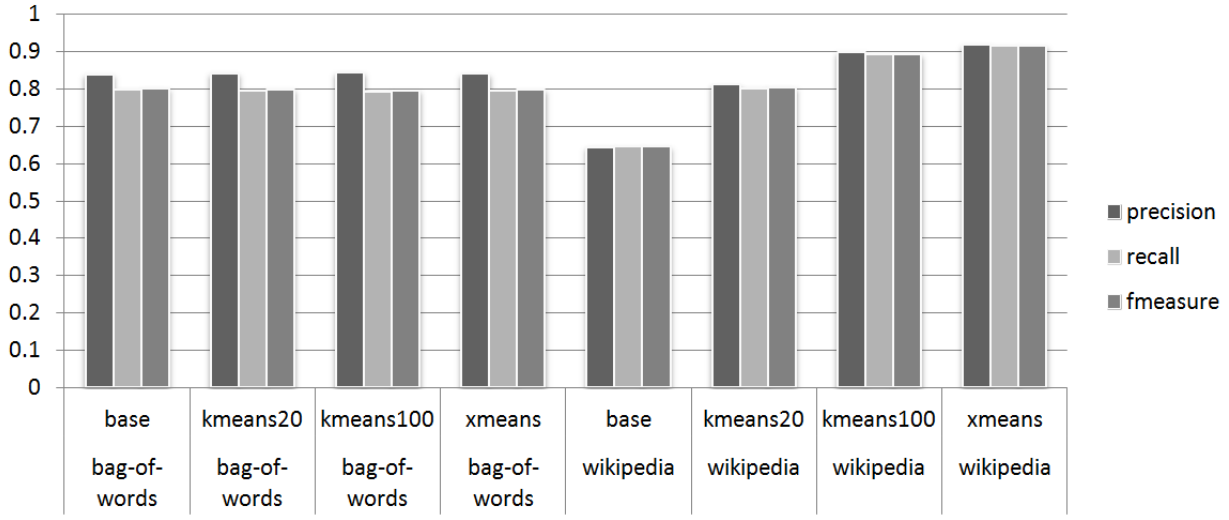

Рис. 4.12. Целевые классы: перекрёстная проверка; алгоритм построения деревьев решений J48

Глава 5

Технические решения

Для реализации всех алгоритмов классификации и кластеризации используется открытая библиотека Weka[1]. Основным языком реализации была выбрана Java[2] в силу своей распространенности и мультиплатформенности. Для работы с “Википедией” применяется библиотека wikixmlj[3]. Для работы с “Твиттером” применяется библиотека twitter4j[4].

Итоговое решение можно разделить на следующие части: часть для работы с “Твиттеров”, часть для работы с “Википедией”, часть содержащая алгоритм классификации и тестирования. Для реализации контекстного классификатора был создан класс ContextClassifier, реализующий класс Classifier библиотеки Weka. В качестве параметров данного классификатора выступают фабрика[5] для создания базовых классификаторов, фабрика алгоритмов кластеризации, модель выделения признаков из текста. Модель выделения признаков из текста реализована в виде наследника класса Filter библиотеки Weka.

Диаграмму классов алгоритмов классификации можно увидеть на Рис. 5.1, кластеризации на Рис. 5.2, выделения признаков из текста на Рис. 5.3.

[1] http://www.cs.waikato.ac.nz/ml/weka/

[2] http://java.com

[3] http://code.google.com/p/wikixmlj/

[4] http://twitter4j.org/en/index.html

[5] http://ru.wikipedia.org/wiki/Абстрактная_фабрика_(шаблон)

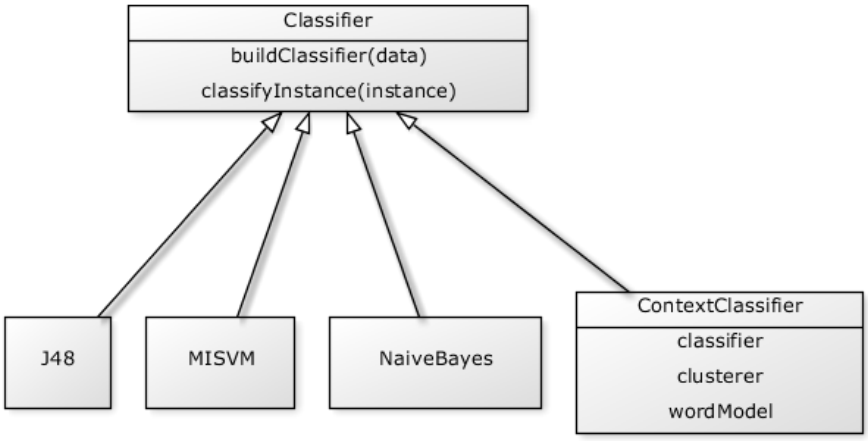

Рис. 5.1. Классы для классификации

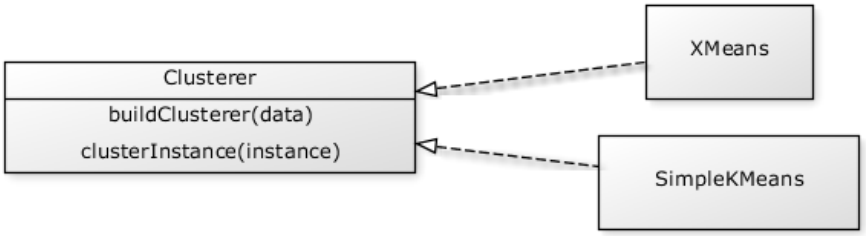

Рис. 5.2. Классы для кластеризации

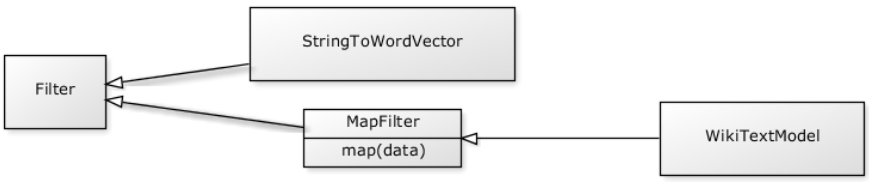

Рис. 5.3. Классы для выделения признаков из текста

Заключение

5.1. Результаты

В результате данной работы была изучена предметная область, придуман и реализован алгоритм классификации записей из микроблогов. Реализованный алгоритм имеет несколько особенностей: он использует контекст записей, а также “Википедию”, как источник дополнительных знаний.

Данный алгоритм был протестирован на нескольких вариантов тестовых данных. В некоторых случаях алгоритм показал хорошие результаты и продемонстрировал существенное улучшение в сравнении с простым подходом для классификации записей.

5.2. Дальнейшая работа

Большим пространством для дальнейших исследований представляется улучшение алгоритма выделения признаков из текста с использованием “Википедии”. Стоит отметить, что возможно стоит попробовать другие алгоритмы кластеризации, например, k-medoids[6], как не требующий представления документов в пространстве признаков (данный алгоритм требует только расстояние между элементами выборки). Имеет смысл попробовать применить латентно-семантический анализ, для предобработки признаковых представлений объектов, для улучшения результата в задачах классификации и кластеризации. Также стоит подготовить бóльшие обучающие и тестовые выборки для более полного тестирования полученных алгоритмов.

Кроме того разработанный алгоритм работает очень долго, что силь-

[6] http://en.wikipedia.org/wiki/K-medoids

но осложняет его тестирование. Было проведено профилирование, в ходе которого было обнаружено, что наибольшую часть времени занимает процесс построения кластеров и непосредственно классификации. В частности, проблема заключается в том, что код библиотеки Weka является однопоточным. Можно, к примеру, распараллелить алгоритмы кластеризации, контекстной классификации, а также тестирование методом перекрёстной проверки.

Приложение А

Пользователи Твиттера, выбранные для тестовых данных

А.1. Для тематических классов

Пользователь	Класс
carnage4life	programming
stevenf	programming
jasonfried	programming
codinghorror	programming
PhysicsNews	physics
materion	physics
CERN	physics
Math_Bits	math
mathguide	math
AlgebraFact	math
ChemistryBooks	chemistry
NatureChemistry	chemistry
ChemistryWorld	chemistry
tlemberger	biology
BioscienceNews	biology
CHoytPhD	biology
pzmyers	biology

A.2. Для разделения на персональный/новостной/принадлежит компании

Пользователь	Класс
ladygaga	user
johnbreslin	user
PerezHilton	user
tony	user
davejmatthews	user
saturdaystar	news
HurriyetDaily	news
MoscowTimes	news
latimes	news
TelegraphMG	news
arielgabriel	company
xemmy	company
myeverydaymoney	company
dealnaydotcom	company
compudah	company

Литература

1. Twitter. Twitter Blog: Twitter turns six. 2012. [Online; accessed 26-May-2012]. URL: http://blog.twitter.com/2012/03/twitter-turns-six.html.

2. Diakopoulos N. A., Shamma D. A. Characterizing debate performance via aggregated twitter sentiment // Proceedings of the 28th international conference on Human factors in computing systems. CHI '10. New York, NY, USA: ACM, 2010. Pp. 1195–1198. URL: http://doi.acm.org/10.1145/1753326.1753504.

3. Asur S., Huberman B. A. Predicting the Future with Social Media // CoRR. 2010. Vol. abs/1003.5699.

4. Genc Y., Sakamoto Y., Nickerson J. V. Discovering context: classifying tweets through a semantic transform based on wikipedia // Proceedings of the 6th international conference on Foundations of augmented cognition: directing the future of adaptive systems. FAC'11. Berlin, Heidelberg: Springer-Verlag, 2011. Pp. 484–492. URL: http://dl.acm.org/citation.cfm?id=2021773.2021833.

5. Wikipedia. Задача классификации — Википедия. http://ru.wikipedia.org/w/index.php?title=Задача_классификации&oldid=377 2012. [Online; accessed 16-May-2012].

6. Gabrilovich E., Markovitch S. Text categorization with many redundant features: using aggressive feature selection to make SVMs competitive with C4.5 // Proceedings of the twenty-first international conference on Machine learning. ICML '04. New York, NY, USA: ACM, 2004. Pp. 41–. URL: http://doi.acm.org/10.1145/1015330.1015388.

7. Wikipedia. Кластерный анализ — Википедия. http://ru.wikipedia.org/w/index.php?title=Кластерный_анализ&oldid=4408989 2012. [Online; accessed 17-May-2012].

8. Aloise D., Deshpande A., Hansen P., Popat P. NP-hardness of Euclidean sum-of-squares clustering // Machine Learning. 2009. Vol. 75. Pp. 245–248. 10.1007/s10994-009-5103-0. URL: http://dx.doi.org/10.1007/s10994-009-5103-0.

9. Arthur D., Manthey B., Röglin H. k-Means has Polynomial Smoothed Complexity // CoRR. 2009. Vol. abs/0904.1113.

10. Pelleg D., Moore A. W. X-means: Extending K-means with Efficient Estimation of the Number of Clusters // Seventeenth International Conference on Machine Learning. Morgan Kaufmann, 2000. Pp. 727–734.

11. Ramage D., Dumais S., Liebling D. Characterizing Microblogs with Topic Models // ICWSM. 2010. URL: http://www.stanford.edu/~dramage/papers/twitter-icwsm10.pdf.

12. Go A., Bhayani R., Huang L. Twitter Sentiment Classification using Distant Supervision // Processing. 2009. Vol. 150, no. 12. Pp. 1–6. URL: http://www.stanford.edu/~alecmgo/papers/TwitterDistantSupervision09.pdf.

13. Gabrilovich E., Markovitch S. Wikipedia-based semantic interpretation for natural language processing // J. Artif. Int. Res. 2009. — . Vol. 34, no. 1. Pp. 443–498. URL: http://dl.acm.org/citation.cfm?id=1622716.1622728.

14. Milne D., Witten I. H. Learning to link with wikipedia // Proceedings of the 17th ACM conference on Information and knowledge management.

CIKM '08. New York, NY, USA: ACM, 2008. Pp. 509–518. URL: http://doi.acm.org/10.1145/1458082.1458150.

15. Sriram B., Fuhry D., Demir E. et al. Short text classification in twitter to improve information filtering // Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval. SIGIR '10. New York, NY, USA: ACM, 2010. Pp. 841–842. URL: http://doi.acm.org/10.1145/1835449.1835643.

16. Horn C. Analysis and Classification of Twitter messages Analyse und Kategorisierung von Twitter Nachrichten // Arbeit. 2010. no. April. URL: http://know-center.tugraz.at/wp-content/uploads/2010/12/Master-Thesis-Christopher-Horn.pdf.

17. Xu T., Oard D. W. Wikipedia-based topic clustering for microblogs // Proceedings of the American Society for Information Science and Technology. 2011. Vol. 48, no. 1. Pp. 1–10. URL: http://dx.doi.org/10.1002/meet.2011.14504801186.

18. Noel J. C. G. Wikipedia-based Text Classification. 2010. URL: http://cs.anu.edu.au/student/projects/10S2/Reports/Joseph%20Noel.pdf.