Московский Государственный Университет им. М. В. Ломоносова

Факультет Вычислительной Математики и Кибернетики

Выпускная квалификационная работа

на тему:

«Частичное машинное обучение в задачах классификации текстов»

Подготовила:

студентка 431 группы

Чучвара Александра Ивановна

Научный руководитель:

доцент кафедры ММП, к.ф.-м.н.

Дьяконов Александр Геннадьевич

Москва, 2009

Содержание

Аннотация 4

Введение 5

1 Постановка задачи 8

2 Подходы к решению 9

2.1 Алгоритм №1 (PSSF) 11

2.2 Алгоритм №2 (TPN²) 13

2.3 Алгоритм №3 (LLGC) 17

3 Программная реализация и эксперименты 19

3.1 Экспериментальные данные 20

3.2 Метрика, используемая для оценки качества 21

3.3 Эксперименты 23

Заключение 28

Список литературы 29

Аннотация

Данная работа посвящена применению методов частичного машинного обучения в задачах классификации текстов. В частности, рассматривается задача персонализированной фильтрации спама на стороне почтового сервера, которая была предложена в рамках международного соревнования ученых-прикладников на конференции по машинному обучению ECML-PKDD в 2006 году. Основной задачей данной работы являлось исследование алгоритмов, занявших первые места в соревновании, для получения полной картины о качестве их работы. Для решения данной задачи были реализованы три из предложенных алгоритмов. Алгоритмы были протестированы на реальных данных. В работе представлено экспериментальное исследование и сравнительный анализ алгоритмов. Предметом исследования стали такие важные характеристики алгоритмов, как скорость и точность работы, а также обобщающая способность алгоритмов. Используя полученные результаты, можно модифицировать алгоритмы и улучшить их производительность, или произвести эффективный выбор алгоритма фильтрации.

Введение

Автоматическая классификация текста, т.е. определение принадлежности входного текста к некоторой категории, является весьма актуальной и интересной задачей в условиях постоянно растущего объема используемой информации. Принадлежность к тому или иному классу может определяться общей тематикой текста, употреблением определенных понятий, другими условиями. Классификация текста применяется в решении многих практических задач: поиске документов, навигации в больших информационных ресурсах, фильтрации спама, подборе контекстной рекламы, составлении интернет-каталогов и других.

Классические методы машинного обучения, используемые в задачах классификации текста, такие как байесовские методы, нейронные сети, методы опорных векторов *(Support Vector Machine – SVM)* [11], принадлежат к группе методов обучения с учителем. Для построения классификатора эти методы используют набор текстов, уже распределенных по классам (обучающее множество). Классификатор строится на основе автоматического анализа этого множества. Применение методов обучения с учителем очень эффективно при наличии представительного обучающего множества, отражающего характер распределения классифицируемых данных. Однако часто бывает сложно или невозможно получить такое множество на практике. Проблемы возникают из-за недостаточного количества размеченных примеров, наличия ошибок, несоответствия распределений обучающего и классифицируемого множеств. Таким образом, из-за трудностей формирования качественного обучающего набора примеров точность распознавания методов обучения с учителем сильно ухудшается.

Автоматическая классификация текста является ярким примером задач, для которых довольно сложно получить непротиворечивое, достаточно представительное обучающее множество, и в то же время, сравнительно легко собрать большой объем неразмеченных документов. В таких условиях, когда нельзя полагаться только на обучающее множество, улучшить качество классификации можно с помощью имеющихся неразмеченных примеров. Методы, использующие в процессе обучения

как размеченные, так и неразмеченные данные, называются методами частичного обучения *(semi-supervised learning)*.

Традиционно в машинном обучении существуют два основных подхода:

– Обучение без учителя (*unsupervised learning*). Имеется множество объектов $X = \{x_1, \dots, x_n\}$. Требуется обнаружить внутренние взаимосвязи, зависимости, закономерности, существующие между объектами множества X.

– Обучение с учителем (*supervised learning*). Дано множество объектов $X = \{x_1, \dots, x_n\}$ и множество возможных ответов $Y = \{1, \dots, c\}$. Известно обучающее множество пар "объект-ответ" $\{(x_1, y_1), \dots, (x_n, y_n)\}$. На основе этих данных требуется восстановить зависимость, то есть построить классификатор, способный для любого объекта определить к какому классу он принадлежит.

Частичное обучение занимает промежуточное положение между обучением с учителем и без учителя. В задачах частичного обучения имеется лишь небольшое количество примеров $X_l = \{x_1, \dots, x_l\}$, для которых известна правильная классификация $Y_l = \{y_1, \dots, y_l\}$, но большой объем неразмеченных данных $X_u = \{x_{l+1}, \dots, x_{l+u}\}$. Размеченные и неразмеченные данные используются одновременно, при этом можно сказать, что неразмеченные данные дают представление о структуре классифицируемого множества объектов, тогда как размеченные данные определяют классификацию в пределах этой структуры. При определенных условиях использование неразмеченных данных позволяет значительно повысить качество и обобщающую способность алгоритма.

Областью применения методов частичного обучения являются задачи, в которых неразмеченные данные имеются в изобилии, такие как обработка изображений, биоинформатика и классификация текстов. За последнее время появилось большое количество работ, посвященных частичному обучению [12]. Немалый интерес вызывают исследования возможностей адаптации существующих методов для решения различных прикладных задач. Одно из таких исследований проводилось в рамках международного соревнования ученых-прикладников на конференции по машинному обучению ECML-PKDD в 2006 году [1]. Участникам предлагалось разработать алгоритм для решения задачи персонализированной фильтрации спама на стороне почтового сервера.

Спам – одна из главных проблем современного Интернета. Огромное количество бесполезной информации вынуждает пользователей тратить все больше времени на прочтение и отсеивание ненужной корреспонденции. В настоящее время существует множество технологий борьбы со спамом. Наиболее эффективными считаются методы фильтрации, основанные на разборе и анализе текста письма. Большинство этих методов являются методами машинного обучения с учителем. Такие методы имеют возможность проводить более гибкую фильтрацию и персонализировать процесс обработки почты. Провайдеры могут фильтровать спам для своих клиентов. Однако из-за массового характера поступления почты на стороне провайдера алгоритмы, осуществляющие анализ текста письма, практически не применимы в чистом виде, т.к. в процессе обучения фильтра известны предпочтения только усредненного пользователя. Проблема состоит в том, что большими почтовыми службами пользуется весьма разнообразная аудитория и то, что для одного пользователя не является спамом, другой классифицирует как спам. Так, например, если в обучающем множестве много туристического спама, то все письма, например, из турагентства могут быть отнесены к спаму. По этой же причине не всегда эффективен и механизм обратной связи с пользователем.

Таким образом, возникает вопрос о возможности применения методов, анализирующих текст письма, для организации персонализированной фильтрации спама на стороне почтового сервера. В процессе обучения такого фильтра может быть использовано некоторое множество размеченных писем, полученное из общедоступных ресурсов. При этом для достижения высокого качества фильтрации необходимо каким-то образом адаптировать фильтр к характеристикам отдельного пользовательского ящика. Такая адаптация должна быть осуществима без использования обратной связи с пользователем. В данном случае возможным решением будет использование неразмеченного множества пользовательских писем и применение методов частичного обучения.

1 Постановка задачи

Требуется организовать качественную фильтрацию спама для каждого пользователя почтового сервера, но предполагается, что фильтры обучаются и используются на стороне сервера. При этом для обучения используется комбинированное множество размеченных писем, полученное из общедоступных ресурсов. Предполагается, что кроме размеченного множества в процессе обучения также доступны собственные письма пользователя, но для них не известна правильная классификация. Таких неразмеченных писем довольно много, и они могут быть использованы для адаптации к характеристикам пользовательского ящика с помощью частичного обучения. Однако данная постановка задачи отличается от классической задачи частичного обучения, в силу зашумленности обучающего множества. Более формально задачу можно поставить следующим образом.

Формальная постановка задачи

Пусть L – общее размеченное множество писем и U_1, ..., U_n – неразмеченные множества пользовательских писем, где U_i соответствует множеству писем i-го пользователя. Предполагается, что распределения писем в L и U_i неизвестны и отличны друг от друга. Письмо представляется в виде вектора признаков, размерность которого равна размеру словаря[1] V, $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{i|V|})$, где $x_{ij} = 1$ если i-е письмо содержит j-е слово и $x_{ij} = 0$ – иначе. Требуется на основе этих данных построить фильтр $F_i: U_i \to \{+1, -1\}$, который классифицирует письмо $\mathbf{x}$ пользователя i как спам (класс с положительными исходами) или обычное письмо (класс с отрицательными исходами).

Качество полученного классификатора оценивается по результатам фильтрации пользовательских писем. Результаты этой фильтрации сравниваются с заранее известными значениями. Для оценки качества применяются F-мера и AUC[2] – площадь под ROC кривой *(Receiver Operating Characteristic)*. Требуется, чтобы средние по всем пользователям значения AUC и F-меры были как можно больше.

---

[1] Из словаря удалены слишком редко встречающиеся слова (менее 4-х раз). Всего в словаре около 150’000 слов.

[2] Подробно в разделе *«Метрика, используемая для оценки качества».*

2 Подходы к решению

В рамках соревнования рассматривались два варианта поставленной задачи: **A** и **B.** Участниками был предложен ряд подходов к решению, в большинстве из которых были использованы различные варианты методов частичного обучения, такие как графовые методы *(graph-based)* [2], максимизирующие зазор алгоритмы *(large-margin-based)* [3, 5], методы самообучения *(self-training)* [4, 6], одноклассовое обучение *(positive-only learning)* [3]. В таблицах 1 и 2 приведены результаты для задач **A** и **B** соответственно (только первые три места).

Место	*Средняя AUC*	*Команда*
1	0.950667	**Khurram Nazir Junejo, Mirza Muhammad Yousaf, Asim Karim** Lahore University of Management Sciences, Pakistan
1	0.949094	**Bernhard Pfahringer** University of Waikato, New Zealand
1	0.948666	**Kushagra Gupta, Vikrant Chaudhary, Nikhil Marwah, Chirag Taneja** Inductis India Pvt Ltd
2	0.936457	**Nikolaos Trogkanis**, National Technical University of Athens, Greece **Georgios Paliouras**, National Center of Scientific Research "Demokritos", Greece
3	0.927839	**Chao Xu, Yiming Zhou** Beihang University, Beijing, China

Таблица 1. Первые три места в задаче **A**

Место	*Средняя AUC*	*Команда*
1	0.946469	**Gordon Cormack** University of Waterloo, Canada
2	0.918251	**Nikolaos Trogkanis**, National Technical University of Athens, Greece **Georgios Paliouras**, National Center of Scientific Research "Demokritos", Greece
3	0.907398	**Kushagra Gupta, Vikrant Chaudhary, Nikhil Marwah, Chirag Taneja** Inductis India Pvt Ltd

Таблица 2. Первые три места в задаче **B**

Для экспериментального исследования были выбраны следующие алгоритмы:

1. *PSSF (*Khurram Nazir Junejo, Mirza Muhammad Yousaf, Asim Karim: *1-е место в задаче A);*
2. *TPN² (*Nikolaos Trogkanis, Georgios Paliouras: *2-е место в задаче A, 2-е место в задаче B);*
3. *LLGC (*Bernhard Pfahringer: *1-е место в задаче A).*

Алгоритм *PSSF* [6] основан на статистическом анализе распределения признаков письма, а классификация письма зависит от относительной частоты слов, входящих в текст. Данный алгоритм является хорошо масштабируемым и обладает высокой скоростью работы. Это важно, поскольку алгоритм планируется применять для обработки больших объемов данных.

Алгоритм *TPN²* [3] представляет собой комбинацию нескольких методов машинного обучения. Классификатор, полученный с помощью обучающего множества, применяется для разметки очень небольшого количества писем, которые классифицируются с большой уверенностью. На последующих шагах обучающее множество игнорируется, и методы частичного обучения применяются только к пользовательским письмам.

В основе алгоритма *LLGC* [2] лежит одноименный метод *LLGC (Learning with Local and Global Consistency)* – один из наиболее известных графовых методов частичного обучения [7]. Идея метода состоит в сопоставлении двух потенциально противоречивых условий: близкие примеры должны относиться к одному классу (локальное соответствие), полученное разбиение на классы не должно противоречить обучающим данным (глобальное соответствие).

Во всех алгоритмах можно условно выделить два этапа:

1. Первоначальная разметка;
2. Итеративное уточнение правил классификации.

Ключевым моментом является получение первоначальной разметки пользовательских писем с помощью обучающего множества. Довольно сложно получить точную классификацию из-за несоответствия распределений обучающего множества и множеств писем пользователей. Несмотря на это, использование неразмеченных данных позволяет снизить количество ошибок.

Подробное описание алгоритмов приведено в следующих трех разделах.

2.1 Алгоритм №1 (PSSF)

Псевдокод алгоритма PSSF
Вход: • обучающее множество L • неразмеченные множества U_1, ..., U_n **Выход:** • фильтр для каждого пользователя • классифицированные множества U_1, ..., U_n
Для обучающего множества L: 1. Построить статистическую модель // ...***(1)*** 2. Построить классификатор // ...***(2)*** **Для каждого пользовательского почтового ящика U_i:** 3. **Повторять** (один или более раз) // ...***(3)*** 4. Разметить письма пользовательского ящика 5. Построить новую статистическую модель 6. **Конец** 7. Классифицировать письма

(1) **Статистическая модель**

Статистическая модель строится по размеченному множеству писем (это может быть обучающее множество L или классифицированное множество U_i). Обозначим количество появлений слова i в спаме и в обычных письмах этого множества как n_i^S и n_i^N соответственно. Определим множество слов, характерных для спама (Z^S), и множество слов, характерных для обычных писем (Z^N), как $Z^S = \{j|(n_j^S/n^S - n_j^N/n^N) > 0\}$ и $Z^N = \{j|(n_j^N/n^N - n_j^S/n^S) > 0\}$, где n^N и n^S это количество обычных писем и количество спама в размеченном множестве писем соответственно. Каждому слову $j \in \{Z^S \cup Z^N\}$ сопоставим вес:

$$w_j = \begin{cases} (n_j^S/n^S)/(n_j^N/n^N), & j \in Z^S \\ (n_j^N/n^N)/(n_j^S/n^S), & j \in Z^N \end{cases}$$

Таким образом, чем больше вероятность появления слова $j \in Z^S$ в спаме относительно вероятности его появления в обычном письме, тем больший вес ему приписывается. Аналогично для слов $j \in Z^N$.

(2) Классификатор

Решающее правило выглядит следующим образом:

$$F(\mathbf{x}) = S * \sum_{j \in Z^S} w_j x_j - \sum_{k \in Z^N} w_k x_k,$$

где S – это масштабирующий множитель, x_j – значение j-го элемента вектора $\mathbf{x}$ и w_j – веса, полученные в предыдущем пункте. Если $F(\mathbf{x}) > 0$, то классифицируем письмо как спам, если $F(\mathbf{x}) < 0$ – обычное письмо. Масштабирующий множитель необходим, чтобы учесть тот факт, что количество слов, характерных для обычных писем, а следовательно, и их взвешенная сумма обычно больше, чем количество слов, характерных для спама, и их взвешенная сумма. Масштабирующий множитель S выбирается так, чтобы минимизировать ошибку классификации размеченного множества.

(3) Адаптация классификатора

На этом этапе происходит адаптация общей статистической модели, к индивидуальным характеристикам почтового ящика конкретного пользователя:

- классифицируем письма пользователя, используя общую статистическую модель;
- так же как в первом пункте строим новую (адаптированную) статистическую модель, при этом в качестве размеченного множества берем классифицированные письма пользователя.

Процедуру адаптации классификатора можно повторить несколько раз (уже после двух повторений достигается довольно высокая точность классификации). На последнем шаге окончательно размечаем пользовательские письма.

2.2 Алгоритм №2 (TPN^2)

Псевдокод алгоритма TPN²
Вход: • обучающее множество L • неразмеченные множества U_1, ..., U_n • классифицируемое множество U_i • вес классифицируемого множества w • порог T • вероятность появления спама p **Выход:** • фильтр для пользователя i • классифицированное множество U_i
1. Построить объединенное множество писем всех пользователей U, при этом классифицируемое множество U_i добавляется w раз // ...***(1)*** 2. По обучающему множеству построить классификатор // ...***(2)*** 3. Классифицировать множество U и выделить множество «сильного» спама POS // ...***(2)*** 4. Увеличить множество спама // ...***(3)*** 5. Построить классификатор с помощью полученного множества спама // ...***(4)***

(1) **Объединенное множество писем**

Объединяем все имеющиеся письма пользователей U_1, ..., U_n в общий набор писем U, при этом классифицируемое множество U_i добавляется w раз:

$$U := \sum_{j \neq i} U_j + w * U_i.$$

(2) **Множество «сильного» спама**

Из набора неклассифицированных данных U нужно выделить небольшое количество писем, про которые с большой вероятностью можно сказать, что они

являются спамом. Для этого по обучающему множеству L строим байесовский классификатор *(Naïve Bayes Multinomial – NBM [8])*:

Дано множество размеченных документов L. Обозначим через PL (NL) множество положительных (отрицательных) примеров в L ($L = PL \cup NL$). Вероятность появления письма из класса $P(c)$ оценивается как:

$$\hat{P}(-1) = \frac{|NL|}{|L|} \; ; \hat{P}(+1) = \frac{|PL|}{|L|}$$

где $|X|$ – мощность множества X.

Условная вероятность появления слова $P(w_i|c)$ оценивается как:

$$\hat{P}(w_i|-1) = \frac{N(w_i,NL)}{N(NL)} \; ; \hat{P}(w_i|+1) = \frac{N(w_i,PL)}{N(PL)}$$

где $N(w_i,X)$ – количество писем множества X, содержащих слово w_i, и $N(X)$ – количество уникальных слов, содержащихся в письмах множества X.

Письмо $\mathbf{x}$, состоящее из n слов $(w_1, \dots, w_n)$, относится к наиболее вероятному классу в предположении условной независимости появлений слов $w_1, \dots, w_n$

$$\mathrm{NMB}(\mathbf{x}) = \underset{c \in \{-1,+1\}}{\operatorname{argmax}} \hat{P}(c) \prod_{i=1}^{n} \hat{P}(w_i|c).$$

Классифицируем набор пользовательских писем и составим множество «сильного» спама POS из тех писем, которые были классифицированы как спам с уверенностью большей, чем некоторый порог T.

(3) Итеративное расширение множества спама

Имея множество сильного спама и много неразмеченных примеров, применим одноклассовый байесовский классификатор *(Positive Naïve Bayes – PNB [8])*, чтобы выявить больше спама:

Предполагается, что оценка вероятности появления положительного примера $\hat{P}(+1)$ дана, тогда оценка вероятности появления отрицательного примера считается как $\hat{P}(-1) = 1 - \hat{P}(+1)$. Также дано множество положительных примеров POS. Обозначим E – множество неразмеченных примеров $(E = U - POS)$.

Условная вероятность появления слова $P(w_i|c)$ оценивается как:

$$\hat{P}(w_i|+1) = \frac{N(w_i,POS)}{N(POS)}; \hat{P}(w_i|-1) = \frac{P(w_i)-\hat{P}(w_i|+1)\times\hat{P}(+1)}{1-\hat{P}(+1)}$$

где оценка вероятность появления слова $P(w_i)$ считается следующим образом:

$$\hat{P}(w_i) = \frac{N(w_i,E)}{N(E)}.$$

Письмо $\mathbf{x}$, состоящее из n слов $(w_1, \ldots, w_n)$, относится к классу

$$\text{PNB}(\mathbf{x}) = \underset{c\in\{-1,+1\}}{\text{argmax}}\, \hat{P}(c) \prod_{i=1}^{n} \hat{P}(w_i|c).$$

Данный метод был выбран из соображений, что обучающее множество точнее отражает характеристики спама. Действительно, спам менее специфичен, чем письма конкретного пользователя, отражающие его индивидуальные интересы. И, следовательно, различие между спамом в обучающем множестве и спамом, полученным пользователями, вероятно, будет менее заметно.

Построенный с помощью начального множества «сильного» спама классификатор далее применяется к неразмеченному множеству E. Те примеры, которые будут классифицированы как спам, составят новое множество POS, которое используется на следующей итерации для построения нового классификатора *PNB*.

(4) Построение классификатора по множеству спама

Определив большую часть спама, применим к множеству U_i другой алгоритм одноклассового обучения, идея которого состоит в определении множества писем максимально отличных от спама. Например, можно использовать итеративный алгоритм *PEBL (Positive Example-Based Learning)*:

1. Определить множество «сильных» отрицательных примеров из множества неразмеченных писем;
2. Обучить классификатор (например *SVM*), используя множество положительных и множество «сильных» отрицательных примеров, полученное на предыдущем шаге;
3. Применить классификатор к множеству оставшихся неразмеченных данных;
4. Добавить примеры, классифицированные как отрицательные, к множеству отрицательных примеров и заново обучить классификатор;

5. Остановиться, если не нашлось новых отрицательных примеров, и классифицировать оставшиеся неразмеченные примеры как положительные.

2.3 Алгоритм №3 (LLGC)

Псевдокод алгоритма LLGC
Вход: • обучающее множество L • неразмеченные множества $U_1, \ldots, U_n$ • параметр α • количество соседей k • число итераций $iter$ **Выход:** • фильтр для каждого пользователя • классифицированные множества $U_1, \ldots, U_n$
Для каждого пользовательского почтового ящика U_i: 1. Для каждого отдельного письма из пользовательского ящика стоится свой классификатор и определяется первоначальная классификация письма // ...***(1)*** 2. К обучающему множеству L и первоначально размеченному множеству U_i применить $iter$ итераций метода *LLGC* // ...***(2)***

***(1)* Первоначальная классификация**

Для каждого отдельного письма пользовательского ящика стоится свой классификатор (например *SVM*), при этом множество слов обучающего набора проектируется на множество слов, содержащихся в письме. Классификатор обучается, используя преобразованное обучающее множество, таким образом, уменьшается размерность данных, и классификатор фокусируется на словах, фактически присутствующих в письме. Количество слов в отдельном письме значительно меньше общего количества возможных слов и находится в пределах от десятка до двух тысяч, тогда как большинство писем содержит до сотни слов.

***(2)* Алгоритм LLGC**

На вход алгоритма подается множество точек $X = \{x_1, \ldots, x_l, x_{l+1}, \ldots, x_n\} \in R^m$ и множество классов $C = \{1, \ldots, c\}$. Для первых l точек x_i $(i \leq l)$ известен класс

$y_i \in C$, к которому они принадлежат, для оставшихся точек $x_u\,(l+1 \le u \le n)$ класс не известен (в нашем случае письмам из множества U_i также соответствует классификация, полученная на предыдущем шаге).

Алгоритм состоит из следующих шагов:

1. Посчитать матрицу W, задающую попарную схожесть объектов входного множества, где $W_{ij} = \exp\left(-||x_i - x_j||^2/(2\sigma^2)\right)$ если $i \neq j$ и $W_{ii} = 0$;
2. Посчитать матрицу $S = D^{-1/2}WD^{-1/2}$, где D – диагональная матрица, (i,i)-й элемент которой равен сумме элементов i-ой строки матрицы W (таким образом нормируем матрицу W, чтобы обеспечить сходимость);
3. Определить матрицу Y размером $n \times c$, где n – количество примеров, c – количество классов и $Y_{ij} = 1$, если пример i принадлежит классу j, все остальные элементы матрицы Y равны нулю, таким образом, каждому неразмеченному примеру соответствует нулевая строка матрицы;
4. Инициализировать $F(0) = Y$;
5. Повторять $F(t+1) = \alpha * S * F(t) + (1-\alpha) * Y$, где $\alpha \in (0,1)$ – параметр, определяющий, в какой пропорции учитывается информация от соседних точек и начальная информация.

Считается, что для задач классификации текста в качестве функции расстояния лучше всего подходит косинус угла между векторами признаков. Поэтому элементы матрицы W вычисляются как $W_{ij} = \exp\left(-\left(1 - \langle x_i, x_j\rangle/\|x_i\|\|x_j\|\right)/(2\sigma^2)\right)$.

Большинство элементов матрицы W содержат значения близкие к нулю. Следовательно для сокращения используемой памяти и увеличения скорости работы алгоритма можно разредить матрицу W, обнулив все элементы кроме k ближайших соседних для каждого объекта, где $k << n$.

3 Программная реализация и эксперименты

Описанные алгоритмы были реализованы в среде MatLab. В ходе реализации некоторые алгоритмы были модифицированы для повышения скорости и/или качества работы. Так в оригинальной версии алгоритма *TPN²* вводится параметр w (вес), определяющий сколько раз каждое письмо из классифицируемого пользовательского ящика будет добавлено в общий набор писем. Предполагается, что это поможет классификатору лучше сфокусироваться на характеристиках именно классифицируемого множества. На самом деле, влияние этого параметра на качество работы алгоритма не велико, а подобрать оптимальный параметр для каждого пользовательского ящика автоматически не представляется возможным. Поэтому на первом шаге алгоритма просто объединяем письма всех пользователей в одно множество. Далее, на последнем шаге алгоритма применяется алгоритм *PEBL*. Это довольно медленный алгоритм, так как на каждой итерации строится классификатор *SVM*, скорость обучения которого сравнительно низка особенно для большого количества обучающих примеров (как в задаче **A**). Поэтому алгоритм *PEBL* был заменен несколькими итерациями алгоритма *PNB*, что значительно ускорило работу алгоритма и положительно сказалось на качестве классификации. Повышение качества классификации можно объяснить тем, что алгоритму *PNB* для обучения не требуется множество отрицательных примеров, которое в наше случае оказывается ненадежным. В алгоритме *LLGC* наиболее долгим оказывается этап первоначальной разметки писем, так как в данном случае для каждого отдельного письма строится свой классификатор. Вместо этого применим для всех писем классификатор *PNB* и на втором этапе в качестве входных размеченных данных оставим только те примеры, которые были классифицированы очень уверенно. Кроме того, в ходе предварительных экспериментов было установлено, что качество работы алгоритма оказывается выше, если на втором этапе игнорировать обучающее множество, т.е. применять метод *LLGC* только к множеству пользовательских писем. Алгоритм *PSSF* был реализован без изменений.

3.1 Экспериментальные данные

Целью экспериментального исследования было сравнение эффективности предложенных методов, а также возможностей их применения в рамках реальной задачи персонализированной фильтрации спама на стороне почтового сервера. Алгоритмы были протестированы на двух наборах данных задач **A** и **B**, которые доступны на сайте конференции [9]. Каждый набор состоит из обучающего множества (L) и нескольких пользовательских почтовых ящиков (U_i). Распределение писем в обучающем множестве, которое представляет собой комбинированное множество писем, полученных из общедоступных ресурсов, отличается от распределения писем, полученных определенным пользователем. Количество писем в обучающем множестве и в пользовательских ящиках гораздо больше в наборе **A**, чем в наборе **B**. Более того, в обучающем множестве набора **B** содержится меньше писем, чем в пользовательских ящиках. Поэтому задача **B** представляется более сложной. В обоих наборах пользовательские почтовые ящики, как и обучающие множества, содержат 50% спама и 50% обычных писем. Для обучающих множеств обычные письма набирались из новостных рассылок, спам – с помощью «спам-ловушек». Все письма представлены в виде списка слов, встречающихся в тексте письма, включая заголовок. Для всех наборов используется один и тот же словарь. Спецификации тестовых наборов представлены в таблице 3.

	Задача A	***Задача B***
Количество размеченных обучающих писем	**4000**	**100**
Количество писем в пользовательском почтовом ящике	**2500**	**400**
Количество пользовательских ящиков	**3**	**15**
Процент спама в обучающем множестве писем	**50%**	**50%**
Процент спама в пользовательском почтовом ящике	**50%**	**50%**

Таблица 3. Спецификация тестовых наборов задач **A** и **B**

3.2 Метрика, используемая для оценки качества

Для оценки качества работы алгоритмов будем использовать AUC – площадь под ROC-кривой [10]. ROC-кривая (*Receiver Operator Characteristic*) наиболее часто используется для представления результатов бинарной классификации. Поскольку классов два, один из них называется классом с положительными исходами, второй – с отрицательными исходами. При этом предполагается, что у классификатора имеется некоторый параметр, варьируя который, мы будем получать то или иное разбиение на два класса. В зависимости от него будут получаться различные величины ошибок I и II рода. ROC-кривая показывает зависимость количества верно классифицированных положительных примеров от количества неверно классифицированных отрицательных примеров. ROC-кривая строится следующим образом:

1. Имеется параметр T, меняющийся от 0 до 1 с шагом dx. Для каждого значения параметра рассчитываются значения чувствительности

$$Sensitivity = \frac{TP}{TP + FN}$$

и специфичности

$$Specificity = \frac{TN}{TN + FP},$$

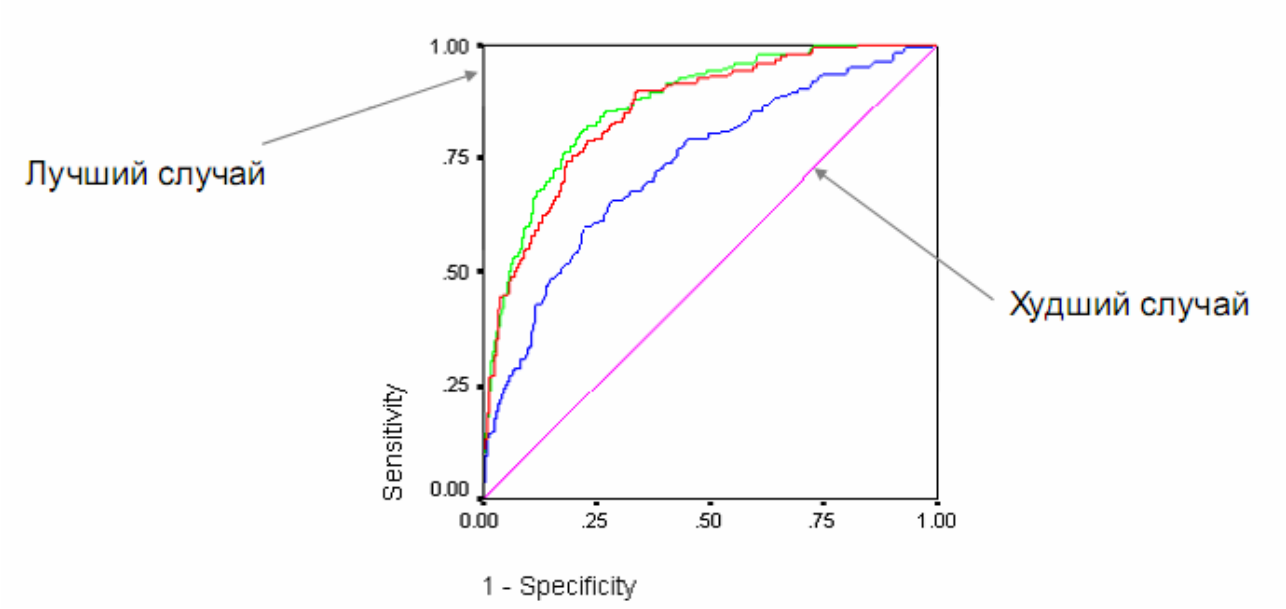
где

- TP (*True Positives*) – верно классифицированные положительные примеры;
- TN (*True Negatives*) – верно классифицированные отрицательные примеры;
- FN (*False Negatives*) – положительные примеры, классифицированные как отрицательные (*ошибка I рода*);
- FP (*False Positives*) – отрицательные примеры, классифицированные как положительные (*ошибка II рода*).

2. Строится график зависимости: по оси X откладывается $1 - Specificity$, по оси Y – чувствительность $Sensitivity$.

Для идеального классификатора график ROC-кривой проходит через верхний левый угол, где доля истинно положительных случаев составляет 1.0, а доля ложно

положительных примеров равна нулю. Поэтому чем меньше изгиб кривой, и чем ближе она расположена к диагональной прямой, тем менее эффективна модель.



Методом сравнения ROC-кривых является оценка площади под кривыми. Она изменяется от 0.5 («бесполезный» классификатор) до 1.0 («идеальная» модель). Можно считать, что чем больше показатель AUC, тем лучшей прогностической силой обладает модель. В литературе иногда приводится следующая шкала для значений AUC, по которой можно судить о качестве модели:

Интервал AUC	***Качество модели***
0.9-1.0	Отличное
0.8-0.9	Очень хорошее
0.7-0.8	Хорошее
0.6-0.7	Среднее
0.5-0.6	Неудовлетворительное

Следует заметить, что показатель AUC предназначен скорее для сравнительного анализа нескольких моделей и не содержит никакой информации о чувствительности и специфичности модели. Поэтому введем также такую оценку качества как F-мера:

$$F = \frac{2}{\frac{1}{precision} + \frac{1}{recall}},$$

где $recall = |TP|/(|TP| + |FN|)$ – полнота и $precision = |TP|/(|TP| + |FP|)$ – точность.

3.3 Эксперименты

Для исследования качества работы алгоритмов было проведено сравнение результатов классификации двух тестовых наборов (**A** и **B**). Эксперименты проводились следующим образом. На вход алгоритма подается обучающее множество и набор неразмеченных пользовательских писем. На выходе получаем классифицированные пользовательские письма. После чего сравниваем полученный результат с заранее известными значениями. В ходе экспериментов была произведена настройка параметров. В таблицах 4, 5, 6 представлены результаты классификации на разных этапах обучения, для следующих значений параметров:

- PSSF: количество итераций = 3;
- TPN[2]: $T = 0.99$, $p = 0.5$;
- LLGC: $\alpha = 0.85$, $k = 10$, количество итераций = 20.

Алгоритм **PSSF**

Номер почтового ящика	***Задача B***				***Задача A***			
	Этап 1		***Этап 2***		***Этап 1***		***Этап 2***	
	AUC	***F-мера***	***AUC***	***F-мера***	***AUC***	***F-мера***	***AUC***	***F-мера***
U_1	0.9252	0.5754	0.9937	0.9720	0.9631	0.9241	0.9987	0.9811
U_2	0.9117	0.5500	0.9920	0.9567	0.9609	0.9272	0.9992	0.9767
U_3	0.9639	0.7805	0.9990	0.9771	0.9396	0.8778	0.9980	0.9154
U_4	0.9435	0.8994	0.9962	0.9340				
U_5	0.8592	0.6689	0.9665	0.8760				
U_6	0.8797	0.5632	0.9192	0.8284				
U_7	0.9025	0.6755	0.9607	0.8827				
U_8	0.9274	0.5507	0.9966	0.9724				
U_9	0.9320	0.6186	0.9989	0.9798				
U_{10}	0.9474	0.6090	0.9978	0.9701				
U_{11}	0.9338	0.7197	0.9776	0.9219				
U_{12}	0.9405	0.6990	0.9760	0.9219				
U_{13}	0.9547	0.7607	0.9969	0.9673				
U_{14}	0.7939	0.4791	0.9839	0.8500				
U_{15}	0.8402	0.6174	0.9893	0.9449				
Среднее значение	**0.9104**	**0.6511**	**0.9829**	**0.9303**	**0.9545**	**0.9097**	**0.9986**	**0.9577**

Таблица 4. Результаты алгоритм PSSF (количество итераций = 3)

Алгоритм **TPN²**

Номер почтового ящика	*Задача B*				*Задача A*			
	Этап 1		*Этап 2*		*Этап 1*		*Этап 2*	
	AUC	*F-мера*	*AUC*	*F-мера*	*AUC*	*F-мера*	*AUC*	*F-мера*
U_1	0.8005	0.7315	0.9727	0.9602	0.8522	0.7972	0.9830	0.9734
U_2	0.7762	0.7013	0.9850	0.9630	0.9078	0.8661	0.9867	0.9709
U_3	0.8933	0.7974	0.9997	0.9950	0.9487	0.8550	0.9957	0.9881
U_4	0.9444	0.8280	0.9455	0.9224				
U_5	0.7541	0.6752	0.9390	0.9065				
U_6	0.7385	0.7258	0.8406	0.8463				
U_7	0.8770	0.7892	0.9739	0.9359				
U_8	0.8368	0.7716	0.9739	0.9600				
U_9	0.7706	0.6941	0.9834	0.9682				
U_{10}	0.8582	0.7824	0.9895	0.9756				
U_{11}	0.8828	0.7907	0.9065	0.8831				
U_{12}	0.8346	0.7970	0.8735	0.8842				
U_{13}	0.8954	0.7571	0.9754	0.9773				
U_{14}	0.6718	0.6366	0.9708	0.9420				
U_{15}	0.8033	0.6994	0.9794	0.9604				
Среднее значение	**0.8225**	**0.7452**	**0.9539**	**0.9387**	**0.9029**	**0.8394**	**0.9884**	**0.9775**

Таблица 5. Результаты алгоритм TPN² ($T = 0.99$, $p = 0.5$)

Алгоритм **LLGC**

Номер почтового ящика	*Задача B*				*Задача A*			
	Этап 1		*Этап 2*		*Этап 1*		*Этап 2*	
	AUC	*F-мера*	*AUC*	*F-мера*	*AUC*	*F-мера*	*AUC*	*F-мера*
U_1	0.8447	0.7990	0.9801	0.8764	0.9503	0.8234	0.9714	0.9555
U_2	0.8545	0.7843	0.9855	0.8981	0.9543	0.8353	0.9835	0.9563
U_3	0.9337	0.8606	0.9804	0.9062	0.9513	0.8630	0.9959	0.9784
U_4	0.9745	0.8910	0.9916	0.9247				
U_5	0.8732	0.8170	0.9856	0.8995				
U_6	0.8873	0.8232	0.8795	0.8311				
U_7	0.8230	0.7772	0.9600	0.8539				
U_8	0.9394	0.8766	0.9831	0.9406				
U_9	0.8822	0.8120	0.9984	0.9418				
U_{10}	0.8740	0.8128	0.9981	0.9276				
U_{11}	0.9115	0.8300	0.9493	0.8846				
U_{12}	0.9082	0.8457	0.9162	0.8796				
U_{13}	0.9252	0.8434	0.9902	0.8802				
U_{14}	0.8867	0.7658	0.9305	0.9124				
U_{15}	0.8537	0.7626	0.9118	0.8629				
Среднее значение	**0.8915**	**0.8201**	**0.9627**	**0.8946**	**0.9520**	**0.8406**	**0.9836**	**0.9634**

Таблица 6. Результаты алгоритм LLGC ($\alpha = 0.85$, $k = 10$, количество итераций = 20)

Как видно из таблиц, предложенные методы показывают высокое качество классификации, несмотря на то, что на этапе первоначальной разметки множества пользовательских писем качество классификации в большинстве случаев не является удовлетворительным.

На рис.1 представлены средние значения AUC различных классификаторов. Классификаторы, построенные без использования неразмеченных данных *(SVM, NBM),* показывают слабое качество классификации в задаче **A**, а в задаче **B** и вовсе бесполезны.

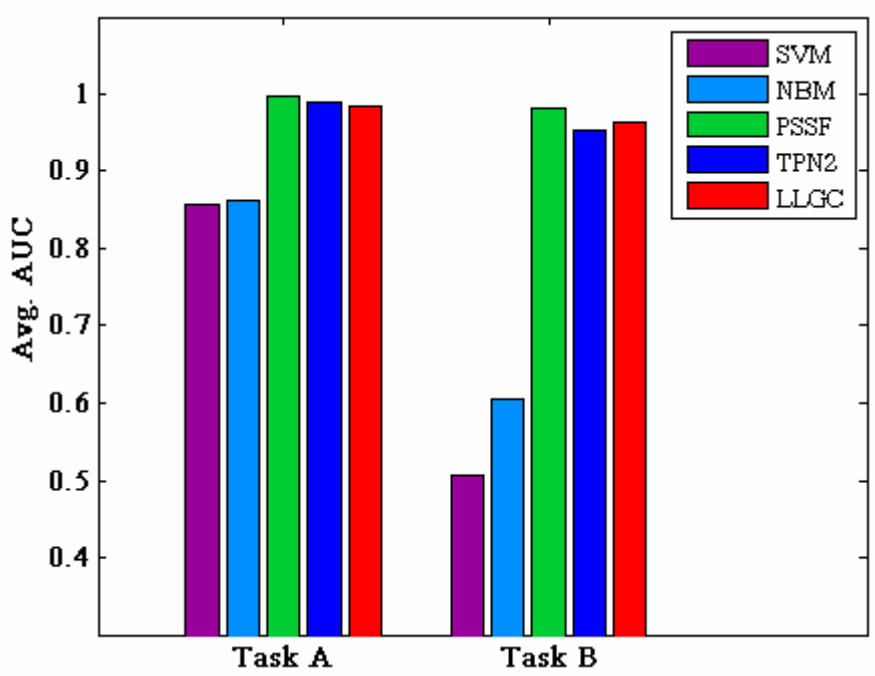


Рис.1. Сравнение величины AUC различных классификаторов

В целом эффективность работы описанных алгоритмов примерно одинакова. На тестовом наборе **A** все алгоритмы достигают отличного качества классификации. Что касается тестового набора **B**, то для алгоритмов *PSSF* и *TPN²* оценка качества также довольно высока, а для *LLGC* – несколько ниже.

При сравнении результатов классификации наборов **A** и **B**, видно, что каждый алгоритм работает примерно одинаково для всех пользовательских ящиков набора **A**, тогда как для набора **B** эффективность алгоритмов заметно меняется от ящика к ящику (таблицы 4, 5, 6). Вероятнее всего это зависит от степени несоответствия обучающего и классифицируемого множества. В целом задача **B** из-за небольших размеров обучающего и классифицируемых множеств является более сложной задачей. Чтобы исследовать влияние размера обучающего множества на эффективность алгоритмов, проведем эксперимент, используя модифицированное

обучающее множество, в которое добавим по 20 писем из каждого ящика (получим 400 писем). Результаты эксперимента приведены в таблице 7.

	PSFF	***TPN²***	***LLGC***
Среднее значение F-меры	0.9451	0.9503	0.9401
Среднее значение AUC	0.9880	0.9647	0.9801

Таблица 7. Результаты классификации (размер обучающего множества = 400)

Результаты показывают, что за счет небольшого улучшения обучающего множества качество классификации заметно повысилось, это подчеркивает необходимость иметь достаточное количество писем в обучающем множестве.

Помимо качества классификации на тестовом наборе данных одной из важнейших характеристик алгоритма является его обобщающая способность, т.е. способность выявить как можно больше объективных закономерностей при минимальном количестве ложных закономерностей. После того как классификатор обучен, используя размеченные и неразмеченные данные, он применяется для классификации новых данных. По эффективности работы на этих «невидимых» данных можно судить об обобщающей способности алгоритма.

Для получения оценки обобщающей способности алгоритмов, разделим множество пользовательских писем на две части: часть 1 содержит письма, используемые в процессе обучения, а часть 2 – новые при обучении письма. Результаты эксперимента показаны на рис.2 и рис.3. По оси x отложен размер части 2 в процентах от общего количества писем в пользовательском ящике, по оси y – среднее значение F-меры классификации части 1 и части 2.

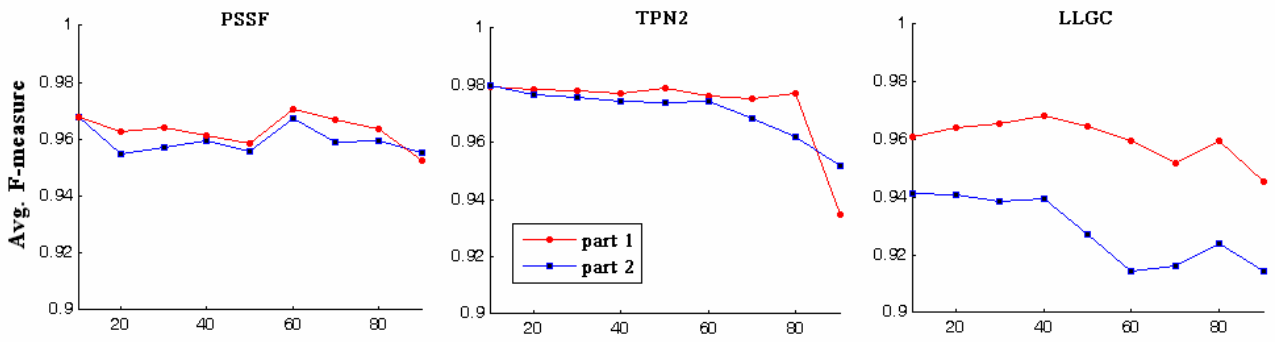


Рис.2. Оценка обобщающей способности алгоритмов (набор **A**)

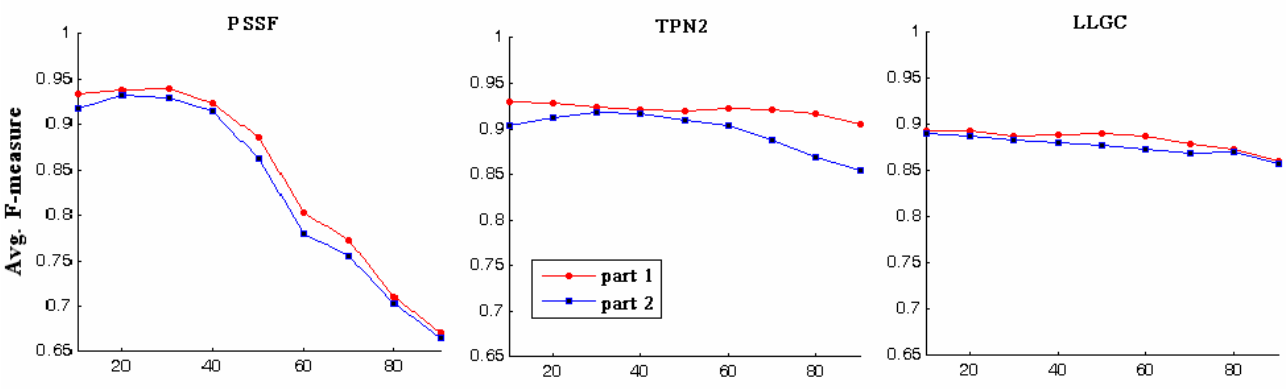


Рис.3. Оценка обобщающей способности алгоритмов (набор **B**)

Как видно, среднее значение F-меры классификации части 2 меньше, чем части 1, но в основном разница составляет менее 0.05. Таким образом, полученные классификаторы обеспечивают низкую выходную ошибку как на множестве, используемом в процессе обучения, так и на «невидимых» данных, т.е. алгоритмы обладают хорошей обобщающей способностью.

Для тестового набора **B** с увеличением размера части 2 (при уменьшении части 1) эффективность классификации алгоритма *PSSF* заметно снижается, тогда как для алгоритмов *TPN²* и *LLGC* такого снижения качества не происходит, таким образом, с их помощью можно получить качественный фильтр даже при наличии небольшого количества пользовательских писем.

Заключение

В данной работе рассматривалась задача классификации текстов с помощью частичного машинного обучения на примере задачи персонализированной фильтрации спама. Целью было изучение существующих подходов к решению поставленной задачи и оценка качества их работы. В ходе данной работы были получены следующие результаты:

- Изучена предметная область.
- Реализованы три алгоритма в среде MatLab, предложенные для решения поставленной задачи.
- Предложены модификации некоторых алгоритмов, позволившие увеличить скорость и качество их работы.
- В работе представлено экспериментальное исследование алгоритмов. Предметом исследования стали такие важные характеристики алгоритмов, как скорость и точность работы, а также обобщающая способность алгоритмов.

В процессе исследования были получены следующие основные результаты. Качество классификации алгоритмов довольно высокое даже при наличии небольшого количества неразмеченных примеров. Алгоритмы обладают хорошей обобщающей способностью. Также стоит отметить высокую скорость работы алгоритмов с большими объемами данных. В целом исследованные алгоритмы показали неплохие результаты, на основании которых можно сделать вывод о возможности их успешного применения для персонализированной фильтрации спама на стороне сервера и других задач.

Список литературы

1. Bickel S. *ECML/PKDD Discovery Challenge 2006 Overview*. [PDF] (http://www.ecmlpkdd2006.org/discovery_challenge2006_overview.pdf).
2. Pfahringer B. *A semi-supervised spam mail detector*. ECML-PKDD Discovery Challenge Workshop, 2006. [PDF] (http://www.ecmlpkdd2006.org/pfahringer.pdf).
3. Trogkanis N. and Paliouras G. *TPN²: Using positive-only learning to deal with the heterogeneity of labeled and unlabeled data*. ECML-PKDD Discovery Challenge Workshop, 2006. [PDF] (http://www.ecmlpkdd2006.org/trokanis.pdf).
4. Junejo K., Yousaf M., and Karim A. *A two-pass statistical approach for automatic personalized spam filtering*. ECML-PKDD Discover Challenge Workshop, 2006. [PDF] (http://www.ecmlpkdd2006.org/karim.pdf).
5. Pfahringer B., Leschi C., Reutemann P. Scaling up semi-supervised learning: an effcient and effective LLGC variant, 2007. [PDF] (http://www.cs.waikato.ac.nz/~fracpete/projects/collective-classification/downloads/2007-pfahringer-scaling_up_semisupervisedpaper927.pdf).
6. Junejo K. and Karim A. *PSSF: A Novel Statistical Approach for Personalized Service-side Spam Filtering*, 2007. [PDF] (http://suraj.lums.edu.pk/~junejo/index_files/junejo_wi07.pdf).
7. Zhou D., Bousquet O., Lal T.N., Weston J., and Schlkopf B. *Learning with local and global consistency.* Cambridge, Mass., 2004. [PDF] (http://books.nips.cc/papers/files/nips16/NIPS2003_AA41.pdf).
8. F. Denis, R. Gilleron and M. Tommasi. *Text classification from positive and unlabeled examples.* IPMU-02., 2002. [PS] (http://www.cmi.univ-mrs.fr/~fdenis/ipmu02.ps).
9. ECML-PKDD: Discovery challenge. [HTML] (http://www.ecmlpkdd2006.org/challenge.html).
10. Паклин Н. *Логистическая регрессия и ROC-анализ - математический аппарат.* [HTML] (http://www.basegroup.ru/library/analysis/regression/logistic).
11. 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).
12. Zho X. *Semi-Supervised Learning Literature Survey*. Technical Report 1530, University of Wisconsin- Madison, 2005. [PDF] (http://pages.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf)