Математические структуры и моделирование
2010, вып. 21, с. 65–81

УДК 004.93'1

АВТОМАТИЧЕСКАЯ КЛАССИФИКАЦИЯ ТЕКСТОВЫХ ДОКУМЕНТОВ

А.С. Епрев

В данной статье приводится обзор некоторых актуальных методов и подходов, применяемых при решении задач классификации текстовых документов.

Введение

Классификация (категоризация, рубрикация) текстовых документов является задачей автоматического определения документа в одну или несколько категорий (рубрик, тематик) на основании содержания документа. В зарубежной литературе получил широкое распространение термин Text Categorization [1,2].

В настоящее время мы имеем дело с постоянно увеличивающимся объемом обрабатываемой и накапливаемой информации, что делает задачу классификации все более актуальной. Использование классификаторов, позволяет ограничить поиск необходимой информации относительно небольшим подмножеством документов.

Помимо сужения области поиска в поисковых системах задача классификации имеет практическое применение в следующих областях:

- фильтрация спама;
- составление тематических каталогов;
- контекстная реклама;
- системы документооборота;
- автоматический перевод текстов (снятие омонимии).

Данная работа посвящена обзору методов и подходов, применяемых на различных этапах решения задачи классификации текстов.

Омский государственный университет им. Ф.М. Достоевского.
E-mail: anton@eprev.me

1. Формализация задачи

Задача классификации текстов может быть формализована [2] как задача аппроксимации неизвестной функции $\Phi : D \times C \to \{0, 1\}$ (каким образом документы должны быть классифицированы) через функцию $\hat{\Phi} : D \times C \to \{0, 1\}$, именуемую *классификатором*, где $C = \{c_1, \ldots, c_{|C|}\}$ — множество возможных категорий, а $D = \{d_1, \ldots d_{|D|}\}$ — множество документов.

$$\Phi(d_j, c_i) = \begin{cases} 0, & \text{если } d_j \notin c_i; \\ 1, & \text{если } d_j \in c_i. \end{cases}$$

Документ d_j называют *положительным примером* категории c_i, если $\Phi(d_j, c_i) = 1$, и *отрицательным* в противном случае.

Если в задаче каждому документу $d_j \in D$ может соответствовать только одна категория $c_i \in C$, то имеет место *однозначная классификация*, а если произвольное количество категорий $0 < n_j < |C|$ — *многозначная классификация*. Далее будет рассматриваться случай многозначной классификации, если не сказано иного.

Выделяют особый вид классификаторов — *бинарные* (двоичные), множество категорий которых состоит из двух элементов (c_i и его дополнения $\bar{c}_i$). Бинарный классификатор для $\{c_i, \bar{c}_i\}$ определяется функцией $\hat{\Phi}_i : D \to \{0, 1\}$, которая является аппроксимацией неизвестной функции $\Phi_i : D \to \{0, 1\}$.

Нахождение классификатора для множества категорий $C = \{c_1, \ldots, c_{|C|}\}$ обычно рассматривается как поиск $|C|$ бинарных классификаторов $\{c_i, \bar{c}_i\}$, где $i = 1, \ldots, |C|$. Таким образом, классификатор $\hat{\Phi}$ представляет собой множество бинарных классификаторов.

2. Автоматическая классификация

При решении задач автоматической классификации текстовых документов используются методы информационного поиска (Information Retrieval, IR) [3, 4] и машинного обучения (Machine Learning, ML) [4–6].

Документы на есстественном языке преобразовываются в удобную для машинной обработки форму — *индексируются*. В процессе индексирования происходит выделение признаков из документа.

Классификатор для категории c_i автоматически создается в *процессе обучения*, при котором просматривается множество документов с заранее определенными категориями c_i или $\bar{c}_i$ и подбираются такие характеристики классификатора, чтобы новый (ранее не просмотренный) документ, отнесенный к категории c_i, соответствовал им. Чтобы построить классификатор C, необходимо множество документов D, для которых значения функции $\Phi(d_j, c_i)$ известны для каждой пары $\langle d_j, c_i \rangle \in D \times C$.

В экспериментах обычно все множество документов D разделяется на два непересекающихся подмножества [2]:

- набор для обучения (обучающая выборка) $\mathcal{L}$;

- набор для проверки (тестирующая выборка) $\mathcal{T}$.

На обучающем множестве $\mathcal{L}$ строится классификатор и определяются значения его параметров, при которых классификатор выдает лучший результат. На тестовом наборе $\mathcal{T}$ происходит *вычисление эффективности* построенного классификатора. Индексирование документов, построение классификатора и вычисление его эффективности являются темами следующих разделов.

3. Индексирование документов

Индексирование документов в задачах классификации текстов не представлено разнообразием методов и подходов. Обычно документ после индексации представляется как вектор в некотором пространстве (*пространстве признаков*), в котором каждому терму (признаку) ставится в соответствие его вес (значимость):

$$\vec{d_j} = \langle \omega_{1j}, \ldots, \omega_{|T|j} \rangle,$$

где T — словарь, т.е. множество термов, которые встречаются в $|\mathcal{L}|$ обучающих классификатор документах, и $0 \leqslant \omega_{kj} \leqslant 1$ определяет значимость терма t_k в документе d_j.

Подходы к индексированию различаются в методе *определения термов* и способе *вычисления весов*.

Обычно термами являются слова, встречающиеся в документе (за исключением так называемых *стоп-слов*, т.е. нейтральных слов, таких как союзы, предлоги, местоимения и т.п.). Слова, как правило, подвергаются морфологическому разбору или стеммингу (специальный алгоритм для определения морфологического корня слова [7]). В классификации текстов также используется подход к использованию не отдельных слов в качестве термов, а словосочетаний, которые выделяются при помощи синтаксического разбора предложений или статистически, что придает таким термам семантически большую значимость [9, 10].

Вес ω_{kj} может просто определять наличие терма в документе, в таком случае $\omega_{kj} \in \{0, 1\}$. Но обычно в качестве весовых значений используются вещественные числа из диапазона $0 \leqslant \omega_{kj} \leqslant 1$. Такие веса имеют статистическую или вероятностную природу и зависят от метода построения классификатора.

Для определения веса терма можно привлекать дополнительную информацию. Например, если терм находится в заголовке документа, то его вес можно увеличить на несколько процентов. Документы в наборе, как правило, имеют разную длину, поэтому полученные веса принято *нормализовывать*. С обзором подходов к вычислению весов термов можно ознакомиться в статье [11].

3.1. Класс весовых функций $tf * idf$

Наиболее популярным классом статистических весовых функций является $tf * idf$, в котором определены два интуитивных правила: чем чаще терм t_k встречается в документе d_j, тем более значим он в нем (*term frequency*); чем

в большем количестве документов терм t_k встречается, тем менее отличительным он является (*inverse document frequency*). Существует множество вариантов $tf * idf$. Приведем один из них:

$$\omega_{ij} = \frac{tf_{ij} \cdot idf_i}{\sqrt{\sum_k (tf_{kj} \cdot idf_k)^2}},$$

где ω_{ij} — вес i-го терма в документе d_j, tf_{ij} — частота встречаемости i-го терма в рассматриваемом документе (term frequency), $idf_i = \log N/n$ — логарифм отношения количества документов в коллекции к количеству документов, в которых встречается i-ый терм (inverse document frequency). Веса, вычисленные по этой формуле, нормализованы таким образом, что сумма квадратов весов каждого документа равна единице.

4. Уменьшение размерности пространства признаков

Эмпирический закон Хипса [13] связывает рост количества обрабатываемых документов с ростом словаря T. Вычислительная сложность различных методов классификации напрямую зависит от размерности пространства признаков. Поэтому в задачах классификации часто прибегают к сокращению числа используемых термов, т. е. к уменьшению значения $|T|$ до $|T'| \ll |T|$.

Побочным эффектом уменьшения размерности пространства признаков (УР) является *переобучение* (overfitting), когда классификатор слишком хорошо работает на документах из обучающего набора и достаточно плохо на документах, не участвующих в обучении.

Для УР можно прибегнуть к выбору термов из существующих (feature selection) или к созданию искусственных (feature extraction).

4.1. Выбор признаков

Существуют различные методы выбора термов: оставлять наиболее встречающиеся термы в документах или выбирать наиболее значимые, используя различные функции полезности.

Наиболее простой подход к уменьшению размерности пространства признаков заключается в нахождении значений $df(t_k)$ (document frequency — количество документов из $\mathcal{L}$, в которых встречается терм t_k) и выборе наиболее встречающихся. Янг в своей работе [14] показывает возможность уменьшения размерности пространства признаков в 10 раз без потери эффективности и отмечает, что уменьшение в 100 раз приводит к незначительным ухудшениям работы классификатора.

Рассмотрим теперь некоторые функции полезности $f(t_k, c_i)$, характеризующие значимость терма t_k в некотором документе для категории c_i. Чтобы вычислить значимость терма для всей коллекции C, можно, например, найти

среднее значение

$$f(t_k) = \sum_{i=1}^{|C|} P(c_i) f(t_k, c_i)$$

или максимальное

$$f(t_k) = max_{i=1}^{|C|} f(t_k, c_i).$$

Широкое распространение получили функции полезности «прирост информации», «взаимная информация» и «метод хи-квадрат».

Прирост информациии (Information Gain, IG) определяется по формуле

$$IG(t_k, c_i) = \sum_{c \in \{c_i, \bar{c}_i\}} \sum_{t \in \{t_k, \bar{t}_k\}} P(t, c) \cdot \log \frac{P(t, c)}{P(t) \cdot P(c)},$$

где, например, $P(\bar{t}_k, c_i)$ — вероятность того, что терм t_k не встречается в некотором документе d и документ d определен в категорию c_i.

Взаимная информация (Mutual Information, MI):

$$MI(t_k, c_i) = \log \frac{P(t_k, c_i)}{P(t_k) \cdot P(c_i)}.$$

И критерий хи-квадрат:

$$\chi^2 = \frac{|\mathcal{L}| \cdot [P(t_k, c_i) \cdot P(\bar{t}_k, \bar{c}_i) - P(\bar{t}_k, c_i) \cdot P(t_k, \bar{c}_i)]^2}{P(t_k) \cdot P(\bar{t}_k) \cdot P(c_i) \cdot P(\bar{c}_i)}.$$

С подробным сравнением этих методов уменьшения размерности можно ознакомится в статье [14].

4.2. Извлечение признаков

Для уменьшения размерности пространства признаков могут применяться методы кластеризации термов и латентно-семантическое индексирование, в результате которых образуются (извлекаются) новые признаки, способствующие увеличению эффективности классификации [1].

При **кластеризации признаков** происходит объединение в группы термов с высокой попарной семантической близостью, представления этих групп или их центроиды используются в качестве признаков для уменьшения размерности пространства.

Бейкер и Маккалум в своей работе [15] описывают метод кластеризации, при котором уменьшение размерности пространства в 1000 раз приводит к потере эффективности классификации всего на 2%.

В качестве исходных данных в **латентно-семантическом индексировании** (Latent Semantic Indexing, LSI) используется матрица *термы-на-документы*. Столбцы этой матрицы – документы, а строки – термы. Элементами этой матрицы являются веса термов в документах. Задача уменьшения размерности пространства заключается в нахождении сингулярного разложения матрицы.

Разложение матрицы $A \in \mathbb{R}^{m \times n}$ в произведение двух ортогональных матриц $U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times n}$ и диагональной матрицы $D = diag(\sigma_1, \ldots, \sigma_p) \in \mathbb{R}^{m \times n}$, где $p = min(m, n)$, называется сингулярным [16]:

$$A = UDV^T.$$

Элементы $\sigma_i \geq 0$ диагональной матрицы D являются корнями собственных чисел матрицы AA^T. Если в матрице D оставить только k наибольших чисел, то произведение полученной диагональной матрицы D' и двух ортогональных U, V является самым близким приближением матрицы A ранга k:

$$A' = UD'V^T.$$

Теперь каждый документ и признак можно представить как линейную комбинацию k линейно-независимых столбцов и строк соответственно.

5. Методы построения классификаторов

В литературе можно встретить различные методы построения классификаторов. Некоторые из них строят двоичные функции $\hat{\Phi} : D \times C \to \{0, 1\}$, а некоторые — вещественные функции $CSV : D \times C \to [0, 1]$ (Categorization Status Value). Если используются первые, то имеет место *точная классификация*, если вторые — *пороговая классификация*. Для последних необходимо определить множество пороговых значений τ_i для $i = 1, \ldots, |C|$ (вычисляются экспериментально на обучающем наборе), которые позволяют рассматривать вещественные значения CSV как двоичные:

$$\hat{\Phi}(d_j, c_i) = \begin{cases} 0, & \text{если } CSV_i(d_j) < \tau_i; \\ 1, & \text{если } CSV_i(d_j) \geqslant \tau_i. \end{cases}$$

Стоит отметить, что в некоторых приложениях вещественные функции могут с успехом использоваться без необходимости преобразования к двоичным. Например, классификаторы с такими функциями могут строить «рейтинг» категорий для документа, просматривая который человек утверждает определенные из них.

Далее будут рассмотрены некоторые из классических методов построения текстовых классификаторов, которые могут служить отправной точкой для разработки более эффективных методик.

5.1. Деревья решений

Классификатор на базе деревьев решений (Decision Trees) [17] для категории c_i представляет собой дерево, узлами которого являются термы t_k, каждое ребро обозначено условием $\geqslant v_k$ или $< v_k$, а листья помечены как c_i или $\bar{c}_i$. Чтобы классифицировать документ d_j в категорию c_i или $\bar{c}_i$, необходимо пройти по

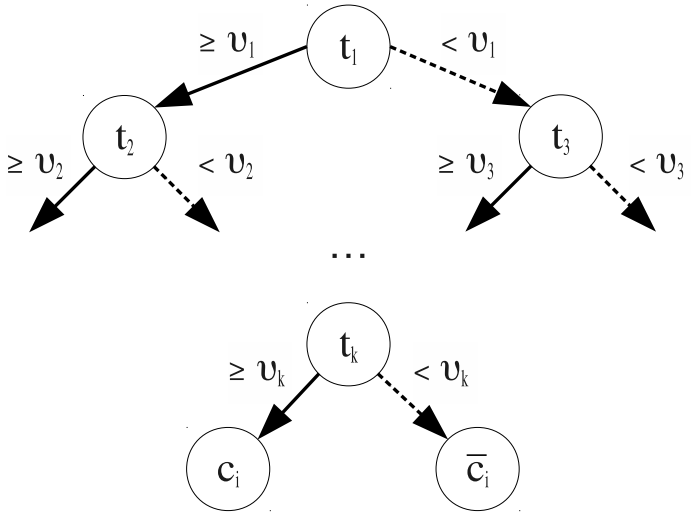

Рис. 1. Дерево решений для категории c_i.

узлам дерева начиная с корня, сравнивая веса терма в документе ω_{kj} со значениями υ_k на ребрах. На практике обычно используют бинарные деревья решений, в которых принятие решения перехода по ребрам осуществляется простой проверкой наличия терма в документе.

Один из способов автоматического построения деревьев решений заключается в последовательном разбиении множества обучающих документов $\mathcal{L}$ на классы, до тех пор пока в классе не останется документов, определенных только в одну из категорий c_i или $\bar{c}_i$. На каждом этапе в качестве узла дерева выбирается терм t_k и определяется υ_k, затем множество документов разбивается на два класса: $\omega_k \geqslant \upsilon_k$ и $\omega_k < \upsilon_k$.

Обычно построенное дерево решений является сильно детализированным (эффект переобучения), поэтому применяются различные алгоритмы усечения дерева. Широкое применение получили алгоритмы ID3 [18] и C4.5 [19].

5.2. Решающие правила

Этот класс классификаторов строит правила классификации вида «если выполняется формула, то категория». В статье [21] можно ознакомиться с системой классификации документов CONSTRUE. Эта система использовалась агентством Reuters для классификации новостных сообщений. Правила классификации составлялись экспертами вручную. Вот одно из таких правил:

```
if   (wheat & farm) or
     (wheat & commodity) or
     (bushels & export) or
     (wheat & tonnes) or
     (wheat & whinter and (¬ soft))
then
     WHEAT
else
     (¬ WHEAT).
```

Одним из способов автоматического построения правил классификации является перебор всевозможных правил в виде формул определенного вида. В статье [22] описывается алгоритм, который применяется при решении задач медицинской направленности. Алгоритм строит формулы вида:

$$C = I_1 \cup I_2 \cup \ldots \cup I_q,$$

где I_i – это конъюнкция p_i признаков. Пространство признаков размерностью порядка нескольких десятков состоит из различных медицинских показателей. Время работы алгоритма экспоненциально зависит от размерности пространства признаков.

Другой подход заключается в построении формул на основе деревьев решений. Каждому пути от корня до листа в дереве решений соответствует правило, где условиями будут являться проверки из узлов, встретившихся на пути. Чтобы получить формулу для категории c_i, нужно объединить все правила для путей, листьями которых является c_i.

5.3. Метод наименьших квадратов

В методе наименьших квадратов (Linear Least-Squares Fit, LLSF) [23] с каждым документом d_j связывают два вектора: входной вектор весов термов $I(d_j)$ размерности $|T|$ и выходной вектор весов категорий $O(d_j)$ размерности $|C|$. Задача классификации сводится к определению вектора $O(d_j)$ по вектору $I(d_j)$:

$$MI(d_j) = O(d_j),$$

где M — матрица размера $|C| \times |T|$.

Построение классификатора заключается в нахождении матрицы $\hat{M}$ минимизирующей норму $\|\hat{M}I - O\|_F$, где I — матрица размера $|T| \times |\mathcal{L}|$, столбцы которой являются векторами $I(d_j)$, O — матрица размера $|C| \times |\mathcal{L}|$, столбцы которой являются векторами $O(d_j)$, и $\|V\|_F$ — норма Фробениуса для матрицы размера $n \times m$:

$$\|V\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m} v_{ij}^2}.$$

Метод наименьших квадратов позволяет минимизировать корень из суммы квадратов всех ошибок при обучении классификатора. Матрицу $\hat{M}$ получают

через сингулярное разложение матрицы, построенной на обучающем множестве, а элементы m_{ik} определяют связь между категорией c_i и термом t_k.

5.4. Адаптивные линейные классификаторы

В классе линейных классификаторов категории и документы представлены векторами $\vec{c}_i = \langle \omega_{1i}, \ldots, \omega_{|T|i} \rangle$ и $\vec{d}_j = \langle \omega_{1j}, \ldots, \omega_{|T|j} \rangle$ соответственно. В качестве $CSV_i(d_j)$ используется значение косинуса угла между векторами $\vec{d}_j$ и $\vec{c}_i$:

$$cos(\vec{d}_j, \vec{c}_i) = \frac{\sum\limits_{k=1}^{|T|} \omega_{ki} \cdot \omega_{kj}}{\sqrt{\sum\limits_{k=1}^{|T|} \omega_{ki}^2} \cdot \sqrt{\sum\limits_{k=1}^{|T|} \omega_{kj}^2}}.$$

Адаптивные линейные классификатора (On-Line Linear Classifiers) [24–26] осуществляют построение классификатора на первом просмотренном документе и постоянно совершенствуют его на последующих.

Один из методов автоматического построения классификатора для категории c_i заключается в следующем. Изначально $\vec{c}_i = \langle 1, \ldots, 1 \rangle$. Затем для каждого документа $d_j \in \mathcal{L}$, представленного вектором $\vec{d}_j$ с двоичными весами, вычисляется $CSV_i(d_j)$ и вносятся поправки в $\vec{c}_i$, если результат классификации неверен : если $d_j \in c_i$, тогда веса ω_{ki} для k, таких, что $\omega_{kj} = 1$, увеличивают на некоторую фиксированную величину $\alpha > 0$ и понижают на эту же величину в том случае, если $d_j \notin c_i$.

Такой подход позволяет продолжить обучение классификатора после его построения. Кроме того, термы t_k с маленькими весами ω_{ki}, которые не оказывают влияния на результаты классификации, можно выбросить из рассмотрения, тем самым уменьшить размерность пространства признаков.

5.5. Метод Rocchio

Использовать метод Rocchio [27] для построения линейного классификатора впервые было предложено в работе [28]. Для каждой категории c_i вычисляем вектор $\vec{c}_i = \langle \omega_{1i}, \ldots, \omega_{|T|i} \rangle$ по формуле

$$\omega_{ki} = \beta \cdot \sum_{d_j \in D_i^+} \frac{\omega_{kj}}{|D_i^+|} - \gamma \cdot \sum_{d_j \in D_i^-} \frac{\omega_{kj}}{|D_i^-|},$$

где ω_{kj} — вес терма t_k в документе d_j, $D_i^+ = \{d \in \mathcal{L} \mid \Phi(d_j, c_i) = 1\}$ и $D_i^- = \{d \in \mathcal{L} \mid \Phi(d_j, c_i) = 0\}$. Параметры β и γ определяют значимость положительных и отрицательных примеров. В случае, когда $\beta = 1$ и $\gamma = 0$, вектор $\vec{c}_i$ будет являться центроидом положительных примеров категории c_i.

5.6. Метод k-NN

Метод k-ближайших соседей (k-Nearest Neighbours, k-NN) [29] в отличие от других не требует обучения. Для того чтобы найти категории, соответствующие документу d_j, классификатор сравнивает d_j со всеми документами из обучающей выборки $\mathcal{L}$: для каждого $d_z \in \mathcal{L}$ вычисляется «расстояние» $\rho(d_j, d_z)$. Далее из обучающей выборки выбираются k документов, ближайших к d_j. Для категорий вычисляются функции ранжирования $CSV_i(d_j)$ по формуле

$$\sum_{d_z \in \mathcal{L}_k(d_j)} \rho(d_j, d_z) \cdot \Phi(d_z, c_i),$$

где $\mathcal{L}_k(d_j)$ — это ближайшие k документов из $\mathcal{L}$ к d_j.

Параметр k обычно выбирается в интервале $20 \ldots 50$. Документ d_j определен в категории, для которых $CSV_i(d_j) \geqslant \tau_i$. Данный метод дает высокую эффективность, но при этом требователен к вычислительным ресурсам на этапе классификации.

5.7. Метод опорных векторов

Метод опорных векторов (Support Vector Machine, SVM) [31] заключается в нахождении гиперплоскости в пространстве признаков $\mathbb{R}^{|T|}$, разделяющей его на две части: положительные примеры в одной и отрицательные в другой — у которой минимальное расстояние до ближайших примеров максимально.

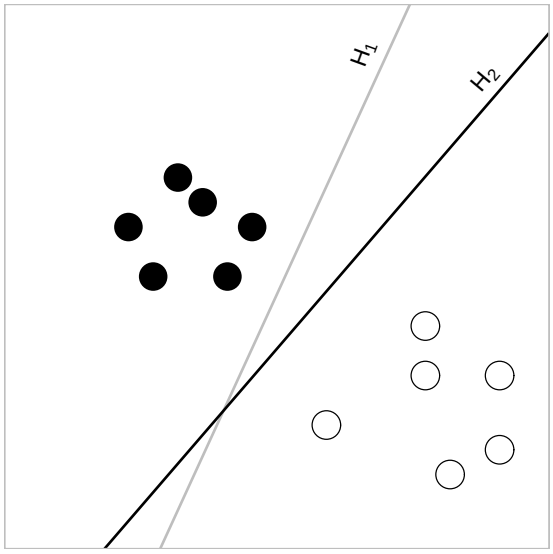

Рис. 2. Метод опорных векторов. Две классифицирующих разделяющих прямых (гиперплоскости), но только H_2 разделяет два множества с большим отступом

В том случае, когда такой разделяющей гиперплоскости не существуют (проблема линейной неразделимости), используют подход, заключающийся в пе-

реходе от исходного пространства признаков к новому, в котором обучающая выборка окажется линейно разделимой.

Для классификаторов на базе метода опорных векторов, как правило, не требуется уменьшать размерность пространства признаков; они довольно устойчивы к переобучению и хорошо масштабируются [31]. Подробную информацию о методе опорных векторов можно найти в работах [32, 33].

5.8. Метод Байеса

В вероятностном классификаторе [34] используется векторное представление документов, а функции $CSV_i(d_j)$ рассматриваются в терминах условных вероятностей $P(c_i|\vec{d_j})$ (вероятность того, что документ, представленный вектором $\vec{d_j}$, соответствует категории c_i). Работа вероятностного классификатора заключается в вычислении значений $P(c_i|\vec{d_j})$ для $i = 1 \dots |C|$ и нахождении наибольшей такой вероятности:

$$H(\vec{d_j}) = \arg\max_{c_i \in C} P(c_i|\vec{d_j}).$$

Условную вероятность $P(c_i|\vec{d_j})$ согласно теореме Байеса можно переписать как

$$P(c_i|\vec{d_j}) = \frac{P(\vec{d_j}|c_i) \cdot P(c_i)}{P(\vec{d_j})}, \tag{1}$$

где $P(c_i)$ — это априорная вероятность того, что документ определен в категорию c_i, $P(\vec{d_j}|c_i)$ — вероятность найти документ, представленный вектором $\vec{d_j}$, в категории c_i и $P(\vec{d_j})$ — вероятность того, что случайно выбранный документ будет иметь вектор $\vec{d_j}$.

По сути $P(c_i)$ является отношением количества документов из обучающей выборки $\mathcal{L}$, отнесенных в категорию c_i, к количеству всех документов из $\mathcal{L}$:

$$P(c_i) = \frac{|\{d_j \in \mathcal{L} \mid d_j \in c_i\}|}{|\mathcal{L}|}.$$

Чтобы вычислить $P(\vec{d_j}|c_i)$ (и $P(\vec{d_j})$) необходимо сделать допущение о том, что вхождение термов в документ зависит от категории, но не зависит от других термов этого документа. Таким образом, $P(\vec{d_j}|c_i)$ можно записать как

$$P(\vec{d_j}|c_i) = \prod_{k=1}^{|T|} P(\omega_{kj}|c_i).$$

В свою очередь $P(\omega_{kj}|c_i)$ можно определить как отношение количества документов из обучающей выборки $\mathcal{L}$, отнесенных в категорию c_i и содержащих терм t_k, к общему количеству всех документов из $\mathcal{L}$, отнесенных в категорию c_i:

$$P(\omega_{kj}|c_i) = \frac{|\{d_j \in \mathcal{L} \mid d_j \in c_i, t_k \in d_j\}|}{|\{d_j \in \mathcal{L} \mid d_j \in c_i\}|}.$$

Из-за предположения о независимости признаков такой классификатор называют «наивный байесовский» (Naive Bayes Classifier) [9, 20, 31, 35].

5.9. Другие методы

В литературе можно встретить и другие методы построения классификаторов, такие как байесовские [36,37] и нейронные [24–26,38–40] сети, методы с использованием генетических алгоритмов [41,42] и N-грамм [43].

6. Оценка эффективности

Эффективность классификатора $\hat{\Phi}_i$ является качественной оценкой результатов его работы на тестирующей выборке $\mathcal{T}$. Эффективность используется для сравнения различных методов классификации.

При однозначной классификации эффективность классификатора можно рассматривать как отношение верно классифицированных документов к общему количеству документов. Но в случае использования бинарных классификаторов (при многозначной классификации) такое отношение не пригодно для оценки эффективности. Причиной этому является то, что категории c_i и $\bar{c}_i$ обычно не сбалансированы, то есть одна из них содержит намного больше документов, чем другая. В таком случае построение классификатора, который бы давал высокую эффективность, является тривиальной задачей — достаточно, чтобы классификатор сопоставлял всем документам наиболее часто встречающуюся категорию. Но о практической применимости такого классификатора не может быть и речи.

Как результат, для бинарных классификаторов часто применяется оценка эффективности как комбинация *точности* (precision) и *полноты* (recall). Точность p_i — доля верно классифицированных в c_i документов, а полнота r_i — отношение верно классифицированных в c_i документов к общему количеству документов, которые должны были быть классифицированы в c_i.

Например, в категорию c_i было отнесено 10 документов, из них только восемь оказались правильно классифицированы. Таким образом, точность $p_i = 8/10$. Но в тестирующей выборке было 12 документов, отнесенных в c_i, получаем, что $r_i = 8/12$.

В задачах многозначной классификации эффективность, которая заключается в вычислении точности и полноты для каждой категории индивидуально, необходимо усреднить. Существует два подхода усреднения (таблица 1): микроусреднение (которое учитывает «вес» категории) и макроусреднение (для которого все категории равны). Наиболее часто используется макроподход, потому что именно он позволяет рассмотреть категории независимо от их встречаемости в корпусе документов.

Классификаторы можно настроить на увеличение точности в ущерб простоте (и наоборот) изменением значений τ_i в $CSV_i : D \to \{0, 1\}$ [1]. Поэтому только комбинация точности и полноты может служить хорошей оценкой эффективности. Наиболее популярным способом объединения этих двух оценок является функция

$$F_\beta = \frac{(\beta^2 + 1)pr}{\beta^2 p + r},$$

Таблица 1. Усреднение точности и полноты; $D_i = \{d_j \in \mathcal{T} \mid \Phi(d_j, c_i) = 1\}$, $\hat{D}_i = \{d_j \in \mathcal{T} \mid \hat{\Phi}(d_j, c_i) = 1\}$

	Точность, p	Полнота, r
Микроусреднение	$p = \dfrac{\sum_{i=1}^{\lvert C\rvert} \lvert \hat{D}_i \cap D_i\rvert}{\sum_{i=1}^{\lvert C\rvert} \lvert \hat{D}_i\rvert}$	$r = \dfrac{\sum_{i=1}^{\lvert C\rvert} \lvert \hat{D}_i \cap D_i\rvert}{\sum_{i=1}^{\lvert C\rvert} \lvert D_i\rvert}$
Макроусреднение	$p = \dfrac{1}{\lvert C\rvert} \sum_{i=1}^{\lvert C\rvert} \dfrac{\lvert \hat{D}_i \cap D_i\rvert}{\lvert \hat{D}_i\rvert}$	$r = \dfrac{1}{\lvert C\rvert} \sum_{i=1}^{\lvert C\rvert} \dfrac{\lvert \hat{D}_i \cap D_i\rvert}{\lvert D_i\rvert}$

где $0 \leqslant \beta \leqslant \infty$. Обычно в качестве значения β используется единица, и функция F_β принимает, вид в котором точность и полнота находятся в равных весовых категориях:

$$F_1 = \frac{2pr}{p + r}.$$

Заметим, что для классификатора, который назначает всем документам часто встречающуюся категорию, $p = 1$ и $r = 0$ таким образом, $F_\beta = 0$ для любого β. Аналогично и для классификатора, который всем документам будет назначать все категории, в таком случае $p = 0$ и $r = 1$, а $F_\beta = 0$ для любых β.

В некоторых приложениях классификации текстов, таких как фильтрация спама (бинарный классификатор для определения электронного письма в одну из двух категорий – «спам» и его дополнения «не спам»), точность имеет большее значение, чем полнота, потому что отнесение «хорошего» письма в категорию «спам» является большей ошибкой, чем принятие нежелательного сообщения как «не спам». Оценкой эффективности может служить F_β, где $0 \leqslant \beta < 1$, что позволяет больше внимания уделять точности чем полноте. А выбором значения $1 < \beta < \infty$ внимание уделяетеся полноте в ущерб точности.

7. Сравнение классификаторов

Сравнение методов построения классификаторов является довольно сложной задачей по причине того, что разные входные данные могут приводить к различным результатам. Чтобы провести сравнение различных классификаторов, необходимо выполнить их построение и вычисление эффективности на одинаковых наборах документов для обучения и тестирования. Широкое распространение получил корпус текстов Reuters-21578 [44], для которого существуют фиксированные разбиения на обучающее и тестирующее множества.

Таблица 2. Сравнение эффективности различных классификаторов [40]

Метод	Микро r	Микро p	Микро F_1	Макро F_1
Метод опорных векторов	.8120	.9137	.8599	.5251
Метод k-ближайших соседей	.8339	.8807	.8567	.5442
Метод наименьших квадратов	.8507	.8489	.8498	.5008
Нейронная сеть	.7842	.8785	.8287	.3765
Метод Байеса	.7688	.8245	.7956	.3886

В таблице 2 приведены результаты эксперименто, опубликованных в работе [40]. Построение классификаторов и оценка их эффективности проводилась с использованием разбиения «ModApte» коллекции документов Reuters-21578. Это разбиение задает 90 категорий, 9603 документа содержатся в обучающем наборе и 3299 документов в тестирующем.

С результатами других экспериментов можно ознакомиться в [10] (сравниваются байесовские сети, деревья решений, методы Байеса и опорных векторов) и [31] (сравниваются методы Баейса, Rocchio, k-ближайших соседей, опорных векторов и деревья решений). Автор статьи [40] отмечает, что его результаты немного отличаются от опубликованных в [31], но классификатор, построенный на базе метода опорных векторов, также имеет небольшое преимущество перед остальными.

8. Ансамбли классификаторов

Использование комбинации классификаторов позволяет повысить точность классификации [45]. Идея заключается в построении k классификаторов $\hat{\Phi}_1, \ldots, \hat{\Phi}_k$ и объединении их результатов классификации. В машинном обучении широкое распространение получили методы «bagging» и «boosting» [46], которые основаны на изменении обучающего множества.

В методе «bagging» построение k классификаторов $\hat{\Phi}_i$ осуществляется независимо друг от друга на обучающих множествах, полученных из исходного случайной заменой документов (размер обучающего множества остается прежним, просто одни документы отсутствуют, а другие встречаются несколько раз). Результат классификации определяется простым большинством голосов элементов ансамбля.

Идея метода «boosting» заключается в последовательном построении k классификаторов, при котором на классификатор $\hat{\Phi}_i$ оказывают влияние $\hat{\Phi}_1, \ldots, \hat{\Phi}_{i-1}$. Классификатор $\hat{\Phi}_i$ строится на исходном обучающем множестве, документы d_j которого участвуют в обучении с некоторыми весовыми коэффициентами h_j^i. После обучения классификатор $\hat{\Phi}_i$ проверяется на исходной обучающей выборке и происходит пересчет коэффициентов. Коэффициент h_j^{i+1} уменьшается, если документ d_j классифицирован верно, и увеличивается в противном случае. В методе «boosting» используется взвешенная линейная комби-

нация голосов элементов ансамбля.

В работе [47] приводятся теоретическое обоснование и результаты эксперимента, которые показывают, что комбинации независимых классификаторов наиболее эффективны. В последнее время большую популярность получили методы, в которых обучение отдельных элементов ансамбля осуществляется независимо на различающихся подмножествах признаков [48, 49].

Литература

1. Sebastiani F. Machine Learning in Automated Text Categorization // ACM Computing Surveys. 2002. V. 34, N. 1. P. 1–47.
2. Sebastiani F. Text Categorization // Text Mining and Its Applications. WIT Press, Southampton, UK, 2005. P. 109–129.
3. Manning C., Raghavan P., Schutze H. Introduction to Information Retrieval. Cambridge University Press, 2008. 544 p.
4. Adam Berger. Statistical Machine Learning for Information Retrieval. Carnegie Mellon University, 2001. 143 p.
5. Witten I. H., Frank E. Data Mining: Practical Machine Learning Tools and Techniques (Second Edition). Morgan Kaufmann, 2005. 525 p.
6. Paliouras G., Karkaletsis V., Spyropoulos C. D. Machine Learning and Its Applications: Advanced Lectures (Lecture Notes in Computer Science / Lecture Notes in Artificial Intelligence). Springer, 2001. 325 p.
7. Bill Frakes. Stemming algorithms // Information Retrieval: Data Structures and Algorithms. Englewood Cliffs, US. 1992. P. 131–160.
8. Thorsten Joachims. A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization // Proceedings of International Conference on Machine Learning (ICML). 1997. 26 p.
9. Lewis D.D. An evaluation of phrasal and clustered representations on a text categorization task // Proceedings of SIGIR-92, 15th ACM International Conference on Research and Development in Information Retrieval. ACM Press, US. 1992. P. 37–50.
10. Dumais S.T., Platt J., Heckerman D., Sahami M. Inductive learning algorithms and representations for text categorization // Proceedings of CIKM-98, 7th ACM International Conference on Information and Knowledge Management, Bethesda, MD. 1998. P. 148–155.
11. Salton G, Buckley C. Term-Weighting Approaches in Automatic Text Retrieval // Information Processing and Management. 1988. P. 513–523.
12. Lewis D.D., Ringuette M. A comparison of two learning algorithms for text categorization // Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, US. 1994. P. 81–93.
13. Heaps H.S. Information Retrieval: Computational and Theoretical Aspects. Academic Press, 1978. 368 p.
14. Yang Y. Pedersen J.O. A comparative study on feature selection in text categorization // Proceedings of ICML-97, 14th International Conference on Machine Learning. Morgan Kaufmann Publishers, San Francisco, US: Nashville, US. 1997. P. 412–420.

15. Baker L.D., McCallum A.K. Distributional clustering of words for text classification // Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval, Melbourne, Australia. 1998. P. 96–103.
16. Meltzer T. SVD and its Application to Generalized Eigenvalue Problems. 2004. 16 p.
17. Mitchell T. M. Machine Learning. McGraw Hill, New York, 1997. 414 p.
18. Quinlan J. Induction of decision trees // Machine Learning. 1998. V. 1, N. 1. P. 81–106.
19. Quinlan J. C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993. 302 p.
20. Li Y. H., Jain A. K. Classification of Text Documents // The Computer Journal. 1998. V. 41, N. 8. P. 537–546.
21. Hayes P.J., Weinstein S.P. Construe: A System for Content-Based Indexing of a Database of News Stories // Proceedings of the Second Annual Conference on Innovative Applications of Intelligence. 1990.
22. Marshall R.J. Generation of Boolean classification rules // Proceedings of Computational Statistics, Utrecht, The Netherlands. 2000. P. 355–360.
23. Yang Y., Chute C. G. An example-based mapping method for text categorization and retrieval // ACM Trans. Inform. Syst. 1994. V. 12, N. 3. P. 252–277.
24. Schutze H., Hull D. A., Pedersen J. O. A comparison of classifiers and document representations for the routing problem // Proceedings of SIGIR-95, 18th ACM International Conference on Research and Development in Information Retrieval, Seattle. 1995. P. 229–237.
25. Ng H. T., Goh W. B., Low K. L. Feature selection, perceptron learning, and a usability case study for text categorization // Proceedings of SIGIR-97, 20th ACM International Conference on Research and Development in Information Retrieval, Philadelphia. 1997. P. 67–73.
26. Dagan I., Karov Y., Roth D. Mistake-driven learning in text categorization // Proceedings of EMNLP-97, 2nd Conference on Empirical Methods in Natural Language Processing, Providence, RI. 1997. P. 55–63.
27. Rocchio J.J. Relevance feedback in information retrieval // The SMART Retrieval System: Experiments in Automatic Document Processing. 1971. P. 313–323.
28. Hull D. A. Improving text retrieval for the routing problem using latent semantic indexing // Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval, Dublin, Ireland. 1994. P. 282–289.
29. Yang Y. Expert network: effective and efficient learning from human decisions in text categorisation and retrieval // Proceedings of SGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval, Dublin, Ireland. 1994. P. 13–22.
30. Vapnik V., The Nature of Statistical Learning Theory. Springer-Verlag, 1995. 188 p.
31. Joachims T. Text categorization with support vector machines: learning with many relevant features // Proceedings of ECML-98, 10th European Conference on Machine Learning, Chemnitz, Germany. 1998. P. 137–142.
32. Воронцов К. В. Лекции по методу опорных векторов. 2007. 18 с. URL: http://www.ccas.ru/voron/download/SVM.pdf (дата обращения: 12.12.2009)
33. Cristianini N., Shawe-Taulor J. An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, 2000. 189 p.
34. Lewis, D.D. Naive (Bayes) at forty: The independence assumption in information retrieval // Proceedings of ECML-98, 10th European Conference on Machine Learning,

Chemnitz, Germany. 1998. P. 4–15.
35. Koller D., Sahami M. Hierarchically classifying documents using very few words // Proceedings of ICML-97, 14th International Conference on Machine Learning, Nashville. 1997. P. 170–178.
36. Heckerman D. A Tutorial on Learning With Bayesian Networks // Learning in graphical models. 1999. P. 301–354.
37. L. M. de Campos, A. E. Romero. Bayesian network models for hierarchical text classification from a thesaurus // International Journal of Approximate Reasoning. 2009. V. 50, N. 7. P. 932–944.
38. Lam S.L., Lee D.L. Feature reduction for neural network based text categorization // Proceedings of DASFAA-99, Taiwan. 1999. P. 195–202.
39. Ruiz M., Srinivasan P. Hierarchical Text Categorization Using Neural Networks // Information Retrieval. 2002. V. 5, N. 1. P. 87–118.
40. Yang Y., Liu X. A re-examination of text categorization methods // Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval, Berkeley, CA. 1999. P. 42–49.
41. Wong M. L., Cheung K. S. Data Mining Using Grammar Based Genetic Programming and Applications. Kluwer Academic Publishers, 2002. 228 p.
42. Lankhorst M. Automatic Word Categorization with Genetic Algorithms // Proceedings of the ECAI'94 Workshop on Applied Genetic and other Evolutionary Algorithms. 1994.
43. Cavnar W. B., Trenkle J. M. N-Gram-Based Text Categorization // Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval. 1994. P. 161–175.
44. Lewis D. The Reuters-21578 text categorizationtest collection. 1999. URL: http://www.daviddlewis.com/resources/testcollections/reuters21578/ (дата обращения: 12.12.2009)
45. Dietterich T. G. Machine learning research: four current directions // AI Magazine. 1997. V. 18. P. 97–136.
46. Quinlan J. R. Bagging, Boosting, and C4.5 // Proceedings of AAA/IAAI. 1996. P. 725–730.
47. Oza N. C., Tumer K. Decimated input ensembles for improved generalization // Proceedings of the International Joint Conference on Neural Networks, Washington, DC. 1999.
48. Bryll R. Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets // Pattern Recognition. 2003. V. 36. P. 1291–1302.
49. Bay S. D. Nearest neighbor classifiers from multiple feature subsets // Intelligent data analysis. 1999. V. 3. P. 191–209.