Классификация текстов

Ю. Лифшиц*

23 ноября 2005 г.

Содержание

1 Постановка задачи, подходы и применения

1.1 Неформальное описание

Задача состоит в том, чтобы **автоматически классифицировать** набор текстов по категориям. Сам процесс должен быть именно **автоматическим**: классификация происходит без помощи специально обученных экспертов, которые могут определить, попадает тот или иной документ в определенный раздел, никакого подбора правил вручную. **Классификацию** не нужно путать с кластеризацией, то есть объединением текстов. Задача автоматической кластеризации имеет отдельный интерес, но мы не будем ее затрагивать.

Нас будет интересовать только синтаксическая сторона вопроса. Категории - это всего лишь символические ярлыкы, никакой дополнительной информации, описания. Документ - набор слов без каких-либо внешних данных, к примеру, даты публикации, типа документа, источников и т.д.

Будут использоваться два метода:

- Информационного поиска (Information Retrieval)
- Машинного обучения (Machine Learning)

1.2 Формальная постановка задачи

Определим задачу формально. Пусть заданы некоторое конечное множество категорий $\mathfrak{C} = \{c_1, \ldots, c_{|\mathfrak{C}|}\}$, конечное множество документов $\mathfrak{D} = \{d_1, \ldots, d_{|\mathfrak{D}|}\}$ и неизвестная целевая функция Φ, которая для каждой пары <документ, категория> определяет, соответствуют ли они друг другу: $\Phi : \mathfrak{D} \times \mathfrak{C} \to \{0, 1\}$.

*Законспектировал А. Киракозов.

Задача состоит в том, чтобы найти максимально близкую к функции Φ функцию Φ'. Функцию Φ' называют **классификатором**.

Машинное обучение основывается на начальной коллекции документов $\Omega = \{d_1, \ldots, d_{|\Omega|}\} \subset \mathfrak{D}$. При этом, значение целевой функции Φ известно для каждой пары $< d_j, c_i > \in \Omega \times \mathfrak{C}$. Документы из Ω разделяют на две непересекающиеся коллекции:

- **"учебную"** $Tr = \{d_1, \ldots, d_{|Tr|}\}$. Коллекция документов, с помощью которой создается классификатор Φ'. Φ' обучаться индуктивно, основываясь на замеченных характеристиках этих документов.
- **"тестовую"** $Te = \{d_{|Tr|+1}, \ldots, d_{|\Omega|}\}$. Коллекция документов, на которой тестируется эффективность построенного классификатора. Каждый "тестовый"документ подается на вход классификатору Φ', а затем сравнивается результат классификатора $\Phi'(d_j, c_i)$ с известным значением функции $\Phi(d_j, c_i)$. Классификатор считается тем эффективнее, чем чаще эти значения совпадают.

Документ $d \in \Omega$ называется **положительным** или **отрицательным** примером для категории c, если значение функции $\Phi(d, c)$ равно 1 или 0, соответственно.

1.3 Виды классификации

Существует несколько различных видов классификации. В зависимости от ответа, классификация бывает:

- точная : $\Phi' : \mathfrak{C} \times \mathfrak{D} \to \{0, 1\}$
- ранжированная : $\Phi' : \mathfrak{C} \times \mathfrak{D} \to [0, 1]$

Выше мы рассматривали только **точную классификацию**, которая каждой паре <документ, категория> сопоставляло булево значение 1 или 0 (соответствует документ категории или нет). Теперь предположим, что классификатор Φ' умеет определять степень принадлежности документа к категории, причем эта степень являться числом, лежащем в диапозоне от 0 до 1 (чем больше степень принадлежности, тем больше документ подходит этой категории). Такой вид классификации называется **ранжированным**.

Существует два различных порядка обработки документов. Первый, когда есть набор категорий, и поступает новый документ, для которого нужно определить список подходящих ему категорий. Второй, когда, наоборот, есть коллекция документов, и создается новая категория. Необходимо найти все документы, соответствующие данной новой категории. Эти два вида классификаций довольно симметричны, хотя существуют методы, которые работают для одного из них и не работают для другого, но это скорее исключение, чем правило.

Между множеством категорий $\mathfrak{C}$ возможны следующие соотношения:

- категории не пересекаются,

- категории могут пересекаться,
- две непересекающиеся категории.

Наибольшей интерес представляет последнее из них, которое имеет самостоятельное название **бинарная классификация**. К бинарной классификации можно свести все остальные: для любой категории $c \in \mathfrak{C}$ будем определять, принадлежит документ категории c или ее дополнению $\overline{c}$.

1.4 Применения классификации текстов

- Фильтрация документов, распознавание спама
- Автоматическая аннотирование
- Снятие неоднозначности (автоматические переводчики)
- Составление интернет-каталогов
- Классификация новостей
- Распределение рекламы
- Персональные новости

1.5 Основные шаги

Классификацию можно разделить на три основных этапа:

- индексация документов,
- обучение классификатора,
- оценка качества классификации.

Индексацией называется процесс приведения документов к единному формату. Зачастую приходиться иметь дело с большими объемами информации, поэтому при индексации страраются экономить и выкидывать все лишнее. К примеру, некоторые слова (вроде предлогов) могут очень часто встречаться во всех документах, но при этом они не несут никакой смысловой нагрузки.

Обучение классификатора состоит в выборе общей формы классифицирующего правила со множеством различных параметров. Далее производится настройка параметров на "обучающем" наборе документов.

Существует два подхода к **оценки качества классификации**. Первый - это сравнение классификаторов между собой, второй - абсолютная оценка качества. Вообще говоря, довольно сложно оценить качество классификации. Часто даже опытные эксперты не могут определить к какой категории отнести конкретный документ, но для описанной выше формальной системы можно предложить несколько способов оценки абсолютного качества классификации. Каждый из них имеет свои "сильные" и "слабые" стороны. Рассмотрим три этих этапа более подробно.

2 Индексация документов

2.1 Стандартный подход

Будем считать, что каждый документ - это просто набор слов (термов). Множество всех термов обозначим за $\mathcal{T}$. Каждый терм $t_i \in \mathcal{T}$ имеет вес w_{ij} по отношению к документу $d_j \in \mathfrak{D}$. Таким образом, каждый документ можно представить в виде вектора весов его термов $\overrightarrow{d}_j =< w_{1j}, \ldots, w_{|\mathcal{T}|j} >$. Веса документов нормируют так, чтобы $0 \leq w_{ij} \leq 1$ для $\forall i, j : 0 \leq i \leq |\mathcal{T}|, 0 \leq j \leq |\mathfrak{D}|$.

Вес терма в документе можно определить следующим образом:

$$w_{ij} = TF_{ij} \cdot IDF_i$$

Зесь TF_{ij} - это отношение числа термов t_i в документе d_j к общему числу термов в этом документе, а IDF_i - число, обратное количеству документов, в котором встречается терм t_i. Таким образом, чем чаще слово встречается в документе, но реже встречается вообще во всех документах, тем больше вес этого терма в данном документе.

Нормализовать вес терма в документе можно следующим стандартным способом:

$$w_{ij} = \frac{TF_{ij} \cdot IDF_i}{\sqrt{\sum_{s=1}^{|\mathcal{T}|}(TF_{sj} \cdot IDF_s)^2}}$$

2.2 Уменьшение размерности

Документы обычно состоят из очень большого числа слов. Существует ряд методов, которые позволяют уменьшить их количество. Для каждой категории можно использовать ее собственный метод, но чаще используют один единый метод для всех категорий. Прежде всего можно выкинуть бесполезные слова. Слова, которые встречаются очень часто скорее всего являются "мусором" (предлоги, союзы и т.д.), от них можно избавиться. То же самое касается слов, которые встречаются слишком редко, т.к. скорее всего они не несут никакой смысловой нагрузки. Таким образом следует оставлять только "средне-встречающиеся" слова.

Другая техника состоит в определении "коэффициентов полезности" термов. Пусть мы имеем дело с бинарной классификацией. Предположим, что вероятность встретить терм t в документе d, если документ d принадлежит категории c близка к вероятности встретить терм t в документе d, если документ d принадлежит категории $\overline{c}$. Тогда, если документ содержит терм t, то он (документ) может равновероятно попасть как в категорию c, так и в ее дополнение. Поэтому можно считать такие термы бесполезными. "Коэффициент полезности" в данном случае можно определить как модуль разности этих двух условных вероятностей.

Следующая идея состоит в создании искусственных термов. Несколько термов можно объединять в один (**кластеризация термов**). Например,

все однокоренные слова можно заменить на одно слово. Можно использовать словарь синонимов, или, в общем случае, определять метрики синонимичности слов. Так же можно использовать технику **сингулярного** разложения, которая была рассмотрена на предыдущей лекции.

3 Построение и обучение классификатора

3.1 Ранжирование и четкая классификация

Предположим, что для каждой категории c_i построена функция $CSV_i : \mathfrak{D} \to [0,1]$, которая задает степень принадлежности документа категории. Задача состоит в том, чтобы от ранжированной функции перейти к точной классификации. Наиболее простой способ - для каждой категории c_i выбрать пороговое значение τ_i. Если $CSV_i(d) > \tau_i$, то документ d соответствует категории c_i. Другой метод: для каждого документа d выбирать k ближайших категорий, т.е. k категорий, на которых $CSV_i(d)$ принимают наибольшие значения.

3.2 Метод Rocchio

Некоторые классификаторы используют, так называемый, **профайл** (или прототип документа) для категории. Профайл - это список взвешенных термов, присутствие или отсутствие которых наиболее хорошо отличают категорию c_i от других категорий. Метод Rocchio используется для линейных классификаторов, то есть классификаторов которые представляют каждый документ в виде вектора весов термов (см. 2.1). Профайл $\overrightarrow{c_i} =< w_{1i}, \ldots, w_{|\mathcal{T}|i} >$ для категории c_i в методе Rocchio рассчитывается по следующей формуле:

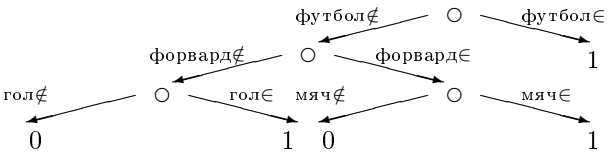
$$w_{ki} = \beta \cdot \sum_{\{d_j \in POS_i\}} \frac{w_{kj}}{|POS_i|} - \gamma \cdot \sum_{\{d_j \in NEG_i\}} \frac{w_{kj}}{|NEG_i|},$$

где w_{kj} - это вес терма t_k в документе d_j, $POS_i = \{d_j \in Tr | \Phi(d_j, c_i) = 1\}$ и $NEG_i = \{d_j \in Tr | \Phi(d_j, c_i) = 0\}$. В этой формуле, β и γ - контрольные параметры, которые характеризуют значимость положительных и отрицательных примеров. Например, если $\beta = 1$ и $\gamma = 0$, $\overrightarrow{c_i}$ будет центром масс всех положительных примеров категории c_i.

Функция $CSV_i(d)$ определяется как величина обратная расстоянию от документа $d \in \mathfrak{D}$ до профайла категории c_i. Метод Rocchio дает хорошие результаты, когда документы из одной категории близки друг другу (в смысле расстояния). Но если, например, категории c_i соответствуют тексты на двух разных языках (английском и русском), то расстояние от профайл категории c_i до документа на любом языке будет очень большим, поэтому значение функции CSV_i будет маленьким, и ни один документ не попадет в эту категорию.

3.3 Разрешающие деревья

Идея данного метода состоит в построении **разрешающего дерева** на "обучающем" наборе документов. Дерево строится по следующему правилу: выбираем терм, документы содержащие его кладем направо, остальные налево. Таким образом, документы разделились на две непересекающиеся коллекии. Для каждой коллекции выбирается новый терм и повторяется описанная выше процедура. Так продолжается до тех пор пока не получится однородная коллекция, то есть коллекция, в которой либо все документы соответствуют категории, либо все документы соответствуют ее дополнению. Для большей ясности рассмотрим пример для категории "ФУТБОЛ".



Из рисунка видно, что все "обучающие" документы, содержащие слово "футбол", соответствуют категории. Если в документе не встречается слово "футбол" , но встречаются слова "форвард" и "мяч", то документ тоже подходит данной категории, хотя одного слова "форвард" не достаточно. Это означает, что есть "обучающие" документы, в которых есть слово "форвард", но они относятся к дополнению категории. Наконец, документ может не содержать слов "футбол" и "форвард", но если в нем встречаеться слово "гол", то его так же отнесут к категории "ФУТБОЛ".

В примере дерево очень маленькое, реально оно может сильно разрастаться, если множество "обучающих" документов велико. К тому же и "переобучать" большое дерево затруднительно. Встает вопрос, как же правильно выбирать термы, чтобы дерево не разрасталось? Для выбора термов можно использовать "коэффициент полезности" терма, который мы уже рассматривали в пункте 2.2.

3.4 Метод k соседей

Вернемся вновь к линейным классификаторам. Метод k соседей состоит в том, чтобы определять функцию $CSV_i(d)$ через ближайшие (в смысле расстояния) к d "обучающие" документы.

$$CSV_i(d) = \sum_{d_z \in Tr_k(d)} \frac{\Phi(c_i, d_z)}{\rho(d, d_z)},$$

где $Tr_k(d)$ - k ближайших к документу d "обучающих"документов, а $\rho : \mathfrak{D} \times \mathfrak{D} \to \mathcal{R}$ - функция расстояния (метрика) в пространстве документов. Рекомендуется брать k $\sim$ 20-50.

3.5 Другие методы

Существует еще ряд методов, которые не были затронуты на лекции:

- вероятностные классификаторы,
- нейронные сети,
- Support Vector Machines.

О них будет рассказано на следующей лекции.

3.6 Комитеты классификаторов

Допустим уже построено несколько классификаторов. Хотелось бы уметь объеденять эти классификаторы в один более эффективный. Встает резонный вопрос, какому классификатору отдавать предпочтение, если их результаты различны? Существует несколько методов объеденения:

- **Выбор большинства** - берем результат, который дают большинство классификаторов.
- **Взвешенная линейная комбинация** - для каждого классификатора Φ'_i зададим натуральное число n_i, которое будет характеризовать нашу степень доверия классификатору (чем больше число, тем мы считаем классификатор ближе к Φ, и тем его голос более весомен). Затем подсчитываем уже взвешенное число голосов по формуле $\sum_i n_i \cdot \Phi'_i(d, c)$.
- **Динамический выбор классификатора** - для каждого классификатора определяем категории, в которых он наиболее "компетентен". В зависимости от категории выбираем тот или иной классификатор.
- **Динамическая комбинация классификаторов** - объеденение "взвешенной линейной комбинации" и "динамического выбора классификатора", то есть в зависимости от категории определяем степень доверия классификатору. А затем, как и ранее, подсчитываем число голосов.

4 Оценка качества классификации

4.1 Методы из информационного поиска

Для оценки качества классификации используются метрики из информационного поиска, которые были рассмотрены на прошлой лекции.

- **Полнота:** отношение количества найденных документов из категории к общему количеству документов категории.

- **Точность:** доля документов действительно из категории в общем количестве найденных документов.
- **Benchmarks:** показатели системы на контрольных запросах и специальных коллекциях документов.
- **Аккуратность:** доля верно соотнесенных документов во всех документах.

К примеру, пусть было 100 документов, и получились следующие результаты.

	$\Phi(d,c)=1$	$\Phi(d,c)=0$
$\Phi'(d,c)=1$	10	15
$\Phi'(d,c)=0$	5	70

- Полнота равна $\frac{10}{10+5}=\frac{2}{3}$.
- Точность равна $\frac{10}{10+15}=\frac{2}{5}$.
- Аккуратность равна $\frac{70+10}{100}=\frac{4}{5}$.

Представим теперь, что классификатор всегда будет возвращать 0, то есть отправлять все документы в дополнение категории c, тогда значение аккуратности будет равно $\frac{80+5}{100}=\frac{17}{20}$. Как можно заметить, аккуратность может быть большей для абсолютно бессмысленного классификатора, поэтому ее нужно использовать с осторожностью.

Классификатор может совершить две разные ошибки, отнести документ к категории, когда документ не принадлежит категории, и, наоборот, сказать, что документ не соответствуют категории, когда он ей принадлежит. Значимость этих ошибок может быть разной, в зависимости от нее выбирают ту или иную метрику.

4.2 Сравнение двух методов

Вообще говоря, сравнение двух разных классификаторов довольно сложная задача, так как в зависимости от разных входных данных они могут давать различные результаты. Поэтому обычно сравнивают классификаторы с одинаковыми индексациями, на одинаковых коллекциях документов, с одинаковым "обучающим" набором. Тем не менне, все равно остается не ясным, каков будет результат, если мы расширим коллекцию документов. Вполне возможно, что "отстающий" классификатор обгонит "лидирующего" на новом наборе документов.

Другой метод состоит в том, чтобы создать некий "эталонный" примитивный документ, а затем сравнивать все классификаторы с ним.