УДК 004.912

С.В. Шанов, П.Г. Чупин, А.Ю. Афонин

ПРИМЕНЕНИЕ БАЙЕСОВСКОГО КЛАССИФИКАТОРА ДЛЯ ОПРЕДЕЛЕНИЯ ТЕМАТИКИ ТЕКСТА

ФГПУ ВПО Пензенский государственный университет, Пенза, Россия

Актуальность исследования обусловлена потребностью современного общества в автоматической классификации данных. В данной работе рассмотрен байесовский алгоритм на примере определения тематики текста. Целью работы является разработка, выявление и решение проблем, возникающих во время реализации и непосредственной работы классификатора, а также оценка его эффективности. Выявлены проблемы арифметического переполнения и появления нулевой вероятности в результате. Предложено их решение с помощью сглаживания Лапласа и свойства логарифмов. Также представлены подходы по оптимизации и увеличению скорости работы программного модуля. В результате был реализован байесовский классификатор. Его обучение проводилось на базе наборов статей 10 различных тематик. На основе полученных данных проведена тестовая классификация и выполнена проверка корректности данной операции. Материалы статьи представляют практическую ценность для тех, кто собирается применить рассмотренный алгоритм или подобные ему в своих исследованиях.

Ключевые слова: наивный байесовский классификатор, Text Mining, алгоритм, теорема Байеса, анализ документа.

Введение. С каждым днем объем текстовой информации постоянно увеличивается. При таком стремительном росте доступных данных классификация и анализ полезной информации становятся практической необходимостью. Актуальность исследования обусловлена потребностью современного общества в автоматической классификации данных.

Задача классификации применяется в следующих областях:

- автоматический рубрикатор;
- таргетированная реклама;
- фильтрация спама;
- документооборот.

Для решения таких задач используются методы информационного поиска и машинного обучения.

В связи с этим целью данной работы является разработка байесовского классификатора, выявление и решение проблем, возникающих во время реализации, а также оценка его эффективности. Данный алгоритм позволит проводить классификацию текстовой информации с хорошей точностью и высокой скоростью.

Постановка задачи. Была принята следующая постановка задачи. Документы на естественном языке приводятся к единому стандарту с

помощью методов Text Mining – интеллектуальный анализ текстовых данных [1]. Задача классификации текстов может быть формализована как задача, аппроксимации неизвестной функции f: D×C →{0,1}. Где C={c_1, ... ,$c_{|C|}$} – множество возможных категорий, а D={d_1, … , $d_{|D|}$} – множество документов [2].

Классификатор для категории C_i создается заранее. Работа с классификатором происходит с множеством документов D, которое можно разделить на два набора [2]:

- набор для обучения (обучающая выборка) L;
- набор для проверки (тестирующая выборка) K.

Создание классификатора происходит путем обучения его на множестве L, в результате определяются его параметры. Проверка эффективности работы полученного классификатора происходит на множестве K.

Математические особенности алгоритма. Существует множество различных подходов для классификации данных, однако, несмотря на постоянные разработки и совершенствование текущих алгоритмов, наивный байесовский классификатор (НБК) является одним из наиболее популярных алгоритмов по причине простоты реализации и минимальных человеческих и финансовых затрат при внедрении в информационные системы конечных потребителей.

Подробно рассмотрим математическую модель НБК и обозначим ключевые особенности.

Наивный Байесовский классификатор – вероятностный классификатор, который делает предсказания, основываясь на списке определенных классов. Он основан на теореме Байеса, которая позволяет рассчитать вероятность некоторого события, на основании факта, что произошло некоторое другое событие [3]. Формула Байеса(1):

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}, (1)$$

где:

- $P(c|d)$ – вероятность, что документ “d” принадлежит классу “c”;
- $P(d|c)$ – вероятность встретить документ “d” среди всех документов класса “c”;
- $P(c)$ – безусловная вероятность встретить документ класса “c” в корпусе документов;
- $P(d)$ – безусловная вероятность документа “d” в корпусе документов.

Конечной задачей является определение к какому классу принадлежит документ, поэтому необходимо получить не саму вероятность, а наиболее вероятный класс. Для этого НБА производит

оценку апостериорного максимума. Другими словами, рассчитывается вероятность для всех классов и выбирается тот, который обладает максимальной вероятностью [3]

Безусловная вероятность документа $P(d)$является константой, следовательно, не может повлиять на результат, поэтому знаменатель исключается. Формула принимает вид (2):

$$c_{map} = arg\max_{c \in C}[P(d|c)P(c)], (2)$$

В натуральном языке вероятность появления слова сильно зависит от контекста. Алгоритма Байеса использует подход “bag of words model”. В этой модели документ представляется в виде мультимножества его слов, игнорируя грамматику и даже порядок слов, но сохраняя множественность, т.е. документ представляется набором отдельных слов, у которых вероятность их появления не зависит друг от друга. Именно поэтому классификатор называют наивным. Исходя из этого, условная вероятность документа аппроксимируется произведением условных вероятностей всех слов, входящих в документ (3) [3]:

$$P(d|c) \approx P(\omega_1|c)P(\omega_2|c) \dots P(\omega_n|c) = \prod_{i=1}^{n} P(\omega_i|c), (3)$$

Подставив данное выражение в формулу получим(4):

$$c_{map} = arg\max_{c \in C}\left[P(c)\prod_{i=1}^{n} P(\omega_i|c)\right], (4)$$

На практике, зачастую, в произведении вероятностей будет большое количество очень маленьких чисел, что может привести к арифметическому переполнению. Арифметическое переполнение специфичная для компьютерной арифметики ситуация, когда при арифметическом действии результат становится больше максимально возможного значения для переменной, использующейся для хранения результата [4]. Используем свойство логарифма, чтобы избежать переполнения. Логарифм является монотонной функцией, поэтому применив его к обеим частям выражения изменится только его численное значение, а параметры при которых достигается максимум останутся прежними. Логарифмические значения вероятностей гораздо удобнее для анализа, т.к. логарифм от числа близкого к нулю будет числом отрицательным, но в абсолютном значении существенно большим чем исходное число. Формула примет вид (5):

$$c_{map} = \arg\max_{c \in C}[\log P(c) + \sum_{i=1}^{n} \log P(\omega_i|c)], (5)$$

Оценка вероятностей $P(c)$ и $P(\omega_i|c)$ осуществляется на обучающей выборке. Вероятность класса выражается как (6):

$$P(c) = \frac{D_c}{D}, (6)$$

где:

- D – общее количество документов;
- D_c – количество документов, принадлежащих классу "c" в обучающей выборке.

Оценка вероятности слова в классе $P(\omega_i|c)$определяется как(7):

$$P(\omega_i|c) = \frac{W_{ic}}{\sum_{i' \in V} W_{i'c}}, (7)$$

где:

- W_{ic} – количество того, сколько i-ое слово встречается в документах класса "c";
- V – словарь уникальных слов корпуса документов.

Другими словами, числитель показывает сколько раз, включая повторы, слово встречается в документах конкретного класса, а знаменатель – суммарное количество уникальных слов, которые встречаются во всех документах этого класса.

На практике часто встречается следующая проблема. Если во время непосредственной классификации в документе встретится слово, которого не было на этапе обучения, то вероятность слова $P(\omega_i|c)$ будет равна нулю. Это приведет к тому, что такой документ не получится отнести ни к одному классу. Даже если увеличить число документов на этапе обучения, избавиться от этой проблемы не получится. Невозможно составить обучающую выборку, содержащую все возможные слова, т.к. в тексте встречаются опечатки, синонимы и различные неологизмы. Решением данной проблемы является применение сглаживания Лапласа. Идея такого аддитивного сглаживания проста, считается, что каждое слово встречалось на один раз больше, чем есть на самом деле, т.е. прибавляется единица к частоте каждого слова (8):

$$P(\omega_i|c) = \frac{W_{ic} + 1}{\sum_{i' \in V}(W_{i'c} + 1)} = \frac{W_{ic} + 1}{|V| + \sum_{i' \in V} W_{i'c}}, (8)$$

Логически данный подход смещает оценку вероятностей в сторону менее вероятных исходов. Таким образом, слова, которые не встречались на этапе обучения модели получают пусть маленькую, но не нулевую вероятность [3].

В результате получена окончательная формула по которой происходит байесовская классификация (9):

$$c_{map} = \arg\max_{c \in C}[\log\frac{D_c}{D} + \sum_{i=1}^{n} \log\frac{W_{ic} + 1}{|V| + \sum_{i' \in V} W_{i'c}}, (9)$$

Реализация. Реализация классификатора была выполнена с использование языка программирования Python. Необходимо было запрограммировать процедуру обучения, модель НБК и непосредственно сам процесс классификации.

Под моделью НБК понимается совокупность информации, которая вычисляется на основе данных полученных во время обучения классификатора, а именно:

- общее количество документов, участвовавших в обучении;
- количество документов по каждому классу;
- количество уникальных слов в обучающем наборе;
- общее количество слов по каждому классу.

На этапе обучения будем не просто сохранять слова в БД и связывать их с определенной категорией, а также выполнять сохранение статистики по количеству документов и слов, участвовавших в процессе. Перед сохранением важно уделить время предобработке текста. Во-первых, следует удалить все символы, кроме букв и пробелов, в том числе и знаки препинания. Во-вторых, необходимо унифицировать слова, т.е. привести их к определенной, типовой форме. Самым распространенным решением является использование разнообразных «стеммеров», т.е. у слова удаляется окончания и остается только корневая часть. Однако, из-за такого подхода, различные по смыслу слова часто интерпретируются как одинаковые. Поэтому будем использовать другой подход, а именно, нормализацию формы слова. Все слова приводятся в именительный падеж единственного числа. Для этого нами используется свободная библиотека pymorphy2 – морфологический анализатор для русского языка использующий словари из OpenCorpora [5]. Помимо этого, из входного набора были исключены «стоп-слова» и слова меньшие двух символов. К «стоп-словам» можно отнести предлоги, суффиксы, причастия, междометия, цифры, частицы и т. п. Они не несут в себе важного для классификации смысла, а только засоряют обучающую выборку и замедляют скорость работы. Перечисленные действия улучшат точность алгоритма и увеличат скорость работы программы.

На этапе классификации необходимо для каждого класса рассчитать значение следующего выражения и выбрать класс с максимальным значением (10) [3]. Формула в упрощенном виде:

$$\log\frac{D_c}{D} + \sum_{i \in Q} \log\frac{W_{ic} + 1}{|V| + L_c}, (10)$$

где:

- L_c – суммарное количество слов в документах класса "c" в обучающей выборке.

Если возникнет необходимость представить в качестве результата не просто наиболее вероятный класс, а выделить процентное распределение вероятных категорий, то в данном случае, сравнения логарифмических значений ни к чему не приведут, т.к. они не удовлетворяют формальным свойствам вероятностных оценок. Для формирования вероятностного пространства все оценки должны быть больше нуля и меньше одного, и их сумма равна единице.

Выполним нормирование суммы по единице и избавимся от логарифмов возведением экспоненты в степень оценки (11) [3]:

$$P(c|d) = \frac{e^{q_c}}{\sum_{c' \in C} e^{q_{c'}}}, (11)$$

где:

- q_c – это логарифмическая оценка алгоритма класса.

Здесь также необходимо выполнить алгебраическую оптимизацию, чтобы избежать арифметического переполнения при делении экспонент (12). Для этого сократим экспоненту:

$$P(c|d) = \frac{1}{1 + \sum_{c' \in C\{c\}} e^{q_{c'} - q_c}}, (12)$$

.

Оптимизация. Реализовав представленную математическую модель, был получен прототип НБК. Его необходимо было протестировать и оценить с точки зрения производительности и корректности работы. Однако перед этим необходимо обучить созданный классификатор. Для этого были использованы статьи с портала cyberleninka.ru. Возьмем основные категории статей, такие как: Биология, Вычислительная техника, География, Государство и право, Информатика, История, Математика, Машиностроение, Физика, Химия. Используем по 100 статей для обучения каждого раздела.

После этого была выполнена проверка производительности прототипа, для этого использовалась статья объемом 1000 слов. Количество уникальных слов в обучающей выборке – 44285. Время на анализ тестового документа составило 352 секунды. Такой результат является недопустимым, поэтому необходимо было оптимизировать программный код классификатора.

С увеличением обрабатываемых документов словарь термов растет, что и приводит к задержке анализа документа. Такую динамику называют эмпирическим законом Хипса. Это эмпирическая закономерность распределения числа уникальных слов в документе как функция от длины

документа. Вычислительная мощность классификации напрямую зависит от размерности словаря. Поэтому можно выполнить сокращение числа термов. Один из самых простых методов сокращения словаря – это выбор часто встречаемых или более значимых слов. Даже сокращение пространства термов в 10 раз не повредит работе классификатора, при этом скорость работы возрастет.

Опытным путем выяснилось, что рубрика-образующие слова встречаются в статье не меньше 5 раз. Поэтому, во время обучения, остальные слова были исключены из обработки. Далее классификатор был переобучен с новыми критериями отбора. После выполненных действий размер словаря уникальных слов составил 8857, что в 5 раз меньше предыдущего. Повторив анализ тестового документа было получено время выполнения процесса равное 44 секундам, что намного лучше прошлого результата, но все равно недостаточно быстро.

Для еще большего увеличения производительности алгоритма можно прибегнуть к использованию параллельного программирования. Реализуем асинхронную проверку принадлежности документа к каждой рубрике. Для этого пригодятся стандартные модули python’a такие как queue – для очередей и threading – для распараллеливания. Пройдя циклом по всем рубрикам, было выполнено наполнение очереди категориями queue.put(category) и запущены параллельные процессы обработки threading. Thread (target= process_category).start(). В каждом таком процессе программа забирает рубрику queue.get() и начинает ее обработку, после завершения который осуществляется вызов queue.task_done(), сигнализирующий о том, что обработка конкретной категории завершена. Чтобы синхронизировать потоки использовался вызов queue.join(), который ожидает завершение выполнения всех запущенных работ. Помимо этого, рекомендуется использовать СУБД, которая поддерживаем множественное обращение. В данном случае использовалась PostgreSQL. После сделанных изменений время тестового анализа составило 12.2 секунды.

Последним этапом оптимизации было исключение одиночных слов документа из проверки во время непосредственной классификации. Если размер документа будет большим, можно исключить и более частотные слова. Такой лимит можно вычислять автоматически, в зависимости от размера документа. Например, если размер документа 1000 слов, можно исключать слова, которые повторяются не более 1-2 раз, если 3000 слов, то исключаем 3х частотные слова и т.д.

Результаты и обсуждение. В итоге, в обработке тестового документа участвовало 495 слов, вместо 1000. Время выполнения составило 6.48 секунды, что является приемлемым результатом. Благодаря

оптимизации кода и применению параллельного программирования получилось ускорить классификацию в 55 раз.

Однако одной скорости недостаточно, необходимо было проверить точность классификатора. Для это возьмем по 20 статей (не участвовавших в обучении) каждой категории с cyberleninka.ru и определим их рубрику через классификатор.

Получен следующий результат. Тематика текста определена корректно в 161 случае из 200. Точность алгоритма составила 80%. Классификатор показал хороший результат, однако ожидалась большая точность. Изучив материал, по которому производилось обучение, более подробно, можно сделать вывод, что статьи на данном портале не всегда точно соответствуют указанной рубрике, а их тематика может быть равноправно отнесена к нескольким категориям. Из-за этого, обучение получилось не совсем корректным. Поэтому, для более качественного обучение, рекомендуется выбирать только узко ориентированные данные.

Заключение. В результате разработано программное обеспечение, базирующееся на байесовском алгоритме, которое позволяет определять тематику текста на основе данных, полученных во время обучения классификатора. Решены проблемы арифметического переполнения и появление нулевой вероятности с помощью сглаживание Лапласа и свойства логарифмов. Также представлены подходы по оптимизации и увеличению скорости работы программы, такие как параллельное программирование и уменьшение числа термов.

Проведен и проанализирован машинный эксперимент по обучению на основе статей различных тематик. НБА классификация выполняется без особых сложностей и с учетом оптимизации довольно быстро. НБА требует меньший объем обучающих данных, чем другие подобные алгоритмы и лучше работает с категорийными признаками. Эти особенности следует учитывать при выборе классификатора для своих исследований.

ЛИТЕРАТУРА

1. Text Mining [Электронный ресурс]. – Режим доступа: *https://sites.google.com/site/upravlenieznaniami/tehnologii-upravlenia-znaniami/text-mining-web-mining/text-mining* Управление знаниями_ – (Дата обращения: 04.02.2018).
2. А. С. Епрев Автоматическая классификация текстовых документов. // Математические структуры и моделирование 2010, вып. 21, с.65 - 81
3. Наивный байесовский классификатор [Электронный ресурс]. – Режим доступа: *http://bazhenov.me/blog/2012/06/11/naive-bayes* – (Дата обращения: 04.02.2018).

4. А. А. Алексеев, А. С, Катасёв, А. Е. Кириллов, А. П. Кирпичников Классификация текстовых документов на основе Text Minig // Вестник технологического университета. 2016. Т.19, №18 стр 116 – 119.
5. Морфологический анализатор pymorphy2 [Электронный ресурс]. – Режим доступа: *https://pymorphy2.readthedocs.io/en/latest/* – (Дата обращения: 04.02.2018).

S.V. Shanov, P.G. Chupin, A. Y. Afonin

APPLICATION OF THE BAYESOV CLASSIFIER FOR THE DEFINITION OF THE THEMATICS OF THE TEXT

Penza State University, Penza, Russia

The relevance of the study is conditioned by the need of modern society in the automatic classification of data. In this paper, we consider a Bayesian algorithm for the case of determining the subject matter of a text. The purpose of the work is to develop, identify and solve problems arising during the implementation and work of the classifier, as well as to evaluate its effectiveness. Identified problems of arithmetic overflow and the appearance of zero probability as a result. Their solution is proposed by means of Laplace smoothing and the properties of logarithms. Approaches to optimizing and increasing the speed of the program module are also presented. As a result, a Bayesian classifier was implemented. His study was conducted on the basis of sets of articles of 10 different subjects. Based on the results of analytical and test verification. The materials of the article are of practical value for those who are going to apply the algorithm considered or to them in their research.

Keywords: naive Bayesian classifier, Text Mining, algorithm, Bayes theorem, document analysis.

REFERENCES

1. Text Mining. – Access mode: https://sites.google.com/site/upravlenieznaniami/tehnologii-upravlenia-znaniami/text-mining-web-mining/text-mining Knowledge management – (Date of circulation: 04.02.2018).
2. S. Eprev Automatic classification of text documents. // Mathematical structures and modeling 2010, vol. 21, p.65 - 81
3. Naive Bayesian Classifier [Electronic resource]. – Access mode: http://bazhenov.me/blog/2012/06/11/naive-bayes – (Date of circulation: 04.02.2018).
4. A. Alekseev, A. S. Katasev, A. E. Kirillov, A. P. Kirpichnikov Classification of Text Documents Based on Text Minig // Bulletin of the Technological University. 2016. Vol. 19, No. 18 pages 116-119.
5. Morphological analyzer pymorphy2 [Electronic resource]. - Access mode: https://pymorphy2.readthedocs.io/en/latest/ – (Date of circulation: 04.02.2018).