Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовой проект по курсу «Дискретный анализ»

Студент: А. В. Барсов
Преподаватель: С. Сорокин
Группа: М8О-307Б-19

Дата: 18.11.2021
Оценка:
Подпись:

Москва, 2021

Курсовой проект

Задача: Реализуйте систему, которая на основе базы вопросов и тегов к ним, будет предлагать варианты тегов, которые подходят к новым вопросам.

Формат запуска программы в режиме обучения:
`./prog learn --input <input file> --output <stats file>`

Ключ	Значение
--input	входной файл с вопросами
--output	выходной файл с рассчитанной статистикой

Формат запуска программы в режиме классификации:
`./prog classify --stats <stat file> --input <in file> --output <out file>`

Ключ	Значение
--stats	файл со статистикой полученной на предыдущем этапе
--input	входной файл с вопросами
--output	выходной файл с тегами к вопросам

Формат входных файлов при обучении:
```
<Количество строк в вопросе [n]>
<Тег 1>,<Тег 2>,...,<Тег m>
<Заголовок вопроса>
<Текст вопроса [n строк]>
```

Формат входных файлов при запросах:
```
<Количество строк в вопросе [n]>
<Заголовок вопроса>
<Текст вопроса [n строк]>
```

Формат выходного файла:
Для каждого запроса в отдельной строке выводится предполагаемый набор тегов, через запятую.

1 Описание алгоритма

Наивный байесовский классификатор – простой вероятностный классификатор, основанный на применении теоремы Байеса со строгими (наивными) предположениями о независимости.

Теорема Байеса:

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

, где
$P(c|d)$ — вероятность, что документ d принадлежит классу c, именно её нам надо рассчитать;
$P(d|c)$ — 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.

Цель классификации состоит в том, чтобы понять, к каким классам больше подходит документ, поэтому нам нужны не сами вероятности, а набор наиболее вероятных классов. Байесовский классификатор использует оценку апостериорного максимума *(MAP)* для определения наиболее вероятного класса. Грубо говоря, это класс с максимальной вероятностью.

$$c_{MAP} = \arg\max_{c \in C} \frac{P(d|c)P(c)}{P(d)}$$

Поскольку знаменатель (вероятность документа) является константой и никак не может повлиять на ранжирование классов, в нашей задаче мы можем его игнорировать. То есть нам надо рассчитать вероятность для каждого класса и выбрать тот, который обладает максимальной вероятностью.

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

Предположение условной независимости. Байесовский классификатор представляет документ как набор слов, вероятности которых условно не зависят друг от друга. Исходя из этого предположения условная вероятность документа аппроксимируется произведением условных вероятностей всех слов входящих в документ.

$$P(d|c) \approx P(w_1|c)P(w_2|c)...P(w_n|c) = \prod_{i=1}^{n} P(w_i|c)$$

Подставив эту формулу в предыдущую, получим:

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

Проблема арифметического переполнения. При достаточно большой длине документа количество различных слов в нём становится очень большим, и, соответственно, условные вероятности многих слов становятся очень маленькими числами, перемножение которых для вычисления условной вероятности документа приводит к арифметическомму переполнению снизу базовых вещественных типов данных. Во избежание этого логарифмируем формулу, тем самым изменив только её численное значение, но не параметры, при которых достигается максимум, так как логарифм функция монотонная.

$$c_{MAP} = \arg\max_{c \in C} \left[\log P(c) + \sum_{i=1}^{n} \log P(w_i|c) \right]$$

Оценка параметров

Оценка вероятности класса:

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

, где D_c – количество документов принадлежащих классу c,
D – общее количество документов в обучающей выборке.

Оценка вероятности слова в классе:

Здесь будет приведена полиномиальная модель наивного байесовского классификатора:

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

, где W_{ic} — количество раз, сколько i-ое слово встречается в документах класса c;
V — словарь корпуса документов (список всех уникальных слов).

Сглаживание Лапласа. Если на этапе классификации встретится слово, которое не встречалось на этапе обучения, то значения W_{ic}, а следственно и $P(w_i|c)$, будут равны нулю. Это приведет к тому, что документ с этим словом нельзя будет классифицировать, так как он будет иметь нулевую вероятность по всем классам. Идея сглаживания Лапласа заключается в том, что берётся расчёт, будто встречали каждое уникальное слово на этапе обучения на определённое количество l раз больше.

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

Итог. Подставив выбранные формулы оценки в главную, получим:

$$c_{MAP} = \arg\max_{c \in C} \left[\log \frac{D_c}{D} + \sum_{i=1}^{n} \log \frac{W_{ic} + l}{l\,|V| + \sum_{i' \in V} W_{i'c}} \right]$$

Реализация классификатора. На этапе классификации необходимо для каждого класса рассчитать значение следующего выражения и выбрать класс с максимальным значением. (В случае нашего задания, выбираются по несколько классов для заголовка и текста вопроса и удаляются повторения.)

$$\log D_c - \log D + \sum_{i \in Q} \left(\log \left(W_{ic} + l \right) - \log \left(l\,|V| + L_c \right) \right)$$

, где D_c — количество документов в обучающей выборке принадлежащих классу c;
D — общее количество документов в обучающей выборке;
$|V|$ — количество уникальных слов во всех документах обучающей выборки;
L_c — суммарное количество слов в документах класса c в обучающей выборке;
W_{ic} — сколько раз i-ое слово встречалось в документах класса c в обучающей выборке;
Q – множество слов классифицируемого документа (включая повторы).

Сложность алгоритма

Функция обучения (высчеты статистики): $O(d * m * n * l)$,
где d - количество документов,
m - количество тегов,
n - количество слов в документе,
l - длина слова.

Функция классификации: $O(f * d * m * n)$,
где f - количество фич.

2 Описание программы

Файлы

Файл	Описание
main.cpp	Главный файл с алгоритмом согласно заданию
Data.hpp	Объявление используемых структур данных и функций обработки информации и сохранения данных
Data.cpp	Реализация функций обработки информации с потока ввода и сохранения данных в поток вывода
NBC.hpp	Библиотека Наивного байесовского классификатора
NBC.cpp	Реализация алгоритмов обучения и классификации модели
train.txt	Корпус для обучения модели
stats.txt	Промежуточные вычесленные статистических данных для классификации на основе обучающей выборки
test.txt	Корпус для классификации данных обученной моделью
predict.txt	Результат классификации тестового корпуса

Функции и методы

Функция	Описание
const Dataset get_trainset(std::ifstream&)	Распарсить поток ввода на датасет обучающей выборки
const Dataset get_testset(std::ifstream&)	Распарсить поток ввода на датасет тестовой выборки
void save_stats(std::ofstream&, const Stats&)	Вывести собранную статистику в поток вывода
const Stats load_stats(std::ifstream&)	Загрузить статистику из потока ввода
void save_feature(std::ofstream&, const Feature&)	Вывести фичу в поток вывода
const std::string clear_text(const std::string&)	Очистить текст от лишних символов
void fill_tags(std::ifstream&, Feature&)	Распарсить теги из потока ввода в фичу
void fill_title(std::ifstream&, Feature&)	Распарсить заголовок из потока ввода в фичу
void fill_text(std::ifstream&, Feature&, int)	Распарсить текст из потока ввода в фичу
NBC::NBC(const Stats&&)	Построить модель из статистики
const Stats& NBC::get_stats() const	Получить статистику модели
void NBC::fit(const Feature&, const Feature&)	Обучить модель на множестве признаков и целевой фиче
template <size_t SZ> const Feature predict(const std::array<std::reference_wrapper<const Feature>, SZ>&) const	Шаблонный метод для классификации данных на основе определённого количества признаков

3 Консоль

```
[artem@IdeaPad Naive Bayes(KP1.1)]$ make
g++ -std=c++17 -c main.cpp
g++ -std=c++17 -c Data.cpp
g++ -std=c++17 -c NBC.cpp
g++ -std=c++17 main.o Data.o NBC.o -o NBC
[artem@IdeaPad Naive Bayes(KP1.1)]$ ls
Data.cpp  Data.hpp  Data.o  main.cpp  main.o  Makefile  NBC  NBC.cpp
NBC.hpp  NBC.o  predict.txt  stats.txt  test.txt  train.txt
[artem@IdeaPad Naive Bayes(KP1.1)]$ ./NBC learn --input train.txt --output
stats.txt
[artem@IdeaPad Naive Bayes(KP1.1)]$ ./NBC classify --stats stats.txt
--input test.txt --output predict.txt
[artem@IdeaPad Naive Bayes(KP1.1)]$ cat predict.txt
geography, regions, food
tools, software, philosophy, politics
philosophy, science, geography
science, geography, philosophy
sport, history
regions, geography, food, cooking
philosophy, history
```

4 Выводы

Была разработана система классификации документов, которая вычисляет относительный порядок подходящих классов к предоставленному документу и выбирает наиболее вероятные из них. В частности, данная модель полностью выполняет поставленную задачу подбора подходящих вариантов тегов к новым вопросам на основе базы вопросов и тегов к ним. Модель была протестирована на статьях с Wikipedia, и, хотя в расчёт не берётся зависимость слов от контекста, результаты оценки оказались вполне адекватными.

Список литературы

[1] Scikit-learn: *Machine Learning in Python,* Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[2] Блог *разработчика программного обеспечения.*
URL: `http://bazhenov.me/blog/2012/06/11/naive-bayes.html` .