Курсовой проект по курсу дискретного анализа: Классификация документов

Выполнил студент группы М8О-312Б-22 МАИ *Юрков Евгений*.

Условие

Необходимо реализовать наивный байесовский классификатор, который будет обучен на первой части входных данных и классифицировать им вторую часть.

Метод решения

Рассмотрим классификатор, который сопоставляет один класс каждому документу.

Классификатор использует формулу Байеса для предсказаний:

$$P(c|d) \sim P(c) \prod_{1 \le k \le n_d} P(t_k|c),$$

где

- $P(c|d)$ - вероятность того, что документ d пренадлежит классу c,
- $P(c)$ - вероятность появления документа класса c,
- $P(t_k|c)$ - вероятность того, что слово t_k появится в документе класса c.

Класс выбирается по формуле:

$$c_{map} = \arg\max_{c \in C} \widehat{P}(c) \prod_{1 \le k \le n_d} \widehat{P}(t_k|c).$$

При перемножении вероятностей может получиться очень маленькое число, при котором тип `double` может переполниться. Чтобы вычисления были более точными логарифмируем формулу:

$$c_{map} = \arg\max_{c \in C} [\log \widehat{P}(c) + \sum_{1 \le k \le n_d} \log \widehat{P}(t_k|c)].$$

Оценка вероятности класса равна отношению количества документов класса к общему количеству документов:

$$\widehat{P}(c) = \frac{N_c}{N}$$

Оценка вероятности слова в классе рассчитывается по формуле:

$$\widehat{P}(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}},$$

где

- T_{ct} - количество появлений слова t в документах класса c,
- $\sum_{t' \in V} T_{ct'}$ - сумма появлений всех слов в документах класса c,

Эта формула плохо справляется со словами, которые появляются только в своём классе. В таком случае для другого класса резульлтат был бы равен нулю, тогда произведение в формуле Байеса стало бы так же равно нулю, хотя вероятности для других слов могли быть достаточно большими. Нужно чтобы вероятность таких слов была мала, но больше нуля. Для этого формула преобразуется следующим образом:

$$\widehat{P}(t|c) = \frac{T_{ct} + 1}{\sum_{t' \in V}(T_{ct'} + 1)} = \frac{T_{ct} + 1}{(\sum_{t' \in V} T_{ct'}) + B'},$$

где B' - общее количество слов в словаре (без повторений).

Чтобы предсказывать сразу несколько тегов для документа, вычисляется среднее значение вероятности каждого из теги и выбираются те, чья вероятность больше, чем средняя.

Описание программы

Для предсказаний используется класс `NaiveBayes`, в котором реализованы функции:

- `void learn(const std::vector<std::string>& document, const std::vector<int>& classes)` - обучение модели на массиве предложений `document`, в котором каждое предложение соответствует своему классу.
- `std::vector<int> predict(const std::vector<std::string>& document) const` - осуществляет предсказание типа для каждого предложения из массива `document`.

Для обработки аргументов программы использовалась библиотека boost/program_options.

Дневник отладки

1. Написан класс для предсказания двух классов: 0 или 1.
2. Исправлено вычисление вероятности для каждого слова.
3. Класс принимает заранее указанное число классов n.

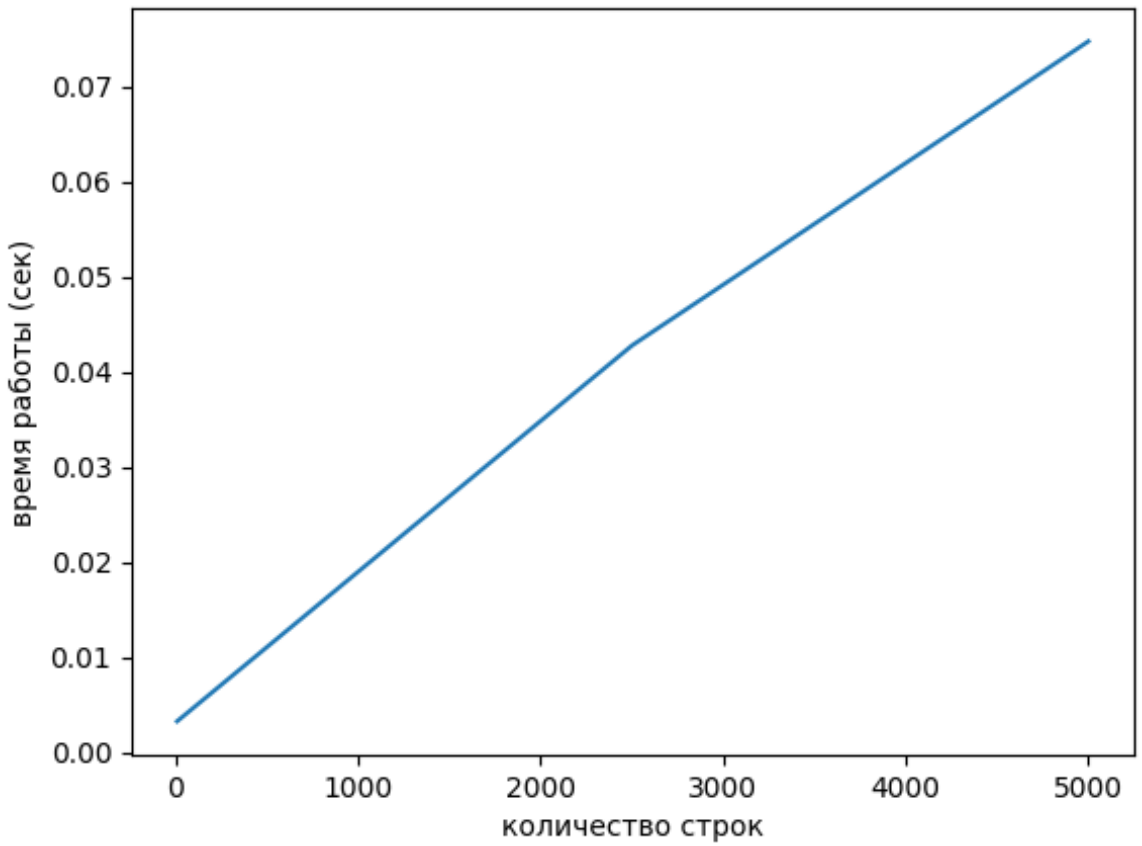

Рис. 1: График зависимости времени работы программы от количества запросов в файле

Тест производительности

Алгоритм обучения использует разбиение предложения на массив из слов, подсчёт количества слов в предложении, а потом для каждого класса заполняет `unordered_map` вероятностей слов, то есть сложность обучения $O(n \cdot m \cdot c)$, где n - количество документов, поданных для обучения, m - длина документа, c - количество предсказываемых классов.

Алгоритм предсказания рассчитывает формулу для каждого из классов и выбирает максимум, то есть сложность предсказания тоже $O(n \cdot m \cdot c)$.

Точность

Для оценки точности модели использовалась выборка данных из 5572 элементов, в которой каждому документу сопоставлялся тег спам/не спам. В этой выборке предложения со спамом составляли 13%.

Precision: 0.9726027397260274 **Recall**: 0.9594594594594594

Выводы

В ходе курсового проекта я реализовал программу для классификации документов с использованием наивного Байесовского классификатора. Я узнал теоретические основы его работы и укрепил навыки работы с текстовыми данными.

В ходе выполнения проекта сложности возникали при правильной организации структуры данных для классификации. Также была важной задача корректной работы с вероятностями, особенно в случае редких слов в тексте.

Полученные навыки полезны для дальнейшей работы с текстовыми данными и их обработкой, включая задачи анализа данных, создания систем рекомендаций или автоматического распределения текстов по категориям, например, для определения спама.