Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт № 8 информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Курсовой проект по курсу**

**«Дискретный анализ»**

**Классификация документов**

Студент: Пермяков Н. А.

Группа: М80 – 208Б-19

Преподаватель: *Сорокин C.*

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

**Содержание**

1. Постановка задачи
2. Метод и алгоритм решения
3. Описание программы
4. Дневник отладки
5. Вывод

**Постановка задачи**

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 <stats file> \

--input <input file> \

--output <output 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.

**Метод и алгоритм решения**

Наивный байесовский классификатор - основан на предположении о независимости функций. Модель обучения использует теорему Байеса с предположениями о полной независимости всех особенностей наблюдаемых случайных величин.

где c - класс, e - вектор наблюдаемых характеристик. Затем это представляет нам

с простым правилом классификации апостериорного максимума (MAP):

C = \argmax\_{C '} P (C') \* \prod\_{i = 1}^{n} P (e\_i | C ')

Здесь классификатор обучается назначая категории заданному текстовому набору. Он разработан для обучения в режиме реального времени, то есть новые типы текстов/категорий могут добавляться на лету. Модель набора слов используется для оценки (частотной) вероятности отдельных признаков. Может встретиться слово, которого не было на этапе обучения системы. Это может привести к тому, что оценка будет равна нулю и документ нельзя будет отнести ни в одну из категорий. Чтобы этого избежать необходимо применить сглаживание - поправки во все вероятности вхождения слов в документ. Выбирается параметр α = 1 - сглаживание Лапласа.

В генерации нового документа заключается разница между мультиномиальный (multinomial) и многомерный (multivariate) Наивными Байесовскими классификаторами.

В многомерной модели документ - это вектор бинарных атрибутов, показывающих, встретилось ли в документе то или иное слово. Когда мы подсчитываем правдоподобие документа, мы перемножаем вероятности того, что встретилось каждое слово из документа и вероятности того, что не встретилось слово. Наивное предположение в том, что события «встретилось слово» предполагаются независимыми. Для применения требуется зафиксировать словарь, а количество повторений каждого слова теряется.

В мультиномиальной модели документ - это последовательность событий. Каждое событие - это случайный выбор одного слова. Когда мы подсчитываем правдоподобие документа, мы перемножаем вероятности того, что мы достали из мешка те самые слова, которые встретились в документе. Наивное предположение в том, что мы достаем из мешка разные слова независимо друг от друга. Получается модель, которая учитывает количество повторений каждого слова, но не учитывает, каких слов нет в документе. Этот вариант реализован в работе.

**Описание программы**

**main.cpp -** файл с консольным интерфейсом согласно условиям задачи

**base.cpp -** файл с общими функциями, выполняющие парсинг одних и тех же частей со входных файлах

**classify.cpp -** файл с реализацией чтения из файла промежуточных данных после обучения и алгоритмом классификации

**learn.cpp -** файл с реализацией чтения входных данных и процессом обучения модели

**result.txt -** файл с результатами классификации

**stat.txt -** файл с результатами промежуточных вычислений

**test.txt -** файл с тестовыми данными

**train.txt -** файл с данными для обучения

get\_num\_line - функция считывания количества строк

get\_title - функция считывания заголовка

get\_text - функция считывания текста вопроса

total\_training\_learn - количество итераций обучения

classes\_learn - структура для каждого тега

set\_empty\_input\_learn - функция очистки начальных значений считывающих переменных

get\_tags - функция считывания тегов

prepare\_learn - функция подготовительной компоновки данных

start\_learn - функция запуска обучения отдельно заголовка и текста

learn - функция обучения

write\_statistic - - функция записи промежуточных подсчетов

total\_training\_classify - количество итераций классифицирования

classes\_classify - структура для каждого тега

RESULT - результат вычислений

read\_statistic - считывание промежуточного файла

get\_token - функция считывания токенов из строки с определенной структурой

get\_tokens\_words - функция считывания токенов из строки с разделителем

set\_empty\_input\_classify - функция очистки начальных значений считывающих переменных

prepare\_classify - функция подготовки данных к классификации

start\_classify - функция начала классифицирования отдельно заголовка и вопроса

classify - функция классифицирования

write\_result - функция записи результата вычислений в файл

// разделение строки статистики на токены

if ((pos = line.find(s)) != std::string::npos) {

token = line.substr(0, pos);

line.erase(0, pos + 1);

}

// считаем сглаживание Лапласа

for (auto& item\_word\_count : word\_counts) {

if (!cls\_stat.words.count(item\_word\_count.first)) {

++laplace\_smoothing;

}

}

// считаем метрику для каждого класса

for (auto& item\_word\_count : word\_counts) {

word = item\_word\_count.first;

count = item\_word\_count.second;

if (cls\_stat.words.count(word)) {

metric\_class += count \* (log(cls\_stat.words[word]) - log(laplace\_smoothing));

}

else {

metric\_class += count \* (-log(laplace\_smoothing));

}

}

// считаем статистические данные по каждому слову

if (classes\_learn.count(cls)) {

cls\_stat = classes\_learn[cls];

}

else {

cls\_stat.count\_outdoor = 0;

cls\_stat.words.clear();

}

++cls\_stat.count\_outdoor;

std::stringstream ss(text);

std::string token;

while (ss >> token) {

++cls\_stat.words[token];

++cls\_stat.total\_words;

}

**Вывод**

В результате работы была реализована система, которая на основе базы вопросов и тегов к ним, предлагает варианты тегов, которые подходят к новым вопросам. Соблюден формат входных и выходных данных. Использовал алгоритм Наивного Байесовского классификатора с предположениями о полной независимости всех особенностей наблюдаемых случайных величин.

Была изучена разница между мультиномиальный (multinomial) и многомерный (multivariate) Наивными Байесовскими классификаторами.

Рассчитаем точность в пределах класса (доля документов действительно принадлежащих данному классу относительно всех документов которые система отнесла к этому классу), и полноту системы (долю найденных классификатором документов принадлежащих классу относительно всех документов этого класса в тестовой выборке).

TP — истино-положительное решение;

TN — истино-отрицательное решение;

FP — ложно-положительное решение;

FN — ложно-отрицательное решение;

TP = 4

FP = 2

FN = 1

precision = 0.(6)

recall = 0.8