КЛАССИФИКАЦИЯ ТЕКСТОВЫХ ФАЙЛОВ НА ПЛАТФОРМЕ ANDROID

А. С. Гусейнова, В. О. Ефимик
Кафедра многопроцессорных систем и сетей, Белорусский государственный университет
Минск, Республика Беларусь
E-mail: efimikvitaliy@gmail.com, huseinavaas@bsu.by

В данной работе рассматривается построение классификатора текстов на платформе Android с помощью метода опорных векторов. Так же рассматриваются проблемы, возникающие при реализации такого классификатора, и способы их преодоления.

Введение

Использование мобильных устройств – актуальная тема на сегодняшний день. Она находит отражение во многих сферах человеческой деятельности от коммуникаций и общения до развлечений и информационного обеспечения. На сегодняшний день количество памяти мобильных устройств достигает десятки гигабайт. Таким образом, появляется возможность содержать на своих мобильных устройствах большое количество различных файлов. В том числе, в процессе пользования своим мобильным устройством конечный пользователь накапливает значительное количество текстовых файлов. Например, таких файлов как книги, документы или просто интересные статьи. Но при большом объеме файлов и папок довольно сложно выбрать нужную и необходимую информацию или же проверить наличие некоторой информации на своем устройстве. Таким образом, возникает проблема классификации текстовых документов на мобильных устройствах. Для решения задачи классификации эффективным подходом является метод опорных векторов.

I. Метод опорных векторов

Метод опорных векторов (SVM, support vector machine) – набор схожих алгоритмов обучения с учителем, использующихся для задач классификации и регрессионного анализа. Принадлежит к семейству линейных классификаторов [1].

Основная идея метода опорных векторов – перевод исходных векторов в пространство более высокой размерности и построение разделяющей гиперплоскости. Особым свойством метода опорных векторов является непрерывное уменьшение эмпирической ошибки классификации и увеличение зазора между гиперплоскостью и граничными элементами обучающей выборки. Разделяющей гиперплоскостью будет гиперплоскость с максимальным зазором.

II. Представление текста в виде вектора

Документы с текстом, которые обычно представляют собой строки символов, должны быть преобразованы в подходящую для классификатора форму. Перед запуском основного алгоритма производится стемминг файла и удаляются стоп-слова.Слова в тексте выступают в качестве представительских единиц, а их упорядочение в документе имеет второстепенное значение [2]. Для точности вычислений используется не само слово, а его основа, т.к. у слова могут быть разные лексические формы. Для выделения основы слова (стемминга) применен алгоритма Портера. Таким образом, текст можно представить в форме вектора, компоненты которого будут содержать структуру “ключ-значение”. В настоящей реализации в качестве ключа выступает само слово, а в качестве значения – количество раз, которое слово встретилось в данном документе.

III. Реализация классификатора

В качестве обучающей выборки послужила текстовая база RCV1-v2/LYRL2004, которая в себе содержит коллекцию классифицированных текстовых документов. В данной базе определено 103 категории типа текста. Для обучения (построения разделяющих плоскостей для каждого типа текста) было использовано 150 000 текстов из данной текстовой базы. Темы в текстовой базе представлены в виде дерева, т.е. у некоторых тем есть дочерние темы. Дочерние темы имеют более узкую специализацию, нежели родительские. Поэтому, если текст не принадлежит некоторой теме, то все дочерние темы не рассматривались.

В силу того, что аппаратное обеспечение мобильных устройств по производительности уступает обеспечению ЭВМ, обучение проводилось на ЭВМ, а результат работы записывался в файл и сохранялся в памяти мобильного устройства. Таким образом, мобильное устройство, при работе программы, может считать из файла уже готовую разделяющую плоскость и не обучаться каждый раз при новом запуске приложения. Данный способ значительно уменьшил расход оперативной памяти и увеличил скорость работы приложения.

Классификатор в приложении реализуется параллельно, для реализации была выбрана “модель с равноправными узлами”. Каждый по-

ток может взять себе некоторое “задание” из общей очереди и положить в очередь новые “задания”. “Задание” состоит в проверке определенного множества текстов на принадлежность конкретной теме.

IV. Реализованное приложение

Приложение было реализовано на платформу Android. Для реализации приложения были использованы языки Java и C++. Для построения разделяющей плоскости между векторами была применена библиотека SVM-Light, реализованная и поддерживаемая Техническим Дортмундским Университетом.

Реализованное приложение является классификатором текстов и предназначено для классификации текстовой информации на мобильных устройствах с операционной системой Android. Реализованное приложение может быть использовано пользователем в целях структуризации информации и оперативной очистки памяти на своем мобильном устройстве. Приложение предоставляет пользователю следующие возможности:

1. выбор файлов для классификации;
2. классификация файлов и построение по ним виртуального файлового менеджера;
3. сохранение модели виртуального файлового менеджера для будущего использование;
4. загрузка модели виртуального файлового менеджера из памяти устройства.

Виртуальный файловый менеджер имеет древовидную структуру, в которой текста распределены по соответствующим темам, и предоставляет следующие возможности:

1. просмотр виртуальной файловой структуры;
2. просмотр текстового файла с помощью сторонних программ;
3. удаление текстового файла из виртуального файлового менеджера и памяти устройства.

V. Анализ и тестирование программы

Было произведено сравнение времени работы параллельного алгоритма с 4 потоками на Android OS и такого же алгоритма с 4 потоками на ЭВМ. Результаты приведены в таблице 1 и таблице 2.

Заключение

В данной работе реализован классификатор текстов для мобильного устройства с OS Android, реализован классификатор текстов на ЭВМ. Исследована скорость классификации текстов на OS Android. Произведено сравнение по скорости мобильного устройства и ЭВМ. Как показали вычислительные эксперименты, параллельный алгоритм на ЭВМ работает примерно в 10 раз быстрее, чем на мобильном устройстве.

1. CS299 Lecture notes // Standford University [Electronic resource] - Mode of access : http://cs229.stanford.edu/notes/cs229-notes3.pdf. – Date of access : 19.03.2015.
2. Йохимс Т. Классификация текста с использованием SVM / Дортмундский Университет, 1997. - 18 с.

Таблица 1 – Время работы на мобильном устройстве - 4 потока, 4 ядра частотой 1,83 ГГц

Положение текстов в дереве	Общее время, мс	Время чтения модели, мс Максимум	Время чтения модели, мс Минимум	Время чтения модели, мс Среднее	Время классификации, мс Максимум	Время классификации, мс Минимум	Время классификации, мс Среднее
Одна ветвь	444115	205951	3193	56665	2	близко к 0	0.04
Один текст	447246	238525	3465	72811	1	близко к 0	0.075
Две ветви	523008	257660	3465	76603	2	близко к 0	0.12

Таблица 2 – Время работы на ЭВМ - 4 потока, 4 ядра частотой 2,40 ГГц

Положение текстов в дереве	Общее время, мс	Время чтения модели, мс Максимум	Время чтения модели, мс Минимум	Время чтения модели, мс Среднее	Время классификации, мс Максимум	Время классификации, мс Минимум	Время классификации, мс Среднее
Одна ветвь	40308	22435	270	6000.5	близко к 0	близко к 0	близко к 0
Один текст	40417	23394	219	6052	близко к 0	близко к 0	близко к 0
Две ветви	59357	23862	530	8380	близко к 0	близко к 0	близко к 0