Министерство образования и науки Российской Федерации
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет информатики
Кафедра прикладной информатики

УДК 519.237.8

ДОПУСТИТЬ К ЗАЩИТЕ В ГАК
Зав. кафедрой, проф., д.т.н.
_____ С.П. Сущенко
«___» _____ 2011 г.

Запрягаева Елена Александровна

КЛАССИФИКАЦИЯ ТЕКСТОВ ПО ЗАДАННЫМ ПРИЗНАКАМ СТИЛЕЙ НА ОСНОВЕ ДЕРЕВЬЕВ РЕШЕНИЙ

Выпускная квалификационная работа на соискание степени бакалавра

Научный руководитель,
профессор, д.т.н. В.В. Поддубный

Исполнитель,
студ. гр. 1471 Е.А. Запрягаева

Электронная версия выпускной квалификационной работы помещена
в электронную библиотеку. Файл
Администратор

Томск – 2011

Реферат

Выпускная квалификационная работа на соискание степени бакалавра 30 с., 6 рис., 1 таблица, 10 источников, 2 приложения

КЛАССИФИКАЦИЯ, ДЕРЕВЬЯ РЕШЕНИЙ, ТЕКСТЫ, СРАВНЕНИЕ АЛГОРИТМОВ, C#

Объект исследования – деревья решений.

Цель работы – изучение алгоритмов построения деревьев решений применительно к текстам и их сравнительный анализ.

Методы исследования – экспериментальный (на ЭВМ).

Основные результаты – изучены алгоритмы построения деревьев решений на основе частотных и номинальных признаков. Проведен сравнительный анализ результатов работы алгоритмов. Разработан программный продукт, позволяющий производить классификацию текстов на построенных деревьях решений.

Оглавление

Введение

В современном мире, с ростом количества текстов, представленных в цифровом формате, все более остро встает проблема определения авторства того или иного текста. Так же с давних пор существуют сомнения в авторстве некоторых произведений.

Данная проблема может быть решена сравнением стилей написания произведений различных авторов.

Сравнение стилей текстов на схожесть проводили Менденхолл Т., Морозов Н.А., Марков А.А., Мортон А., Фоменко Т.Г. и Фоменко В.П. [1], Хмелев Д.В.,. и др.

Деревья решений как средство классификации используются уже давно. Впервые определений деревьев решений дано в работе C. I. Hovland and E. B. Hunt «Programming a Model of Human Concept Formation», опубликованной в 1961 году. Но для решения проблем авторства того или иного текста деревья решений стали применять относительно недавно.

В частности, классификацию текстов по автору на основе частотных признаков с помощью деревьев решений проводили Шевелев О.Г. и Петраков А.В. [2], Романов А.С. [3] и др.

Однако в представленных ранее работах рассматривались только частотные признаки, то есть основанные на частотах встречаемости анализируемых слов в тексте. Хотелось бы так же рассмотреть, как повлияет на качество классификации использование при построении дерева решений номинальных признаков, то есть перечисляемых, значение которых известно заранее. А так же сравнить популярные алгоритмы построения деревьев решений и качество классификации на построенных с их помощью деревьях.

Таким образом, целью данной работы является сравнительный анализ алгоритмов классификации текстов на основе деревьев решений.

Можно выделить следующие задачи:

1. Изучение алгоритмов классификации на основе деревьев решений;
2. Программная реализация алгоритмов на языке C# в среде Microsoft Visual Studio .NET;
3. Тестирование алгоритмов на материале электронных библиотек текстов;
4. Сравнение эффективности работы алгоритмов в задачах классификации текстов;
5. Выработка рекомендаций по практическому применению алгоритмов.

1 Деревья решений

Дерево решений (Рис. 1) – это древовидный граф, состоящий из узлов и листьев, соединенных между собой дугами. В узлах графа происходит принятие решений, а листья указывают на классы. Граф дерева решений должен быть ациклический, иначе он перестает быть древовидным и его использование для принятия решений вызовет большие трудности. [4]

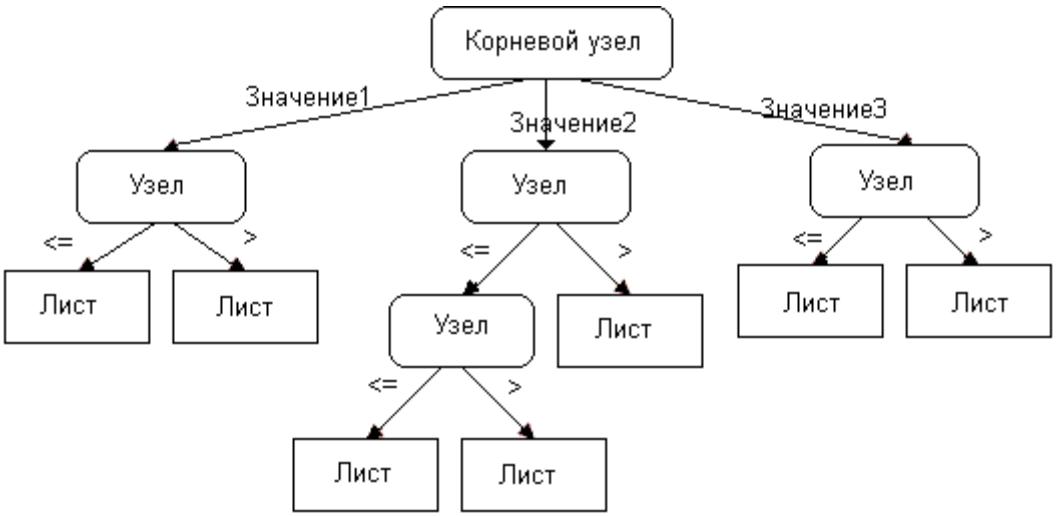


Рисунок 1 - Дерево решений с произвольным количеством потомков

Первое упоминание деревьев решений было в работе C. I. Hovland and E. B. Hunt «Programming a Model of Human Concept Formation», опубликованной в 1961 году. Но полное определение было дано И. Хантом в работе «Эксперименты в индукции» ("Experiments in Induction") в 1966 году.

Деревья решений можно разделить на два типа: деревья классификации и деревья регрессии. Если целевой признак имеет дискретные значения, то получено дерево классификации. Если же целевой признак принимает непрерывные значение, то подучено дерево регрессии. [5]

Преимущества использования деревьев решений. [4], [5]

1) Простота восприятия. Результат построения дерева решений легко интерпретируется пользователем. Дерево решений наглядно поясняет, почему конкретный объект отнесен к тому или иному классу.
2) Алгоритм построения дерева решений не требует выбора входных атрибутов. Для построения используются все атрибуты, и алгоритм сам выбирает наиболее значимые и строит на их основе дерево решений.
3) Быстрота обучения.

4) Для построения дерева требуется малый объем информации, поэтому они занимают мало места в памяти.
5) Относительная гибкость. Деревья решений позволяют работать с непрерывными и символьными целевыми признаками. Во многих алгоритмах построения деревьев решений имеется возможность обработки пропущенных значений. Это позволяет применять деревья решения в самых разных задачах.

Стоит отметить и недостатки деревьев решений.[5]

1) Сложность обучения дерева в оперативном режиме.
2) Проблема усечения больших деревьев.

2 Алгоритмы построения деревьев решений

В данном разделе будут рассмотрены три популярных алгоритма построения деревьев решений ID3, C4.5 и CART. Прежде всего, необходимо дать основные определения и обозначения, который будут использоваться далее.

2.1 Основные определения и обозначения.

Пусть T есть множество текстов, а их количество – есть мощность данного множества |T|. Так же имеется множество классов

$$C = \{C_1, C_2, \ldots, C_k\} \quad (1)$$

и множество признаков

$$A = \{A_1, A_2, \ldots, A_m\} \quad (2)$$

Далее будут использоваться следующие определения.

Числовой признак – это признак, значениями которого могут быть только числа.

Номинальный признак – это признак, значения которого являются перечисляемыми и известны заранее.

Следует отметить, что значениями номинального признака могут быть как числа, так и строки.

Частотный признак – вычисляемый признак, значениями которого являются частоты встречаемости слов в тексте.

Целевой признак – признак, по которому проводится классификация.

Обучающая выборка – набор текстов, для которых известно значение целевого признака. С помощью этого набора текстов происходит обучение дерева.

Далее, следует определить формат ввода обучающей выборки. Данные должны быть представлены в виде таблицы (Таблица 1)

Таблица 1 - Структура входных данных алгоритмов.

	$Признак_1$	$Признак_2$	…	$Признак_m$	Целевой признак
$Текст_1$	$Значение_{11}$	$Значение_{12}$	…	$Значение_{1m}$	$Метка\ класса_1$
$Текст_2$	$Значение_{21}$	$Значение_{22}$	…	$Значение_{2m}$	$Метка\ класса_2$
…	…	…	…	…	…
$Текст_n$	$Значение_{n1}$	$Значение_{n2}$	…	$Значение_{nm}$	$Метка\ класса_k$

В таблице данные представлены следующим образом. В строках – названия текстов, в столбцах – признаки, по которым происходит разбиение, причем, их должно быть дискретное число. Последний столбец – целевой признак, содержащий метку класса, к которому принадлежит тот или иной текст.

2.2 Алгоритм ID3

Данный алгоритм впервые был представлен Дж. Р. Куинланом (J. Ross Quinlan) в книге «Машинное обучение» [6] («Machine Learning»), опубликованной в 1992 году.

Алгоритм ID3 работает рекурсивно, разбивая по выбранному признаку в каждом узле множество текстов на подмножества, начиная с корня дерева, в котором содержаться все тексты.

Далее, следует рассмотреть математический аппарат алгоритма.

По каждому признаку A_i можно разбить множество T на подмножества T_1, T_2, …, T_n.

Рассмотрим, каким образом выбирается разбиение. [7]

Пусть $f(C_j, T)$ - количество текстов из некоторого множества T, принадлежащих одному классу C_j. Тогда вероятность того, что случайно выбранный текст из множества T принадлежит классу C_j

$$P_j = \frac{f(C_j, T)}{|T|} \tag{3}$$

Тогда энтропия множества T имеет вид

$$H(T) = -\sum_{j=1}^{k} P_j \log_2(P_j) \tag{4}$$

Условная энтропия множества текстов T при рассматриваемом признаке X есть

$$H(T \mid X) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} H(T_i) \tag{5}$$

Далее для каждого из признаков вычисляется количество информации

$$I(X) = H(T) - H(T \mid X) \tag{6}$$

Существует 2 варианта разбиения.

Если A_i – номинальный строковый признак, то количество подмножеств T будет равно количеству значений признака A_i.

Если A_i – числовой признак, то множество T разбивается на два подмножества. При этом необходимо выбрать некий порог разбиения, с которым будут сравниваться все значения признака. Обозначим множество значений признака A_i

$$v = \{v_1, v_2, \ldots, v_n\} \tag{7}$$

Сначала следует отсортировать значения. Тогда любое значение, лежащее между v_i и v_{i+1}, делит все примеры на два множества и в качестве порога можно выбрать среднее между значениями v_i и v_{i+1}.

$$t_i = \frac{v_i + v_{i+1}}{2} \tag{8}$$

Таким образом, имеется n-1 потенциальное пороговое значение.

Так как для номинального признака существует один вариант разбиения, а для числового признака количество вариантов разбиения равно количеству возможных порогов, т.е. n - 1. Таким образом, если имеется u номинальных признаков и v числовых, то в каждой вершине разбиение можно провести O способами, где

$$O = u + v(n - 1) \tag{9}$$

Формулы (3) – (6) применяются ко всем возможным разбиениям, и в качестве рабочего выбирается наиболее информативное разбиение.

2.3 Алгоритм C4.5

Так как алгоритм C4.5 [7] является усовершенствованием ID3, то формулы (3), (4), (5), (6) верны и для этого алгоритма. Отличие появляется в критерии разбиения множества на подмножества.

Критерий разбиения (6) имеет один недостаток – он "предпочитает" признаки, которые имеют много значений, так как при разбиении по такому признаку получаются подмножества, содержащие минимальное число текстов. Проблема решается введением некоторой нормализации. По аналогии с формулой (7) энтропия разбиений есть

$$splitH(X) = -\sum_{i=1}^{n} \frac{|T_i|}{|T|} \log_2 \left(\frac{|T_i|}{|T|}\right), \tag{10}$$

где $X = \{T_1, T_2, \ldots, T_n\}$ и $\sum_{i=1}^{n} T_i = T$

Выражение (10) оценивает потенциальную информацию, полученную при разбиении множества T на n подмножеств.

Тогда новым критерием разбиения будет

$$newI(X) = \frac{I(X)}{splitH(X)} \tag{11}$$

2.4 Алгоритм CART

Алгоритм CART [8] (Classification and Regression Tree), как видно из названия, решает задачи классификации и регрессии. Он разработан в 1974-1984 годах профессорами Leo Breiman, Jerry Friedman, Charles Stone и Richard Olshen.

Алгоритм строит бинарные деревья, имеющие двух потомков в каждом узле дерева. На каждом шаге построения дерева правило, формируемое в узле, делит заданную обучающую выборку на две части – часть, в которой выполняется правило (левый потомок) и часть, в которой правило не выполняется (правый потомок). Для выбора оптимального правила используется функция оценки качества разбиения.

Функция оценки качества разбиения основана на идее уменьшения неопределенности в узле. Например, выборка состоит из 100 текстов, наибольшая неопределенность возникает при разбиении выборки на 2 потомка по 50 текстов, а наименьшая – при разбиении на 100 и 0 текстов.

Если набор текстов T содержит данные k классов, тогда индекс *Gini* определяется как:

$$Gini(T) = 1 - \sum_{j=1}^{k} p_j^2, \qquad (12)$$

где p_j вычисляется по формуле (3).

При разбиении множества текстов на два подмножества показатель качества разбиения будет равен

$$Gini_{split}(T_1, T_2) = \frac{|T_1|}{|T|} Gini(T_1) + \frac{|T_2|}{|T|} Gini(T_2) \qquad (13)$$

Наилучшим считается разбиение, для которого индекс $Gini_{split}$

Обозначив количество текстов i-го класса в левом и правом потомке через l_i и r_i оценить разбиении можно по формуле

$$Gini_{split}(T_1, T_2) = \frac{|T_1|}{|T|}\left(1 - \sum_{i=1}^{k}\left(\frac{l_i}{|T_1|}\right)^2\right) + \frac{|T_2|}{|T|}\left(1 - \sum_{i=1}^{k}\left(\frac{r_i}{|T_2|}\right)^2\right) \to \min \qquad (14)$$

Преобразовывая и упрощая формулу (14), можно получить следующее

$$Gini_{split}(T_1, T_2) = \frac{1}{|T|}\left(|T_1|\left(1 - \frac{1}{|T_1|^2}\sum_{i=1}^{k} l_i^2\right) + |T_2|\left(1 - \frac{1}{|T_2|^2}\sum_{i=1}^{k} r_i^2\right)\right) \to \min \qquad (15)$$

Так как умножение на константу не влияет на минимизацию, имеем

$$Gini_{split}(T_1, T_2) = |T_1| - \frac{1}{|T_1|}\sum_{i=1}^{k} l_i^2 + |T_2| - \frac{1}{|T_2|}\sum_{i=1}^{k} r_i^2 \to \min \qquad (16)$$

$$Gini_{split}(T_1, T_2) = |T| - \left(\frac{1}{|T_1|} \sum_{i=1}^{k} l_i^2 + \frac{1}{|T_2|} \sum_{i=1}^{k} r_i^2 \right) \rightarrow \min \qquad (17)$$

В итоге критерий разбиения имеет вид

$$Gini_{split}(T_1, T_2) = \frac{1}{|T_1|} \sum_{i=1}^{k} l_i^2 + \frac{1}{|T_2|} \sum_{i=1}^{k} r_i^2 \rightarrow \max \qquad (18)$$

Теперь, следует определить, как формируется разбиение.

Если признак числовой, то разбиение происходит аналогично алгоритму ID3 и правило в узле имеет вид

$$v_i \leq t_i \qquad (19)$$

Если признак номинальный строковый, то правило имеет вид

$$v_i \; R(v_i), \qquad (20)$$

где $R(v_i)$ – некоторое непустое подмножество множества значений признака A_i в обучающей выборке.

Алгоритм последовательно сравнивает все возможные разбиения и выбирает наилучший признак и наилучшее разбиение для него.

3 Программная реализация алгоритмов

3.1 Входные данные

В данном разделе будет рассмотрено, как и в каком формате хранятся входные данные алгоритма и его основные классы.

3.1.1 Тексты

Каждый текст хранится в виде отдельного файла с расширением .ctxt. Структуру можно отобразить на примере.

Пример

```
<Автор:String/Шолохов Михаил/>
<Название:String/Тихий Дон_Книга1/>
<Жанр:String/Роман/>
697601
      * ЧАСТЬ ПЕРВАЯ *
I  Мелеховский двор - на самом краю хутора.  Воротца  со
скотиньего  база ведут на  север  к  Дону.  Крутой
восьмисаженный  спуск  меж  замшелых  в прозелени меловых глыб,
и вот берег: перламутровая россыпь ракушек,  серая изломистая
кайма нацелованной волнами гальки и дальше -  перекипающее  под
ветром вороненой рябью стремя Дона. На  восток,  за  красноталом
гуменных плетней,  -  Гетманский  шлях,  полынная  проседь,
истоптанный  конскими копытами бурый, живущий придорожник,
часовенка  на  развилке;  за  ней  - задернутая текучим маревом
степь.
```

В начале файла хранятся значения номинальных признаков текста в формате <Название признака:тип значения/значение>. Каждое значение признака хранится на отдельной строке. После признаков хранится приблизительное количество символов в тексте. Далее уже непосредственно текст.

Класс, в котором содержатся данные о текстах, имеет структуру, описанную ниже.

Класс ***Text*** имеет следующие поля: *text* – массив символов, в котором содержится непосредственно текст, *path* – переменная типа *string*, в которой хранится путь к файлу текста, *atributes* – список номинальных признаков текста, *formText* – ссылка на форму, отображающую содержимое класса. Следует отметить, что признаки текстов описываются в отдельном классе *Atribute.*

Следует так же указать основные методы класса. Методы *Read()* и *SaveToFile()* работают с файлом текста. Методы *AddAtribute(), DeleteAtribute()* и *ShowAtribute()*

используются для работы с признаками текста, добавляют, удаляют и отображают их на форму признаков текста, *FormShow()* и *FormUpdate()* – методы, работающие с формой текста, обеспечивают отображение содержимого класса на форму текста и обновление данных на ней, метод *FormClosed* удаляет ссылку на форму при ее закрытии.

3.1.2 Признаки

Каждый признак хранится в отдельном файле с расширением .sgn. Файл имеет следующий формат:

```
Название признака
Тип признака
Значение признака
```

Признаки бывают четырех типов *Set(Atribute)* – номинальный строковый признак, *Number(TextWord)* – частотный признак, значения которого вычисляются подсчетом частот встречаемости в тексте отдельных слов, *Number(TextSubstring)* – признак аналогичен предыдущему, но подсчитываются частоты встречаемости подстроки или буквосочетания, *Number(Atribute)* - номинальный числовой признак. Так же в отдельном файле хранится целевой признак, структура файла у него такая же, как и у обычных признаков.

Класс, содержащий всю информацию о признаках, называется ***Sign.***

Основные поля класса: *name* - переменная типа *string*, хранит имя признака, *type* – переменная, хранящая тип признака, *value* – переменная типа *string*, хранящая значение признака, *path* – переменная типа *string*, хранящая путь к файлу признака. Дополнительные поля класса: *formSign* - ссылка на форму, отображающую содержимое данного класса, *register* – логическая переменная, которая определяет, учитывать ли регистр при поиске признака в тексте.

Свойства класса дают доступ к данным класса.

Класс *Sign* имеет следующие методы: *Read* – считывает данные из файла признака, *SaveToFile()* – сохраняет данные в файл признака, *FormShow()* – отображает содержимое признака на форму, *FormClosed()* – удаляет ссылку на форму при ее закрытии.

3.1.3 Проекты

Проекты хранятся в файлах с расширением .prt. Структура файла следующая.

```
Количество текстов
Ссылка на файл с текстом
Ссылка на файл с текстом
```

```
......
Ссылка на файл с текстом
Количество признаков
Ссылка на файл с признаком
Ссылка на файл с признаком
.....
Ссылка на файл с признаком
Количество целевых признаков
Ссылка на файл с целевым признаком
```

Класс, в котором хранятся данные о проекте называется ***Project***. Данный класс является основным, так как он логически объединяет в себе признаки и тексты. Однако, стоит отметить, что данный класс не связан напрямую с классом ***Text*** и сам текст не хранится в классе ***Project***, но текст можно получить в любой момент, так как в классе хранится путь к каждому тексту.

Основные поля класса ***Project***: *path* – переменная типа *string,* хранящая путь к файлу проекта, *signs* – список признаков, *textsPath* – список путей к файлам текстов, *targetSign* – целевой признак. Дополнительные поля класса: *showTextAtribute* – логическая переменная, которая определяет отображать на форму названия текстов или нет, *formProject* – ссылка на форму проекта. Рассмотрим основные методы класса ***Project***. Методы *Read()* и *SaveToFile()* работают с файлом признака, первый читает, второй сохраняет. Метод *UsingSignSelected()* используется для выбора признаков, которые будут участвовать в классификации и строит таблицу обучающей выборки и дерево решений. Остальные методы работают с формой проекта отображают все необходимые для пользователя данные.

3.2 Построение деревьев решений

Приложение имеет иерархическую структуру (Рис. 2), то есть старшие классы знают о младших, а младшие о старших не знают.

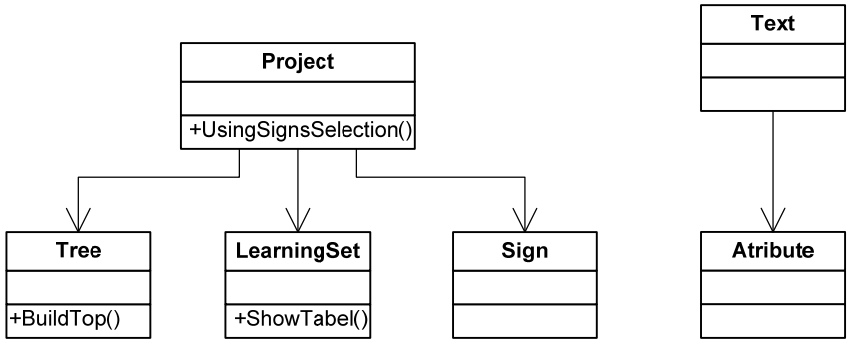


Рисунок 2 - Основные классы

В данном разделе будут рассмотрены основные классы, участвующие в построении дерева решений.

Класс *Searcher*. Статический класс, который используется для подсчета частот встречаемости слов. Для поиска признаков в тексте используется алгоритм Кнута-Морриса-Пратта[9] поиска подстроки в строке, который реализован в методе KMPSubstring(). Этот метод возвращает позицию первого символа найденной подстроки методу KMP(), который проверяет найденное вхождение и увеличивает значение признака частот.

Класс *LearningSet*. Данный класс содержит в себе обучающую выборку, которая представляет собой список столбцов обучающей выборки, и отображает ее в виде таблицы. Причем для частотных признаков, в таблице отображаются абсолютные частоты, а для вычислений используются относительные частоты (по отношению к количеству символов в тексте), так как абсолютные частоты не имеют практического значения, ведь автор может писать и романы, и рассказы, а стиль автора остается прежним. Задачу построения таблицы выполняет метод *ShowTabel()*.

Класс *TreeTop*. Этот класс используется для построения и хранения дерева решений. Класс представляет собой одну вершину, в которой при вызове конструктора проверяется необходимость разбиения (проверка на лист). Если разбиение необходимо, то происходит перебор всех возможных разбиений и выбор наиболее информативного. После чего тексты делятся между потомками. Далее в каждом потомке снова происходят те же действия. Методы: *ID3(), C4.5.(), CART()* – строят деревья решения по алгоритму ID3, C4.5, CART соответственно, *DrawTop()* отрисовывает вершину, а метод *ShowForm()* позволяет отобразить дерево на форму. А классификация с помощью построенного дерева осуществляется в методе *RunClassification()*.

4 Тестирование построенных алгоритмов

Тестирование алгоритмов проведено на реальных текстах, взятых из электронной библиотеки Максима Мошкова. [10]

Для тестирования разработанного приложения были составлены несколько выборок текстов различных объемов и содержания.

Одна из выборок состоит из пятидесяти пяти текстов и сорока одного признака. Приведем пример работы приложения на этой выборке.

Дерево, построенное алгоритмом ID3, имеет вид (Рис. 3).

Рисунок 3 - Дерево решений ID3

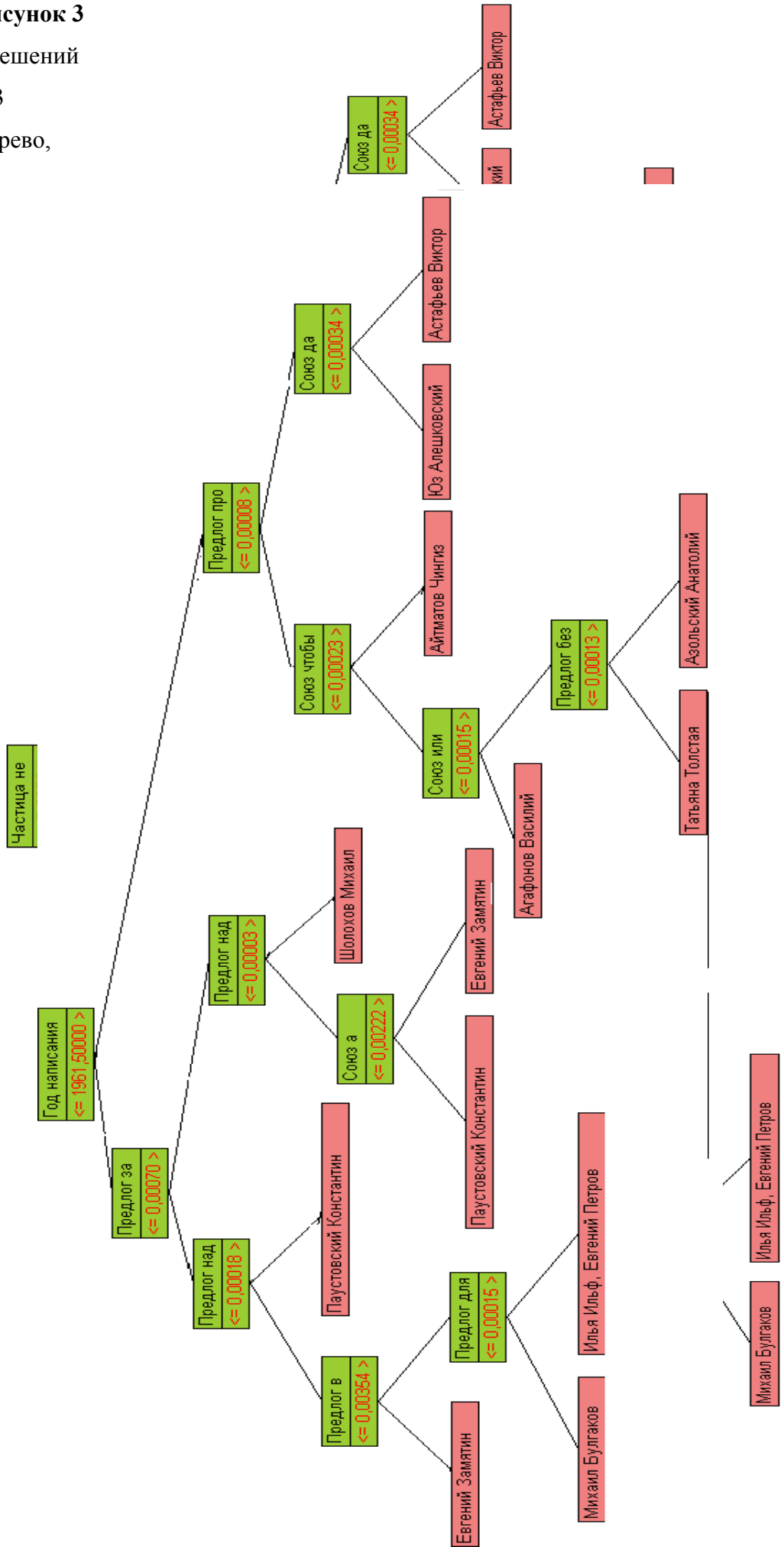
Дерево, построенное алгоритмом C4.5 на той же выборке (Рис. 4).

Рисунок 4 - Дерево решений C4.5

Дерево, построенное алгоритмом CART (Рис. 5).

Рисунок 5 - Дерево решений, построенное алгоритмом CART

Строит так



же рассмотреть процесс классификации.

Просеивание по дереву решений происходит следующим образом. Для классифицируемых текстов вычисляются значения признаков, аналогично построению обучающей выборки. В итоге получается таблица, соответствующая таблице 1 (см. раздел 2.1), но метки классов в получившейся таблице отсутствуют.

Далее каждый классифицируемый текст передается в корень дерева решений. Значение признака текста сравнивается с порогом разбиения (для числового признака), либо со значением номинального признака и текст передается соответствующему потомку вершины. Эти действия повторяются, пока текст не окажется в листе дерева. Значение целевого атрибута листа и присваивается тексту как метка класса.

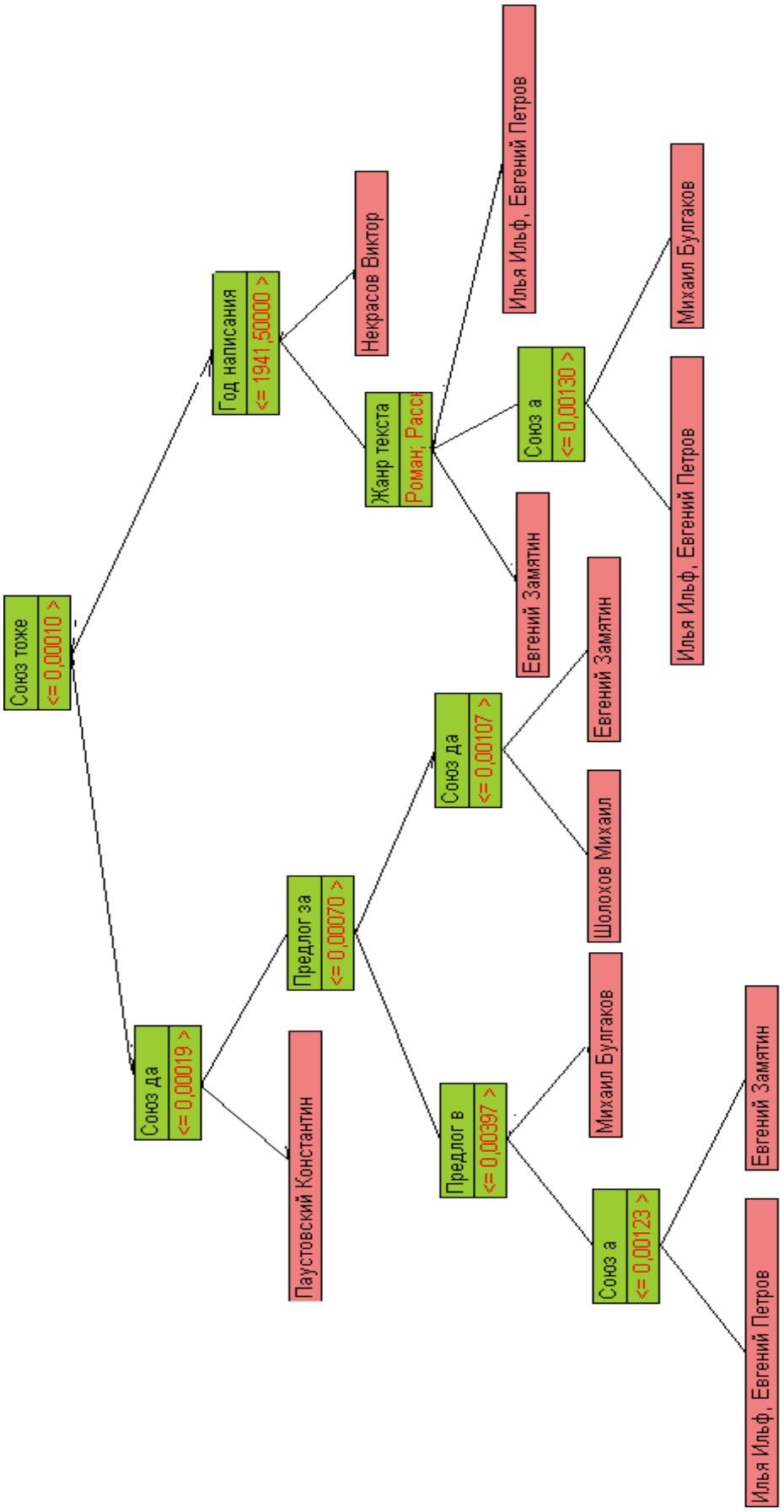
Для проверки правильности построенных деревьев, был проведен следующий тест. В качестве текстов для просеивания берутся тексты из обучающей выборки. Все алгоритмы показали правильность работы и всех тексты обучающей выборки классифицировали правильно.

После этого была проведена классификация текстов, которые не участвовали в построении дерева решений. Были получены следующие результаты. Дерево, построенное с помощью алгоритм ID3, правильно классифицировало шесть текстов из одиннадцати предложенных, для алгоритма C4.5 этот результат лучше – восемь правильно классифицированных текстов из одиннадцати, для алгоритма CART верно классифицировано семь текстов из одиннадцати.

Второй вариант обучающей выборки, по которой строилось дерево решений, состоит из восьмидесяти двух текстов и сорока одного признака.

Дерево решений, построенное алгоритмом ID3 (Рис. 6)

Рисунок 6 – Ветвь дерева решений,



построенного алгоритмом ID3 для второй обучающей выборки, в которой в разбиении участвует номинальный признак

В ходе классификации текстов были получены следующие результаты. Дерево решений, построенное алгоритмом C4.5 правильно классифицировало восемь текстов из одиннадцати. Дерево решений, построенное алгоритмом CART, показало такой же результат. Результат классификации дерева, построенного алгоритмом ID3 не изменился, правильно классифицировано шесть текстов из одиннадцати.

На основе данных, полученных в результате построения деревьев и классификации текстов можно сделать вывод, что номинальные признаки мало пригодны для применения в задачах определения авторства текстов, так как, используя их, дерево показывает более низкое качество классификации.

Однако так же было выяснено, что большой информативностью обладает признак Год написания произведения, не смотря на то, что является номинальным, и его можно использовать наряду с частотными признаками в обучающей выборке.

Так же было проведено сравнение алгоритмов и для классификации текстов можно рекомендовать использовать алгоритм C4.5, так как он дает хорошие результаты при классификации неизвестных текстов.

Заключение

В ходе работы были изучены алгоритмы построения деревьев решений, реализован программный продукт, позволяющий классифицировать неизвестного автора с помощью полученного дерева решений с использованием как числовых, так и номинальных признаков. Работоспособность приложения проверена на реальных текстах библиотеки Максима Мошкова [10].

Так как провести широкомасштабное тестирование в рамках данной работы не представляется возможным, так как ее целью не ставилось обширное лингвистическое исследование, то был проведен обзорный анализ-тестирование, в результате которого были сделаны следующие выводы.

На небольших выборках текстов, состоящих из пятидесяти-шестидесяти текстов и десяти классов, хороший результат дает дерево решений, построенное алгоритмом C4.5, давая правильный ответ в 70% случаев, в отличие от алгоритмов ID3 (55%) и CART(60%)

При увеличении размера выборки до восьмидесяти текстов и добавлении классов, алгоритм CART улучшил свой результат до 70%, алгоритм C4.5 показал такой же результат, дерево решений, построенное алгоритмом ID3 дало 53% правильных ответов.

Так же была определена актуальность использования номинальных признаков при классификации текстов, и как показало исследование, оправданным является использование признака Год написания произведения, так как он оказывает положительное влияние на качество классификации по построенному дереву решений.

Список использованных источников и литературы

1. Фоменко В.П., Фоменко Т.Г. Авторский инвариант русских литературных текстов. Предисловие А.Т. Фоменко // Фоменко А.Т. Новая хронология Греции: Античность в средневековье. Т. 2. М.: Изд-во МГУ, 1996.
2. Шевелев О. Г., Петраков А. В. Классификация текстов с помощью деревьев решений и сетей прямого распространения // Вестник Том. гос. ун-та, 2006. - № 290. - С. 300-307.
3. Романов А.С. Методика и программный комплекс для идентификации автора неизвестного текста: Автореф. дис. канд. тех. наук / Томский гос. ун-т систем управления и радиоэлектроники – Томск, 2010. – 26 с.
4. Деревья классификации и регрессии [Электронный ресурс]. – URL http://www.williamspublishing.com/PDF/978-5-8459-1170-4/part.pdf (дата обращения: 02.06.2011)
5. Методы классификации и прогнозирования. Деревья решений. [Электронный ресурс]. – URL http://www.intuit.ru/department/database/datamining/9/1.html (дата обращения: 02.06.2011)
6. Quinlan R. C4.5: Programs for Machine Learning. - San Mateo, CA: Morgan Kaufmann, 1993. – 302 p.
7. Деревья решений – C4.5 математический аппарат. [Электронный ресурс]. – URL http://www.basegroup.ru/library/analysis/tree/math_c45_part1/ (дата обращения: 02.06.2011)
8. Деревья решений – CART математический аппарат. [Электронный ресурс]. – URL http://www.basegroup.ru/library/analysis/tree/math_cart_part1/ (дата обращения: 03.06.2011)
9. Кормен Т. Алгоритмы: построение и анализ : пер. с англ. / Т. Кормен, Ч. Лейзерсон, Р. Ривест ; под ред. А. Шеня. – 2-е изд., стереотип. – М. : МЦНМО: БИНОМ. Лаборатория знаний, 2004. – 960 с. : ил.
10. Библиотека Максима Мошкова [Электронный ресурс]. – URL http://lib.ru/ (дата обращения: 04.06.2011)

Приложение А. Руководство пользователя

После запуска программы следует выполнить следующую последовательность действий. В главном меню выбрать Файл -> Создать -> Текст. В открывшемся окне задать путь к файлу с текстом и название создаваемого файла. В поле «Текст» ввести текст.

На рисунке 1 изображена форма для отображения текстов. Помимо текста на ней отображаются характеристики текста, которые можно описать в форме задания атрибутов текста. Открыть ее можно, нажав на кнопку, обведенную кружком на рисунке. После нажатия на кнопку «Сохранить», текст записывается в файл.

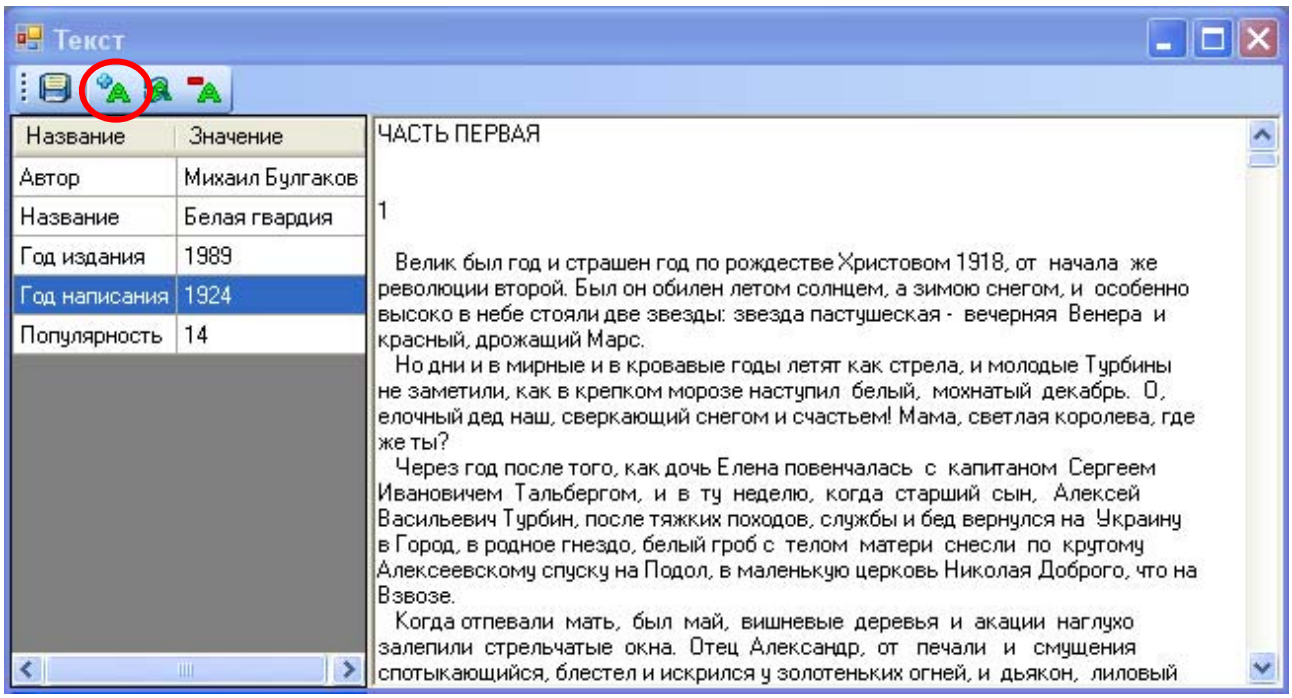


Рисунок А.1 - Форма для отображения текстов

А на рисунках 2 и 3 изображены форма, в которой описываются атрибуты текстов и признаки.

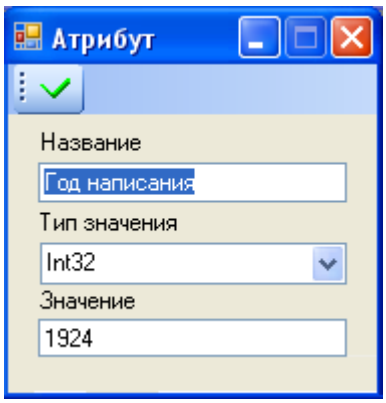


Рисунок А.2 - Форма для задания признаков текста

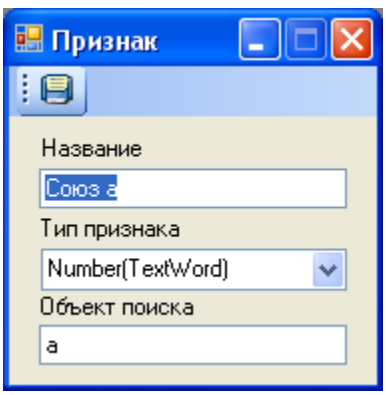


Рисунок А.3 - Форма для задания признака

Для создания признака следует выбрать Файл -> Создать -> Признак. В появившемся окне необходимо задать путь к файлу признака и задать его название. В поле «Название» - название признака, в поле «Тип» выбрать тип создаваемого признака, в поле «Объект поиска» указать слово, которое будет искаться в тексте.

Когда все тексты и признаки созданы, следует создать проект. Для этого следует выбрать Файл -> Создать -> Проект, задать адрес сохранения файла и его имя. Чтобы добавить тексты или признаки в проект следует кликнуть правой кнопкой мыши на поле обучающей выборки или признака и, выбрав «Добавить», добавить созданные признаки и тексты в проект.

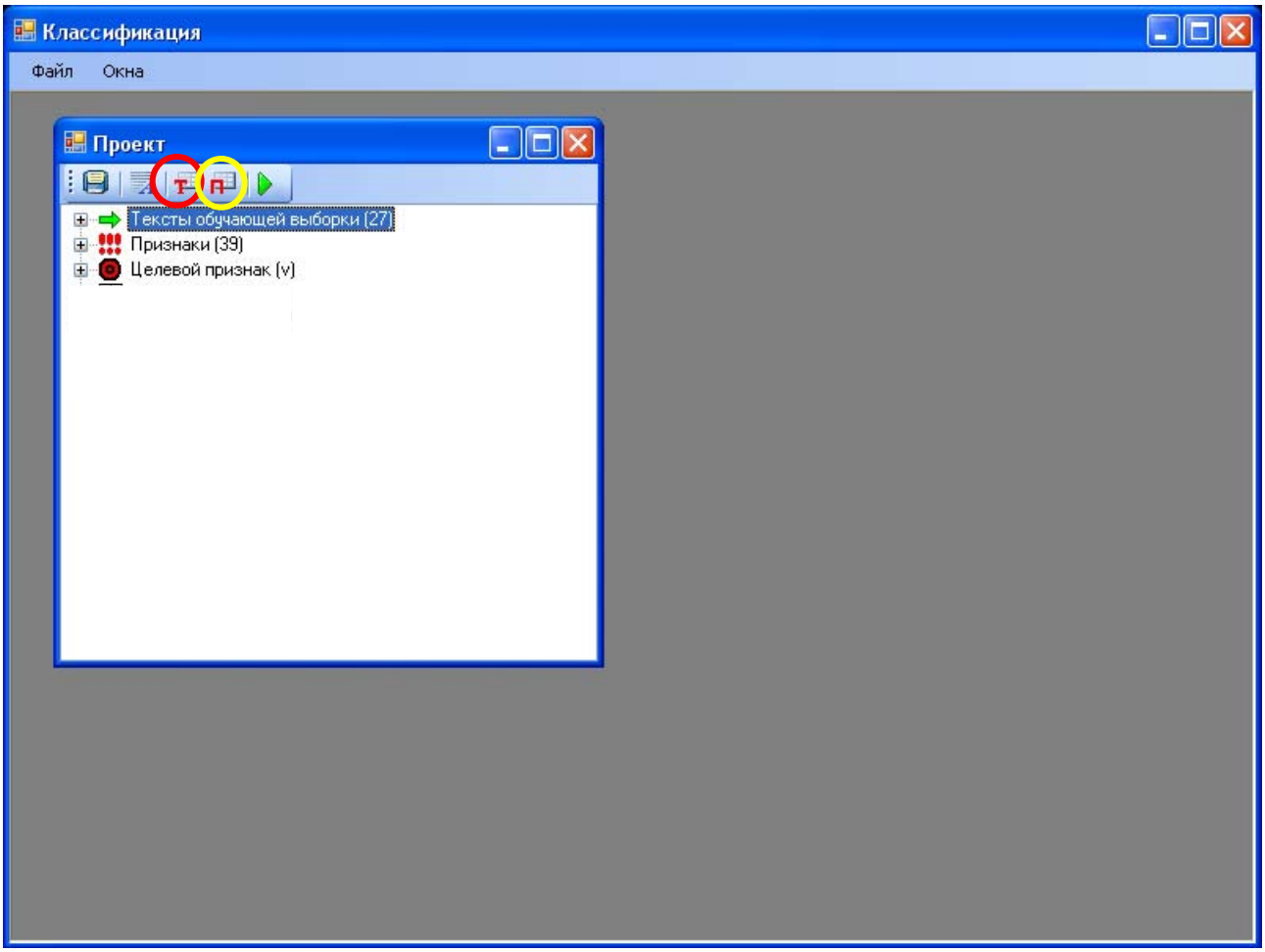


Рисунок А.4 - Главная форма и форма проекта

Нажав на кнопку, выделенную на рисунке красным цветом, пользователь получит сводную таблицу текстов, в которой указаны все заданные признаки текста. Нажав на кнопку, выделенную на рисунке желтым цветом, пользователь получит сводную таблицу признаков, в которой указаны все признаки и их свойства.

В результате этих действий задаются входные данные, необходимые для построения дерева решений.

Если проект, тексты или признаки уже созданы, то нет необходимости создавать их вновь, нужно просто открыть их (например, для проекта Файл -> Открыть -> Проект)

Далее, при нажатии на кнопку пуска, начинается процесс построения обучающей выборки. В появившейся таблице (Рис. 5) необходимо выбрать алгоритм, по которому будет строиться дерево решений, и нажать на кнопку «Пуск».

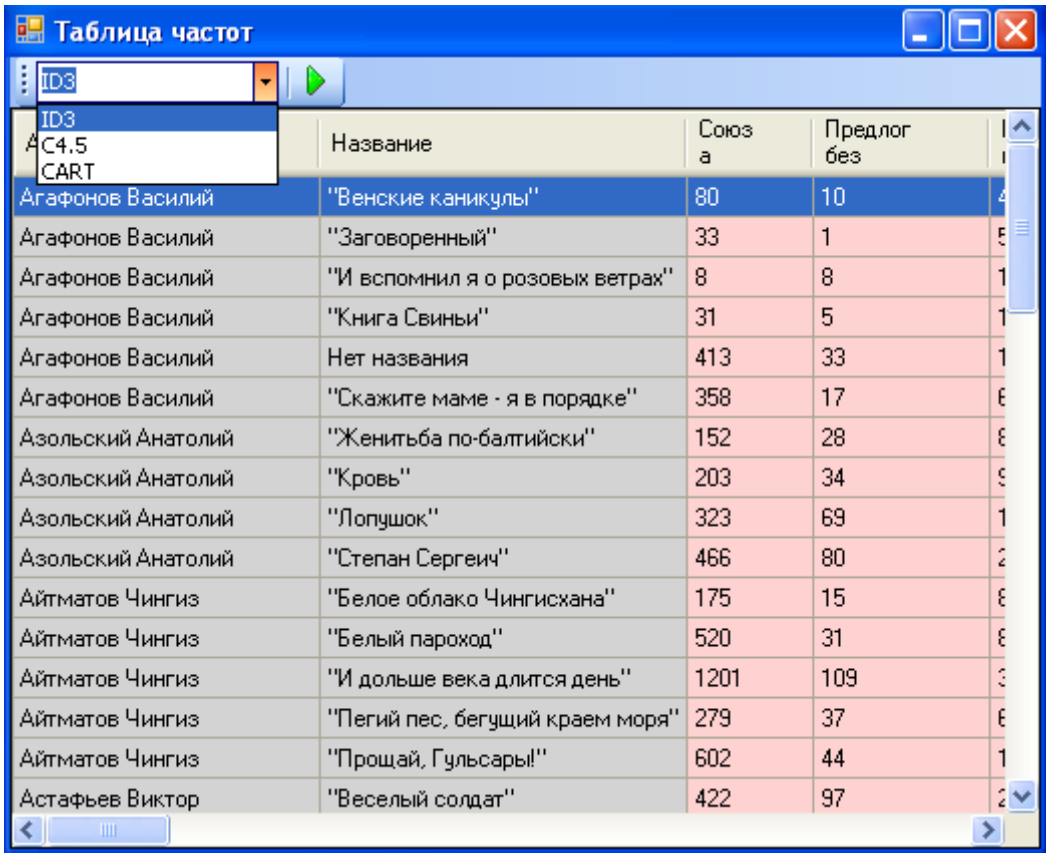


Рисунок А.5 - Таблица обучающей выборки

Нажав на кнопку пуска на форме дерева решений, можно провести классификацию текста неизвестного автора. Для этого на форме классификации следует добавить тексты для анализа, нажав на кнопку, выделенную на рисунке 6 красным кружком (при нажатии на кнопку, выделенную желтым, текст удаляется из списка анализируемых).

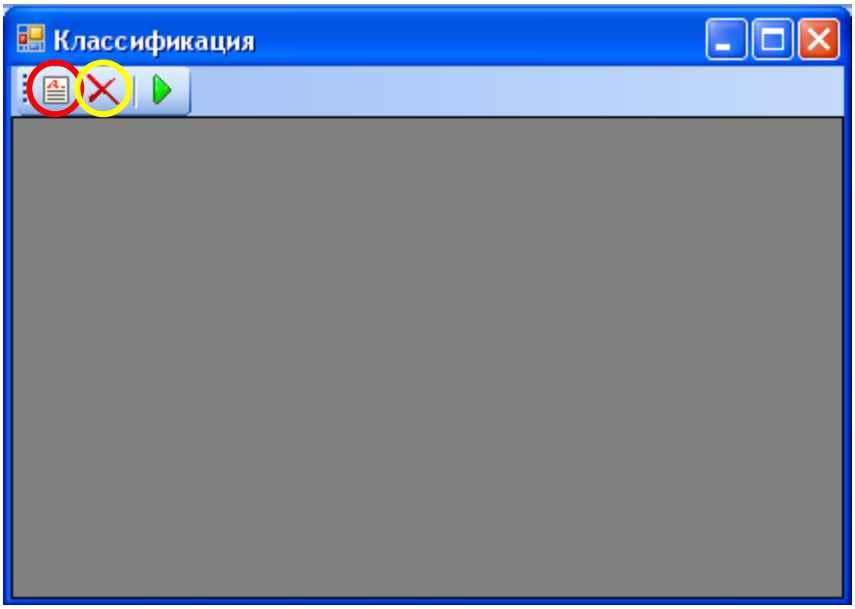


Рисунок А.6 - Форма классификации

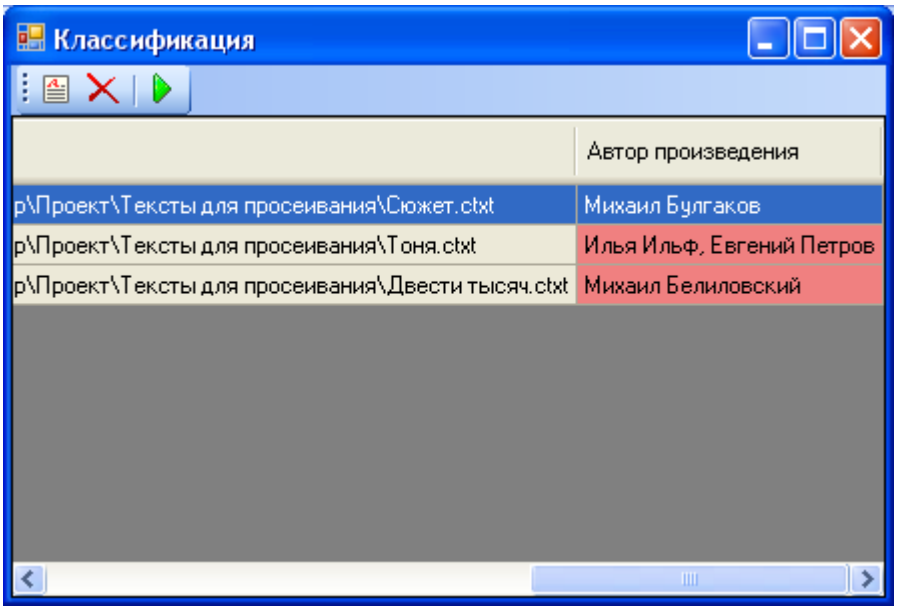


Рисунок А.7 - Классификация

Приложение Б. Руководство программиста

Классы, описанные в разделе 3.1, не имеет смысла рассматривать в данном приложении, так как они довольно просты и не требуют более подробных пояснений. Следует, более подробно остановится на классах, которые не были указаны ранее, либо было дано недостаточно полное пояснение для них.

Indication – служебный класс, который представляет собой столбец таблицы обучающей выборки.

Основные поля класса:

type – тип признака, который описывает данный столбец;

values - список значений. Для частотных признаков – относительные частоты, для номинальных числовых – значения признаков без изменений, для номинальных строковых – индекс списка перекодировок;

name – имя столбца;

absoluteFrequency – массив абсолютных частот (существует только для частотных призноков);

indicationValues – список перекодировок (существует только для номинальных строковых признаков). Его значениями является неповторяющиеся значения признака, а индексы записываются вместо значений в список values.

Данный класс имеет два конструктора следующего вида:

Indication(Double[] Values, Int32[] AbsoluteFrequency, String Name) – конструктор для числовых признаков.

Indication(String[] Values, String Name) – конструктор для строковых признаков.

LearningSet – класс обучающей выборки.

Основные поля класса:

indication – список столбцов обучающей выборки;

target – столбец целевого признака;

textPath – список путей к текстам обучающей выборки;

Основные методы класса:

SaveToFile() – сохранияет обучающую выборку в файл;

ReadFromFile() – считывает обучающую выборку из файла;

ShowTabel() - отображает обучающую выборку на форму.

TreeTop – класс для построения дерева. Представляет собой одну вершину дерева решений, коотрая имеет ссылки на помков.

Основные поля класса:

isLeaf – логическая переменная, указывающая на лист дерева;

textPathIndex – список индексов текстов, используются при делении обучающей выборки между потомками узла дерева;

signIndex – индекс признака, по которому проводистя разбиение;

limit – порог разбиения, используется для числовых признаков;

children – список потомков узла;

picture – хранит изображени дерева;

lerningSet – ссылка на обучающую выборку;

childrenLerningSet – обучающие выборки потомков;

childrenTarget – список, используемый для деления значений целового признака между потомками.

Основные методы класса

ID3(LearningSet LearningSet) – рекурсивный метод построения дерева решений алгоритмом ID3. В качестве параметра принимет обучающую выборку, которая каждый раз делится между потомками.

C4_5(LearningSet LearningSet) – рекурсивный метод построения дерева решений алгоритмом C4.5.

CART()

DrawTop() – метод возвращающий изображение дерева.

BuildTabel() – строит таблицу для классификации неизвестного текста.

RunClassification() – метод, выполняющий классификацию текста на основе построенного дерева.

Так же можно коротко рассмотреть вспомогательные классы

Converters – используется для преобразования типов объектов из обычных в специфические типы проекта и обратно.

ProjectStatistics – используется для составления сводных таблиц текстов и признаков.

Как уже говорилось, классы имеют иерархическую структуру. И так как младшие классы этой иерархии не могут напрямую общаться с классами стоящими на уровне выше, то для организации связи между ними использовался паттерн Событие. То есть младшие классы сообщают о своих изменениях с помощью событий. В частности паттерн Событие использовался в классе Atribute, для сообщения об изменениях классу Text, в классах Text и Sign для сообщения об изменениях классу Project. Так же данный паттерн используется для сообщения классу Project, какой алгоритм выбран для построения дерева решений.

Так же при разработке использовался паттерн Одиночка. При помощи этого паттерна реализована главная форма приложения.