Московский Государственный Университет
имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра системного программирования

Курсовая работа

Исследование и разработка методов определения кинокартины, использующих информацию о сюжете

Выполнил:
Студент 327 группы
Пархоменко Павел Андреевич

Научный руководитель:
Недумов Ярослав Ростиславович

Москва
2014

Оглавление

Аннотация

Данная работа посвящена исследованию современных методов определения кинокартины, использующих информацию о сюжете, а также разработке и реализации собственного метода, позволяющего решать данную задачу.

Разработана диалоговая система, основанная на вероятностной модели.

Введение

Уже на протяжении нескольких десятилетий кинокартины являются неотъемлемой частью жизни человека. И с каждым годом их значимость все увеличивается и увеличивается. На сегодняшний день уже тяжело представить свою жизнь без такого важного искусства, как кинематограф.

Очень часто можно столкнуться с ситуацией, когда возникает желание пересмотреть какую-нибудь кинокартину, просмотренную много лет назад. Но не всегда это легко можно сделать. Человеку свойственно забывать названия кинолент, а узнать фильм, помня только общие детали сюжета (или, что еще хуже, отрывок) не является тривиальной задачей. На данный момент, ни один поисковый сервис не умеет эффективно решать данную задачу.

Основная сложность, возникающая в ходе решения данной задачи, заключается в том, что не существует ресурсов, содержащих подробные описания сюжетов кинокартин. Также при разработке системы, решающей подобную задачу, необходимо учитывать тот факт, что кинематограф является очень динамичной отраслью: каждый год выпускаются сотни, а то и тысячи новых фильмов. Следовательно, система должна уметь собирать и обрабатывать информацию о новых фильмах.

Целью курсовой является исследование и разработка метода определения кинокартины, использующего информацию о сюжете.

1 Постановка задачи

Целью данной работы является исследование и разработка метода определения кинокартины, использующего информацию о сюжете.

Можно выделить следующие основные задачи данной курсовой работы:

- Исследовать существующие методы определения кинокартины
- Разработать систему, агрегирующую описания кинокартин способом, позволяющим затем эффективно осуществлять их поиск.
- Найти коллекцию описания фильмов на русском языке.
- Подготовить тестовый набор для проверки качества работы системы.

2 Обзор существующих решений

Системы, решающие данную задачу можно выделить в 2 большие группы:

1. Системы, реализующие поиск в базе данных по заданным значениям параметров.
2. Системы, являющиеся интерфейсом для общения пользователей и экспертов в данной области.

Рассмотрим данные системы более подробно.

2.1 Системы, реализующие поиск в базе данных по заданным значениям параметров

Данные системы предоставляют интерфейс, с помощью которого пользователь передает известную информацию о фильме. Система ищет в базе данных наиболее подходящие под данное описание фильмы и возвращает их пользователю. Как правило, интерфейс таких систем состоит из полей, которые нужно заполнить. Каждое поле соответствует какой-то специфичной информации (например, жанр, имя главного героя, актер, продюсер, режиссер, и т.д.).

Некоторые системы учитывают информацию о сюжете. Например, расширенный поиск сайта «Кинопоиск»[1] позволяет искать фильм по заданным терминам. В процессе данного поиска находятся фильмы, краткое описание которых содержит наибольшее количество общих слов с введенным запросом.

Тем не менее, данный метод является не очень продуктивным, так как он не обрабатывает семантику запроса.

Главный недостаток данной системы – неэффективная обработка информации о сюжете.

Достоинством данного подхода является его полная автоматизация. Также, в случаях, когда пользователь помнит достаточно информации о самом фильме (актеров, героев, страну производства и т.д.), система показывает хорошее качество работы.

2.2 Системы, являющиеся интерфейсом для общения пользователей и экспертов в данной области

Данные системы предоставляют интерфейс, помогающий общаться пользователю и эксперту. Пользователь предоставляет всю известную информацию о фильме эксперту, а тот, в свою очередь, пытается вспомнить фильм. Чаще всего такие системы реализованы в виде форума.

[1] http://www.kinopoisk.ru/s/

Достоинство такого подхода заключается в том, что эксперт может вспомнить фильм по очень маленькому описанию.
Недостаток – человеческие знания сильно ограничены, и практически невозможно знать про все кинокартины. Также недостаток данных систем заключается в том, что они не являются автоматизированными.

2.3 Вывод

Существующие на данный момент решения поставленной задачи не совсем эффективно используют информацию о сюжете. В частности, семантика запроса пользователя чаще всего никак не обрабатывается. Необходимо реализовать систему, которая решит данную проблему.

3 Исследование и построение решения задачи

Для построения решения поставленной задачи было проведено исследование предметной области. Изучались русскоязычные запросы на специализированных форумах, в которых пользователи искали фильмы, предоставляя всю имеющуюся информацию о них. В ходе обзора запросов пользователей были выявлены следующие тенденции:

- Пользователь часто помнит страну, в которой был произведен фильм, временные рамки, в которых разворачиваются события, жанр кинокартины и актеров, игравших в ней.
- Пользователь практически никогда не помнит примерное название картины (какие-то слова из названия) и имена героев.
- Очень часто единственной имеющейся информацией оказывается знание о сюжете (или о каких-то отрывках) произведения.
- Когда пользователю задавали наводящие вопросы, касающиеся кинокартины, он вспоминал какую-то новую информацию.

Опираясь на полученные данные, было решено реализовать систему, задающую вопросы о кинокартине пользователю, получающую на них ответы, и, на основе собранных данных, определяющую наиболее правдоподобный фильм.

Для полного решения поставленной задачи, необходимо решить следующие подзадачи

- Заполнить базу данных существующими фильмами.
- Сгенерировать вопросы, которые будут узнавать информацию у пользователя.
- Определять вопрос, который будет задан на очередной итерации.
- Узнавать наиболее вероятный фильм, соответствующий ответам на вопросы.

3.1 Заполнение базы данных фильмами

Для заполнения базы данных фильмами использовался сайт «Кинопоиск»[2]. Этот ресурс содержит большинство известных отечественных и зарубежных кинокартин, информация о которых использовалась как в заполнении базы данных, так и в дальнейшем обучении.

Например, каждый фильм содержит следующую информацию:

- Год создания
- Страна, в которой фильм снимался
- Режиссер
- Жанр
- Бюджет
- Снимавшиеся актеры

[2] www.kinopoisk.ru

Это неполный список имеющейся информации по каждому фильму. При создании вопросов, узнающих данную информацию о фильме, можно использовать данную информацию для автоматической генерации тренировочной выборки.

3.2 Генерация вопросов

Сгенерированные вопросы можно условно разделить на две части:

1. Вопросы, узнающие общую информацию (жанр, главные герои, актеры и т.д.) о фильме
2. Вопросы, узнающие информацию о сюжете

Пример вопросов, узнающих общую информацию:

- «Является ли фильм комедией?»
- «Снимался ли в фильме Юрий Никулин?»

С помощью подхода, описанного в предыдущем пункте, можно генерировать тренировочную выборку, использую такие вопросы.

Пример вопросов, узнающих информацию о сюжете:

- «Связан ли фильм с пиратами?»
- «Действие фильма происходит в Москве?»

С помощью таких вопросов система узнает данные о сюжете фильма.

3.3 Определение результата

Для определения наиболее правдоподобного фильма по уже имеющимся ответам использовался метод машинного обучения (классификация[]). В качестве классификатора был выбран наивный байесовский классификатор[]. В данной системе он работает следующим образом.

Предположим, что пользователю уже было задано некоторое количество вопросов, на которые он дал ответ. Если пользователь на какой-то вопрос отвечал «Не знаю», информация об этом ответе никак не использовалась при классификации. Это было сделано для того, чтобы отбросить шум, который мог возникать из-за отсутствия данных.

Введем следующие обозначения:

$F -$ загаданный фильм

$Q -$ заданный вопрос

$A -$ ответ на вопрос ("да" или "нет")

$< Q, A >$ $-$был задан вопрос Q, на него был получен ответ A

Для получения результата необходимо для каждого фильма F посчитать вероятность того, что он искомый при известных ответах на некоторые вопросы:

$$P(F \mid < Q_1, A_1 >, \dots, < Q_n, A_n >) \quad (1),$$

где n – количество заданных вопросов.

Фильм с максимальным значением условной вероятности (1) при известных ответах будет считаться искомым.

Для вычисления условной вероятности фильма применяется теорема Байеса:

$$P(F \mid < Q_1, A_1 >, \dots, < Q_n, A_n >) =$$

$$= \frac{P(< Q_1, A_1 >, \dots, < Q_n, A_n > \mid F)\, P(F)}{\sum_{F'} P(< Q_1, A_1 >, \dots, < Q_n, A_n > \mid F')\, P(F')} \quad (2)$$

где F' - фильм. Сумма в знаменателе считается по всем известным фильмам.

Теперь задача состоит в нахождении аргумента F, максимизирующего формулу (2). Так как знаменатель в (2) не зависит от того, для какого фильма считается условная вероятность, следовательно, его можно отбросить, так как он никак не повлияет на результат.

Таким образом, задача сводится к нахождению фильма F_0, заданного формулой:

$$F_0 = \underset{F}{\operatorname{argmax}}\, P(< Q_1, A_1 >, \dots, < Q_n, A_n > \mid F)\, P(F) \quad (3)$$

$P(F)$ - априорная вероятность того, что именно фильм F искомый. Она может быть вычислена как отношение количества раз, когда пользователь искал именно фильм F, к общему количеству запусков программы.

Для вычисления условной вероятности $P(< Q_1, A_1 >, \dots, < Q_n, A_n > \mid F)$, необходимо предположить условную независимость ответов на вопросы при заданном фильме:

$$P(< Q_1, A_1 >, \dots, < Q_n, A_n > \mid F) = \prod_{i=1}^{n} P(< Q_i, A_i > \mid F) \quad (4)$$

$P(< Q_i, A_i > \mid F)$ – условная вероятность того, что на вопрос Q_i пользователь даст ответ A_i, при условии, что фильм F искомый. Данная вероятность может быть вычислена как отношение количества раз, когда пользователь искал именно фильм F и на вопрос Q_i ответил A_i, к общему количеству запусков программы, в которых искал фильм F и отвечал на вопрос Q_i.

Таким образом, если считать, что пользователь отвечает на вопросы независимо друг от друга, то задачу можно свести к нахождению фильма F_0, заданного формулой:

$$F_0 = \underset{F}{\operatorname{argmax}} \prod_{i=1}^{n} P(< Q_i, A_i > \mid F)\ P(F) \tag{5}$$

Если пересчитывать результаты после каждого ответа пользователя, то каждый такой пересчет будет занимать $O(|F|)$ времени, где $|F|$ - общее количество фильмов.

Выбор наивного байесовского классификатора был продиктован тем, что он устойчив к ошибочному ответу пользователя на небольшое количество вопросов, так как данный классификатор основан на вероятностной модели. Еще одним преимуществом является простота дообучения.

3.4 Определение очередного вопроса

Для определения вопроса, который будет задан пользователю, использовался вероятностный подход.

На каждой итерации выбирается тот вопрос, который уменьшает энтропию распределения $P(F \mid < Q_1, A_1 >, \dots, < Q_n, A_n >)$. Тогда каждый новый вопрос будет устранять как можно больше неопределенности.

Для этого необходимо посчитать вероятность каждого варианта ответа («да» и «нет») для каждого вопроса:

$$P_{Q,A} = \sum_{F} P(< Q, A > \mid F)\ P(F \mid\ < Q_1, A_1 >, \dots, < Q_n, A_n >) \tag{6}$$

Применяя формулу Байеса, можно переписать формулу (6) в следующем виде:

$$P_{Q,A} = \sum_{F} P(< Q, A > \mid F) *$$

$$* \frac{P(< Q_1, A_1 >, \dots, < Q_n, A_n > \mid F)\ P(F)}{\sum_{F'} P(< Q_1, A_1 >, \dots, < Q_n, A_n > \mid F')\ P(F')} \tag{7}$$

Предполагая независимость ответа на очередной вопрос от предыдущих, и, избавляясь от одинаковых знаменателей, получаем формулу (7) в виде:

$$P_{Q,A} = \sum_F P(< Q, A > \mid F) \prod_{i=1}^{n} P(< Q_i, A_i > \mid F)\ P(F) \tag{8}$$

Для решения подзадачи выбора следующего вопроса, необходимо вычислить условную энтропию при известном ответе для каждого вопроса. Энтропия имеет вид:

$$H(Q \mid A) = H(F \mid < Q, A >, < Q_1, A_1 >, \dots, < Q_n, A_n >) =$$

$$= \sum_{i=1,2} P(F \mid < Q, A_i >, < Q_1, A_1 >, \dots, < Q_n, A_n >) *$$

$$* \log P(F \mid < Q, A_i >, < Q_1, A_1 >, \dots, < Q_n, A_n >) * P_{Q,A_i} \tag{9}$$

Очередным вопросом, который будет задан пользователю, нужно выбирать вопрос, минимизирующий условную энтропию при известном ответе.

$$Q' = \operatorname{argmin}_Q H(Q\,|A) \tag{10}$$

Используя результаты предыдущих итераций, можно выбирать новый вопрос за время $O(|F| * |Q| * |A|)$.

3.5 Результаты тестирования

// Написать после тестирования

4 Описание практической части

4.1 Обоснование выбранного инструментария

Был разработан прототип на языке программирования Java[3]. Java – кроссплатформенный объектно-ориентированный язык программирования, обладающий огромным числом свободных библиотек, позволяющих ускорять разработку исходного кода системы.

В качестве библиотеки алгоритмов машинного обучения была использована библиотека Weka[4]. Данная библиотека содержит большое число эффективных реализаций алгоритмов машинного обучения, в том числе наивный Байесовский классификатор.

Для работы с реляционной базой данных была использована библиотека Hibernate[5]. Данная библиотека предоставляет фреймворк для отображения объектно-ориентированной модели данных в реляционные базы данных.

Для создания веб-интерфейса системы использовалась JSP(JavaServer Pages) технология. Данная технология позволяет создавать Web страницы, содержащие как статические, так и динамические компоненты. JSP является платформонезависимой, переносимой и легко расширяемой технологий для разработки веб-приложений.

4.2 Общая схема работы

В начале работы система предлагает выбрать один из 4 режимов:

- Основной режим
- Тренировочный режим
- Режим добавления вопроса в базу данных
- Режим добавления фильма в базу данных

4.2.1 Основной режим

Схема работы системы в основном режиме:

1. Установление подключения к базе данных. Инициализация данных. Создание пустого вектора признаков. Загрузка обученной модели.
2. Пока пользователь не остановит программу, или пока не пройдет N итераций, выполняются пункты 3-5.
3. Из множества всех вопросов, которые еще не были заданы пользователю, выбирается вопрос с минимальной условной энтропией. Текстовое представление выводится на экран.
4. Считывается ответ пользователя. Дополняется вектор признаков.

[3] http://www.java.com/
[4] http://www.cs.waikato.ac.nz/ml/weka/
[5] http://hibernate.org/

5. Вектор признаков передается классификатору. Результат классификации выводится на экран.
6. Если в ходе работы системы искомый фильм становится известным, ответы и искомый фильм сохраняются в базу данных для дальнейшего обучения.

На рисунке 1 отображена схема работы системы в основном режиме.

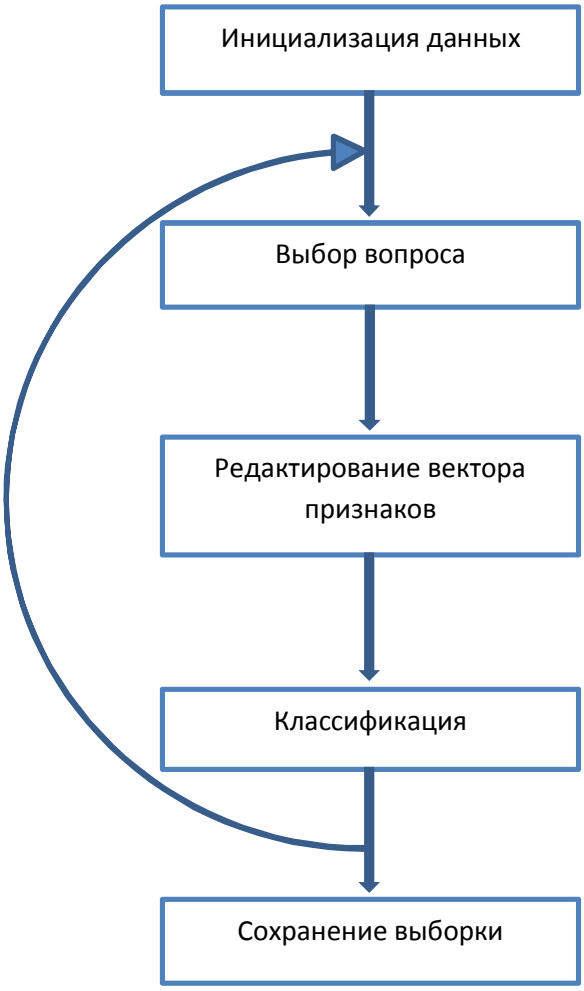

Рис. 1: Схема работы системы в основном режиме

4.2.2 Тренировочный режим

Схема работы системы в тренировочном режиме:

1. Установление подключения к базе данных. Инициализация данных.
2. Пока в базе данных остались неиспользованные выборки, выполняется пункт 3.
3. Загрузка из базы данных признаков (ответов на вопросы), относящихся к очередной выборке. Создание вектора признаков с соответствующими значениями.

4. Передача векторов признаков, полученных в пункте 3, классификатору для обучения.
5. Сохранение обученной модели.

На рисунке 2 отображена схема работы системы в тренировочном режиме.

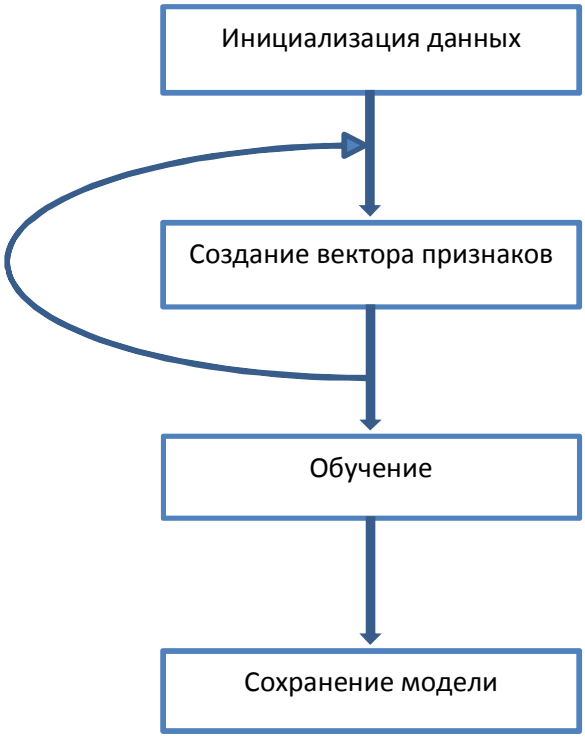

Рис. 2: Схема работы системы в тренировочном режиме

4.2.3 Режим добавления вопроса в базу данных

Схема работы системы в режиме добавления вопроса в базу данных:

1. Считывание текстового представления вопроса, введенного пользователем.
2. Создание объекта *Question* с полем *name*, равным текстовому представлению вопроса.
3. Сохранение объекта в базе данных.

4.2.3 Режим добавления фильма в базу данных

Схема работы системы в режиме добавления фильма в базу данных аналогична работе в режиме добавления вопроса в базу данных. Она состоит из следующих пунктов:

1. Считывание текстового представления фильма, введенного пользователем.
2. Создание объекта *Film* с полем *name*, равным текстовому представлению фильма.
3. Сохранение объекта в базе данных.

4.3 Общая архитектура системы

На рисунке 3 представлена диаграмма классов, описывающая общую архитектуру системы.

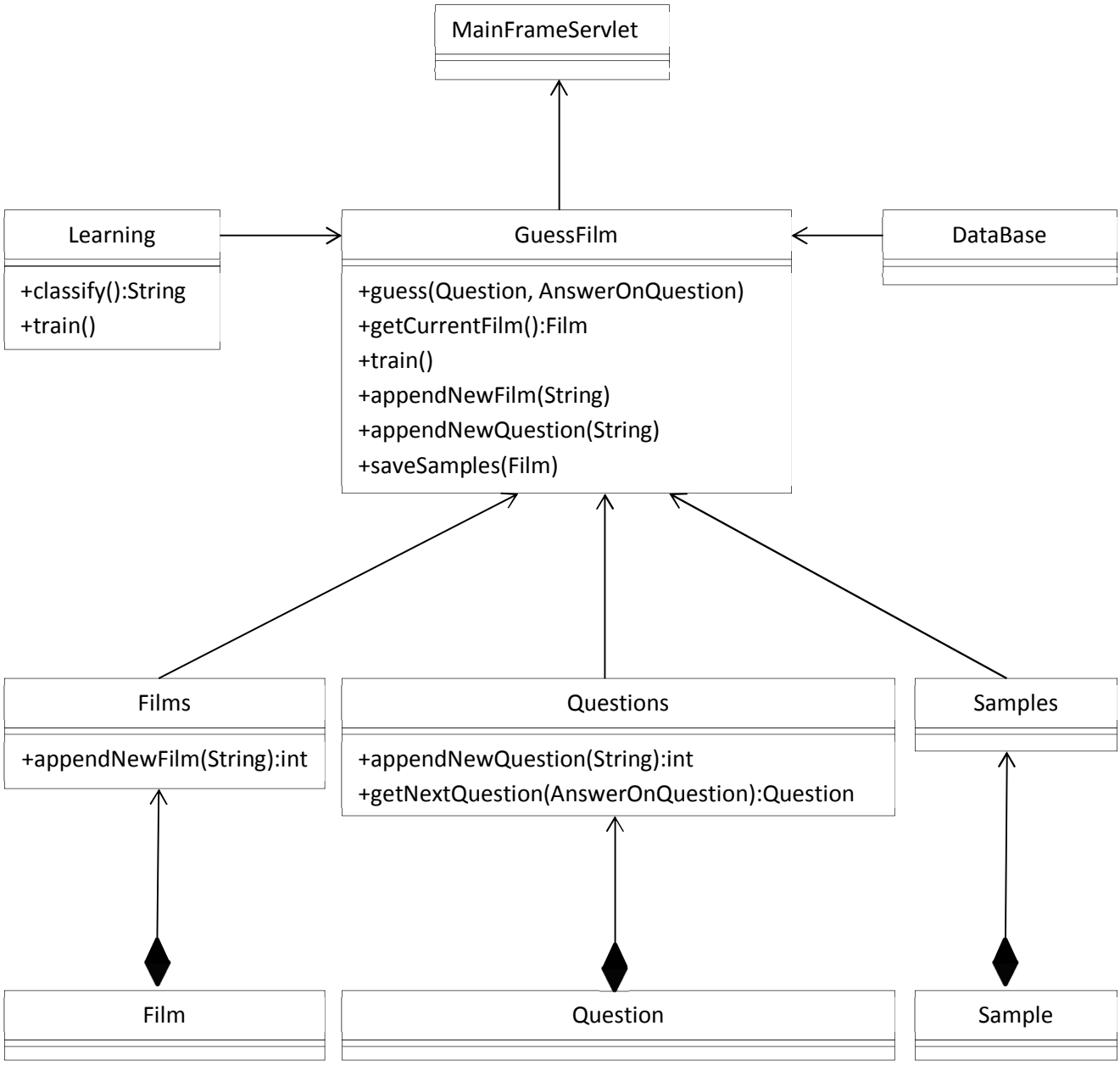

Рис.3: Общая архитектура системы

AnswerOnQuestion – перечислимый тип. Используется для хранения ответа пользователя на заданный ему вопрос. Возможные значения: «YES» (положительный ответ), «NO» (отрицательный ответ), «DO_NOT_KNOW» (ответ неизвестен).

Класс *GuessFlm* является основным классом, координирующим работу системы. Он содержит экземпляры классов *Learning*, *DataBase*, *Questions*, *Films*, *Samples*. Данный класс определяет режим, в котором должна работать система, и передает управление классу, отвечающему за соответствующий режим. Метод *guess(Question, AnswerOnQuestion)* сохраняет информацию о том, какой ответ (аргумент типа *AnswerOnQuestion*) пользователь дал на заданный ему вопрос (аргумент типа *Question*). Метод *getCurrentFilm():Film* возвращает результат классификации. Метод *train()* производит обучение системы на имеющихся данных. Метод *appendNewFilm(String)* предназначен для добавления фильма в базу данных. Метод *appendNewQuestion(String)* используется для добавления вопроса в базу данных. Метод *saveSamples(Film)* предназначен для сохранения ответа пользователя на вопрос, касающийся определенного фильма.

Класс *MainFrameServlet* является посредником между пользователем и классом *GuessFilm.* Он передает данные, полученные от пользователя, основному классу системы.

Класс *Learning* предназначен для работы системы с библиотекой машинного обучения WEKA. Данный класс содержит методы *classify():String* для классификации и *train()* для обучения.

Класс *DataBase* предназначен для работы системы с реляционной базой данных. Данный класс использует библиотеку Hibernate. В нем существуют методы, позволяющие работать с сущностями *Question*, *Film*, *Sample*. Реализованы методы добавления сущности в базу данных, удаление сущности из базы данных, поиск и извлечение сущностей из базы данных.

Класс *Film* содержит информацию о фильме. Хранится его идентификационный номер в базе данных, текстовое представление названия фильма и количество запусков системы в основном режиме, в которых данный фильм был искомым.

Класс *Films* содержит список классов *Film*, соответствующих всем фильмам, хранящимся в базе данных. Метод *appendNewFilm(String):int* позволяет добавлять новый фильм, передавая в качестве аргумента текстовое представление названия фильма.

Класс *Question* содержит информацию о вопросе. Хранится его идентификационный номер в базе данных и текстовое представление вопроса.

Класс *Questions* содержит список классов *Question*, соответствующих всем вопросам, хранящимся в базе данных. Метод *appendNewQuestion(String):int* позволяет добавлять новый вопрос, передавая в качестве аргумента его текстовое представление. Метод *getNextQuestion(AnswerOnQuestion):Question* возвращает экземпляр класса *Question*, соответствующий вопросу, который необходимо задать пользователю в соответствии с

алгоритмом, описанном в главе 3. Аргумент типа *AnswerOnQuestion* хранит ответ пользователя на предыдущий вопрос.

Класс *Sample* содержит информацию об известном ответе пользователя на вопрос. Эта информация необходима для обучения системы. Хранится идентификационный номер в базе данных, идентификационный номер заданного вопроса, идентификационный номер фильма, на вопрос о котором был получен ответ, и, собственно, ответ.

Класс *Samples* содержит список классов *Sample*. Данный класс хранит всю имеющуюся выборку.

4.4 Характеристики функционирования

// Написать после тестирования

Заключение

В рамках данной курсовой работы были получены следующие результаты:

1. Исследованы существующие методы определения кинокартины, использующие информацию о сюжете.
2. Разработан метод определения кинокартины, использующий информацию о сюжете.
3. Создан прототип, реализующий данный метод.
4. Проведено тестирование, подтверждающее работоспособность.

Список литературы