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

Курсовая работа

Исследование и разработка методов построения автоматически обучаемой экспертной системы для поиска кинокартин с автоматической генерацией вопросов

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

Экспертная система – компьютерная система, способная частично заменить специалиста-эксперта в разрешении проблемной ситуации. Фундаментом классической экспертной системы является база знаний – особого рода база данных, разработанная для оперирования знаниями (метаданными). Она составляется на основе экспертных знаний специалистов. Экспертные знания – это сочетание теоретического понимания проблемы и практических навыков ее решения, эффективность которых доказана в результате практической деятельности экспертов в данной области.

Недостатком классической экспертной системы является тот факт, что во многих случаях даже объединение экспертных знаний нескольких специалистов становится недостаточным для эффективного функционирования системы. Для решения данной проблемы были разработаны самообучающиеся экспертные системы.

Преимущество самообучающихся экспертных систем является способность сохранять и обрабатывать знания, полученные от пользователя, тем самым реагируя на изменения в предметной области.

Далее в данной работе будут подробно рассмотрены самообучающиеся экспертные системы «Акинатор»[1] и «20q»[2].

Недостатком самообучающейся экспертной системы является необходимость в информации от достаточно большого количества пользователей для эффективного функционирования. Особенно проблематично найти пользователей на первых этапах жизненного цикла, когда система еще недостаточно эффективно работает. Создание же автоматически обучаемой экспертной системы должно решить данную проблему.

Кроме того, в существующих системах генерация вопросов и объектов поиска происходит вручную. Это трудоемкий процесс, не гарантирующий оптимальный результат. С помощью автоматической генерации можно не только ускорить процесс создания вопросов и объектов, но и поддерживать их актуальность.

Для исследования и разработки экспертной системы была выбрана область кинематография. Эта область является неотъемлемой частью жизни человека. И с каждым

[1] http://ru.akinator.com/
[2] http://www.20q.net/

годом ее важность все увеличивается и увеличивается. Разработанная система должна решать задачу поиска кинокартины: задавать пользователю вопросы о фильме (в том числе и о сюжете), получать от него ответ и, обрабатывая имеющуюся информацию, выдавать предполагаемое название кинофильма.

Важная особенность данной предметной области заключается в сложности обработки сюжета. Сюжет хранит в себе основную, самую запоминающуюся информацию о фильме, но в открытом доступе не существует подробных описаний сюжетов кинокартин. Также при разработке системы необходимо учитывать тот факт, что кинематограф является объемной, динамично развивающейся отраслью: на сегодняшний день существует уже свыше 300000 полнометражных фильмов (по данным IMDb[3]), и их количество постоянно увеличивается. Поэтому система должна эффективно обрабатывать большой объем информации и предоставлять возможность пополнения базы данных фильмами.

Целью курсовой работы является исследование и разработка методов построения автоматически обучаемой экспертной системы для поиска кинокартин с автоматической генерацией вопросов.

[3] http://www.imdb.com/

1 Постановка задачи

Целью данной работы является исследование и разработка методов построения автоматически обучаемой экспертной системы для поиска кинокартин с автоматической генерацией вопросов.

Для достижения поставленной цели необходимо:

- Исследовать существующие решения и качество их работы.
- Разработать систему, агрегирующую описания кинокартин способом, позволяющим затем эффективно осуществлять их поиск.
- Подготовить тестовый набор для проверки качества работы системы.

Требования к системе:

- Автоматическое обучение системы.
- Автоматическая генерация вопросов.
- Возможность пополнения базы фильмов.
- Интерактивная работа с пользователем. Время отклика не более 10 секунд.
- Минимизация количества заданных вопросов.

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»[4] 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.

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

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.

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 Выводы

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 неформальные русскоязычные запросы на специализированных форумах, в которых пользователи искали фильмы, предоставляя всю известную информацию о них. Были рассмотрены 100 запросов.

Результаты исследования:

- во всех 100 запросах пользователи предоставляли информацию о сюжете;
- в 69 запросах был упомянут жанр фильма;
- 24 раза было сказано о дате просмотра;
- 21 раз фигурировала страна, в которой был снят фильм (в 8 из них пользователь обобщал информацию, писав, что фильм иностранный);
- 5 раз пользователь упоминал о каких-то деталях названия кинокартины (название состоит из одного слова, в названии фигурирует имя главного героя и т.д.);
- 8 раз пользователь называл актеров, игравших в фильме;
- ни разу пользователь не предоставлял информацию об именах героях и длительности фильма;
- когда пользователю задавали уточняющие вопросы по фильму, он вспоминал новую информацию.

Используя информацию, полученную в результате исследования, были сделаны выводы о том, какие вопросы необходимо генерировать для более качественной работы системы. В соответствии с результатами исследования, пользователь с большей вероятностью ответит на вопрос о сюжете, чем на вопрос об имени главного героя или длительности фильма.

В качестве ответа на заданный вопрос пользователь может выбрать один из трех вариантов ответа: «да», «нет», «не знаю».

Для полного решения поставленной задачи, необходимо решить следующие подзадачи

- Заполнить базу данных существующими фильмами.
- Сгенерировать вопросы, с помощью которых можно будет узнавать информацию о фильме у пользователя.
- Определять вопрос, который будет задан пользователю на очередной итерации.

- Определять наиболее вероятный фильм, соответствующий ответам на вопросы.

3.1 Заполнение базы данных фильмами

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

Для получения названия фильма, к web-странице, соответствующей фильму, применялось регулярное выражение, с помощью которого извлекалась необходимая информация. Полученное название сохранялось в базу данных.

3.2 Генерация вопросов

Для автоматической генерации вопросов также использовались данные с вышеназванного Интернет-ресурса.

Каждому фильму на данном сайте соответствует web-страница, содержащая следующую общую информацию (метаданные):

- название;
- год создания;
- страна, в которой фильм снимался;
- режиссер;
- жанр;
- бюджет;
- снимавшиеся актеры.

Это неполный список имеющейся информации по каждому фильму. Чтобы система смогла узнать общую информацию о фильме от пользователя, были сгенерированы вопросы, с помощью которых можно узнать метаданные фильма.

Пример вопросов:

- «Является ли фильм комедией?»
- «Снимался ли Юрий Никулин в данном фильме?»
- «Фильм был снят в России?»

Из результатов проведенных исследований были сделаны выводы, что данные о сюжете являются основными данными, которыми располагает пользователь. Для того чтобы система смогла их использовать, необходимо создать вопросы, целью которых является получение информации о сюжете. Для генерации подобных вопросов использовались теги (слов или словосочетания), описывающие детали сюжета фильма. Для каждого фильма на сайте существует web-страница, на которой перечислены теги, связанные с сюжетом. Один и тот же тег может описывать сюжеты нескольких фильмов.

Для каждого существующего тега был сгенерирован вопрос, с помощью которого система имеет возможность узнать, связан ли данный тег с фильмом. Для извлечения тегов с web-страниц использовались регулярные выражения.

При генерации вопросов применялся шаблон: «Присутствует ли в сюжете данного фильма [тег]?»

Примеры вопросов:

- Присутствует ли в сюжете данного фильма оружие?
- Присутствует ли в сюжете данного фильма похищение людей?

3.3 Определение результата

Для определения наиболее подходящего фильма по уже имеющимся ответам использовался метод машинного обучения (классификация). В качестве классификатора был выбран наивный байесовский классификатор[].

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 возможное дообучение.

В качестве признаков, передающихся классификатору, выступали ответы пользователя на вопросы. В ходе классификации не учитывались ответы «не знаю». Это было сделано для того, чтобы отбросить возможный шум и ошибки классификации.

Для создания обучающейся выборки использовались данные, полученные в процессе извлечения метаданных и тегов, связанных с фильмом. Она состояла из положительных ответов на такие вопросы, которые описывают тег, связанный с фильмом. Также

положительные ответы соответствовали вопросам, которые описывают метаданные этого фильма. Все остальные вопросы имели отрицательный ответ.

Кроме того, в ходе функционирования программа регулярно дообучается на ответах пользователей.

Временная сложность пересчета результаты после каждого полученного ответа от пользователя составляет $O(|F|)$ времени, где $|F|$ – количество фильмов в базе данных.

3.4 Определение очередного вопроса

Для минимизации количества вопросов, заданных пользователю, был использован метод максимальной энтропии (Principle of maximum entropy)[]. Этот метод позволяет каждый раз задавать вопрос, ответ на который максимально уменьшает энтропию распределения вероятности фильма при известных ответах на вопросы. Таким образом, каждый заданный вопрос будет максимально устранять неопределенность. Данный метод использует вероятностный подход.

Временная сложность выбора вопроса составляет $O(|F| * |Q|)$, где $|F|$ – количество фильмов в базе данных, $|Q|$ – количество вопросов в базе данных.

3.5 Результаты тестирования

// Написать после тестирования

4 Описание практической части

4.1 Обоснование выбранного инструментария

Был разработан прототип на языке программирования Java. 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[]. 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. 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.
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.

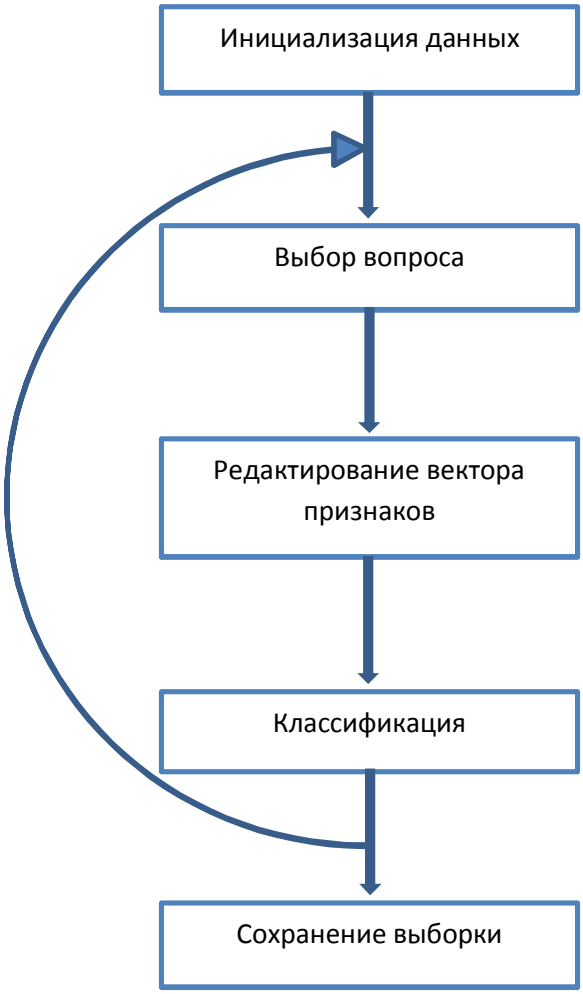

Рис. 1: Схема работы системы в основном режиме

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.

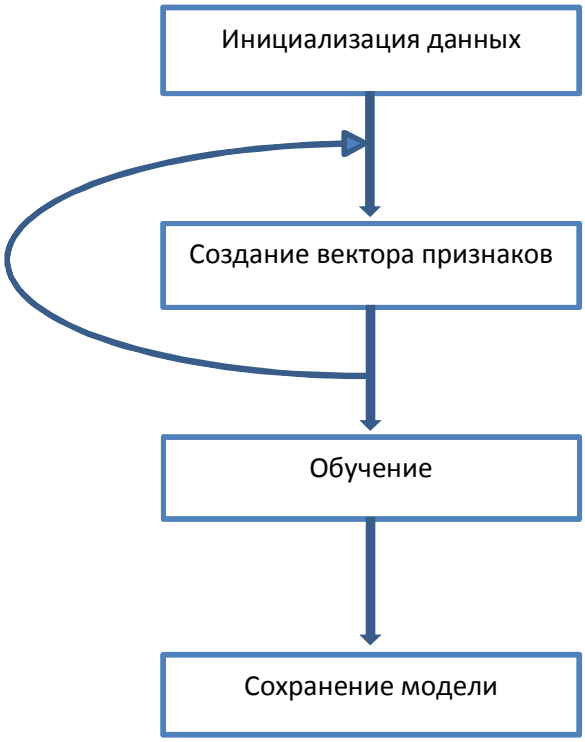

Рис. 2: Схема работы системы в тренировочном режиме

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, описывающая общую архитектуру системы.

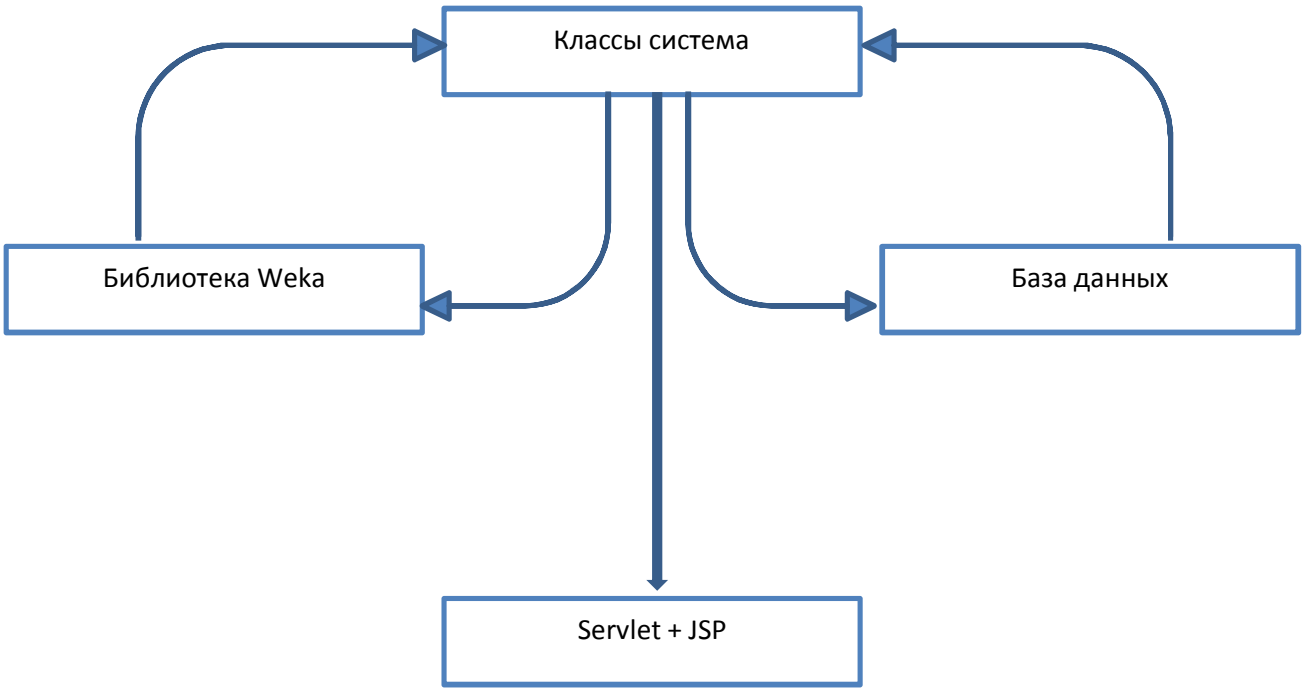

Рис.3: Общая архитектура системы.

На рисунке 4 представлена диаграмма классов системы.

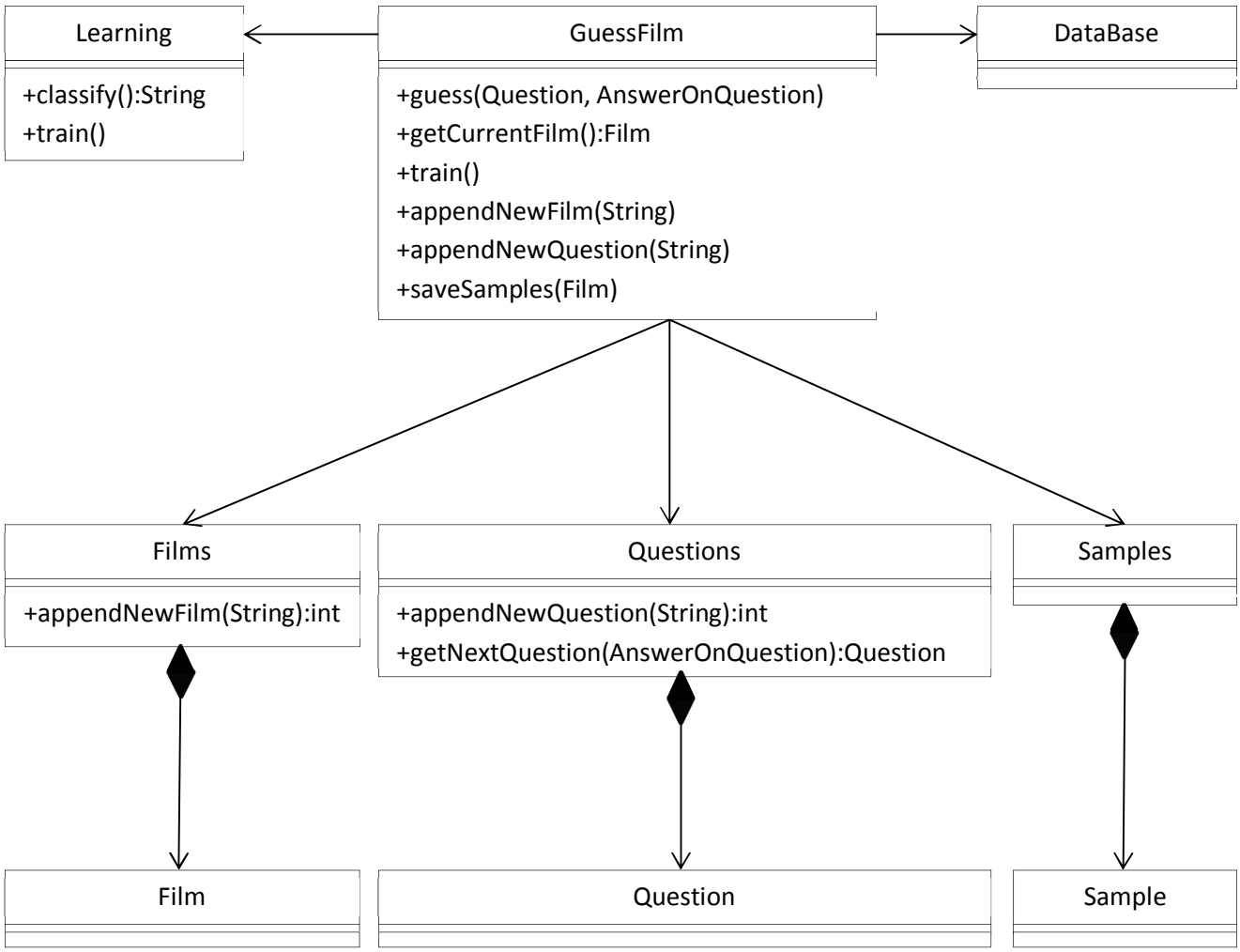

Рис. 4: Диаграмма классов системы.

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 *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