Ebooks searching, reading, managing system

Тузова Екатерина

11 мая 2010 г.

Аннотация

Мотивацией для данного проекта послужило желание облегчить работу пользователей с электронными книгами. Ресурсы, которые сейчас есть в интеренете, позволяют пользователю последовательно сначала находить нужную книгу, затем самостоятельно скачивать и искать программу, которая может работать с файлом в заданном формате.

Основная идея проекта - избавить пользователя от лишних и трудоемких действий, оставив ему самую приятную часть - непосредственно чтение книги. Таким образом, проект призван объединить все стадии подготовки к чтению(поиск, скачивание, открытие файла) в одну.

Содержание

Введение

В последние годы влияние Интернета на жизнь человека становится все сильнее. Это влияние не обошло стороной и такую важную часть повседневной жизни как чтение книг. Если 20 лет назад человек хотел почитать книгу, он шел в библиотеку или в книжный магазин. Если он хочел прочитать определённую книгу - и достаточно было назвать фамилию автора и название книги, а порой и названия было достаточно. Если же человек не знал точно какая книга ему нужна, он мог получить помощь от сотрудника магазина или библиотеки. Сегодня ситуация с библиотеками и книжными магазинами изменилась несильно, хотя количество написанных книг растет с каждым годом и библиотеки переходят на автоматизированный поиск.

Наравне с бумажными книгами появилась возможность читать книги в электронном виде. Большое количество книг оцифровано и хранится в сети Интернет, многие из них в свободном доступе. С одной стороны это должно упрощать процесс поиска книг, не нужно выходить из дома для того, чтобы получить интересующую книгу. С другой стороны, перед пользователем встает новая проблема. Если раньше для того, чтобы прочитать книгу достаточно было пойти в библиотеку и сказать продавцу название книги, то теперь ему необходимо самому искать эту книгу в сети, где количество информации растет с каждым днем. Но на этом трудности пользователя только начинаются. После того, как он нашел нужную книгу - ее нужно скачать, а затем найти программу, которая может работать с файлом в найденном формате.

На настоящий момент в Интернете существует множество электронных библиотек, поиск по которым весьма затруднен из-за того, что не существует системы, которая могла бы объединить всю информацию с этих библиотек. Поэтому пользователям приходится либо искать в каждой из электронных библиотек в отдельности, либо пользоваться существующими поисковыми системами. Вторая проблема, связанная с наличием большого количества электронных библиотек в том, что единый формат для предоставления информации о книгах появился только недавно. Многие библиотеки были вынуждены выдумывать свой формат.

У существующих поисковых систем есть одна очень важная проблема - они обладают ограниченными возможностями в проведении специализированного поиска в сети. Поэтому при поиске пользователь получает всю информацию, которая соответствует поисковому запросу. И дальше уже начинается работа самого пользователя над тем, чтобы эту информацию профильтровать и извлечь оттуда именно то, что ему нужно. Для большей части поисковых запросов данный метод вполне удовлетворителен, так как существует большая вероятность, что на первых нескольких страницах результатов поиска пользователь найдет ту информацию, которая его интересует. Но существует определенная часть запросов, для которых данная система не работает. К ним относится и поиск книг.

В данной работе была разработана система, упрощающая задачу поиска, чтения и управления книгами.

Обзор сущесвующих решений

Для каждой из описанных выше проблем существуют готовые решения.

Например, ebdb.ru(electronics books data base) - это поисковая система, которая обходит интернет и сохраняет у себя ссылки на книги с других ресурсов. Это решает проблему поиска книг, но остаются следующие задачи, которые пользователь вынужден выполнять сам. У этой поисковой системы есть свои минусы. Во-первых, поиск производится в основном в русском сегменте Интернета, плохо реализована интернационализация. Во-вторых, эта система предоставляет информацию только в HTML виде, для скачивания книг необходимо переходить по внешним ссылкам.

Как уже было упомянуто выше, уже существует единый протокол предоставления информации. Это решает проблему различных интерфейсов. Так же сейчас активно развиваются различные клиетские программы, которым нужен единый протокол для получения данных. Большинство таких систем используют OPDS.

Есть системы. решающие сразу все 3 проблемы, а именно, Amazon Kindle, Sony Reader Но эти системы платные, зависимы от девайса и количество доступных через них книг - ограничено теми книгами, которые они хранят/продают.

Описание системы в целом

Проект состоит из серверной и клиентской частей.

Серверная часть собирает в сети Интернет информацию об электронных книгах и представляет собранную информацию для обычных пользователей в виде web-интерфейса и для клиентских программ – в формате OPDS.

Клиентская часть представляет собой программу для удобной работы с сервером, предоставляющим по запросам информацию в формате OPDS. Клиент должен работать как с "родным"сервером, так и с другими серверами, поддерживающими OPDS, например feedbooks.com.

Серверная часть проекта состоит из 3 подпроектов:

- собственно web-сервер, включающий базу данных, поиск по ней, представление информации в web и opds форматах (python, django)
- сборщик информации в сети (crawler) (java)
- анализатор найденной информации, разбирающий информацию о книгах (java)

Клиентских программ написано 2:

- На C++ (с интерфейсом Qt)
- На Java

Клиентская программа на Java более переносима, но для нее необходимо на девайсе иметь java-машину. В свою очередь программа на Qt не требует установки никаких дополнительных библиотек, работает быстрее, но переносимость ниже, чем у java.

Описание компонентов системы и их взаимодействие

Пользователь устанавливает у себя на машине одну из клиентских программ. При поиске книги программа обращается к одному из серверов, поддерживающих протокол OPDS,(OpenSearch?). Сервер обрабатывает запрос, и возвращает данные в в нужном формате. Счастливый пользователь может читать книгу.

Постановка задачи

После того, как книги найдены и добавлены в базу данных появляются новые задачи - это предоставление информации конечному пользователю, а так же верификация и обновление данных. С предоставлением информации пользователю все более-менее ясно. Базу данных нужно все время поддерживать в рабочем состоянии. А информация, поступающая от crawler'а с analyzer'ом зачастую бывает весьма сомнительного качества. Иногда, crawler находит книги, для которых analyzer не может определить автора/название. Тогда эта книга не добавляется в базу, а добавляется только ссылка на файл. Позже мы пытаемся извлечь нужную информацию из самого файла.

Для книг, которые уже находятся в базе происходит поиск дополнительной информации, такой как аннотация, теги и пр.

Структурирование информации

Задача определения жанра книги

Постановка задачи:

У нас есть база данных, заполненная собранной в интернете информацией о книгах. В информацию о книге входят - название, автор, описание, жанр и некоторые другие поля. Обязательными для книги являются поле названия книги и автор (ManyToMany). Для некоторых книг в найденной информации уже указан жанр, для других - нет. Для более удобной навигации пользователя по базе - хочется предоставить возможность различных выборок, в том числе по жанру. Т.к. не у всех книг есть информация о жанре - хочется определить жанр, исходя из имеющейся информации.

Решение:

Для определения жанра книги создан классификатор, построенный на основе наивного байесовского классификатора с использованием метода Фишера для определения вероятности для всего документа. В отличие от наивной байесовсой фильтрации, когда для вычисления вероятности всего документа перемножаются вероятности отдельных признаков, по методу Фишера вычисляется вероятность отнесения к той или иной категории для каждого признака документа, после чего эти вероятности комбинируются и проверяется, насколько получившееся множество похоже на случайное.

В качестве входа классификатор принимает текст. Этот текст разбивается на слова, которые проходят через Stemmer. В итоговый вектор добавляются все слова и пары слов, встреченные в тексте.

Текст - это название книги + ее описание. Если описания в нашей базе нет - ищем описание на Amazone.

Была попытка классифицировать книгу по ее содержимому, если нет описания. Пока что попытка считается проваленной, т.к. во-первых, это оказалось слишком долго, а во-вторых, плохой процент угадывания жанра.

Сейчас классификатор обучается на http://feedbooks.com/ (так же можем обучаться на http://www.smashwords.com/ и http://www.allromanceebooks.com/ , но набор книг на feedbooks наиболее близок к нашему)

Полученный классификатор умеет правильно классифицировать 90% книг(если мы обучались на части книг с сайта feedbooks, а затем классифицировали оставшиеся книги этого сайта).

Классификатор решает еще одну задачу - он пополняет базу дополнительной информацией о книгах.

Информация из формата epub и fb2

Кастомизация админки

В django есть встроенный интерфейс администратора. Этот интерфейс очень удобен, но подходит для решения не всех задач.

Первая задача, которую мы решили - отображение полей типа ManyToMany. В интерфейсе, предоставляемом django - для отображения таких полей есть несколько возможностей:

- Select
- Filter horizontal
- Raw id

Первые два варианта - загружают список всех возможных выборов целиком, что нам не подходит, т.к. загружать список, состоящий из 60 тыс. итемов слишком долго. Третий вариант уже больше похож на то, что можно использовать, но у него есть один недостаток - отображаются в текстовом поле id элементов, связанных с данным. К несчастью, очень маловероятно, что человеку (администратору) будет удобно оперировать в терминах id объектов.

———— Задача

Создать удобный виджет для отображения авторов, связанных с конкретной книгой. Хочется видеть их в формате нескольких checkbox'ов, с возможностью добавлять новые. ————————

При создании интерфейса администратора мы первым делом регистрируем модель. При регистрации есть возможность указать форму, которая будет использоваться при отображении. Так же для поля m2m указываем способ отображения - raw id.

Далее, создадим свою форму, которая наследуется от forms.ModelForm. Внутри формы для каждого из полей модели можно специфицировать используемый виджет. Для нашего m2m мы создадим собственный виджет. Для того, чтобы сохранить нужное поведение мы наследуемся от forms.CheckboxSelectMultiple, ManyToManyRawIdWidget. Осталось переопределить метод render, который генерирует начинку нашего виджета. Внутри этого метода, которому передаются id - нужно обратиться к базе и по id узнать имена авторов. Затем вызвать метод render от checkbox, которому передать список пар (author id, author name).

Теперь мы умеем отображать имеющуюся информацию в удобном нам виде. Но если мы попробуем сейчас сохранить нашу книгу - ничего не выйдет. Это произошло от того, что внутри стандартного метода save используется метод cleaned_ data, который извлекает информацию из форм, затем происходит проверка - изменилась ли эта информация и сохранение. Метод cleaned_ data из checkbox извлекает не только id. Значит нам нужно руками вытащить id и сохранить в словарь. Теперь наша форма умеет правильно сохраняться в базу.

Осталось добавить возможность добавления авторов. Это делается уже с помощью javascript.