Правительство Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования

«Национальный исследовательский университет   
«Высшая школа экономики»

Факультет Бизнес-информатика

Отделение Программной инженерии

Кафедра Управление разработкой программного обеспечения

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

На тему: Программа персонализации новостной подписки

|  |
| --- |
| Студент группы №472ПИ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Гриднев Д. В. /  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2014 г. |
| Научный руководитель ВКР  доцент каф. УРПО, к.т.н.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Брейман А. Д./  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2014 г. |

Москва, 2014

Table of Contents

Аннотация 3

Введение 3

Цели и задачи 3

Актуальность 4

Обзор существующих методов 5

Коллаборативная фильтрация 5

Weighted Slope One 8

Модель Байеса 9

Кластерная модель 10

Факторизация матриц 10

Проблемы существующих подходов 11

Выводы 12

Описание реализации системы 13

Архитектура программы 13

Серверная часть 13

Описание файлов 15

Описание работы серверной части 16

Клиентская часть 19

Описание работы клиентской части 19

Описание файлов 23

Взаимодействие между клиентом и сервером 24

Средства разработки 25

Анализ результатов 25

Тестирование 25

Дальнейшее развитие проекта 26

Заключение 28

Список использованных источников 29

# Аннотация

Данная работа посвящена изучению проблемы персонализации информации, изучению существующих подходов для создания рекомендательных систем, анализу и сравнению алгоритмов персонализации, а также реализации клиент-серверной программы для создания персонализированной ленты новостей (подписки) .

В первой части работы будет проведен анализ существующих решений и подходов персонализации – коллаборативной фильтрации на основе модели и соседстве, метод факторизации матриц и гибридные решения. С учетом специфики конкретной предметной области (новости) будет представлен способ создания рейтинговой системы. Также будет рассмотрен способ выделения ключевых слов из текста новости и создания на его основе вектора “интересов” пользователя.

Вторая часть работы освещает конкретную реализацию различных алгоритмов персонализации, архитектуру и технические особенности клиент-серверной системы, основанной на некоторых выборочных алгоритмов и подходах. В качестве результата, данная программа будет создавать и выводить рекомендационные материалы в виде новостей на клиентской части.

# Введение

## Цели и задачи

Основной целью данной работы является разработка клиент-серверной системы для создания рекомендационных материалов в виде новостных статей. Данная программа подразумевает реализацию клиентской части, через которую пользователь будет просматривать новости и получать готовые рекомендации, и серверной части, которая обрабатывает информацию, приходящую с клиентской части, сохраняет статистику пользователей, выполняет рекомендационные алгоритмы и формирует результат в виде рекомендаций новых новостей.

Для работы рекомендательной системы необходимы данные о пользователях и об объектах, которые будут в последствии предложены. В данном случае это новости, которые берутся с крупного российского новостного ресурса. Одной из важнейших задач является обработка данных о взаимодействии пользователей с новостными элементами таким образом, чтобы эти данные можно было использовать в рекомендационных алгоритмах.  
Также важной задачей является изучение и выбор подходящих способов для создания предсказаний из числа методов коллаборативной фильтрации разных типов, оценки содержимого новости, а также методов на основе факторизации матриц. Каждый из этих подходов обладает своими достоинствами и недостатками, и немаловажной задачей является выбор наиболее подходящих для этого случая методов для создания гибридного решения.

## Актуальность

В современном обществе информация является важным и крайне ценным ресурсом. Наше общество называется информационным и информация в таком обществе занимает главенствующую роль. Но ее объемы растут гораздо быстрее, чем наша способность ее обрабатывать, а также отделять необходимое и интересное от того, что нам вовсе не нужно. Количество информации в интернете настолько велико, что люди просто теряются в таких объемах и не могут получить то, что им действительно нужно. Подобное утверждение в высшей степени верно для новостных агентств и средств массовой информации, вся деятельность которых построена на продаже новостей. В мире каждый день происходит множество событий, поэтому бывает действительно сложно следить за тем, что тебя интересует.

В такой ситуации напрашивающимся решением является создание специальной системы, которая будет показывать те новости, которые могут заинтересовать пользователя. Подобная система будет предлагать новости основываясь на действиях пользователя, на тематике просмотренных новостей и его поведении во время чтения. Это позволит пользователям не тратить время на поиск необходимых новостей, а СМИ – предлагать более персонализированный контент своим клиентам.

Рекомендационными системами обладают почти все крупные сервисы, такие как Amazon, Facebook, Netflix, YouTube. В каждом из них существует своя специфика, которая напрямую зависит от типа данных, которые требуется рекомендовать. Их цель – прогнозировать те предметы, которые могут быть интересны пользователю.

# Обзор существующих методов

Для создания рекомендационных систем существует несколько методов, которые в основном базируются на коллаборативной фильтрации:

* Фильтрация, основанная на схожести пользователей;
* Фильтрация, основанная на схожести объектов (предметов);
* Фильтрация, основанная на модели;
  + Модель Байеса;
  + Регрессионная модель;
  + Кластерная модель;

Также достаточно распространены методы, основанные на факторизации

* Неотрицательная матричная факторизация (NMF);
* Сингулярное разложение (SVD);

## Коллаборативная фильтрация

Коллаборативная фильтрация – это метод, позволяющий предсказать неизвестные предпочтения пользователя на основе известных оценок и/или поведения других пользователей (Segaran 2007). Работа всех видов данного метода основывается на утверждении о том, что пользователи, одинаково оценившие предметы системы, имеют склонность одинаково оценить и другие предметы системы. Также одним из допущений является тот факт, что пользователи дают оценку предмету системы по выборочной шкале, например, оценивая фильм от одной до десяти звезд на сайте imdb.com (рис. 1). Данное допущение является достаточно важным, так как не во всех системах есть возможность явно собирать оценки пользователей. В таких случаях прибегают к неявному сбору информации и оценке поведения, например, записывая просмотренные ролики на YouTube и рекомендуя связанные с ними материалы.



*Рисунок 1. Пример оценки фильма на сайте imdb.com*

Коллаборативная фильтрация традиционно делится на два подхода. Первым, и самый распространенным, является подход, основанный на соседстве (сходстве) пользователей. Его суть заключается в анализе предыдущих оценок или поведения пользователя, поиск других пользователей, имеющих схожую “историю” и вычисление прогноза для неизвестных оценок.[3]

В классическом случае строится матрица пользователей-предметов, значения в которой является оценками конкретного пользователя конкретного предмета. Те ячейки, в которых нет значений, являются неизвестными, то есть для данного предмета пользователь не выставил оценку, т.е. не пользовался (табл. 1).

*Таблица 1*

*Пример матрицы пользователей-предметов*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | “Даллаский клуб покупателей” | “Человек-паук” | “Волк с Уолл-стрит” | “12 лет рабства” | “Free-to-play” |
| Коля | 5 | 3 | 5 | - | - |
| Петя | 4 | - | 4 | - | 5 |
| Ипполит | - | 1 | 5 | 2 | - |

Для определения соседства пользователей применяются несколько различных алгоритмов, таких как:

* Манхэттонское расстояние;
* Евклидово расстояние;
* Коэффициент корреляции Пирсона;

Манхэттонское расстояние или расстояние городских кварталов является одним из базовых метод вычисления расстояния между двумя точками (1):

*где*

*user1, user2 – пользователи и их оценки;*

*n – количество предметов в матрице*

Данное способ имеет недостаток в точности при малом заполнении матрицы, но также имеет простую реализацию и высокую скорость выполнения.

Евклидово расстояние имеет геометрические корни (теорема Пифагора) и вычисляется при помощи следующей формулы (2):

*где*

*user1, user2 – пользователи и их оценки;*

*n – количество предметов в матрице*

Как и расстояние городских кварталов, данный метод имеет проблемы при незаполненной матрицей, но прост в разработке и дешев в выполнении.

Более точный способ определения соседства основан на коэффициенте корреляции Пирсона (3):

*где*

*user1, user2 – пользователи и их оценки;*

*n – количество предметов в матрице;*

Значение *corr(user1, user2)* может быть от -1 до 1, где -1 соответствует абсолютному несовпадению пользователей, а 1 – абсолютному совпадению.

Алгоритмы, основанные на сходстве пользователей, могут быть достаточно полезны – они интуитивно понятны и просты в реализации.

## Weighted Slope One

Одним из самых эффективных алгоритмов в предметно-ориентированной коллаборативной фильтрации является алгоритм Weighted Slope One. Его суть заключается в поиске различий оценок между парами элементов и использовании этих различий для вычисления предсказаний[1]. Вычисление различий между элементами выполняется с помощью следующей формулы (4):

(4)

*где*

*– число пользователей, оценивших и i-й, и j-й элемент;*

*user – оценки пользователя;*

Само предсказание для оценки предмета вычисляется следующим образом (5):

(5)

*где*

*– число пользователей, оценивших и i-й, и j-й элемент;*

*user – оценки пользователя;*

Weighted Slope One является отличным методом построения рекомендательных систем. Имея низкие требования к памяти и большую скорость работы, он показывает большую эффективность при наличии большого числа пользователей. Однако, очевидной проблемой подобного подхода является проблема холодного старта, которая, впрочем, относится ко всем предметно-ориентированным алгоритмам коллаборативной фильтрации.  
Семейство алгоритмов Scope One используется в некоторых известных сервисах, таких как hitflip, сайт рекомендаций DVD и Value Investing News, новостной сайт фондовых бирж.

Вторым крупным видом коллаборативной фильтрации является фильтрация, основанная на модели. Рассмотрим некоторые из них.

## Модель Байеса

Одним из самых известных классификаторов является наивный байесовский классификатор. С его помощью делают рекомендации на категории каких-либо объектов. В его основе лежит вероятностная модель теоремы Байеса [6]. Для работы этого алгоритмов необходимо создать модель Байеса для каждого пользователя, который оценивал какие-либо объекты, на основе содержания этих объектов (Для фильмов это могут быть актеры или жанры, для новостей – ключевые слова и категории). Для нахождения наиболее вероятной категории необходимо вычислить условные вероятности принадлежности какого-либо предмета к каждой категории и выбрать категорию, имеющую наибольшую вероятность (6):

## Кластерная модель

Одним из самых известных алгоритмов в кластерном анализе является метод k-means или k-средних. Он основан на разделении объектов или пользователей на группы – кластеры, которые создаются по некоторым общим признакам, а количество которых задается заранее. Суть алгоритма состоит в случайном выборе k центров кластера и уменьшении суммарного квадратичного отклонения пользователей или объектов от центра кластера. Формально это вычисляется с помощью следующей формулы (7):

*где*

*k – количество векторов,*

*u – центр масс векторов из множества кластеров K*

## Факторизация матриц

Методы коллаборативной фильтрации достаточно наглядны и просты, но с помощью факторизации матриц иногда можно добиться более высоких результатов, так как этот метод позволяет выявить некоторые скрытые факторы, связующие объекты и пользователей. Но также нельзя забывать о том, что математический метод, и для получения адекватных результатов без переобучения требуется настройка данного алгоритма.

Целью неотрицательной факторизации матриц является разложение матрицы на произведение двух других матриц. В случае рекомендательной системы, исходная матрица будет являться матрицей пользователей-объектов, а значения в ячейках – оценками данных пользователей различных объектов. Так как некоторых оценок может не быть, то с помощью факторизации возможно предсказание этих отсутствующих оценок [4].

Математически это вычисляется при помощи получения квадратичной ошибки и вычисления градиента от нее и получения значения матриц P и Q: (8)(9):

*где*

*r – реальное значение исходной матрицы,*

*p и q – значения предполагаемых матриц P и Q*

*где*

*и – новые значения в матрицах P и Q*

*2e – градиент из квадратичной ошибки*

Данный алгоритм требует большого количества итераций (>5000) для получения правильных результатов.

## Проблемы существующих подходов

* *Недостаток информации о пользователях*

Большинство пользователей различных сервисов предпочитает не высказывать свое мнение относительно объектов сервиса, тем самым не ставя им оценки, из-за чего точность предсказаний ухудшается

* *Новые пользователи*

Эта проблема также является достаточно серьезной – новый пользователь не предоставляет достаточно количества информации для создания правильных рекомендация для него, в следствии чего рекомендации составляют либо из популярного в целом контента, либо не составляют вовсе.

* *Поддельные оценки*

На некоторых сервисах, особенно крупных, существует определенное количество пользователей, которые сознательно завышают или занижают оценки всем объектам системы, продвигая те или иные товары, что мешает работе рекомендационных алгоритмов

* *Масштабируемость*

При увеличении количества пользователей и объектов системы увеличивается время выполнения алгоритмов для каждого пользователя. Частично решается усовершенствованием алгоритмов (например, для алгоритма weighted slope one необходимо только следить за отклонениями между парами элементов и общим количеством элементов)

## Выводы

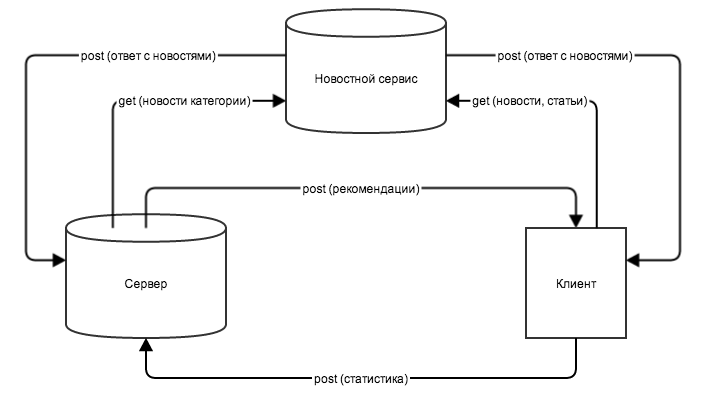
Существующие методы имеют как достоинства, так и недостатки. Одни алгоритмы выигрывают по времени, другие – по потребляемой памяти и точности, но для данной работы было решено выбрать методы, основанные на статистических вычислениях. Создание классификатора не представляется правильным решением, так как ключевые слова не всегда выражают категорию новости, а бесплатные источники новостей не предоставляют теги к новостным объектам. Как и целом оценивание каждого новостного элемента, прочитанного пользователем не является правильным решением по ресурсным соображениям.

Поэтому для реализации рекомендательной системы было решено выбрать алгоритм поиска ближайших соседей и Weighted Slope One.

# Описание реализации системы

## Архитектура программы

Так как “Программа персонализации новостной подписки” включает в себя программу-клиент и программу-сервер, то архитектура является клиент-серверной (рис. 2)

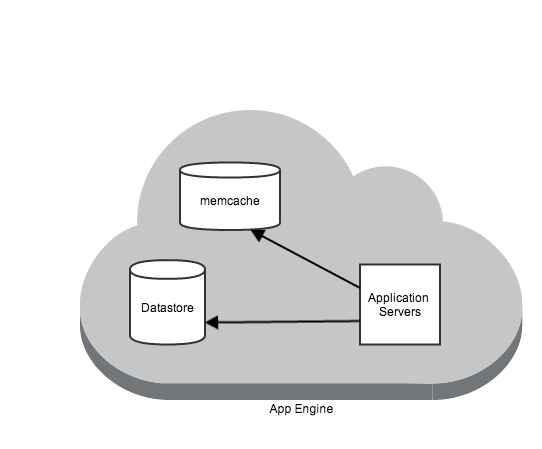


*Рисунок 2 Архитектура системы*

Рассмотрим каждый компонент системы более подробно.

## Серверная часть

Серверная часть написана на языке Python и развернута на облачном сервисе Google App Engine. Так как GAE является веб-фреймворком, это накладывает некоторые ограничения, например, необходимо использовать внутренний Datastore, который не является реляционной базой данных, но, в отличии от традиционных баз данных, имеет распределенную систему управления данными[5] (рис. 3)



*Рисунок 3 Архитектура серверной части (упрощенная)*

AppEngine задает собственную архитектуру приложения, состоящую следующих компонентов:

* *app.yaml –* файл с конфигурациями, состоящий из сопоставлений обработчиков запросов к адресам запросов, объявлений подключаемых модулей и их версий, а также персональной информацией о приложении (application id) и версией языка;
* *\*.py* – файлы в которых находится выполняемый код на языке Python. Классы и методы, объявленные в файлах, отвечают за работу серверного приложения;
* *\*.html* – необязательные файлы с разметкой страниц;

Серверная часть не имеет какого-либо интерфейса или консольного терминала; взаимодействие происходит только по средством запросов на <http://personalized-news.appspot.com/> или на <http://personalized-news.appspot.com/statistics> с определенным user-agent.

### Описание файлов

*requesthandler.py*

* class MainPage
  + def post(self) – обработка POST-запроса на адрес <http://personalized-news.appspot.com/>, создание рекомендаций и их отправка пользователю;
  + def getCategoriesFromUser(self, similarily, userInfo) – парсинг категорий из рекомендаций пользователя;
  + def getSource(self, x) – получение из категорий ссылки на новости;
  + def getNews(self, url, data, typeRecommendation, newsCategoty) - запрос на получение новостей определенной категории и формирование ответа пользователю;
* class Statistics
  + def post(self) – обработка POST-запроса на адрес <http://personalized-news.appspot.com/statistics>, обработка и сохранение статистики, пришедшей от пользователя;
  + def parseCategory(self, category) – парсинг категорий пользователя;
  + def createScore(self, scrollRating, readRating) – преобразование статистики в шкалу оценок;

*usermodel.py*

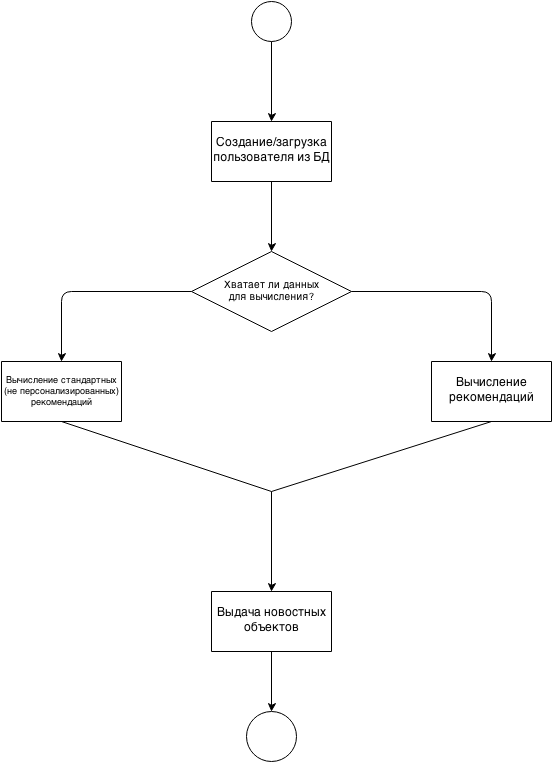
* class AppUser *–* сущность базы данных, представляет собой пользователя;

*algo.py*

* class createRecommendations
  + def getTopUsers(self,prefs,person,n=2*) – получение первых двух похожих пользователей;*
  + *def pearsonSimilarity(self,prefs,p1,p2) – вычисление коэффиециента Пирсона для двух пользователей;*
  + *def computeDeviations(self) – вычисление отклонений между объектами системы;*
  + *def weightedSlopeOne(self, userRatings) – вычисление рекомендаций методом Weighted Slope One;*

### Описание работы серверной части

Работа серверной части заключается в обработке и сохранении данных пользователей и в создании на их основе рекомендаций в виде новостей. На диаграммe деятельности в упрощенном виде показан процесс формирования рекомендаций (рис. 4).



*Рисунок 4 Диаграмма активности*

При запуске клиентской программы делается POST-запрос на сервер по адресу <http://personalized-news.appspot.com/>. В теле запроса присылается uid, который является уникальным идентификатором для каждого пользователя. Если такого пользователя в базе данным Datastore, то он будет создан (с начальными значениями для всех полей 0). При создании нового пользователя в качестве рекомендаций будут выданы более общие, так как никакой информации о новом пользователе не будет.   
Если же такой пользователь существует, то его объект будет получен из базы данных. С помощью алгоритмов коллаборативной фильтрации происходит поиск пользователя, чьи оценки на категории новостей наиболее похожи на пользователя, сделавшего запрос и исходя из его предпочтений будет создан новостной прогноз.

Далее применяется алгоритм семейства Slope One, который является алгоритмом поиска рекомендаций по схожести объектов в системе.

Результаты работы этих алгоритмов комбинируются (исключаются одинаковые категории, если такие были получены) и для полученных категорий выполняются запросы на новостной сервис. Ответ от сервиса обрабатывается, полученные новостные элементы комбинируются, сортируются по дате и отправляются на клиент тому пользователю, который сделал запрос на рекомендации.

Следующим крупным пунктом взаимодействия клиента и сервера является сбор статистики. Для того, чтобы рекомендации были более точными, необходимо иметь информацию о пользователе. В случае с новостным приложением необходимо знать, что за статьи и на какую тему читает пользователь, и насколько они ему нравятся. После каждой прочитанной страницы с клиентского приложения отправляется POST-запрос на адрес <http://personalized-news.appspot.com/statistics> с информацией о прочитанной статье. На диаграмме активности показан процесс получения статистики от клиентского приложения (рис. 5).

После получения статистики, которая состоит из уникального идентификатора пользователя, категории новостной статьи, а также двух параметров, характеризующих успешность чтения статьи (подробнее о них в описании клиентской части). На основе параметров вычисляется итоговая оценка, которую пользователь “поставил” конкретной новостной статье на какую-либо категорию. После этого данные пользователя, отправившего запрос, сохраняются с учетом оценки, которая пришла в запросе (формируется новая средняя оценка).



*Рисунок 5 Процесс получения статистики от пользователя*

Данные о пользователях хранятся в Datastore, NoSQL базе данных обращаться к которой можно с помощью SQL-образного синтаксиса.

Cхема базы данных на сервере (рис. 6):



*Рисунок 6 Схема серверной базы данных*

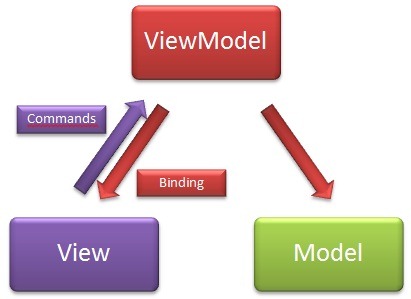
## Клиентская часть

Клиентская часть представляет собой мобильное приложение для платформы Windows Phone. С помощью него пользователь читает новости, после чего на сервер отправляется статистика о прочитанных новостях.  
Существует важное допущение о том, что пользователь во время чтения новостей не ставит оценок новости. Но наличие системы рейтинга обязательно для использования коллаборативной фильтрации, поэтому для создания рейтинговой системы оценивается поведение пользователя – время, проведенное на странице новости и факт скроллинга (пролистывания) до конца.

Приложение построено на базе архитектурного шаблона MVVM.

### Описание работы клиентской части

Клиентская часть основана на архитектурном шаблоне MVVM (Model, View, ViewModel), поэтому логически все приложение разделено на эти три части (рис. 7)



*Рисунок 7 Архитектура MVVM*

* Model – это бизнес-логика приложения, все основные данные приложения. Эквивалентно представлению Model в MVC
* View – отображение данных, пользовательский интерфейс. Эквивалентно представлению View в MVC
* ViewModel же включает в себя с одной стороны преобразованные для показа данные модели, а с другой – абстракцию View. Эта части соединяются при помощи подхода связывания данных (Binding)

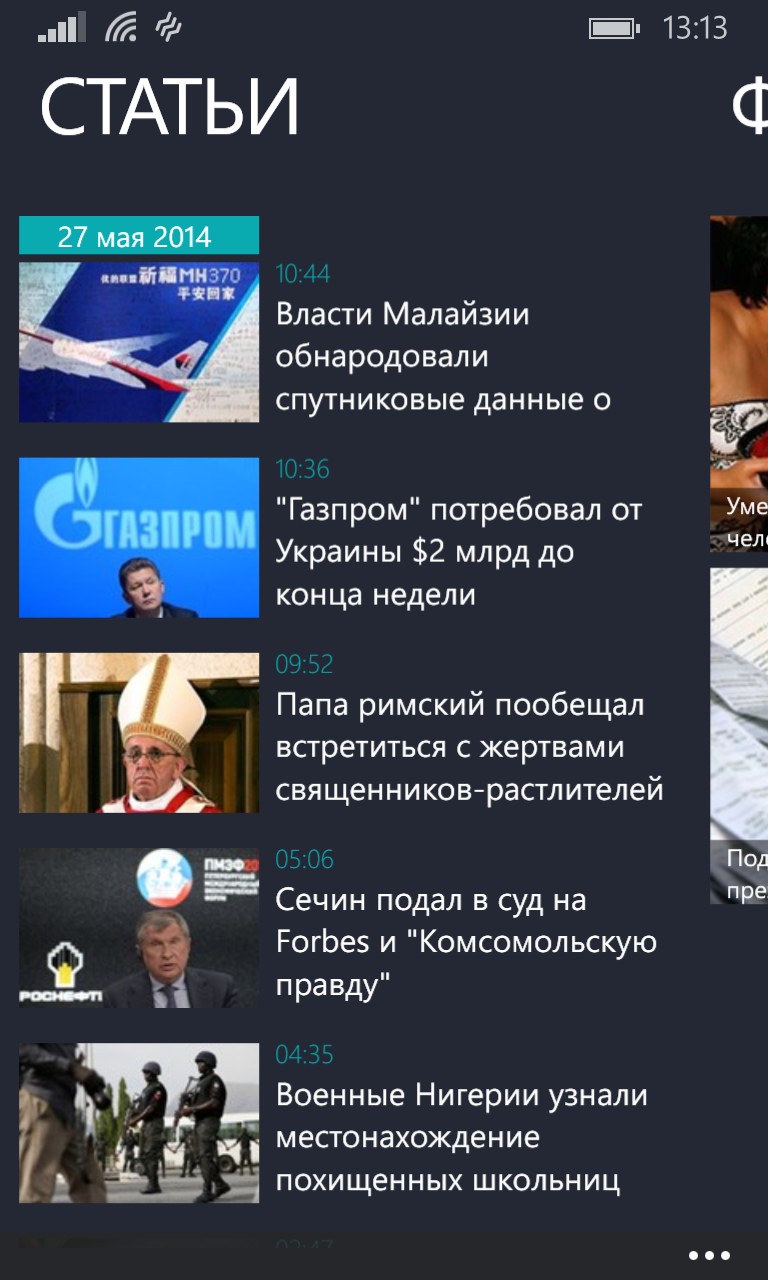
Связывание данных является технологией соединение данных и их визуального отображения в обе стороны (то есть, при изменении одного меняется и другое). Этот подход не вписывается в рамки традиционного MVC, поэтому более правильно использовать паттерн MVVM [2].

При запуске клиентского приложения происходит запрос на сторонний новостной сервис для получения элементов новостей, статей и фотосюжетов. После обработки данных новостные элементы выводятся на экран.

Для создания рекомендаций используются только статьи, так как только этот тип новостных элементов имеет категорию и достаточный объем для создания оценки новости. У каждой статьи может быть до трех разных категорий из следующего списка:

* Экономика
* Общество
* Спорт
* Культура
* Политика

В статьях явно не указана категория. Они ранжированы по времени за последние двое суток (рис. 8)



*Рисунок 8 Статьи в клиентском приложении*

После выбора статьи осуществляется переход на страницу, где представлен полный текст статьи. С этого момента начинает работать счетчик времени, который измеряет то время, которое пользователь провел на странице данной статьи.

После того как пользователь закончил читать статью (нажатие кнопки Назад) происходит вычисление параметров, из которых будет вычислена оценка, “данная” пользователем этой статье.

Параметры являются процентным выражение следующих величин:

* Уровень проматывания (скроллинга) страницы со статьей;
* Отношение количества символов, прочтенных за время нахождения на странице статьи, к общему количеству символов.

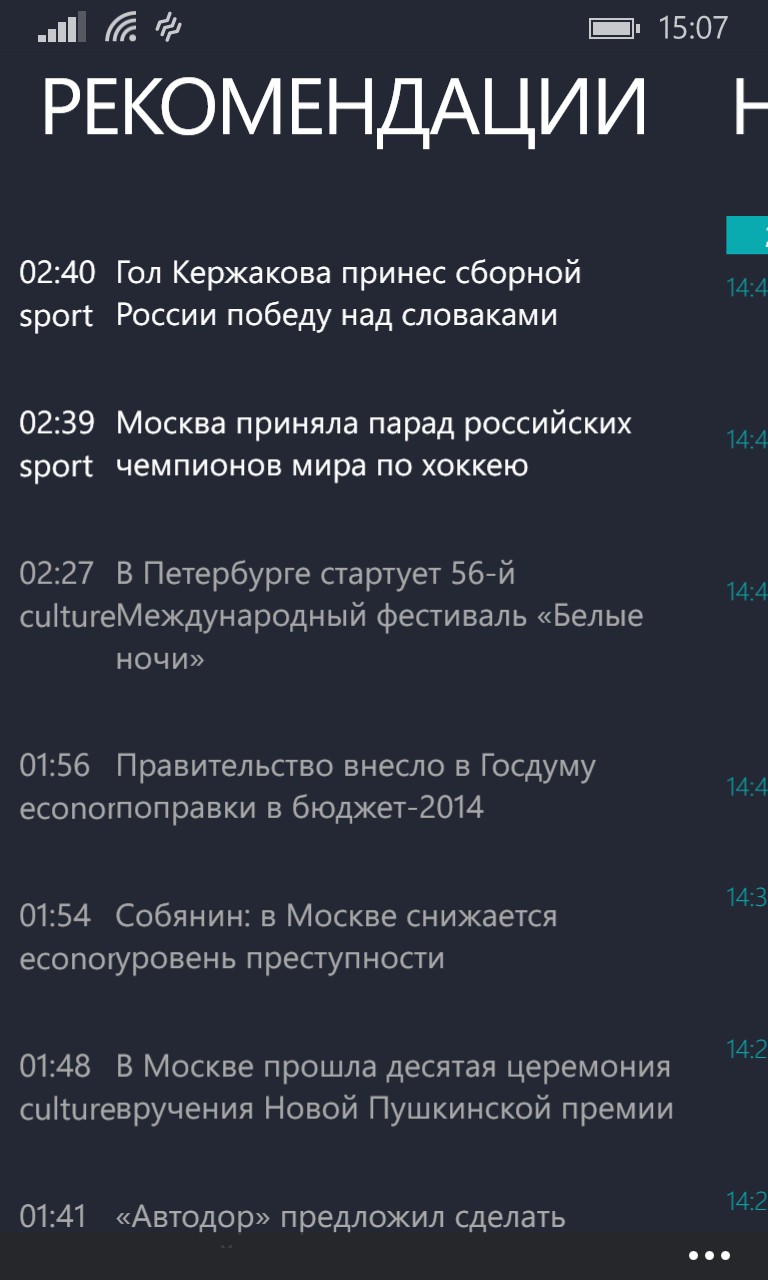
Для вычисления второго параметра была выведена константа, представляющая собой среднее число прочитанных символов в секунду (30).

После вычисления отправляется POST-запрос на адрес <http://personalized-news.appspot.com/statistics> со следующими параметрами:

* scrollRating
* readRating
* uid
* category

Из параметров scrollRating и readRating на серверной части вычисляется итоговая оценка по пятибалльной шкале.

При каждом запуске клиентской программы во вкладке “рекомендации” выводятся рекомендации для конкретного пользователя (рис. 9)



*Рисунок 9 Вкладка “Рекомендации”*

Для отображения оценок категориям новостей в режиме реального времени есть в меню есть пункт “График”. При нажатии на него происходит переход на страницу с графиком, где отображаются текущие средние оценки по категориям (рис 10)

### Macintosh HD:Users:plotkin:Downloads:wp_ss_20140529_0001.jpg

*Рисунок 10 График оценок категорий*

### Описание файлов

*ViewModels*

* Articles
  + ArticleListItemViewModel.cs – класс, представляющий собой ViewModel объекта статьи в общем списке статей;
  + ArticleListViewModel.cs – класс, представляющий собой ViewModel общего списка статей;
  + ArticleViewModel.cs – класс, представляющий собой ViewModel объекта статьи на отдельной странице;
  + RecommendationsViewModel.cs – класс, представляющий ViewModel общего списка рекомендаций;
  + RecommendationPageViewModel.cs – класс, представляющий ViewModel объекта рекомендации на отдельной страницe;
* News
  + NewsListItemViewModel.cs - класс, представляющий собой ViewModel объекта новости в общем списке новостей;
  + NewsListViewModel.cs – класс, представляющий собой ViewModel общего списка новостей;
  + NewsViewModel.cs – класс, представляющий собой ViewModel объекта новости на отдельной странице;
* PhotoStory
  + FreePhotoStoryListViewModel.cs - класс, представляющий собой ViewModel общего списка фотосюжетов;
  + PhotoItemViewModel - класс, представляющий ViewModel объекта фотосюжета на отдельной страницe;

Views

* Articles
  + ArticleList.xaml – класс, представляющий собой View общего списка статей;
  + ArticlePage.xaml – класс, представляющий собой View объекта статьи на отдельной странице;
  + Recommendations.xaml – класс, представляющий Viewобщего списка рекомендаций;
  + RecommendationPage.xaml – класс, представляющий View объекта рекомендации на отдельной страницe;
* News
  + NewsList.xaml – класс, представляющий собой ViewModel общего списка новостей;
  + News.xaml – класс, представляющий собой ViewModel объекта новости на отдельной странице;
* PhotoStory
  + FreePhotoStoryList.xaml - класс, представляющий собой View общего списка фотосюжетов;
  + PhotoStoryPage.xaml - класс, представляющий ViewModel объекта фотосюжета на отдельной страницe;

## Взаимодействие между клиентом и сервером

Взаимодействие между клиентской и серверной частью осуществляется при помощи POST-запросов. Клиент отправляет “сырые” данные о действиях пользователя в формате json, на сервере данные обрабатываются и сохраняются. После выполнения рекомендационных алгоритмов формируется запрос на новостной сервер для получения новостей из конкретных категорий, а после этого новости, которые пришли, отправляются на клиент пользователю.

## Средства разработки

Для разработки данной работы были выбраны следующие программные средства:

* Клиентская часть
  + Visual Studio 2012
  + Blend
  + Windows Phone SDK
* Серверная часть
  + Sublime Text 3
  + Google App Engine SDK
  + Python SDK
  + PyCharm

# Анализ результатов

## Тестирование

При реализации систем использующих рекомендательные алгоритмы необходимо проверять результаты, полученные в результате их работы. Для получения рекомендаций в данной работе были использованы алгоритмы Weighted Slope One и алгоритм нахождения ближайшего соседа. Для проверки работы данных алгоритмов была создана тестовая выборка пользователей системы, состоящая из 200 пользователей, оценки которых задавались случайным числом от 0 до 5. Эта выборка в полной мере эмулирует реальных пользователей, которым могут иметь оценки для каких-либо статей, так и еще не читать их.

В ходе тестирования были проведены серии запусков алгоритмов на данных тестовых выборках. Для проверки результатов необходимо вычисление RMSE (Root Mean Square Error) – параметра, который выражается следующей формулой(10):

*где*

*Rate – прямая оценка, которая была поставлена*

*ExpectedRate – оценка, которая была получена в ходе работы алгоритма*

*n – количество объектов системы*

RMSE позволяет вычислить точность прогноза (нужна ссылка). Для алгоритма Weighted Slope One параметр RMSE составил 0,9454, что является приемлемым для данного алгоритма результатом [7]. Для сравнения, тривиальный алгоритм, выставляющий среднюю оценку для каждой категории рекомендаций имеет RMSE, равный 1,061, что показывает, что результаты Weighted Slope One лучше тривиального алгоритма примерно на 10%.

Для проверки алгоритма поиска ближайших соседей был создан новый тестовый набор пользователей. Часть тестовых данных имела оценки, полностью совпадающие с теми, которые имеет тот пользователь, для которого был осуществлен поиск ближайших соседей. Алгоритм показал данное сходство, показав результат, равный 1, что соответствует полному совпадению пользователей. Те же тестовые наборы, которые имели отличие в оценках категорий с оценками пользователя, показали более низкие результаты (от 0.783 до -0,54), что соответствует правильной работе алгоритма поиска ближайших соседей.

Клиентское приложение приложение, являющееся полноценным новостным клиентом, на данный момент находится в бета-тестировании в магазине Windows Phone Store.

## Дальнейшее развитие проекта

Так как новостные агрегаторы являются крайне востребованными на данный момент программами, а Windows Phone Store имеет некоторый дефицит в качественных приложениях, то развитие проекта будет продолжено. На данный момент существует два главных направления разработки:

* Оптимизация серверной части;
* Дальнейшая разработка клиента;

Для публикации приложения необходимо оптимизировать как работу алгоритмов поиска (кэширование промежуточных результатов), так и работу с запросами в целом, так как при увеличении количества пользователей будет превышен лимит на бесплатные квоты в Google App Engine.  
Для разработки клиента необходимо увеличить количество источников новостей, а также разнообразие новостных рубрик.

Для улучшения рекомендационной составляющей следует обратить внимание на гибридные решения, объединяющие существующее решение с лингвистическими подходами. Лингвистические методы могут увеличить точность определения категорий новостей даже в тех новостных элементах, где категория не проставлена явным образом.

Однако необходимо учитывать, что анализ текста каждой новости не является правильным путем, так как такие операции являются крайне дорогими как с точки зрения времени, так и точки зрения ресурсов серверной части.

# Заключение

В ходе данной работы была реализована программа для персонализации новостной подписки, представляющая собой клиент-серверную систему.

Были решены следующие задачи:

* Был спроектирован и реализован серверный компонент, рассчитанный на множественное использование;
* Была изучена проблема рекомендательных систем, проведен сравнительный анализ методов их создания;
* Были реализованы некоторые методы для создания рекомендаций и объединены в гибридное решение на основе коллаборативной фильтрации;
* Было проведено тестирование методов создания рекомендаций;
* Был спроектирован и реализован клиентский компонент, представляющий собой мобильное приложение на платформе Windows Phone;
* Приложение было запущено в бета-тестирование в магазине Windows Phone Store;

Результатом работы является клиент-серверная программа, которая позволяет следить за актуальными новостями в мире и получать новостные рекомендации на основе предпочтений пользователя и на основе других пользователей.

Данная работа имеет практическую ценность, так как будет использоваться реальными пользователями для чтения новостей. В будущем планируется развитие проекта в качестве новостного агрегатора.

# Список использованных источников

[1] Daniel Lemire, Anna Maclachlan. «Slope One Predictors for Online Rating-Based Collaborative Filtering.» *In SIAM Data Mining (SDM'05)*, 2005.

[2] Hall, Gary. *Pro WPF and Silverlight MVVM.* Apress, 2010.

[3] M. D. Ekstrand, J. T. Riedl and J. A. Konstan. *Collaborative Filtering Recommender Systems.* Minneapolis: University of Minnesota, 2011.

[4] Ott, Patrick. « Incremental Matrix Factorization for Collaborative Filtering.» *Science, Technology and Design* (Anhalt University of Applied Sciences), 2008.

[5] Sanderson, Dan. *Programming Google App Engine.* O'Reilly Media, 2009.

[6] Segaran, Toby. *Programming Collective Intelligence.* O'Reilly Media, 2007.

[7] Yongqiang Wang, Liang Yin, Bing Cheng, and Yong Yu. «Learning to Recommend Based on Slope One Strategy.» 2012.