Санкт-Петербургский Государственный Университет
Математико-механический факультет

Кафедра системного программирования

Трансформация поисковых запросов в распределенных системах web-сервисов

Курсовая работа студентки 361 группы
Солодкой Анастасии Сергеевны

Научный руководитель _____ Новиков Б. А.
д.ф.-м.н., профессор

Санкт-Петербург
2010

Оглавление

Глава 1

Введение

Современное развитие Глобальной Сети характеризуется ростом количества поисковых систем, варьируищихся от таких гигантов, как Google, Yahoo, Яндекс до небольших узкоспециализированных поисковых систем. К последним можно отнести такие системы, как Yahoo Traffic (пробки на дороге), Yahoo Local (разлчичные увеселительные заведения), ZEvent (события), а так же системы поиска и заказа товаров, книг, системы поиска вакансий и прочие.

Такие поисковые системы позволяют пользователю задавать более точные запросы, гарантирующие более успешный поиск, но с другой стороны, они не предоставляют возможности выйти за границы области знаний, для которой они были разработаны. Однако пользователи часто интересуются более комплексными запросами. В результате для получения ответов на них, пользователю требуются навыки в поиске информации. Ему приходится формулировать отдельные запросы к каждой из систем и затем вставлять результаты запросов одной в поисковый запрос другой. Конечно же, идеальным было бы, если бы пользователь мог задать один составной запрос и получить всю необходимую ему информацию, работая с одним общим интерфейсом.

1.1 Новое поколение поисковых систем

Данная работа сфокусирована на исследовании идеи создания общего интерфейса для нескольких поисковых систем, принимающего составные запросы пользователя и производящую поиск соответствующей запросу информации путем объединения ответов на несколько менее общих запросов в узкоспециализированные системы поиска

Примерами пользовательских запросов, для которых необходим подобный подход могут послужить следующие запросы:

- Найти все рестораны китайской кухни в Мурманске, рядом с которыми находятся отделения почтовой службы.
- Купить книги авторов VLDB из Санкт-Петербурга
- Найти все новости, обзор которых проводился The New York Times, Washington Post, The Times и Комсомольской правдой
- Найти все конференции по базам данных в городах со среднемесячной температурой не менее 13C, путешествие на которые обойдется не более, чем в 1000$

Очевидно, что эти запросы могут быть разделены на несколько более простых запросов. К примеру, последний запрос может быть разделен на запросы в следующие поисковые системы: поиск научных встреч, бронирование отелей, бронирование билетов и статистика погоды.

Поисковая система нового поколения должна разбивать сложные запросы на более мелкие, а так же объединять по определенным правилам ответы нескольких поисковых систем друг с другом чтобы предоставить полный составной ответ пользователю.

Подобный подход объединения результатов запросов используется в базах данных. Но для применения его для объединения результатов, возвращаемых поисковыми системами необходим более широкий комплексный анализ: к примеру, зачастую результатами ответов на запросы к поисковым сервисам являются XML-документы, содержащие ранжированные результаты. Для того, чтобы результаты поиска выдавались в порядке релевантности, нужны оптимальные стратегии для объединения результатов с сохранением ранжирования. Кроме того, довольно сложно разбить запросы на языке, близком к естественному, на несколько запросов, понятных узкоспециализированным поисковым системам.

Итак, задача объединения поисковых систем значительно отличается от подобной задачи объединения результатов запросов к таблицам реляционных баз даных и требует более пристального внимания.

1.2 Настоящие разработки: проект NGS

На данный момент разработкой вышеописанной идеи создания общего интерфейса для нескольких поисковых систем занимается итальян-

ская группа ученых DBGroup под руководством профессора Стефана Чери. Первые работы, относящиеся к этой теме были опубликованы в начале 2008-го года. Проект был назван Next Generation Search (NGS) [1]
.

Основная работа в проекте NGS ведется над созданием среды разработки, позволяющей регистрировать поисковые веб-сервисы и "смешивать"их, задавая условия, позволяющие объединять результаты запросов к ним, а так же позволяющие разделять составные запросы пользователя на несколько более узкоспециализированных запросов к сервисам.

Работа по выполнению подобных сложных запросов и объединение сервисов делится на несколько уровней исполнения:

1. Уровень формулирование запроса
2. Уровень выполнения запроса
3. Уровень данных

Одна из неисследованных подзадач - это мэппинг пользовательского запроса в набор запросов к веб-сервисам (от уровня формулирования запроса к уровню выполнения). Исследованию этой подзадачи и посвящена основная часть работы.

Глава 2

Обзор литературы

2.1 Классификация поисковых сервисов [3]

Поисковые сервисы могут быть классифицированы по двум признакам. Первый признак - операции, доминирующая по времени выполнения:

A Операции объединения

B I/O операции (запрос, ответ)

Второй признак - тип ранжирования:

1 Пошаговое ранжирование. Большинство подходящих результатов может быть получено за ограниченое число шагов, т.к. релевантность результатов уменьшается с некоторым фиксированным шагом.

2 Линейное ранжирование. Релевантность результатов уменьшается линейно.

В большинстве случаев сервисы возвращают неявно ранжированные запросы: результат запроса не содержит ранга. Однако этот случай так же покрывается приведенной классификацией, т.к. результаты ответа на запрос сортируются по релевантности, то есть ранжирующая функция просто прозрачна для программы.

Далее будут использоваться обозначения **A**, **B**, **1**, **2**. для указания типа веб-сервисов.

2.2 Алгоритмы для объединение результатов запросов [3]

Опишем основные стратегии объединения запросов.

Последовательность выполнения запросов к двум объединяемым веб-сервисам варьируется в зависимости от типов сервисов. Для начала опишем стратегии выполнения запросов для сервисов, возвращающих явно ранжированные результаты.

TEO - Tile Extraction Optimal. Используется в случае доминирования времени операций объединения (**A**. На каждом шаге выбирается набор результатов для объединения. Выбираются результаты, потенциально дающие "лучшие"(более релевантные) результаты объединения.

R-Shape . Используется в случае доминирования времени операций I/O (**B**). На каждом шаге выбирать какой сервис выбрать следующим. После получения новой группы результатов необходимо произвести объединение кэшированных результатов и новых.

Гораздо больший интерес предоставляют стратегии объединения веб-сервисов, результаты, возвращаемые которыми, не имеют явного ранжирования. Такие веб-сервисы встречаются гораздо чаще, чем сервисы с явным ранжированием. Для объединения таких сервисов существует три стратегии: Nested loop, Merge-scan и S-Shape. Рассмотрим эти стратегии.

Nested loop

Три случая, когда хотя бы один из сервисов имеет тип **1**, использует метод Nested loop:

На первом шаге получаются все наиболее релевантные результаты запроса к сервису, имеющему пошаговое ранжирование. По определению пошагового ранжирования, для этого необходимо выполнить некоторое конечное число запросов. Затем на каждыом шаге выполняется запрос ко второму сервису и происходит объединение нового набора результатов с кэшированными результатами первого сервиса.

Merge-scan

Случай, когда оба сервиса имеют неявное линейное ранжирование (**2**), а так же объединение результатов - более ресурсоемкая операция.

Характеризуется "движением по диагонали". Рассмотрим два сервиса X и Y. Обозначим результаты запроса к сервисам на каждом i-ом шаге соответственно r_{X_i} и r_{Y_i}. Тогда последовательность выполняемых шагов будет следующей:

1. Получаем r_{X_1} и r_{Y_1}, объединяем (r_{X_1}, r_{Y_1})
2. Получаем r_{X_2} и r_{Y_2}, объединяем (r_{X_2}, r_{Y_1}), (r_{X_1}, r_{Y_2})
3. Получаем r_{X_3} и r_{Y_3}, объединяем (r_{X_1}, r_{Y_3}), (r_{X_2}, r_{Y_2}), (r_{X_3}, r_{Y_1})

   ...

2n-1. Получаем $r_{X_{2n-1}}$ и $r_{Y_{2n-1}}$, объединяем $(r_{X_1}, r_{Y_{2n-1}})$, $(r_{X_2}, r_{Y_{2n-2}})$, $\ldots (r_{X_{2n-1}}, r_{Y_1})$

2n. Получаем $r_{X_{2n}}$ и $r_{Y_{2n}}$, объединяем $(r_{X_{2n}}, r_{Y_1})$, $(r_{X_{2n-1}}, r_{Y_2})$, $\ldots (r_{X_1}, r_{Y_{2n}})$

S-shape

Случай, когда оба сервиса имеют неявное линейное ранжирование (**2**), а так же I/O операции наиболее ресурсоемки.

На каждом шаге производится по одному запросу к каждому из веб-сервисов, а затем производится объединение кэшированных результатов с новыми, а так же новых результатов результатов с новыми.

2.3 Семантический разбор пользовательских запросов [6]

Набор запросов, относящихся к одной области поиска обычно характеризуется следующим образом:

- Ограниченый словарь
- Набор шаблонов использования слов, выражений и пр.
- Редкость семантической двусмысленности
- Часто встречаются термины и жаргонные слова

Таким образом, семантический поиск использует следующие стратегии:

- Поиск шаблонных выражений, специфичных для области поиска или же общих (временные промежутки, радиусы, длины и прочее).
- Поиск ключевых слов, таких, как категории, термины, жаргонные слова.

- Использование различных онтологий для расширения области поиска и уточнения

Приведем примеры.

При обработке запросов, относящихся к системам поиска заведений отдыха (к примеру, Yahoo Local Search), имеется возможность указать категорию заведения. Зачастую есть возможность получить все доступные категории и, закешировав их, производить поиск слов, относящихся к категориям для уточнения запроса.

Существует так же возможность расширения запросов за счет онтологий. К примеру, при поиске по событиям научного мира мы имеем возможность расширить указанную пользователем тему событий за счет связанных с ней терминов.

Глава 3

Точное решение и постановка задачи

3.1 Общая постановка задачи

Написать простейшую поисковую систему нового поколения:

- Разбить запрос на языке, близком к естественному на набор запросов, "понятных"исполнителю запросов.
- Разработать простейшую реализацию двух оставшихся уровней для демонстрации результатов.

3.2 Средства решения задачи

- Используемый язык: Java
- Уровень представления: использование спецификации J2EE: Servlets
- Уровень запросов реализован с использованием двух веб-сервисов:
  - Yahoo Local Search - сервис поиска увеселительных заведений на территории США
  - ZEvents - сервис поиска различных мероприятий, происходящих в мире
- Используемые библиотеки: Apache Axiom 1.5.1, Apache Httpclient 1.4

Разработка проводилась на платформе Windows XP с использованием NetBeans IDE.

3.3 Точная постановка задачи

Описание используемых веб-сервисов

Каждый из сервисов характеризуется набором параметров запроса и параметров ответа. Ниже мы перечислим только те параметры, которые используются в нашей программе.

Yahoo Local Search

Yahoo Local Search - сервис, предоставляющий возможность поиска заведений отдыха, питания и прочего в США. Является RESTful-сервисом, поддерживающим протокол SOAP.

URL веб-сервиса: http://local.yahooapis.com/LocalSearchService/V3/localSearch

Параметры запроса:

- query - набор ключевых слов, по которым производится фильтрация заведений.
- street, city, state, zip, longitude, latitude, location - набор параметров, позволяющий определить точку на земном шаре, относительно которой производится поиск
- radius - максимальная удаленность заведения от указанной точки

Параметры ответа:

- title - название заведения
- address, city, state, phone, longitude, latitude - набор параметров, позволяющий определить точку на земном шаре, относительно которой производится поиск
- phone - телефон заведения
- rating - рейтинг заведения (голосование пользователей)
- url - ссылка на страницу с детальным описанием заведения

ZEvents Search

Сервис предоставляет API для поиска мероприятий, происходящих в мире: таких, как концерты, встречи, выставки и прочее.

URL веб-сервиса: http://www.zvents.com/rest/search

Параметры запроса:

- what - набор ключевых слов, по которым производится фильтрация событий
- when - промежуток времени, в котором должно произойти событие. Промежуток указывается в формате "DD/MM/YY to DD/MM/YY"
- where - город, в котором должно произойти событие. Необходимо указать название города и штат. Например, San Francisco; CA
- radius - максимальная удаленность места проведения события от указанного города.

Параметры ответа:

- event (событие)
  - name - название события
  - summary - общее описание события
  - description - детальное описание события
  - url - ссылка на страницу с детальным описанием события вне zevents
  - phone - телефон, по которому можно получить информацию о событии
  - starttime, endtime - время начала и конца события в формате YYYYMMDDHHMM
  - price - цена билета
  - venue_id - идентификатор места проведения события. Список упоминаемых мест проведения событий возвращается вместе с набором событий.
- venue (место проведения события)

– name - название места

– description - детальное описание места

– url - ссылка на страницу с детальным описанием места проведения события вне zevents

– phone - телефон места проведения события

– address, city, state, country, zipcode, latitude, longitude - точное указание координат места проведения события

Параметры объединения

Результаты, возвращаемые веб-сервисами объединяются по полям longitude, latitude (с некоторым отклонением).

Формат пользовательских запросов

В пользовательском запросе необходимо задать город проведения события, ключевые слова для поиска события, ключевые слова для поиска увеселительных заведений, максимальное удаление события от указанного города и период времени, за который необходимо найти все события.

Пример обрабатываемого поискового запроса: "Dance events with greek or pizza restaurants in 15km from phoenix in next month".

Формат представления результатов

В результате выполнения запросов пользователю предоставляется набор описаний пар событие-место проведения, находящихся рядом. К примеру, этот набор может содержать название, время и адрес проведения события, название и адрес увеселительного заведения.

Ожидаемый результат

Ожидаемый результат: веб-приложение, реализующее поисковую систему в двух областях поиска: события и места. Интерфейс позволяет задать составной запрос и демонстрирует пользователю объединенные результаты поиска: информацию о событии и близлежащем месте отдыха.

3.4 Реализация уровня данных

Опишем основные интерфейсы, используемые в программе:

- Service, ServiceParams, ServiceRequest, ServiceResults

- SShapePlan
- JoinerEngine, JoinerResult

Затем опишем их реализации, разработанные соответственно для работы с сервисами Yahoo Local Search и ZEvents.

Представление и реализация уровня данных

Для представления запроса к сервису используется класс **ServiceParams** `<Key>`, параметризованный типом ключа параметра (в большинстве случаев это **String** или **enum**). Класс **ServiceRequest**<**Key**> представляет собой содержание запроса, отправляемого сервису, где Key - тип ключа параметра, **ServiceResults** представляет собой набор результатов, полученных на запрос к сервису. Для представления сервиса испльзуется абстрактный класс **Service** `<PKey, RKey, Params` *extends* `ServiceParams<PKey>, Request` *extends* `ServiceRequest<RKey>, Result` *extends* `ServiceResult>`. Этот класс имеет единственный общедоступный метод `Result` **execute**`(Params)`, реализующий выпонение запроса, сконструированного на основе объекта-представления пользовательского запроса типа `Params`. Полученный набор результатов запроса возвращается в объекте типа `Result` .Рассмотрим немного подробнее класс **ServiceResults** .

ServiceResults имеет три метода, обеспечивающие основную необходимую функциональность. `int` **size**`()`, `Object` **get**`()`, обеспечивают минимальную функциональность, позволяющую использовать объект как хранилище. Кроме того он имеет метод `void` **add**`(ServiceResult)`, производящий простое слияние двух результатов запроса одного сервиса. Такая функциональность необходима, в частности, для реализации кэширования результатов предыдущих запросов к веб-сервису.

Теперь рассмотрим соответствующие реализации, относящиеся к каждому из веб-сервисов.

Yahoo Local Search

Как было указано ранее, Yahoo Local Search - RESTful веб-сервис, поддерживающий протокол SOAP. Для работы с веб-сервисом по протоколу SOAP, мы используем Apache Axis2 API. Для получения SOAP-клиента используется метод класса **RESTReader** `static ServiceClient` **getClient**`(String serviceURL)`. Запрос к сервису по протоколу SOAP

является XML-объектом, представляемым объектом типа (**OMElement**).

Для работы с Apache Axis2 используется набор библиотек Apache Axis2 1.5.1.

Реализации базовых интерфейсов сервисов - это соответственно классы **YahooLocalServiceParams** , **YahooLocalServiceResults** , **YahooLocalService** и **YahooLocalServiceRequest** .

ZEvents Search

В отличии от сервиса Yahoo Local Search, сервис ZEvents Search не поддерживает протокол SOAP. Поэтому реализация отправки запросов (HTTP GET) сделана с использованием библиотки Apache HttpClient, в частности, с использованием классов **HttpClient** и **GetMethod** . Согласно этой реализации, класс **ZEventServiseRequest** содержит в себе экземпляр **HashMap** `<ZEventServiseRequest.AvaliableFields>` - простой набор параметров запроса. Полный список реализаций базовых интерфейсов: **ZEventServiceParams** , **ZEventServiceRequest** , **ZEventServiceResults** и **ZEventService** .

3.5 Реализация уровня запросов

Теперь рассмотрим следующий уровень приложения: уровень, на котором происходит последовательность запросов к веб-сервисам, результаты ответов на которые затем будут объединены.

Для взаимодействия приложения с этим уровнем определен абстрактный класс **JoinerEngine** `<JoinResult extends JoinerResult>`. Этот класс содержит те методы, которые необходимы поисковой системе для взаимодействия с пользовательским интерфейсом. Все эти методы возвращают `List<JoinResult>` - набор объединенных результатов, предоставляемых пользователю. Это методы **getLastList**`()`, **getNextList**`()`, **getList**`(int index)`. В случае, если запрошенные данные еще не получены (номер первого запрошенного элемента больше количества полученных данных), вызывается метод для получения дополнительной порции результатов. Далее рассмотрим алгоритм, непосредственно запрашивающий следующий набор данных.

Как известно (см. 2.2), существует несколько алгоритмов для определения последовательности вызовов и стратегии объединения в зависимости от типа веб-сервиса (см. 2.1).

Оба используемых сервиса относятся к сервисам с неявным ранжированием. Кроме того, время выполнения I/O опреаций для них доминирует над временем выполнения объединений. Таким образом выбранным алгоритмом будет S-Shape.

Опишем более подробно этот алгоритм. Рассмотрим два сервиса X и Y. Обозначим результаты запроса к сервисам на каждом i-ом шаге соответственно r_{X_i} и r_{Y_i}, а кэшированные к i-му шагу результаты выполнения запросов к сервисам X и Y обозначим как R_{X_i} и R_{Y_i}. Пусть $k > 1$ - текущий шаг выполнения алгоритма (выполнение шага 1 очевидно: запрашиваются первые порции данных от каждого из сервисов и обединяются), J - множество уже полученных с помощью объединения результатов.

1. $r_{X_k} := execute(X);\ r_{Y_k} := execute(Y);$
2. $J := J \cap join(R_{X_k}, r_{Y_k})$
3. $J := J \cap join(r_{X_k}, R_{Y_k})$
4. $J := J \cap join(r_{X_k}, r_{Y_k})$
5. $k := k + 1$

Описанный алгоритм представлен абстрактным классом **SShapePlan** . Класс содержит метод `public List<JoinResult>` **fetchNext**`()`, реализующий шаг выполнения алгоритма S-Shape. Класс так же содержит два абстрактных метода, `boolean` **canJoin**`(Object first, Object second)` и `JoinResult` **join**`(Object first, Object second)`, позволяющих потомкам самим определить способы объединения двух записей результата запроса соответственно первого вызываемого веб-сервиса и второго.

3.6 Реализация уровня разбора запросов

Перейдем к этапу разбора пользовательских запросов.

Для разбора запросов на языке, близком к естественному, используется поиск шаблонных выражений. **SemanticPattern** `<ItemValue>` определяет интерфейс для работы с шаблонными выражениями. Метод `String` **getRegex**`()` возвращает регулярное выражение, задающее шаблон. Метод `ItemValue` **parseValue**`(String group)` позволяет преобразовать вы-

ражение в соответствующий объект, хранящий представление содержания шаблона.

Parser - содержит один общедоступный метод `List<Object>` **parse**`(String toParse, SemanticPattern... items)`, разбирающий строку toParse и возвращающий набор объектов выражений, содержащихся в этой строке согласно набору шаблонов выражений `items`.

3.6.1 Поддерживаемые типы запросов

Программа поддерживает следующие типы выражений:

Временные промежутки

Общий формат: $\langle from \| to \| at \| in \rangle \, \langle date \rangle$.

Ключевое слово $\langle from \rangle$ позволяет определить дату, с которой необходимо начать поиск. $\langle to \rangle$, $\langle at \rangle$, $\langle in \rangle$ определяют дату до которой необходимо производить поиск.

Таким образом для того, что бы определить временной промежуток поиска, неободимо задать два выражения: к примеру, *from 21.10.2010* и *to 25.10.2010*. Определение промежутка времени происходит по следующим правилам:

- Если не указана начальная дата, поиск производится с текущей даты.
- Если не указана конечная дата, поиск производится по всем будущим событиям начиная с начальной даты.
- Если не указаны ни начальная дата, ни конечная, поиск производится начиная с текущего момента по всем событиям.

Так же доступны различные форматы $\langle date \rangle$:

- форматы dd.mm.yyyy, dd.mm.yy, dd/mm/yyyy, dd/mm/yy, dd-mm-yyyy, dd-mm-yy
- $[\text{"}this\text{"} \| \text{"}next\text{"}] \, \langle \text{"}day\text{"} \| \text{"}week\text{"} \| \text{"}month\text{"} \| \text{"}year\text{"} \| dayOfWeek \rangle$. dayOfWeek представляет собой день недели, соответственно: sunday, monday, tuesday, wednesday, thursday, friday, saturday. Так же поддерживаются краткие названия дней недели: sun, mon и прочие.

Для реализации вышеописанной функциональности имитировалась лингвистическая онтология. Данному шаблону соответствуют классы **DateIntervalPattern** и **DateValue** .

Адреса

Общий формат: $cityName\left[\langle ;|,|\ \rangle state\right]$

Для обнаружения образцов, связанных с адресом, используется небольшая географическая онтология. Эта онтология содержит все крупные города США и штаты, где они расположены. Так же структура этой онтологии позволяет определить все ZIP-коды, связанные с каждым из городов и краткие (часто употребляемые) названия городов. Сейчас описанная дополнительная информация отсутствует в онтологии, но в случае ее пополнения, будет возможность задавть место поиска так же с помощью ZIP-кодов и употребительных названий городов.

Описанному шаблону соответствуют классы **CityCashePattern** и **CityCasheValue** .

Радиусы

Общий формат: $\langle around|about\rangle\langle number\rangle$

Для нахождения шаблонов радиуса использовалось простое сопоставление с образцом. Соответствующие классы: **RadiusPattern** и **RadiusValue**

Ключевые слова поиска

Самой сложной задачей является определение словосочетаний, относящихся непосредственно к определению событий или мест отдыха. Данный вопрос был решен с помощью введения ключевых слов $\langle local\rangle$ и $\langle event\rangle$. Таким образом общий формат задания параметров поиска, относящихся непосредственно к Yahoo Local Serch будет "$local$"$\langle value\rangle[or\langle value\rangle]^*$, а для ZEvents Search - "$event$"$\langle value\rangle[or\langle value\rangle]^*$.

Соответствующие классы: **StringPattern** , **StringValue** . Для получения необходимого префикса (local/event) в объекте типа StringValue используется метод `String` **getPrefix()**

3.6.2 Примеры

Таким образом, исходный запрос "Dance events with greek or pizza restaurants in 15km from phoenix in next month"3.3 можно задать следующими способами:

[event dance] [to next month] at [Phoenix] [around 5] [local pizza or greek]

[event dance] [from now] [in next month] at [Phoenix, AZ] [about 5] [local pizza or greek]

3.7 Реализация отображения разобранных параметров пользовательского запроса в набор простых запросов к веб-сервисам

Рассмотрим теперь уровень, занимающийся выделением параметров запроса пользователя для каждого конкретного сервиса. Для хранения запроса пользователя введена структура **GlobalQuery ,** в простейшем случае являющаяся списком параметров запроса, выделенным на этапе разбора пользовательского выражения.

На данном этапе развития работы мы не поддерживаем выполнение запросов, содержащих сложные логические связи. Поэтому параметры запроса, относящиеся к каждому из сервисов определяются простейшим образом.

Для определения параметров запроса к сервису Zevents Search используются значения, находящиеся в объектах типа RadiusValue, CityCasheValue, DateValue и StringValue с префиксом "event".

Для определения параметров запроса к сервису Yahoo Local Search используются значения, находящиеся в объектах типа RadiusValue, CityCasheValue и StringValue с префиксом "local".

Глава 4
Выводы

В ходе выполнения курсовой работы была реализована простейшая поисковая система нового поколения, позволяющая производить поиск различных мероприятий в США и увеселительных заведений рядом с местами их проведения.

Основная часть работы включила в себя разбор пользовательского выражения на набор параметров запроса и преобразование полученного набора в параметры запроса к каждому из веб-сервисов. Так же была реализована вспомогательная задача объединения результатов, полученных на запросы к двум веб-сервисам и разработан простейший интерфейс взаимодействия программы с пользователем.

4.1 Дальнейшие исследования

Дальнейшие исследования в области решения общей задачи могут происходить в следующих направлениях:

- Увеличение количества используемых системой веб-сервисов.
- Разработка различных онтологий, использование дополнительных параметров поиска, предоставляемых веб-сервисами (например, категорий) для уточнения или же расширения запроса.
- Проведение анализа запросов и выбора зарегистрированных в системе сервисов, использующихся для выполнения запроса на основе проведенного анализа
- Введение поддержки стандартных логических конструкций (дизъюнкцию, конъюнкцию, отрицание и прочее) в пользовательских запросах

Литература

[1] Daniele Braga, Diego Calvanese, Alessandro Campi, Stefano Ceri, Florian Daniel, Davide Martinenghi, Paolo Merialdo, and Riccardo Torlone. Ngs: a framework for multi-domain query answering. In *Proc. of the Workshop on Information Integration Methods, Architectures, and Systems (IIMAS 2008)*, pages 254–261. IEEE Computer Society Press, 2008.

[2] Daniele Braga, Diego Calvanese, Alessandro Campi, Stefano Ceri, Florian Daniel, Davide Martinenghi, Paolo Merialdo, and Riccardo Torlone. Ngs: A new generation search engine supporting cross domain queries. In *Proc. of the 16th Italian Conf. on Database Systems (SEBD 2008)*, pages 342–349, 2008.

[3] Daniele Braga, Alessandro Campi, Stefano Ceri, and Alessandro Raffio. Joining the results of heterogeneous search engines. *Inf. Syst.*, 33(7-8):658–680, 2008.

[4] Daniele Braga, Stefano Ceri, Florian Daniel, and Davide Martinenghi. Mashing up search services. *IEEE Internet Computing*, 12(5):16–23, 2008.

[5] Marco Brambilla and Stefano Ceri. Engineering search computing applications: vision and challenges. In *ESEC/FSE '09: Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 365–372, New York, NY, USA, 2009. ACM.

[6] Jiwei Zhong, Haiping Zhu, Jianming Li, and Yong Yu. Conceptual graph matching for semantic search. In *ICCS '02: Proceedings of the 10th International Conference on Conceptual Structures*, pages 92–196, London, UK, 2002. Springer-Verlag.