САНКТ–ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Математико–Механический факультет

Кафедра Системного Программирования

Поиск изображений в частично аннотированной базе по текстовому запросу

Дипломная работа студента

Дольника А.С.

Научный руководитель,
профессор,
д. ф.-м. н-к ..... /Б. А. Новиков/
/подпись/

Рецензент,
ст.преподаватель ..... /А. С. Герасимов/
/подпись/

"Допустить к защите"
Заведующий кафедрой,
профессор,
д. ф.-м. н-к ..... /А. Н. Терехов/
/подпись/

Санкт–Петербург
2007

ОГЛАВЛЕНИЕ

1 Введение

За последние годы объем мультимедиа данных возрос в несколько раз. Однако вся эта информация была бы бесполезна без точного, удобного и быстрого поиска по ней.

Эта работа посвящена информационному поиску в изображениях.

Содержание картинки условно подразделяют на три уровня:

1. Низкоуровневые характеристики. В него включаются такие понятия как цветовые гистограммы, геометрия, текстура и прочее. Эти параметры мы можем измерить (и такие методы есть), даже не предполагая, какие реальные объекты присутствуют на изображении.

2. Средний семантический уровень. На этом уровне человек дает описание и обозначает некоторые объекты и/или изображение в целом, при этом он использует свой жизненный опыт и некоторые логические заключения. Примерами описаний семантического уровня являются: машина, кадр из фильма и т.д. Этот уровень является более или менее общим для группы людей.

3. Высокий семантический уровень. Этот уровень связан с чувствами и абстрактными ассоциациями, которые вызывает изображение у конкретного человека. В качестве примера можно привести следующие описания: веселое настроение, радость, грусть, счастливая женщина. Этот уровень является индивидуальным для каждого человека.

Текстовые запросы поиска зачастую содержат понятия среднего и высокого семантического уровня, которые трудно обработать системе, более того, далеко не все параметры уровня низкоуровневых характеристик доступны системе. Вследствие чего, возникает задача сужения "семантического разрыва" при поиске в изображениях.

В настоящее время существует несколько направлений информационного поиска графической информации, это:

- поиск изображений по содержанию (content-based image retrieval – CBIR) [1], который мы будем называть чистым поиском;

[1] В англоязычной научной литературе по компьютерному видению (computer vision) также можно встретить термин query by image content (QBIC) или термин content–based visual information retrieval (CBVIR)

- поиск изображений по так или иначе прилагаемым к ним аннотациям, который делится еще на три вида:
  - автоматически полученные аннотации,
  - полуавтоматические аннотации,
  - полученные вручную аннотации.

К аннотациям, полученным автоматически, относятся: взятые из "около картиночного" текста ключевые слова, различного рода другая мета информация (как время создания, географическое место создания) и прочее.

Под полуавтоматическими способами аннотирования понимают методы, при которых часть аннотаций изображений описывается вручную, остальные автоматически получают анотации изображений, близких по содержанию.

Заголовки изображений и написанные пользователем словесные аннотации – относят к полученным вручную аннотациям.

Системы с ручным аннотированием относят к первому поколению систем текстового поиска изображений (текстового, в смысле запрос в виде текста).

Существуют также разные схемы аннотации:

1. Полнотекстовые аннотации. Картинка полностью описывалась словосочетанием или несколькими предложениями
2. Аннотации ключевыми словами. Картинке приписывается несколько ключевых слов и создается словарь контрольных терминов
3. Онтологии

Анализ систем первого поколения поиска показал, что на простых случаях они работают довольно хорошо, но при усложнении запроса ситуация ухудшается. Более того, в системах первого поколения не учитывается возможность подстройки под пользователя, либо эта подстройка является человечески ресурсоемкой (то есть занимает много сил и времени у пользователя).

Однако, в общем случае глобального поиска, например в сети Интернет, где обычно присутствует множество соответствующих запросу изображений и есть возможность выделить или отфильтровать несущественные ключевые слова путём опроса миллиона пользователей, такие методы работают хорошо. Примером таких систем являются сервисы предоставляемые поисковыми системами Google и Яндекс.

В системах поиска второго поколения текстовые аннотации, полученные вручную, используются наравне с информацией извлекаемой из изображения. Ко второму поколению относятся и системы автоматического аннотирования.

С появлением систем второго поколения, появились запросы, отличные от стандартного текстового запроса. Наиболее распространенные их них следующие запросы:

1. Запрос по изображению-шаблону. В качестве запроса используется данное пользователем изображение.

2. Запрос по рисунку-палитре. Задаются некоторые параметры искомого изображения, такие как цветовая схема. В качестве результатов должны быть выданы образцы с похожей цветовой схемой.

3. Запрос по эскизу. Создается рисунок-набросок, которой и будет использоваться в качестве запроса.

У всех этих запросов есть один существенный недостаток – надо что–то рисовать или находить что–то похожее (человек может совсем не уметь рисовать, а похожей картинки под рукой не найдется), кроме того, такие запросы предполагают, что у пользователя есть некоторые сведения об особенностях работы алгоритмов поиска.

Поэтому пользовательские запросы в виде текста наиболее предпочтительны. В этой работе предпринята попытка соединить лучшие свойства различных систем. Обычно, каждый метод поиска выдает список ранжированных картинок. Предполагается сливать результаты работы нескольких методов.

Таким образом, следуя предположению, повысится точность и полнота поиска, ибо в итоге мы учитываем "реакцию" нескольких систем на приведенный запрос. В тоже время, механизм слияния легко встраивается в схему комбинированного поиска по тексту и содержанию. Схема действия этого механизма состоит в следующем:

- Осуществляется полнотекстовый поиск по аннотациям

- Каждый найденный элемент используем как образец для поиска по содержанию, и притягиваем к нему похожие изображения, получая список.

- В ходе предыдущих этапов был получен список списков, который и предстоит слить в один результат.

Взяв приведенную выше схему за основу, мы постараемся проверить её эффективность и проверить различного рода функции слияния списков.

Для решения задачи поиска изображений в частично аннотированной базе вся работа была разделена на несколько этапов. На первом этапе необходимо выбрать направление исследования. Этому посвящён раздел работы: ”Поиск в частично аннотированной базе данных изображений как комбинирование”.

Второй этап посвящен разработке алгоритмов и проработке деталей схемы. Подробности можно узнать в главе ”Исследование”.

На долю третьего этапа – реализация системы поиска.

Четвёртый этап содержит в себе эксперимент и анализ его результатов. Также во время четвёртого этапа предстоит сравнить предлагаемых подход с похожими подходами.

Наконец пятый, заключительный этап, в котором необходимо подвести итоги и рассказать о возможном расширении области применения методов.

2 Обзор литературы

1. Попытки уменьшить семантический разрыв

Системы первого поколения использовали либо аннотации, которые прилагались к изображениям, либо выделяли их из окружающего текста. Такие методы кластеризации по окружающему тексту широко распространены. Примером может служить метод SRC, описанный в статье [9].

Насколько точно описывает та или иная аннотация картинку – вопрос не простой.

Обычно полагаются на мнение среднестатистического пользователя (то есть усредняют мнения пользователей о системе),– однако такой подход является оправданным, если постановка задачи допускает такое обобщение. Иногда эти ошибки в аннотировании можно условно принять за помеху случайной природы. Запросы к такой системе ставятся в текстовом виде. Они могут быть представлены в виде предложений или в виде набора ключевых слов. Только ограничения задачи могут определять допустимые формы текстового запроса, теоретически ограничений на общую задачу не накладывается. Можно рассматривать пользовательские запросы с булевыми связками ('И', 'ИЛИ'), с применение шаблонов для поиска, с использованием конструкций 'за исключением'.
В идеале ответом на такие текстовые запросы должен быть упорядоченный по релевантности запросу список изображений, как из подмножества изображений с аннотациями из базы, так и без них.

Gertz [2], [5] и другие разработали модель графа аннотаций для нужд искусственного интеллекта (Human Brain Project – HBP): узлы аннотаций служат для соединения так называемых узлов областей интереса изображения с концептуальными узлами управляемыми словарём. Ребра графа обозначают отношения между узлами: например, аннотации или понятия. Они также разработали структуру для запросов к графу, основанную на путевых выражениях и предикатах, которую они протестировали на прототипе системы.

2. Автоматическое аннотирование изображений

С появлением систем второго поколения стали появляться различные новые схемы поиска с текстовым запросом. Так, появилось много работ, связанных с автоматическим аннотированием изображений.

Например, в статье [8] приведен метод автоматического аннотирования изображений. Метод работает следующим образом. Пусть дано изображение без названия. На первом шаге мы используем поиск по содержанию, для поиска мно-

жества похожих изображений используются индексы высокой размерности. Поскольку картинки расположены в глобальной сети, они аннотированы названиями, окружающем текстом (по уже упомянутому методу SRC первого поколения). Далее следует шаг извлечения знаний: результаты поиска, используя технику кластеризации, выдают наивероятнейшие ключевые слова из текста к найденным изображениям. Эти слова и используются для аннотирования изображений без аннотаций. Основанный на современных поисковых методах такой метод не нуждается в предварительной настройке, но даёт весьма хорошие результаты. Потенциально он может работать с неограниченным словарём. Более сложные методы автоматического аннотирования основаны на сегментировании картинки (считается, что каждому сегменту должно соответствовать ключевое слово).

Такой подход использовался в работе [7]. Авторы статьи Herve Glotin и Sabrina Tollari предлагают использовать DIMATEX (дихотомический метод создания текстовых аннотаций). Выделяют 13 различных уровней человеческого восприятия, по 2 состояния на каждом уровне (ярко или не ярко выражен признак). Далее метод сегментирует картинку и приписывает каждому сегменту ключевое слово, основываясь на тренировочном наборе данных. Таким образом, получаются кластеры сегментов с аннотациями. Для аннотирования произвольной картинки, проверяется на каких её сегментах и при каких кластерах достигается максимум.

Также довольно распространены варианты со скрытой марковской моделью (CMM). В статье [6] использовался именно такой подход. На самом деле этот метод чем-то похож на описанный выше сегментированный подход, однако, сегменты в данном случае являются зависимыми друг от друга (совокупность сегментов образует состояние), таким образом, увеличивается точность описания в целом.

Все эти методы используют различные способы притяжения картинок по содержанию.

3. Поиск по цветовым характеристикам

Зачастую, многие авторы (например, DIMATEX) вводят свои пространства уровня человеческого восприятия для конкретных подзадач. Однако в этой дипломной работе, сделана попытка использовать уже готовые системы поиска по низкоуровневым характеристикам для получения высокоуровневого текстового поиска по изображениям.

Для этой цели, в качестве низкоуровневых характеристик, решено было использовались метрическое пространство цветовых характеристик, разработанное

Натальей Васильевой [17] и в статье [18].

В её работе использовалось цветовое пространство RGB, разбитое на 64 куба, для каждой картинки считался размер и положение так называемых цветовых пятен (пиксели изображения, факторизованные по кубам). В качестве функции расстояния использовалась эвклидова метрика.

Таким образом, в процессе комбинирования сочетаются результаты текстового поиска и притяжения по цвету, в результате чего, получаем гибридную систему, в которой в качестве запроса используется текст.

4. Задачи слияния и комбинирования

Существует много работ на тему задач слияния или комбинирования.

Общая идея таких методов состоит в том, чтобы имея некоторые результаты от работы некоторых алгоритмов или систем можно получить один результат, который будет по тем или иным критериям лучше, чем подаваемые на их вход результаты работы алгоритмов:

- либо в любом случае,
- либо в среднем, взятому по некоторому количеству экспериментов.

Критериев таких довольно много, в основном они зависят от задачи, которую необходимо решить.

Ниже будут перечислены некоторые из распространённых поисковых задач.

- Задача уточнения результатов поиска.
  Возникает в случае, когда у нас совпадают области определения алгоритмов, но сами алгоритмы различны.
- Задача дополнения результатов поиска.
  Возникает в случае, когда у нас не совпадают области определения алгоритмов, при этом алгоритмы поиска могут как совпадать, так и различаться.
- Задача сложного комбинированного поиска.
  Возникает в случае, когда области определения алгоритмов совпадают, но для нахождения результата необходимо учитывать результаты работы нескольких различных алгоритмов. Зачастую такие задачи сводятся к тривиальному пересечению областей определения, однако, иногда требуется не только выдать множество, но и расставить ранги его элементов,– в таком варианте задача усложняется.

Хороший обзор и сравнительный анализ по стратегиям приведен также в статье [1]

Таким образом, можно поставить общую задачу алгоритма слияния.

5. Общая задача слияния ранжирующих алгоритмов

Как было заявлено выше, в этом разделе будет поставлена общая задача слияния. Постановка этой задачи будет не слишком формальной, во избежание излишней перегруженности изложения без особой надобности.

Пусть у нас есть некоторое множество объектов X. Также пусть есть некоторое семейство алгоритмов $\{\Im_i\}$, которые соответственно определены и применимы (то есть всегда заканчивают работу) на некоторых подмножествах $U_i \subset X$.

Под равенством алгоритмов понимается условное равенство, то есть если один алгоритм заканчивает работу, то должен заканчивать работу и второй и результаты их на одних и тех же входных данных должны совпадать.

Под обработкой будем понимать результат работы алгоритма $\Im_i$ над данными из U_i.

Результатами работы алгоритма $\Im_i$ должны быть вектора некоторого метрического пространства Y_i. Будем обозначать ρ_i метрику, что задана на пространстве Y_i.

Теперь можно ввести понятие ранга элемента $x \in U_i$ по отношению к элементу $\hat{x} \in U_i$ на множестве объектов из U_i с использованием алгоритма (или метода) $\Im_i$ как значение метрики $\rho(\Im_i(\hat{x}), \rho(\Im_i(x)))$.

Если применить алгоритм ко всему множеству U_i и отсортировать результаты по рангу полученный список можно рассматривать как результат работы поисковой системы $\Im_i$ по тривиальному запросу $\hat{x}$.

Как видно, описанный выше случай обладает рядом недостатков. Перечислим основные из них.

- Потеря информации.
  Зачастую, алгоритм $\Im_i$ теряет часть информации об объекте выделяя у него характеристики только пространства Y_i.

- Проблема сужения области определения.
  Вводя запрос $\hat{x}$ пользователю хочется получить наиболее полный ответ (то есть, чем больше будет область действия определения алгоритма U_i, тем лучше). Но это не всегда возможно по разным причинам, в том числе и техническим. Иногда объекты могут находиться на разных серверах или в разных

базах данных и пользователю не задан алгоритм поиска использующийся на них.

- Проблема идеального метрического пространства по фиксированному критерию.
  До сих пор не известно ни одно такое метрическое пространство, в котором введённая метрика, для определённости скажем объектов мультимедиа данных, идеально бы удовлетворяла нуждам пользователей системы, даже если мы рассматриваем только один аспект информации (например, информацию по цвету). Каждый год предлагаются новые цветовые пространства характеристик и в чём-то они лучше, но в чём-то хуже уже существующих.

- Проблема абстрактного запроса.
  Многие системы вводят алгоритм $\hat{\Im}_i$ теоретически определённый на всем множестве X, который выдаёт вектор характеристик для любого запроса $\hat{x} \in X$. В этом случае возникает проблема, связанная с соответствием алгоритма $\hat{\Im}_i$ и алгоритма $\Im_i$ алгоритма (или проблема продолжения алгоритма $\Im_i$ на все множество), что не всегда возможно. Иногда эта проблема решается путем преобразования или проекции объекта на допустимое множество U_i. Приведём пример абстрактного запроса в области поиска изображений. Допустим, разработан хороший метод поиска по формам и текстуре среди изображений чёрно-белых с оттенками серого. Но пользователь ввёл в качестве запроса цветное изображение. Тогда мы можем преобразовать изображение к чёрно-белому убирая информацию о цвете. Такое преобразование может быть осуществлено несколькими способами и, по сути, не является исходным запросом, а является лишь некой его абстракцией (отсюда и название - абстрактный запрос).

Кроме того, для некоторых случаев недостаточно тривиального запроса. Ниже будет приведён пример такой системы. Вообще, запрос условно можно подразделить на два вида:

1. Тривиальный одиночный запрос
   Состоит из одного понятия-объекта. Ищутся объекты, описываемые этим понятием.

2. Сложный запрос
   Этот запрос состоит из нескольких образцов. Используя различные методы поиска для одиночного запроса, к каждому образцу притягиваются элементы, а потом методами слияния списков получают общий результат. Его можно условно подразделить на два типа:

   - Однородный запрос (гомогенный).
     В этом случае каждый образец воспринимается с одинаковой степенью доверия к нему
   - Неоднородный запрос или запрос с весами (или гетерогенный).
     Каждый элемент в запросе имеет некий вес, который потом учитывается при комбинировании или слиянии.

Для выполнения сложных запросов и устранения некоторых из перечисленных выше проблем используются алгоритмы слияния или смешивания.

На вход этого алгоритма подаются результаты работы некоторых алгоритмов ранжирования – ранжированные списки элементов из множества $H \subset X$, а на выходе получаем один ранжированный список, в котором учтены "мнения" всех ранжирующих алгоритмов.

Однако важен тот факт, что хотя алгоритм слияния и работает с алгоритмами ранжирования как с чёрными ящиками, но для успешной его работы кое-какие знания о входных алгоритмах необходимы. Эти "знания" обычно являются ограничениями задачи на входные алгоритмы для алгоритма смешивания. Ограничения задачи закрепляют вид запроса, методы и области определения методов, результаты обработки которых будут использоваться алгоритмом слияния.

Иногда оставляют некоторую свободу для реализации механизма обратной связи и подстройки.

Только установив правильные ограничения на задачу можно добиться успеха в построении алгоритма слияния.

В литературе рассмотрено множество методов по слиянию (комбинированию) результатов полученных различными способами поиска или поиска по различным данным.

Каждый такой метод поиска в совокупности с данными, с которыми он работает, принято называть свидетелем. Разделяют гомогенных (однородных) и гетерогенных (неоднородных) свидетелей.

В статье [14] (1983 год) разработали модель p-нормы в качестве обобщенного способа обработки булевых запросов. Для запроса, состоящего из n понятий $Q = \{t_1, t_2, \ldots, t_n\}$, предположим ранги понятий в документе d для указанного запроса есть $s_1, s_2, \ldots, s_n$, тогда результирующее значение ранга может быть подсчитано по приведенной ниже OR-подобной формуле,

$$S(d) = \left(\frac{s_1^p + \ldots + s_n^p}{n} \right)^{\frac{1}{p}}$$

В начале 1990-ых годов такая модель была признана не слишком эффективной, а после первой же конференции TREC (`http://trec.nist.gov/`), которая состоялась в ноябре 1992 года, появилась работа Edward A. Fox [3] и [4] в которой были проведены эксперименты по слиянию результатов полученных с помощью различных методов текстового поиска и с использованием различных данных.

Еще через год (1994) Edward A. Fox и Joseph A. Shaw выпустили статью [15] , в которой описывались несколько методов слияния:

1. CombMAX MAX(частных ответов);
2. CombMIN MIN(частных ответов);
3. CombSUM SUM(частных ответов);
4. CombANZ SUM(частных ответов) / (количество списков с ненулевыми ответами);
5. CombMNZ SUM(Individual Similarities) * (количество списков с ненулевыми ответами);
6. CombMED MED(частных ответов) - медиана частных ответов

В статье комбинировались результаты, полученные с использованием различных схем поиска. Результаты говорили в пользу таких методов слияния над p-нормами.

Эта публикация послужила началом целой серии работ по слиянию.

Авторы особенно выделили метод CombSum, как наиболее эффективный. Однако в 1995 году на новых данных TREC5 корейский исследователь Joon Ho Lee в своей статье [10] показал, что метод CombMNZ все-таки выигрывает у CombSum. Для сравнения методов он предложил использовать следующие метрики: коэффициенты релевантных и нерелевантных образцов. Таким образом,

вводились 2 показателя - коэффициент изменения количества релевантных объектов ($R_{overlap}$) и коэффициент изменения нерелевантных объектов $N_{overlap}$ в случае слияния 2-ух источников. Эти коэффициенты рассчитываются по приведенным ниже формулам:

$$R_{overlap} = \frac{R_{common} \times 2}{R_1 + R_2}$$

$$N_{overlap} = \frac{N_{common} \times 2}{N_1 + N_2}$$

Где, $N_{common}(R_{common}), N_1(R_1), N_2(R_2)$ - соответственно количества нерелевантных (релевантных) изображений в результирующем списке, в первом сливаемом списке, во втором сливаемом списке.

В статье [16] предлагают новую функцию смешивания - HSC3D взамен функциям Lee.

Данная функция должна быть линейна. Иными словами, если за f обозначит искомую функцию слияния, а вектор значений рангов входных списков (мнение различных свидетелей о ранге объекта) обозначить за S, то f можно представить в виде:

$$f(S) = \sum_{i=1}^{m} \lambda_i * s_i$$

Авторы ввели несколько свойств, которым должна удовлетворять искомая функция. Перечислим введенные свойства:

- Свойство симметричности
  $S_1 = (s_{1,1}, s_{1,2}, \ldots, s_{1,m}), S_2$ - есть перестановка (перемешивание) S_1, тогда $f(S_1) = f(S_2)$

- Свойство V-монотонности
  $S_1 = (s_{1,1}, s_{1,2}, \ldots, s_{1,m}), S_2 = (s_{2,1}, s_{2,2}, \ldots, s_{2,m})$, тогда из того, что $s_{1,i} \leq s_{2,i}$ для всех $i \in \{1 \ldots m\} \Rightarrow f(S_1) \leq f(S_2)$

- Свойство H-монотонности
  $S_1 = (s_1, s_2, \ldots, s_m), S_2 = (s_1, s_2, .., s_{m+1})$, тогда если $s_{m+1} = 0$ то $f(S_1) = f(S_2)$, иначе $f(S_1) < f(S_2)$

Следуя этим правилам, авторы предлагали взять функцию $f(S) = \sigma(m) * s$, где m - количество сливаемых списков, а $\sigma(i) = \frac{(K+1)*i}{K+i}, K = k_1 * ((1-b) + b * |D|/avdl), k_1, b$ - параметры, $|D|$ - длина документа (списка), $avdl$ - средняя длина документа (списка).

При введении функции слиянии, предлагаемой в дипломной работе также использовались эти свойства. Однако в этом случае также учитывался и тот факт, что сливаемые списки, по сути, являются гетерогенными.

Кроме стандартного комбинирования, существуют и вероятностные подходы, и подходы с использованием нечеткой логики.

Пример вероятностного подхода можно найти в работе [13] и [12]

Если рассматривать слияние гетерогенных источников и данных по поиску в изображениях и видео, то следует упомянуть работу [11].

Авторы использовали методы слияния из статей Lee и Fox'a для выявления наилучшего способа комбинирования в случае видео потоков. В статье показывалось, что методы слияния $CombSum$ и $CombMNZ$ подходят и для гетерогенных свидетелей. Авторы экспериментально показали, что $CombSum$ и $CombMNZ$ работают в среднем лучше, чем при использовании взвешенного среднего или весовых коэффициентов, добавленных в формулы $CombSum$ и $CombMNZ$.

Это служит хорошим аргументом, чтобы сравнивать нашу гетерогенную функцию с не изменёнными $CombSum$ и $CombMNZ$. А именно, рассматривать свидетелей как однородные, без учета весов для применения методов комбинирования $CombSum$ и $CombMNZ$.

В следующем разделе будет поставлена задача, решению которой посвящена данная дипломная работа.

3 Постановка задачи

1. Постановка задачи поиска изображений в частично аннотированной базе

Для организации текстового поиска в частично аннотированной базе предполагается использовать поиск по содержанию и поиск по имеющимся текстовым аннотациям.
Таким образом, нам дана некоторая база изображений. Не у всех изображений в этой базе имеется текстовая аннотация.

В нашем случае – аннотацией является набор ключевых слов.

Более того, возможно эти аннотации не совсем точны и не полностью описывают изображение. [2]

В отличие от пути автоматического аннотирования, был выбран путь комбинирования полнотекстового поиска по аннотациям и поиска по содержанию.

В основе исследования предполагалось использовать следующую схему.

1. Использовать полнотекстовый поиск по аннотациям.

2. Использовать изображения, полученные на первом шаге, в качестве образцов для поиска по содержанию. Тем самым образуется множество списков и каждому списку можно приписать вес равный рангу объекта образца при полнотекстовом поиске.

3. Использовать метод слияния ранжированных списков для комбинирования текстового поиска и поиска по содержанию.

Таким образом, передо мной стояли два вопроса:

- применимость подобного рода схемы комбинирования в общем;

- разработка наиболее эффективного способа комбинирования.

Более детальные сведения про комбинирование приведены в следующем подразделе.

[2] Вообще вопрос точности и полноты довольно сложный, поскольку эти параметры по отношению к описанию изображению трудно поддаются формализации (если такая формализации вообще возможна).

2. Поиск изображений в частично аннотированной базе данных как комбинирование

Итак, нам дана частично аннотированная база данных изображений. Эта база данных состоит из изображений, полученных с крупнейшего американского сайта фото галереи flickr((http://www.flickr.com)) [3]. В ней присутствуют как аннотированные ключевыми словами изображения, так и изображения без аннотаций. Аннотирование происходило пользователями, загружающими картинки на сайт.

Также в распоряжении имелись:

- метод поиска по содержанию изображения,
- метод полнотекстового поиска.

В качестве метода поиска по содержанию использовался метод поиска по цветовым характеристикам, предложенный Натальей Васильевой [18]. В качестве полнотекстового метода поиска использовалась система полнотекстового поиска, предложенная корпорацией Microsoft – компонент полнотекстового поиска MSFTS, включённый в поставку набора MSSQL Server 2005.

Согласно литературе для текстового поиска среди не аннотированных изображений используют следующие методы:

- автоматическое аннотирование изображений без аннотаций;
- поиск похожих изображений с ограничениями, полученными на основе информации изъятой из изображений с текстовыми аннотациями.

Был избран второй путь. Среди множества различных методов решено использовать метод комбинирования работы алгоритма поиска по тексту и имеющегося алгоритма поиска по содержанию.

Данный выбор был обоснован несколькими соображениями.

Во-первых, задача комбинирования (слияния), не зависит от какого–то конкретного метода. То есть, разработав способ, мы сможем его приспособить к другим методам поиска по содержанию.

Во-вторых, он довольно просто расширяем – таким образом, можно использовать более одного метода поиска по содержанию.

В-третьих, при данной формулировке задача легко преобразуется к задаче слияния. Действительно, пусть дополнительный алгоритм (алгоритм получения

[3]Есть ещё база данных изображений Corel и смешанная база данных Corel+flickr, но по ним эксперименты не проводились

абстрактного запроса) – есть полнотекстовый поиск. Тогда мы получим сложный запрос, состоящий из нескольких изображений с весом, равным значению ранга при полнотекстовом поиске.

4 Исследование

1. Научная часть

В разделе постановка задачи (секция 3) для поиска в частично-аннотированной базе данных изображений был выбран путь комбинирования результатов полнотекстового поиска и поиска по содержанию изображений.

Также была предпринята попытка поставить задачу комбинирования таким образом, чтобы она свелась к задаче слияния неоднородных списков.

Для этой цели комбинирование было разделено на три последовательных этапа:

- Этап полнотекстового поиска.
  В ходе этого этапа извлекаются изображение с текстовыми аннотациями и этим изображениям присваивается вес (ранг при полнотекстовом поиске).

- Этап создания сливаемых списков.
  На данном этапе к каждому из найденных изображений притягиваются, при помощи поиска по цветовым характеристикам, изображения из базы данных. Каждому списку присваивается вес, равный весу объекта–образца при полнотекстовом поиске.

- Этап слияния списков.
  Слияние имеющихся списков в один результирующий список. При этом определяются как ранги самих объектов, подаваемых на слияние, так и вес результирующего списка.

Таким образом, на последнем этапе возможно применение методов слияния ранжированных списков с весами, где вес списка принимается равным весу образуещего элемента при полнотекстовом поиске.

В такой схеме используются уже имеющиеся методы полнотекстового поиска и поиска по содержанию. Остаётся только определить метод слияния.

Этому посвящён следующий раздел.

1.1. Методы слияния

Для постоты изложения ввёдем некоторые обозначения.

Ранжированный i-ый список будем обозначать символом α_i.

Ранжированный список состоит из пар вида $(x, r_x^{(\alpha_i)})$, где x - обозначает сам объект, а $r_x^{(\alpha_i)}$ - его ранг в списке α_i.

После слияния ранг элемента x в результирующем списке будет обозначаться как $r_x^{(0)}$.

Обозначение $x \in \alpha_i$ означает, что элемент x принадлежит какой-то паре из α_i и его ранг в этом списке не равен нулю.

Теперь можно приступить к описанию методов слияния.

Идея слияния, как таковая, не нова. В литературе можно найти много информации о подобных методах слияния (см раздел 2), которые используются для различных типов задач уже давно.

В таблице 1 приведено краткое описание некоторых из них.

Название метода слияния	**Расчетная формула или описание алгоритма**	**Авторы метода, источник, другие заметки**
weighted total	$\frac{\sum \omega_{\alpha_i}^a r_{\alpha_i}^b}{\sum \omega_{\alpha_i}^a}$	Наше предположение. Для простоты, положим $a, b = 1$
Random algorithm	Значения рангов выбираются случайным образом, но в рамках выполнимости "естественных условий" (хотя бы, под номерами 1-4 - см ниже)	Теоретически любой осознанный метод должен работать лучше, чем случайный
CombSum	Суммируются ранги элементов (возможно просто суммировать умножая на вес списка)	Analyses of Multiple Evidence Combination Joon Ho Lee* С модернизацией на веса списков
Модернизированный *ombMax* и *ombMin*	Выдается минимум-максимум из значений, сравниваются сперва ранги списков, а потом ранги элементов (если элемент отсутствовал в списке предполагается, что его ранг равен 0)	Analyses of Multiple Evidence Combination Joon Ho Lee* С модернизацией на веса списков
$CombMNZ$	$CombSum \cdot NZ$, где NZ – количество элементов с ненулевым рангом	Analyses of Multiple Evidence Combination Joon Ho Lee* С модернизацией на веса списков

Таблица 1: Методы слияния списков

Не смотря на такое обилие различных методов слияния в литературе, не бы-

ло найдено метода, полностью удовлетворявшего всем условиям нашей задачи,– ибо не существует универсального метода слияния для всех типов задач из–за слишком большой общности постановки задачи слияния.

Таким образом, каждая конкретная задача требует выработки определённых свойств функции слияния, основанных на специфических требованиях к задачи.

Специфические требования нашей задачи состоят из следующих пунктов:

- рассматриваются списки с весами;
- метод должен параметризироваться как для слияния неоднородных (гетерогенных),[4] списков, так и для слияния однородных (гомогенных) списков;
- результат работы метода (ранжированный список) должен рассматриваться как результат работы некоторого метода поиска по содержанию (таким образом он сам может являться объектом слияния и, образовывая функции более высокого порядка, комбинирует слияние однородных и неоднородных источников);
- область задания объектов для всех списков одинакова ($\forall x \in \alpha_i \Rightarrow x \in X$, где X – пространство изображений, и для $\forall\, x \in X \Rightarrow \forall \alpha_i, x \in \alpha_i \wedge \exists (x, 0) \in \alpha_i$);
- каждый список рассматривается как особая точка зрения, которая дополняет общую картину, но не исключает мнения других источников (предполагается, однако, что у нас есть вес списка, который может учитываться как степень доверия той или иной точке зрения).

Приведённые выше неформальные специфические требования были учтены в виде формальных свойств искомой функции.

Некоторым из требуемых свойств удолетворяли и найденные в литературе методы (например, $CombMNZ$ и $HSC3D$ – см табл. 1).

Вот основные из них:

- Свойство симметричности
  $R_1 = (r_x^{(\alpha_1)}, r_x^{(\alpha_2)}, \ldots, r_x^{(\alpha_m)})$, R_2 - есть перестановка (перемешивание) R_1, тогда $f(R_1) = f(R_2)$

[4]Под неоднородностью в данном случае понимается, что различные списки могут быть получены при помощи различных методов поиска по содержанию, однако сопоставимы между собой в соответсвии с указанными весами, и то, что все методы поиска по содержанию определены на всем множестве изображений (или у них имеется соответствующий проектор на рассматриваемое множество)

- Свойство V-монотонности
  $R_1 = (r_x^{(\alpha_1)}, r_x^{(\alpha_2)}, \ldots, r_x^{(\alpha_m)}), R_2 = (r_y^{(\alpha_1)}, r_y^{(\alpha_2)}, \ldots, r_y^{(\alpha_m)})$, тогда из того что $r_x^{(\alpha_i)} \leq r_y^{(\alpha_i)}$ для всех $i \in \{1 \ldots m\} => f(R_1) \leq f(R_2)$

- Свойство H-монотонности
  $R_1 = (r^{(\alpha_1)}, r^{(\alpha_2)}, \ldots, r^{(\alpha_m)}), R_2 = (r^{(\alpha_1)}, r^{(\alpha_2)}, .., r^{(\alpha_{m+1})})$, тогда если $r^{(\alpha_{m+1})} \neq 0$ то $f(R_1) < f(R_2)$

Эти свойства являются необходимыми свойствами для нашего метода слияния. Необходимыми, но не достаточными. Более того, например, $HSC3D$ содержит некоторые особенности реализации, основанные на свойствах текста, которые трудно приспособить к области применения нашей задачи.

Вследствие чего функции типа $HSC3D$ не применимы в чистом виде.

Таким образом нужна другая функция которая удовлетворяет следующему расширенному набору свойств.

Искомая функция должна быть симметричной относительно своих переменных – списков (поскольку все переменные с одинаковым весом для нас равнозначимые), должна быть возможность определить функцию от нескольких переменных (заранее, возможно не известно точное количество переменных) как суперпозиции вложенных функций от 2–ух переменных. Более того, порядок не должен играть роли в формировании результата. Иначе говоря, вводя дополнительные ранжированные списки мы уточняем решение (а не строим его заново). Из вышесказанного следуют два факта:

1. Коммутативность переменных функции

2. Ассоциативность переменных функции

Дальнейшие естественные условия (используя вышесказанное) будем вводить используя 2 списка.

Итак, пусть у нас есть два списка: список α_1 и список α_2 с рангами $r_x^{(\alpha_1)}$ и $r_x^{(\alpha_2)}$ соответственно. [5]

3. Будем считать, что искомой функции для определения ранга элемента x в списке, после слияния необходимо получить на вход набор весов списков и значения ранга для этого элемента в каждом из этих списков. Иначе говоря, результирующее значение ранга для объекта x не зависит от рангов других объектов.

[5] Будем считать, что объект x всегда присутсвует в списке. А в случае его отсутствия положим, что он он присутствует с нулевым рангом.

Введем ещё одно естественное условие.

4. Пусть у нас есть N списков: $\alpha_1, \alpha_2, \ldots, \alpha_N$. В них элементу x сопоставляется следующий набор рангов $r_x^{(\alpha_1)}, r_x^{(\alpha_2)}, \ldots, r_x^{(\alpha_N)}$. Тогда после слияния ранг $r_x^{(0)}$ должен удовлетворять условию:

$$\min\{r_x^{(\alpha_1)}, r_x^{(\alpha_2)}, \ldots, r_x^{(\alpha_N)}\} \leq r_x^{(0)} \leq \max\{r_x^{(\alpha_1)}, r_x^{(\alpha_2)}, \ldots, r_x^{(\alpha_N)}\}$$

Взаиморасположение объектов (порядок) определится, когда будет определен ранг каждого элемента.

Теперь для выполнения остальных неформальных требований и учёта весов необходимо ввести свойство взвешенной стабилизации высокоранговых элементов, которое мы назовём правилом конусов. [6]

Пусть у нас есть некоторый список. Рассмотрим вес данного списка как показатель уверенности (доверия), что значения приведенные в данном списке, соответствуют реальным значениям нашего запроса.

Кроме того, внутри каждого списка введём непрерывную на отрезке [0..1] убывающую от значений ранга функцию g, характеризующую вероятность ошибки в вычислении ранга для метода, с помощью которого был построен этот список.

Для функции g должны выполнятся следующие условия.

- Выполнение граничных условий:

  1. Если есть список с весом, близким к единице и в нем есть объект с рангом почти равным единице, то вероятность того что ранг изменится после слияния очень мала;
  2. При слиянии 2-ух списков, один из которых обладает весом почти равным 0. Он почти не вносит вклада в слияние (за исключением добавления элементов). В результате чего вероятность того что в среднем его элементы сильно изменят свое положение - высока. В то же время, низка вероятность того, что ранги их не изменятся

- Если есть два элемента ($x \in \alpha_1 y \in \alpha_2$) из списков с различными весами ($\omega_1 > \omega_2$) и ранг элемента x равен рангу y в этих списках, то в результирующем списке у них должен быть различный ранг, причем ранг $r_x^{(0)}$ больше чем ранг $r_y^{(0)}$

[6] Идея такого правила была почёрпнута из правила метода $HSC3D$, названного авторами гравитационным правилом в $3D$ пространстве – оттуда идёт часть названия $HSC3D$. Но в чистом виде данное правило не применимо для нашей задачи, однако в дань традиции буква G–gravitation была оставлена в названии нового метода

- Малому приращению веса списка соответствует незначительное изменение его влияния на ранги в результурующем списке
- Чем меньше ранг - тем больше вероятность, что данный ранг будет изменен (больше свобода разброса значения).
- Как уже было сказано, функция g непрерывна на [0..1].
- Если вывести функцию вероятности разброса от веса списка при каком-то зафиксированном ранге - получим непрерывную, монотонную функцию.

5. Вышеперечисленные условия обозначим как условие взвешенной стабилизации высокоранговых элементов или условие конусов.

Пункты (1)–(4) не учитывают веса самих списков. Пункт (5) устраняет этот недочёт.

Сейчас будет предложено решение без учёта условия (5), а далее будет предложено решение с учётом всех условий.

Пусть искомая функция есть:

$$r_x^{(0)} = \max\{r_x^{(\alpha_1)}, r_x^{(\alpha_2)}, \ldots, r_x^{(\alpha_N)}\}.$$

Действительно, такая функция отвечает условиям коммутативности и ассоциативности. Также, естественным образом сооблюдаются условия (3,4).

Однако условие (5) явно нарушается (веса списков нигде не учитываются).

Сейчас введем функцию, которая будет удовлетворять и 5-ому условию.

Для начала вспомним функцию взвешенного среднего для вектора, в которой также выполняются свойства (1)–(4)
$R = (r_x^{(\alpha_1)}, r_x^{(\alpha_2)}, \ldots, r_x^{(\alpha_m)})$.

$$f(R) = \frac{\sum\limits_i r_x^{(\alpha_i)} \cdot \omega(\alpha_i)}{\sum \omega(\alpha_i)},$$

где $\omega(\alpha_i)$ - есть вес списка α_i.

Для выполнения свойства (5) заменим вес элемента $\omega(\alpha_i)$ на значение функции стабилизации высокоранговых элементов $g(r_x^{(\alpha_i)}, \omega(\alpha_i))$. Такую функцию будем называть гравитацинной функцией.

$$f(R) = \frac{\sum_i r_x^{(\alpha_i)} \cdot g(r_x^{(\alpha_i)}, \omega(\alpha_i))}{\sum g(r_x^{(\alpha_i)}, \omega(\alpha_i))}.$$

Поскольку g весовая функция, на неё накладывается условие положительной определенности.

В качестве такой g было предложено взять следующую функцию:

$$g(r_x^{(\alpha_i)}, \omega(\alpha_i)) = \omega^2(\alpha_i) \cdot \left(r_x^{(\alpha_i)} + \frac{1}{12}\right)^4.$$

Такая форма объясняется условиями нашей задачи:

- зависимость ранга и веса должна быть мультипликативной;
- при $\omega \equiv 0$ мы не должны учитывать значение этого списка (элемент сколь высого ранга в нём бы не содержался);
- при $r_x^{(\alpha_i)} \equiv 0$ (означает, что объект x не присутствует в списке (α_i)) тоже должно оказывать на результат некоторое воздействие, а именно, уменьшать значение результирующего ранга, если вес списка не был равен нулю и этом списке данный элемент был бы не с нулевым рангом;
- фунция должна быть мотононна и непрерывна по обоим параметрам.

Константы степеней $(2, 4)$ и сдвига ранга $(\frac{1}{12})$ определяются экспериментальным образом исходя из особенности данных и конкретных применяемых методов.

Таким образом наша функция разрешает ниже приведённые проблемы.

- **Случай 1.**
  - **Ситуация.** Пусть у нас есть список с весом близким 1 и в нем есть элемент с рангом почти равным 1 (то есть, согласно требованиям на выходе у элемента должен быть ранг ~ 1). Однако, предположим, есть список с весом $\sim 0{,}5$ в котором данный элемент отсутствует (ранг $= 0$). Считая по формуле взвешенной суммы получаем при смешивании списков значение $\sim 0.(6)$, что идёт в разрез с пожеланиями к функции – не сильно изменять ранг у высоких элементов.
  - **Причина.** Нарушено условие стабилизации высокоранговых элементов: в знаменателе не рассматривается какой ранг был у элемента в списке $\Rightarrow$ все элементы равноправны, то есть элементы с меньшим рангом имеют "такие же" права влиять на результат.

- **Случай 2.**
  - **Ситуация.** Предположим, имеются два списка для слияния с одинаковыми весами. Формула взвешенной суммы выродится в среднеарифметическую сумму рангов. Предположим отсутствующие элементы будут восприниматься как элементы с рангом равным 0. Предлагается сделать следующее умозрительное построение. Пусть у нас есть элемент только в одном из таких "равнозначных" списков, и пусть его ранг в этом списке = 0,8. При слиянии, например, методом среднего арифметического ранг его будет равен 0,4. Теперь предположим, что этот элемент появится во втором списке с рангом почти равным 0. Однако если следовать формулам типа CombMNZ, CombSum и взвешенное среднее мы получим во втором случае результирующий ранг больше, чем в первом. Информации стало больше. И многие методы работают по схеме: больше информации - более высокий ранг элемента. Бесспорно, в таком правиле есть своя логика - на верх списка попадают элементы о которых нам известно больше. Отчасти, предложенный метод также базируется на этой логике, однако, только когда речь идёт о достаточно высокоранговых элементах вектора. В случае с сильно низкоранговыми элементами ситуация иная,– чем ближе к нулю, тем слабее сила удерживающая элемент на месте. Граница эта не очень чёткая и определяется из опыта путём подбора коэффициентов. Естественно предположить, что результирующий ранг должен быть больше чем в первом случае. Ибо приписываемый нулевой ранг означает лишь отсутсвие элемента в списке, а не реальное значение нулю его ранга. К тому же хочется, чтобы ранги, расположенные в окресности нуля не слишком сказывались на значении результирующего ранга, в случае если у нас есть список с большим весом и этот элемент в нём имеет значительно больший вес. Таким образом, подобные формулы не верны в общем виде.
  - **Причина.** Нарушено условие конусов: в знаменателе не рассматривается какой ранг был у элемента в списке $\Rightarrow$ все элементы равноправны, то есть элементы с меньшим рангом имеют "такие же" права влиять на результат, что и элементы с большим рангом – вопреки условию конусов.

1.2. Алгоритмы и оценки

Однако, поиск значения ранга элемента во всех списках довольно трудоемкая задача.

Максимальное количество предполагаемых сливаемых списков для текстового поиска около 20. А среднее количество элементов в каждом списке – 100.

Формула подсчёта ранга довольно простая.

Поэтому для сливания списков не требуются значительных вычислительных мощностей.

Для вполне приемлемых по времени результатов достаточно изначально отсортировать списки и составить список векторов последовательным сливанием списков (для каждого списка один проход).

Однако для слияния некоторых методов (например, результатов текстурного поиска и поиска по формам) данную формулу можно преобразовать к виду последовательного смешивания 2-ух списков.

$$r_x^{(0)} = \frac{g(r_x^{(\alpha_1)}, \omega(\alpha_1)) \cdot r_x^{(\alpha_1)} + g(r_x^{(\alpha_2)}, \omega(\alpha_2)) \cdot r_x^{(\alpha_2)}}{g(r_x^{(\alpha_1)}, \omega(\alpha_1)) + g(r_x^{(\alpha_2)}, \omega(\alpha_2))}$$

Однако, следует учитывать, что вес результирующего списка после слияния вычисляется по формуле:

$$\omega_0 = (\omega_{\alpha_1}^2 + \omega_{\alpha_2}^2)^{\frac{1}{2}}.$$

Такому методу была присвоена аббревиатура $WTGF$ (weighted total with gravitation function взвешенное среднее с гравитационной функцией)

Оптимальным способом является предварительная сортировка всех списков по идентификатору элемента.

Таким образом мы можем сливать списки последовательным просмотром, а не осуществлять каждый раз поиск ранга элемента для каждого списка.

Однако это можно осуществлять двумя разными способами.

Вариант 1. Поэлементно вычленить из всех списков одинаковые элементы и использовать стандартную формулу. Ниже данная схема представлена схематически.

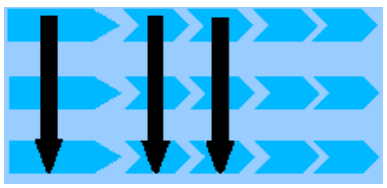


Рис. 1: Схема поэлементного слияния

Вариант 2. Сливать списки сперва попарно, затем попарно из получившегося результата с предыдущего шага и т.д. (см рисунок справа – нижние кружки – исходные списки)

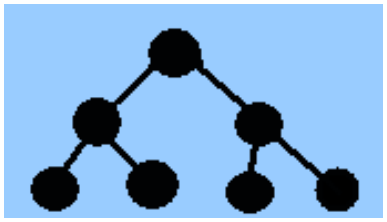


Рис. 2: Схема поэлементного слияния

У обоих вариантов есть свои плюсы и минусы. Оба они поддаются распараллеливанию. Основной минус первого варианта – поиск одинаковых объектов в разных списках медленно работает и сложно реализуем (если реализовывать оптимально). Основной минус второго варианта – хранение "лишних" переменных.

На данный момент используется смешивание слиянием отсортированных списков (вариант 2'). Этот вариант скорее ближе к пункту 2. Основные шаги алгоритма следующие:

1. Производится сортировка элементов списков (по идентификатору объекта)
2. В конец добавляется ограничивающий (максимальный) элемент
3. Производится сортировка слиянием путем соединения 2-ух списков по приведенным выше формулам.

Такая реализация добавляет к методу $WTGF$ суффикс MT. Таким образом полное название метода $WTFG_MT$. Описание пакета, в котором реализован данный алгоритм можно найти в приложении в разделе 1.

Алгоритмы реализованы в классах (подробности описаны в разделе 1. приложения):

- MergeOperation – вариант 2;
- FasterMergeOperation – вариант 2'.

Возможно также стоит проэкспериментировать с алгоритмами распараллеливания. Попробовать в одной нити сливать больше чем 2 списка.

Осталось определится с методами оценки эффективности алгоритмов. В статье Lee ... были предложены коэффициенты $R_{overlap}$ и $N_{overlap}$. Мы будем также использовать эти коэффициенты. Итак, будем считать, что R_0 - есть количество релевантных запросу изображений в некотором объёме результирующего списка, а N_0 - количество нерелевантных в результирующем. Пусть также R_i и N_i - соответственно количество релевантных и не релевантных в таком же объёме i-ого сливаемого списка, который подаётся на вход алгоритма слияния.

И пусть алгоритм слияния использовал M списков. Тогда коэффициенты $R_{overlap}$ и $N_{overlap}$ определяются по формулам:

$$R_{overlap} = \frac{M \cdot R_0}{\sum\limits_{M} R_i};$$
$$N_{overlap} = \frac{M \cdot N_0}{\sum\limits_{M} N_i}.$$

В работе будем использовать эти коэффициенты.

Однако, поскольку теоритечески в списках могут участвовать все изображения в базе будем рассматривать коэффициенты $R_{overlap}(k)$ и $N_{overlap}(k)$, где берутся первые k элементов от списков.

2. Реализация

Вышеописанные алгоритмы были реализованы с использованием современных технологий. Сама база данных была размещена на MSSQL Server 2005 (предоставлено MSDAA).

Алгоритмы были написаны на языке C#. Для интеграции с MSSQL Server 2005 использовались возможности среды $.NET$.

Код разрабатывался на тестовой версии Visual Studio 2005.

Архитектура приложения и схемы базы данных с комментариями по установке и настройке приведены в приложениях...

5 Эксперимент

1. Обзор существующих методов проведения эксперимента

После анализа существующей литературы были предложены методы проведения эксперимента, представленные на таблице 2.

1.1. Описание метода Fuzzy logic evaluation

Метод Fuzzy logic evaluation для оценки работы алгоритмов слияния работает следующим образом:

- На вход методу подается N списков (с набором весов $\{\omega_1, \omega_2, \ldots, \omega_N\}$)
- Вычисляем коэффициент $K = \frac{1}{\sum \omega_i}$
- Коэффициент пропорциональности K используется для построения термов нечетких множеств
- Выделяются основные термы:
  - Изменилось в худшую сторону
  - Нет изменений
  - Изменилось в лучшую сторону
- Дополнительно вводятся оттенки для образования новых термов:
  - Сильно
  - Незначительно
- С пользователя снимаются следующие показания:
  - Количество релевантных картинок (показывается первые 9 картинок списков) – переменная рациональная оценка
  - Качество релевантных картинок – переменная эмоциональная оценка

- Основной вид нечетких множеств (при коэффициенте $K \cdot \omega_i = 1$) представлен на рисунке 3

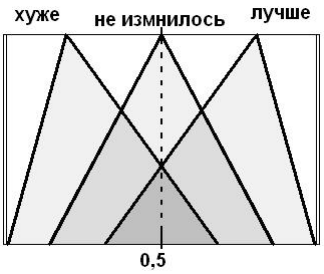


Рис. 3: Основной вид нечетких множеств (при коэффициенте $K \cdot \omega_i = 1$)

В зависимости от коэффициента $K \cdot \omega_i$ изменяется угол при вершинах треугольников (которые лежат не на осях). В зависимости от употребляемого оттенка сдвигается вершина треугольника (которая лежит не на оси, с таким расчетом, чтобы углы при основаниях треугольника (вершины на осях) не были бы тупыми и вершины основания не изменяются) Таким образом, мы вводим значение N лингвистических переменных "качество слияния – рациональная оценка" и N переменных "качество слияния – эмоциональная оценка"

1.2. Описание метода Mark–Recapture evaluation

Предлагается добавить в таблицу объектов (Image) ещё одну колонку – "user annotation". Эта колонка заполняется для тех объектов, у которых отсутствует какая-либо аннотация. Возможно несколько вариантов заполнения данной колонки.

1. Добавлять из уже готовой БД, разбитой по категориям, и аннотировать названиями этих категорий (например, БД Corel)

2. Фильтровать имеющиеся картинки в БД и удалять из них аннотацию. Т.е. пользователь пишет запрос – ему выдается множество картинок – далее идет выбор тех картинок, которые наиболее соответствуют запросу. У этих картинок удаляется аннотация - и добавляется предложение поиска в качестве "user annotation". Однако при таком подходе есть опасность удалить из БД существенно влияющие на поиск образующие объекты.

3. Выбирать из БД случайным образом группу картинок и аннотировать только наиболее характерно-выраженные из них. Этот этап подготовки стоит делать

при помощи человека, ибо только он может оценить насколько аннотация соответствует содержанию.

Эти же слова аннотации будут добавляться в специальную тестовую БД запросов. Когда будет собрано достаточное количество данных можно запускать тестирование. Поясним работу метода. Пусть некоторое множество объектов отнесли к категории A (проаннотировали словом "A"), однако, данные аннотации не участвуют в полнотекстовом поиске. Таким образом, изображения из этого множества могут появиться в результате за счет притяжения, допустим, по цвету. Для каждого списка сразу после притяжения подобных картинок считаем значение традиционных параметров полноты/точности. Понятно, что это будут весьма относительные полнота и точность, но при таком эксперименте все оценки будут весьма относительны. Потому как мы не можем знать, сколько в базе релевантных изображений запросу, у которых нет "user annotation". Однако мы можем предполагать, что если мы проаннотировали "user annotation" достаточное количество изображений и у нас была хорошая выборка, то с большой вероятностью мы можем гарантировать некоторое покрытие этих объектов. Этот метод позволяет сравнивать различных методов в полуавтоматическом режиме.

1.3. Описание метода Relevance–Nonrelevance overlap evaluation

Этот метод оценки использовался для проверки эффективности функций слияний уже давно (см описание литературы 2) благодаря своей простоте и достаточной показательности.

Для того, чтобы вычислись коэффициент перекрытия релевантных изображений $R_{overlap}$ (см формулу 1) и коэффициент перекрытия не релевантных изображений $N_{overlap}$ (см формулу 2) необходимо сосчитать количество релевантных и не релевантных запросу изображений во входных и результирующем списках.

Оценка не релевантных и релевантных изображений происходила следующем образом.

Все списки, как до слияния, так и после, объединялись в теоретико–множественном смысле (первые N картинок от каждого списка).

Далее это множество предлагалось человеку для оценки. Оценка производилась путём выделения релевантных и не релевантных изображений из предлагаемого множества.

Поскольку, человеку трудно оценить за один раз более чем 9 объектов – пользователю желательно показывать не более 9 сущностей для оценки.

Результаты эксперимента заносились в базу данных.

2. Описание эксперимента

Для оценки было решено использовать метод Relevance–Nonrelevance overlap evaluation. Поскольку он довольно прост в реализации, опробован ранее и в то же время наиболее эффективен для оценки качества самого слияния.

Для этих целей было создано специальное приложение – RosaGallery Experiment Test Bench (внешний вид приложения можно увидеть на картинке 4).

Описание системы можно найти в приложении в разделе 3.

В ходе эксперимента была создана тестовая база данных. Которая содержала 100 тестовых запросов. Было принято решение использовать только односложные запросы.

Также была подготовлена база данных изображений с крупнейшего сайта сетевой любительской фото галереи (`www.flickr.com`) – условно будем называть её flickrDB.

Для эффективности проведения эксперимента результаты работы метода поиска по содержанию были закешированы в базе(иначе время ожидания ответа было бы велико).

Эксперимент состоял в следующем: пользователю показывались списки изображений и заранее заготовленное слово запрос. Эти изображение есть перемешанные случайным образом и объединённые в теоретико–множественном смысле результаты обработки запроса различными методами. При этом рассматривались только 15 первых в списке элементов. Далее пользователю предлагалось выбрать один из вариантов ответа для каждого изображения:

- (-1) – не соответствует текстовому запросу,
- (0) – нейтрально текстовому запросу,
- (1) – соответствует текстовому запросу.

Пример приложения можно видеть на рисунках (см рис. 4). Благодаря гибкой системе настройки легко можно сменить дизайн и параметры эксперимента путём редактирования файла конфигурации (см приложение раздел 4.).

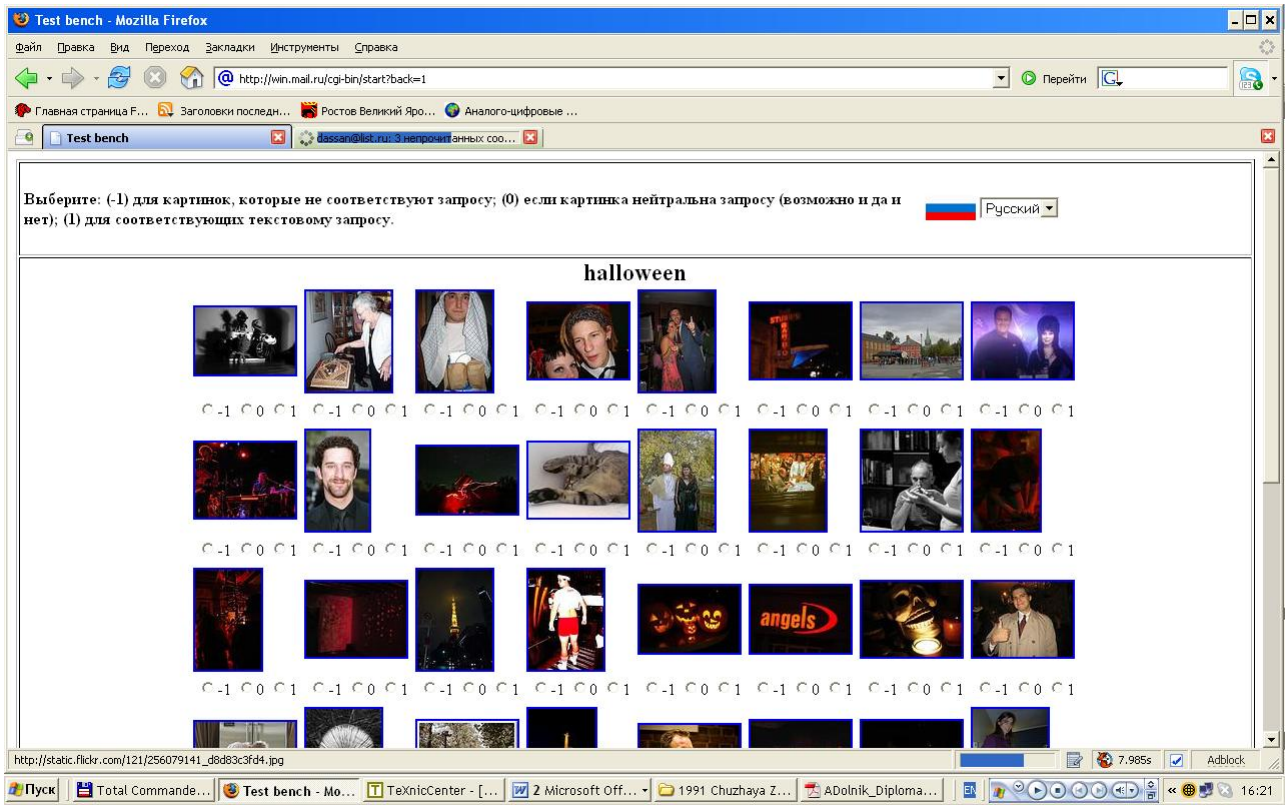


Рис. 4: Система RosaGallery 2 системы настройки

3. Анализ результатов

3.1. Сбор статистики

Продолжительность эксперимента – 2 недели (14 дней).

За это время на сайте зарегистрировались 136 человек. Однако внесли свой вклад в эксперимент (хотя бы один раз прошли тест) – 98 человек.

Всего было поставлено 484 опыта.

Таким образом, в среднем, каждый пользователь дал примерно 5 ответов (точнее 4.93877551).

Распределение количества пользователей заходивших на сайт по времени суток и датам почти равномерное.

В основном опрашиваемые являлись русскоговорящими пользователи интернет.

Возрастная категория опрашиваемых пользователей от 18 до 35 лет.

Сутки были разбиты на 3 временных интервала по 8 часов:

- 07:00 – 15:00 (утреннее и дообеденное время),
- 15:00 – 23:00 (послеобеденное время и вечер),
- 23:00 – 07:00 (ночь).

Тогда, как показал опыт, результаты эксперимента зависят от времени суток. Утром оценки более ‘пессимистичны’, чем в послеобеденное время, а ночью — самые ‘оптимисты’.

Причем, ночью пользователи скорее укажут (0) (нейтрально) или не укажут значение вообще, чем поставят (-1).

Об этом можно судить по сводной гистограмме 5.

Гистограмма была получена следующим способом:

- выбирались лишь те значения эксперимента, которые соотвествовали интересующему временному интервалу;
- для каждого результата эксперимента(один запрос представленный оцененный одним человеком), считался средний ответ пользователя $ans \in [-1 \dots 1]$ по всем имеющимся изображениям, подаваемым на оценку[7];
- полученные коэффициенты ans использовались для построения гистограммы (количество экспериментов со значением ans).

[7]Таким образом если для запроса A, оцененного пользователем $\mathbb{A}$ получалась оценка равная 1 означает, что пользователь $\mathbb{A}$ отмечал только релевантные изображения запросу

По этим данным и составлялась гистограмма 5.

Для гистрограмм были построены аппроксимирующие нормальные распределения.

Заметно, что значения сильно расходятся в зависимости от времени суток.

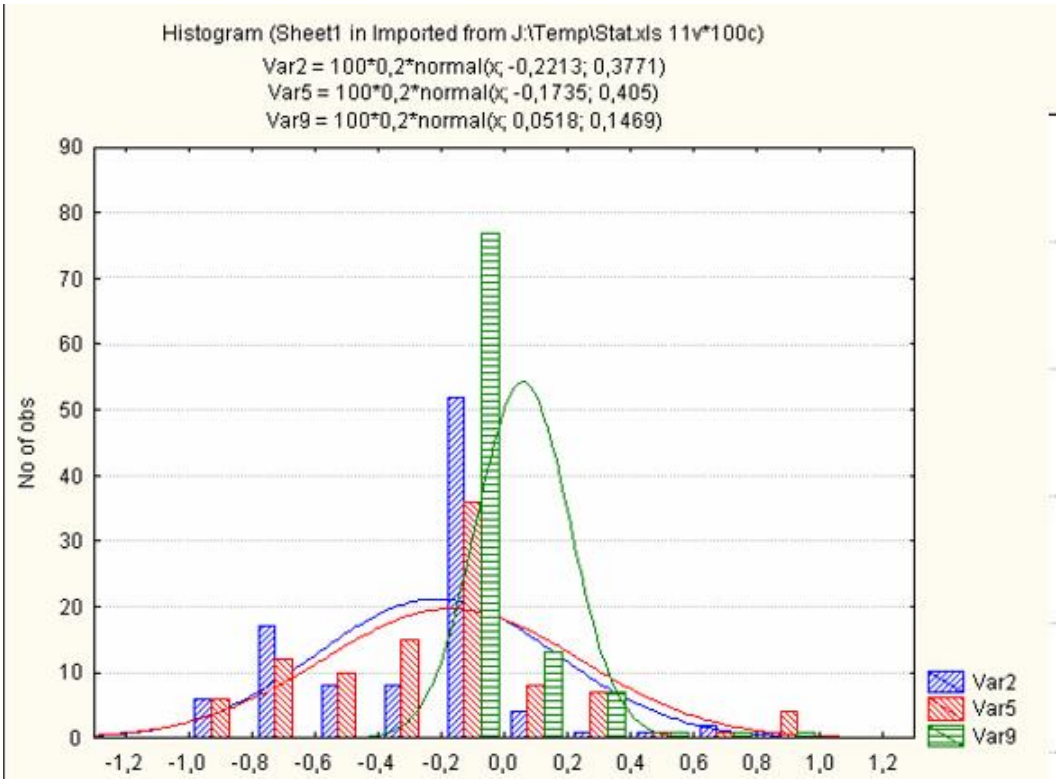


Рис. 5: Гисторамма сравнения ответов пользователей в зависимости от периода суток

Ниже приводится некоторая расшифровка обозначений гистограммы 5:

- $Var2$ – утреннее и дообеденное время;
- $Var5$ – послеобеденное время и вечер;
- $Var9$ – ночь.

3.2. Результаты

Будем использовать нижеследующую формулу 1 для просчёта коэффициента перекрытия ($R_{overlap}$) соответствующих запросу картинок:

$$R_{overlap}(x) = \frac{M \cdot R^{(0)}(x)}{\sum_{i}^{M} R^{(\alpha_i)}(x)}, \tag{1}$$

в которой $R^{(\alpha_i)}(x)$ – количество соответствующих запросу объектов в первых x элементах списка α_i, усреднённое по всем различным объекта, а $R^{(0)}(x)$ – усреднённое по всем различным объектам среднее количество соответствующих запросу в первых x элементах результирующего списка.

Если такой коэффициент больше единицы,– это означает, что в результирующем списке больше релевантных, чем в среднем в тех списках, что подавали на слияние.

Чем больше данный коэффициент – тем лучше.

Второй коэффициент, который использовался для оценки качества методов – есть коэффициент перекрытия не релевантных запросу изображений $N_{overlap}$.

Коэффициент $N_{overlap}$ вычислялся формуле 2 аналогичной формуле 1.

$$N_{overlap}(x) = \frac{M \cdot N^{(0)}(x)}{\sum_{i}^{M} N^{(\alpha_i)}(x)}, \tag{2}$$

где $N^{(\alpha_i)}(x)$ – количество не соответствующих запросу объектов в первых x элементах списка α_i, усреднённое по всем различным объекта, а $N^{(0)}(x)$ – усреднённое по всем различным объектам среднее количество не соответствующих запросу в первых x элементах результирующего списка.

Таким образом чем меньше коэффициент $N_{overlap}(x)$ – тем лучше работает метод.

Однако по коэффициентам $N_{overlap}$ и $R_{overlap}$ стоит сделать одну существенную оговороку.

Поскольку пользователю показывалось лишь ограниченное число изображений [8], нельзя точно вычислить коэффициенты $N^{(\alpha_i)}(x)$ и $R^{(\alpha_i)}(x)$ более чем для 3 первых элементов списка.

Действительно, согласно описанию эксперимента (см табл. 2), пользователю показывались объединённые в теоретико–множественном смысле результаты работы нескольких методов слияния ($CombMNZ$, $WeightedTotal$, $Random$,

[8] В нашем случае не больше 50 изображений за раз

$WTGF\_MT$) и, в среднем, первые 20 элементов от Z–обхода [9] списков, подаваемых на слияние.

Таким образом, например, при слиянии 10 списков в среднем имелись данные только о первых двух элементах в сливаемых списках. Остальные элементы оставались вне поля зрения пользователя.

Хотя для самих результатов работы нам удавалось получить полную оценку по всем 20 первым элементам (в большинстве своём их значения сильно перекрывались).

Следует также учитывать, что кроме как оценки соответствует запросу или не соответствует была введена оценка "нейтрально по отношению к запросу".

И последнее, конечно, следует принимать во внимание вышеупомянутые факты, путём учёта не столько самих значений коэффициентов, сколько характера поведения функций и их взаиморасположение относительно друг друга, ибо указанные характеристики независимы от указанных особенностей.

Стоит также пояснить коэффициент $delta$ для поиска по цветовым характеристикам. Упомянутый параметр используется для установки глубины просмотра при поиске в индексе. Таким образом, если коэффициент $delta$ равен, например, $0,03$,– будут выданы изображения, которые по расстоянию от образца в принятой метрике отличаются не более чем на $0,03$.

Теперь можно перейти собственно к рассмотрению самих графиков.

Графики значения коэффициента $N_{overlap}$ представлены на графике 7 и 8 для 10 и 5 списков.

Ниже приведены графики 6, 9 значения коэффициента $R_{overlap}$ для различных функций слияния в зависимости от количества рассматриваемых элементов от начала списков.

В ходе эксперимента проводилось тестирование при различных параметрах системы поиска по содержанию. Каждый график – есть поведение системы при различных конфигурациях системы поиска по содержанию

На графиках 6(a), 9(a), 7(a), 8(a) представлены результаты эксперимента при сливании списков с параметром $delta = 0.03$.

На графиках 6(b), 9(b), 7(b), 8(b) представлены результаты эксперимента при сливании списков с параметром $delta = 0.07$.

Как можно видеть при 10 сливаемых списках предложенный метод $WTGF\_MT$ выигрывает у методов: $Random$, $WeightedTotal$, $CombMNZ$. Однако, можно заметить, что при различных конфигурациях системы отрыв меняется. Таким образом, чем больше коэффициент $delta$ тем лучше поиск (что

[9] Под Z–обходом подразумевается обход сперва всех первых элементов всех списков, далее всех вторых, потом всех третьих и т.д.

естественно).

Графики на рисунках 9 и 6 показывают общее превосходство метода $WTGF\_MT$ над методами $WeightedTotal$, $Random$, $WeightedTotal$, $CombMNZ$.

Однако присутствуют моменты, когда метод $WTGF\_MT$ проигрывает по эффективности методам $WeightedTotal$.

Ситуация по коэффициенту N_{ovelap}, которую можно увидеть на графиках 8 и 7, несколько хуже для метода $WTGF\_MT$. Вообще, самым оптимальным по показателям N_{ovelap} является график функции $Random$, однако эти данные нельзя считать правдоподобными, поскольку такое положение графика обуславливается тем, что для случайного метода слишком мало вообще хоть как-то оцененных элементов в выходном списке. То есть могут возникать случайные элементы из конца списков которые не предлагались пользователю для оценки.

Также была собрана статистика по количеству релевантных и не релевантных изображений в зависимости от позиции в тексте. Слева на рисунке 10(a) изображены графики релевантных изображений при 10 сливаемых списках, а справо на рисунке 10(b) – не релевантные.

Видно, что при слиянии 10 списков на картинке 10(a) сначала лидирует метод $WTGF\_MT$.

Также заметно, что 10(b) также метод $WTGF\_MT$ принимает минимальное значение до четвёртой позиции.

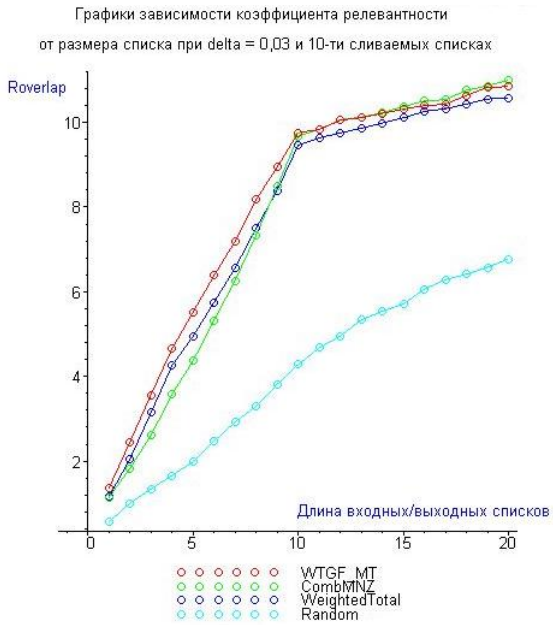
Таким образом, для метода $WTGF\_MT$ свойственна тенденция "перемещать" релевантные изображения к началу списка более чем всем остальным методам.

Таким образом, по различным показателям метод $WTGF\_MT$ действительно можно считать удачным экспериментом.

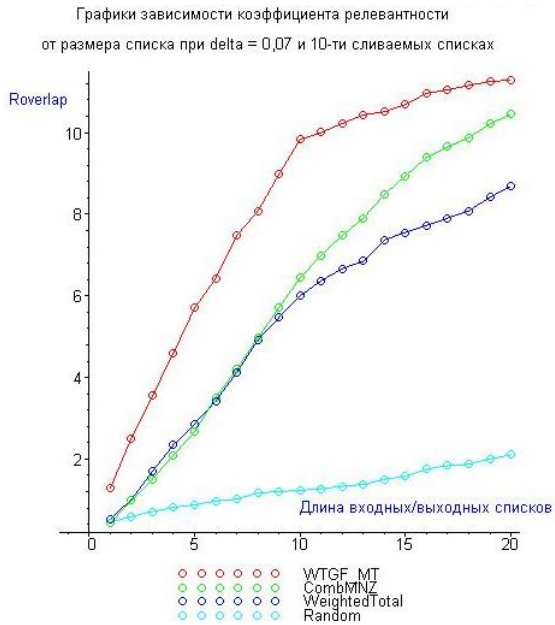
Наши предположения по свойству функций оказались верны, что было подтверждено на практике.

Название	**Описание эксперимента**	**Оценка результата**
Relevance–Nonrelevance Overlap Evaluation	Описание можно найти в разделе 1.3.	Используются приведённые ниже формулы для подсчёта $R_{overlap}$ (см 1) и $N_{overlap}$ (см 2). Большая часть работы перекладывается на пользователя.
Mark–Recapture Method	Прогнать метод на заранее подготовленных запросах. Проверять попала ли картинка подсадки в результирующее множество и какую позицию она в нем заняла. О том, как создавать картинки подсадки – см в разделе 1.2.	Для каждого списка сразу после притяжения подобных картинок считаем значение параметров аналагов традиционных полноты и точности. Понятно, что это будут весьма относительные полнота и точность, однако при таком эксперименте все оценки будут весьма относительны. Потому как мы не можем знать, сколько в базе релевантных изображений запросу, у которых нет user annotation. Этот метод позволяет сравнивать различных методов в полуавтоматическом режиме.
Fuzzy logic evaluation	Метод описан в параграфе 1.1.	N значений лингвистических переменных комбинируются при помощи min. Процедура дефазификации состоит предполагается взять $\min_{arg}(\max(B))$, где B – результирующее нечеткое множество. Для начала оценивается отдельно рациональная и эмоциональная оценка. При необходимости можно преобразовать данную оценку к одному численному значению.
Competition Statistic	Пользователю одновременно предлагается сравнить между собой результаты работы различных методов слияния (сравнение устроено таким образом, что пользователь не знает оценку какого метода он дает). Далее ему предлагается расставить методы по местам.	После накопления некоторой статистики (за каждое место даются баллы – можно определить метод–лидер)

Таблица 2: Способы оценки эффективности методов слияния

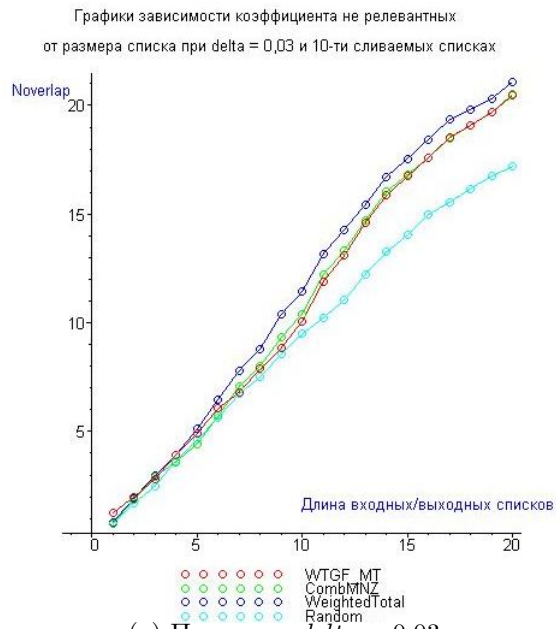


(a) Параметр $delta = 0.03$

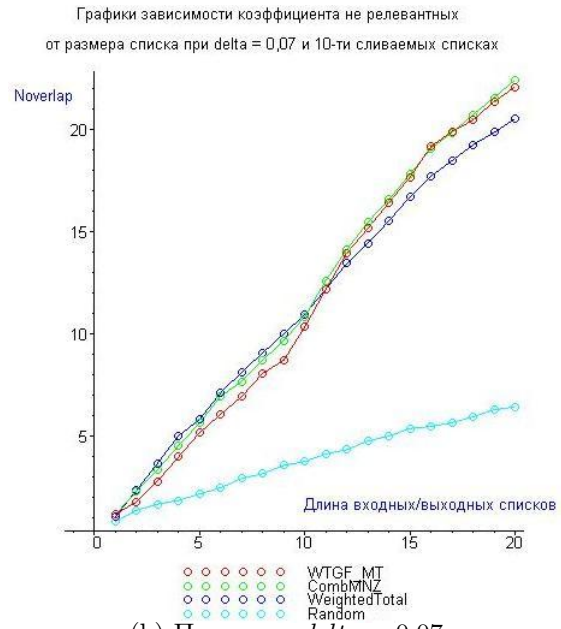


(b) Параметр $delta = 0.07$

Рис. 6: Графики значения коэффициента $R_{overlap}$ для различных функций слияния в зависимости от количества рассматриваемых элементов от начала списков

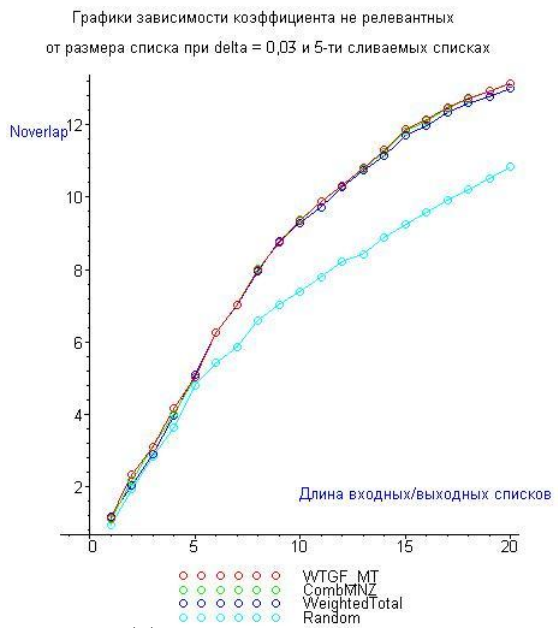


(a) Параметр $delta = 0.03$

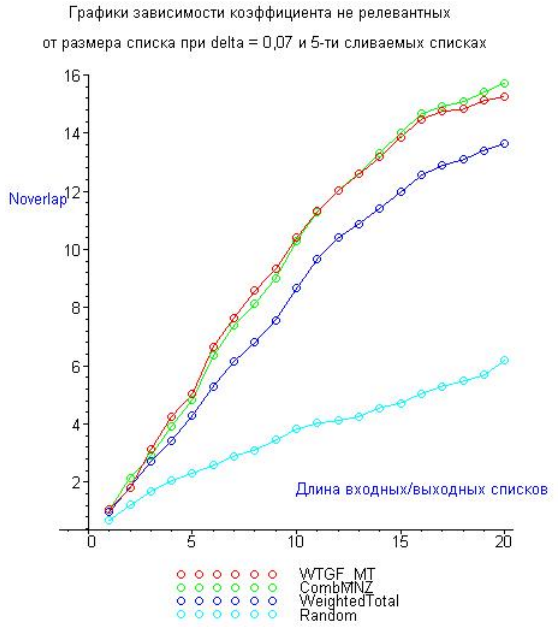


(b) Параметр $delta = 0.07$

Рис. 7: Графики значения коэффициента $N_{overlap}$ для различных функций слияния в зависимости от количества рассматриваемых элементов от начала списков

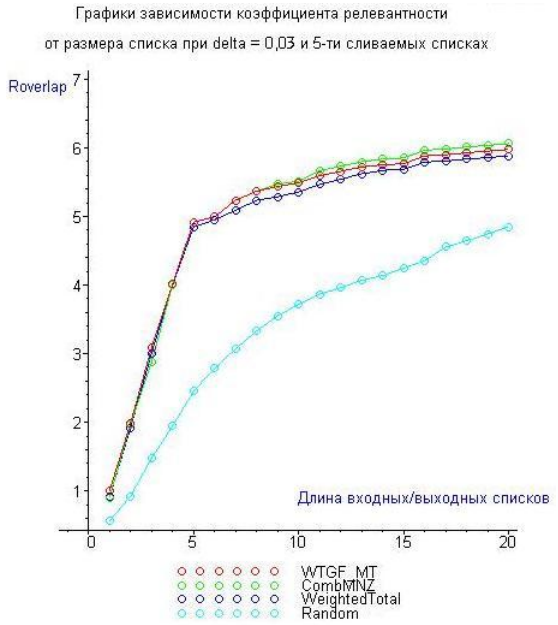


(a) Параметр $delta = 0.03$

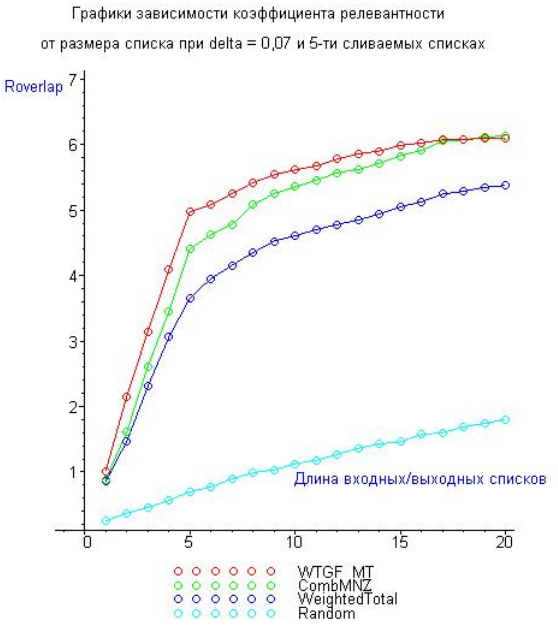


(b) Параметр $delta = 0.07$

Рис. 8: Графики значения коэффициента $N_{overlap}$ для различных функций слияния в зависимости от количества рассматриваемых элементов от начала списков

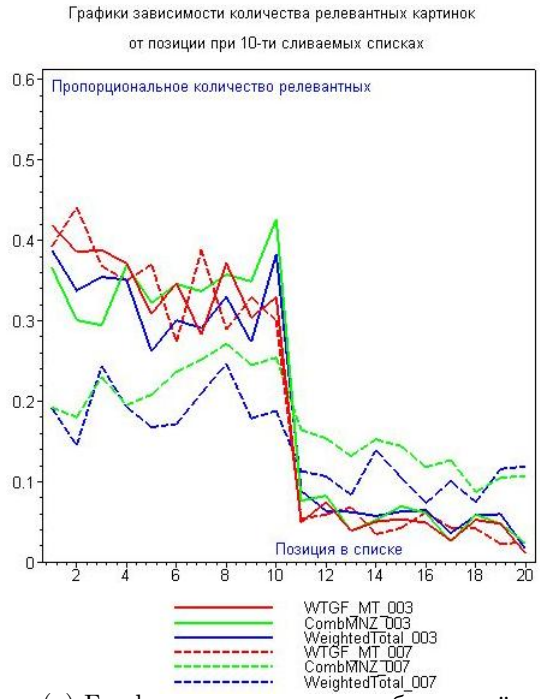


(a) Параметр $delta = 0.03$

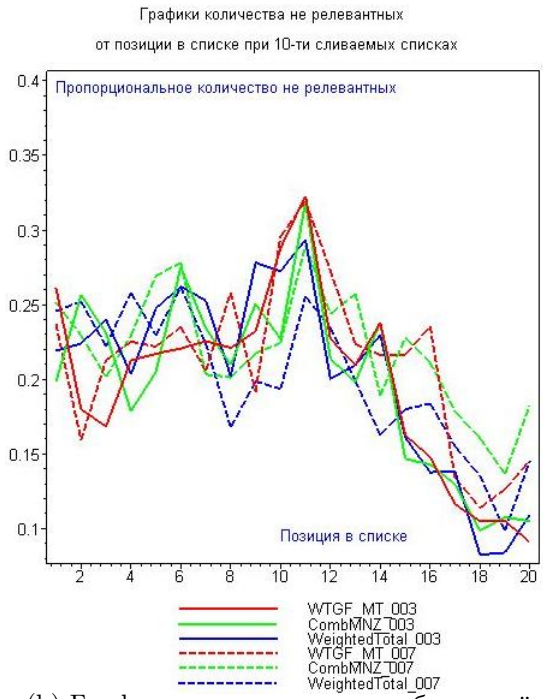


(b) Параметр $delta = 0.07$

Рис. 9: Графики значения коэффициента $R_{overlap}$ для различных функций слияния в зависимости от количества рассматриваемых элементов от начала списков



(a) График релевантных изображений



(b) График не релевантных изображений

Рис. 10: Графики количества релевантных и не релевантных изображений в зависимости от позиции элемента в результирующем списке при слиянии 10 списков

6 Заключение

1. Выводы

- Была разработана система по поиску изображений в частично аннотированной базе данных

- В ходе реализации была проработана схема комбинирования результатов полнотекстового поиска и поиска по содержанию.

- Для комбинирования разработан наиболее эффективный (показатель $R_{overlap}$) метод слияния $WTGF$ по сравнению с классическими методами $CombAVG$, $CombMNZ$ (которые используются в большинстве других систем по слиянию результатов) и методом $Random$ – случайного назначения ранга с выполнением условия $MinMax$.

- В ходе эксперимента были собраны данные по соответствию изображений односложным текстовым запросам, которые в дальнейшем могут быть переиспользованы для оценки качества работы тех или иных методов поиска в изображениях.

- Предлагаемый метод показал хорошие технические характеристики при различных параметрах системы (количестве списков и $delta$).

- Также были предложены и опробованы методы эффективного выполнения путём распараллеливания для алгоритмов слияния. Примером ускоренного метода может служить $WTGF\_MT$.

- Была реализована конфигурируемая система для тестирования подобного рода методов слияния.

- Были разработаны новые методы оценки качества работы функций слияния. Эти методы в дальнейшем возможно применять для настройки функций слияния (поиска оптимальных параметров).

2. Перспективы развития

Данный метод предлагается использовать для слияния не только списков одной природы, но и при комбинировании результатов работы различных систем.

Интересно так же проверить как метод справляется со смешенными базами данных. Так, в будущем, предлагается провести эксперименты со смешенной базой данных – flickrCorelDB [10].

Как было сказано такой метод с небольшими доработками возможно будет использовать для комбинирования однородного слияния и неоднородного слияния.

В часности предполагается использовать метод для слияния результатов поиска по текстуре, цвету, форме.

Причём предполагается использовать несколько различных реализаций поиска по цвету.

Конечно, для каждого вида будут использоваться различные гравитационные функции, значение параметров которых ещё предстоит вычислить.

Однако общий подход к таким функциям комбинирования остаётся.

Такая работа по комбинированию сейчас выполняется в рамках гранта Яндекс.

[10]База данных flickrCorelDB есть уже известная база данных flickr смешенная с частью базы данных изображений предоставляемой фирмой Corel

7 Литература

[1] Analyses of multiple-evidence combinations for retrieval strategies / A. Chowdhury, O. Frieder, D. Grossman, C. McCabe // SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval. — New York, NY, USA: ACM Press, 2001. — Pp. 394–395.

[2] *Gertz, M.* Annotating scientific images: A concept-based approach. `citeseer.ist.psu.edu/528338.html`.

[3] Combining evidence from multiple searches. / E. A. Fox, M. P. Koushik, J. A. Shaw et al. // TREC. — 1992. — Pp. 319–328.

[4] Combining the evidence of multiple query representations for information retrieval / N. J. Belkin, P. Kantor, E. A. Fox, J. A. Shaw // *Inf. Process. Manage.* — 1995. — Vol. 31, no. 3. — Pp. 431–448.

[5] *Gertz, M.* Integrating scientific data through external, concept-based annotations. `citeseer.ist.psu.edu/526357.html`.

[6] *Ghoshal, A.* Hidden markov models for automatic annotation and content-based retrieval of images and video / A. Ghoshal, P. Ircing, S. Khudanpur // SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval. — New York, NY, USA: ACM Press, 2005. — Pp. 544–551.

[7] *Glotin, H.* Fast image auto-annotation with visual vector approximation clusters / H. Glotin, S. Tollari // Proc. of Fourth International Workshop on Content-Based Multimedia Indexing (CBMI2005). — 2005. — june.

[8] Image annotation by large-scale content-based image retrieval / X. Li, L. Chen, L. Zhang et al. // MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia. — New York, NY, USA: ACM Press, 2006. — Pp. 607–610.

[9] *Zeng, H.* Learning to cluster web search results. `citeseer.ist.psu.edu/701862.html`.

[10] *Lee, J. H.* Analyses of multiple evidence combination / J. H. Lee // SIGIR '97: Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval. — New York, NY, USA: ACM Press, 1997. — Pp. 267–276.

[11] *McDonald, K.* A comparison of score, rank and probability-based fusion methods for video shot retrieval. / K. McDonald, A. F. Smeaton // CIVR. — 2005. — Pp. 61–70.

[12] Probability–based fusion of information retrieval result sets / D. Lillis, F. Toolan, A. Mur et al. // Proc. of the 16th Irish Conference on Artificial Intelligence and Cognitive Science. — University of Ulster: AICS 2005, 2005. — Pp. 147–156.

[13] Probfuse: a probabilistic approach to data fusion / D. Lillis, F. Toolan, R. Collier, J. Dunnion // SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. — New York, NY, USA: ACM Press, 2006. — Pp. 139–146.

[14] *Salton, G.* Extended boolean information retrieval / G. Salton, E. A. Fox, H. Wu // *Commun. ACM.* — 1983. — Vol. 26, no. 11. — Pp. 1022–1036.

[15] *Shaw, J. A.* Combination of multiple searches / J. A. Shaw, E. A. Fox // Text REtrieval Conference. — 1994. — Pp. 0–. `citeseer.ist.psu.edu/fox94combination.html`.

[16] *Shuming Shi Ruihua Song, J.-R. W.* Latent additivity: Combining homogeneous evidence: Tech. rep. / J.-R. W. Shuming Shi, Ruihua Song: Microsoft Research, August 2006.

[17] *Vassilieva, N.* A similarity retrieval algorithm for natural images / N. Vassilieva, B. Novikov // Proc. of the Baltic BDBIS'2004 / Ed. by J. Barzdins. — Vol. 672. — Riga, Latvia: Scientific Papers University of Latvia, 2004. — June.

[18] *Vassilieva, N.* Establishing a correspondence between low–level features and semantics of fixed images / N. Vassilieva, B. Novikov // Proc. of the Seventh National Russian Research Conference RCDL'2005. — Yaroslavl, Russia: 2005. — October 04 - 06.

8 Приложения

1. Описание библиотеки алгоритмов слияния ($MergeAlg$)

1.1. Введение

Этот документ содержит описание кода и алгоритмов пакета MergeAlg. После компиляции пакет MergeAlg комбинируется в библиотеку MergeAlg.dll. Пакет MergeAlg содержит алгоритмы по слиянию ранжированных списков, а также дополнительные алгоритмы по обработке списков, такие как, например, фильтрация.

1.2. Настройка и установка

Системные требования Для установки в качестве хранимой процедуры на SQL Server 2005 необходимо:

1. SQL Server 2005 (проверенно на ENT или DEV edition)
2. Устанавливая сборка (MergeAlg.dll, release версия)
3. Скрипты для установки (их можно найти в папке TestScripts):
   (a) ConfigureCLR.sql – скрипт для конфигурирования SQL Server – включения возможности CLR хранимых процедур. Для запуска требуются права администратора БД.
   (b) CreateAssemblersANDProcedures.sql – для установки сборки на SQLServer и создание хранимой процедуры. Для установки необходимо прописать вместо пути
   '$D:\\MyDocs\Alexander\Studing\_Docs\5\Diploma\src\MergeAlg\MergeAlg\bin\Release\MergeAlg.dll$' реальный путь к MergeAlg.dll, также необходимо сменить БД [xcavator] на ту БД, куда планируется устанавливать сборку и создавать процедуру. Хочу заметить, что login должен обладать привелегиями создания UNSAFE ASSEMBLY permissions (по умолчанию ими обладает data owner – dbo пользователь, он может их передавать "по наследству")
   (c) CreateTestData.sql – скрипт для создания тестового набора данных (тестирование)

(d) SimpleFuncTest.sql – скрипт по запуску тестовой процедуры (для подгрузки данных сначала необходимо запустить скрипт CreateTestData.sql). При запуске проверьте используемую БД (строчка USE [xcavator]) [xcavator] заменить на требуемую БД.

Для редактирования и запуска без SQLServer необходимо (ориентировано на обычные приложения):

1. Visual Studio 2005 с установленной на ней поддержкой C# проектов
2. Исходный код проекта.(папка с исходным)
3. Тестовый файл проекта: TestFuncMergeAlg.csproj
4. "DataPreparator.cs" – файл содержит статический класс DataDispatcher. DataDispatcher – диспетчер данных. В его основные задачи входит подготовка данных к использованию (в том числе и подготовка данных для тестирования), выбор и создание необходимого для обработки этих данных алгоритмов (в соответствии с конфигурационной информацией).
5. "ImageElem.cs"
6. "ImageObjComparator.cs"
7. "ListOperAlg.cs"
8. "MergeOperation.cs"
9. "Program.cs" – файл, содержащий функцию Main
10. "Properties/AssemblyInfo.cs" – информация о сборке
11. "RankList.cs" – класс для работы со списком элементов
12. "RankListTable.cs"
13. "DBConnectionDiagram.cd" – диаграмма для описания классов
14. "TestFuncMergeAlg.exe.config" – файл с конфигурационной информацией

1.3. Конфигурирование

Для SQLServer.

1. Скопируйте файл с конфигурационными настройками sqlservr.exe.config в директорию, где находится исполняемый файл sqlservr.exe БД. Он располагается в директории /Binn, находящейся в рабочей папке SQLServer (её можно узнать следующим образом: запустите SQL Server Management studio, правый щелчок мыши по текущему подключению, выберите свойства и посмотрите значение поля "Root Directory" – см рисунок 11 ниже)

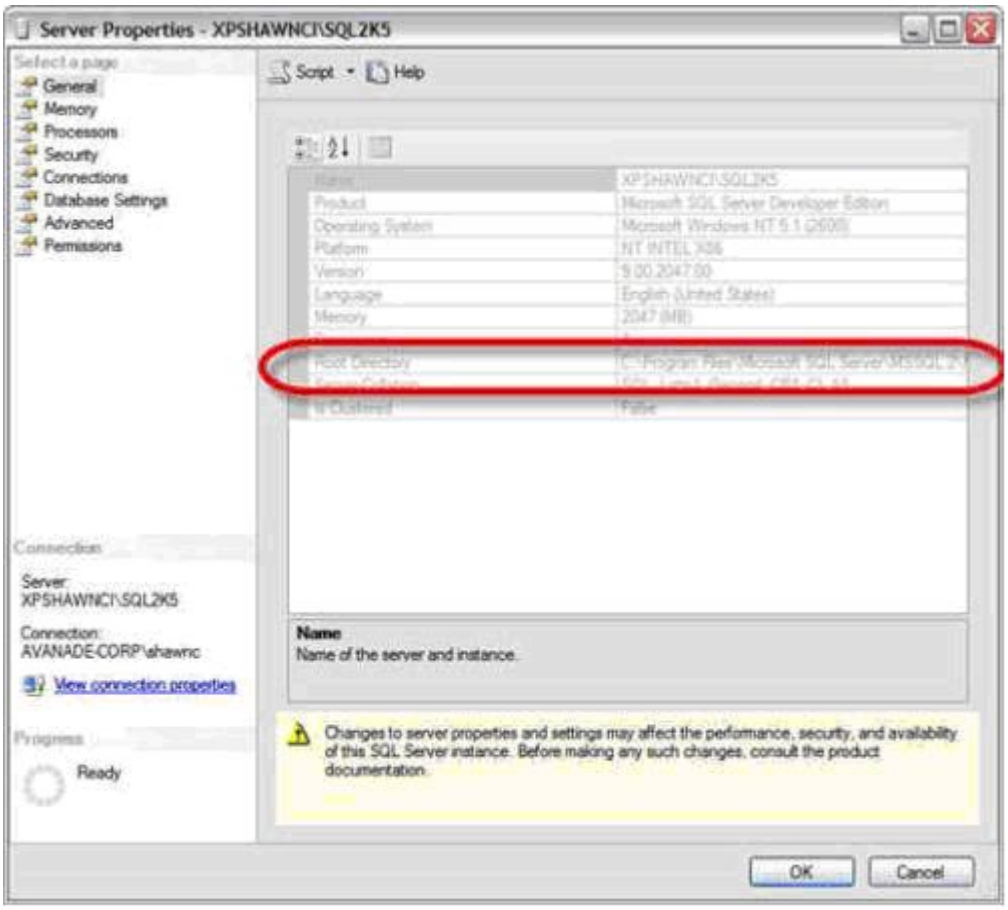


Рис. 11: Настройка MS SqlServer

2. Остановите и запустите службу MSSQLSERVER (это можно сделать через командную строку используя команды: NET STOP MSSQLSERVER для остановки и NET START MSSQLSERVER для запуска) Для Windows приложения (в примере приведено консольное приложение):

3. Скомпилируйте тестовый проект TestFuncMergeAlg.exe

4. Добавьте в директорию к исполняемому файлу конфигурационный файл TestFuncMergeAlg.exe.config

1.4. Архитектура приложения

- Основные классы Основные классы изображены на приведенной ниже схеме 12:

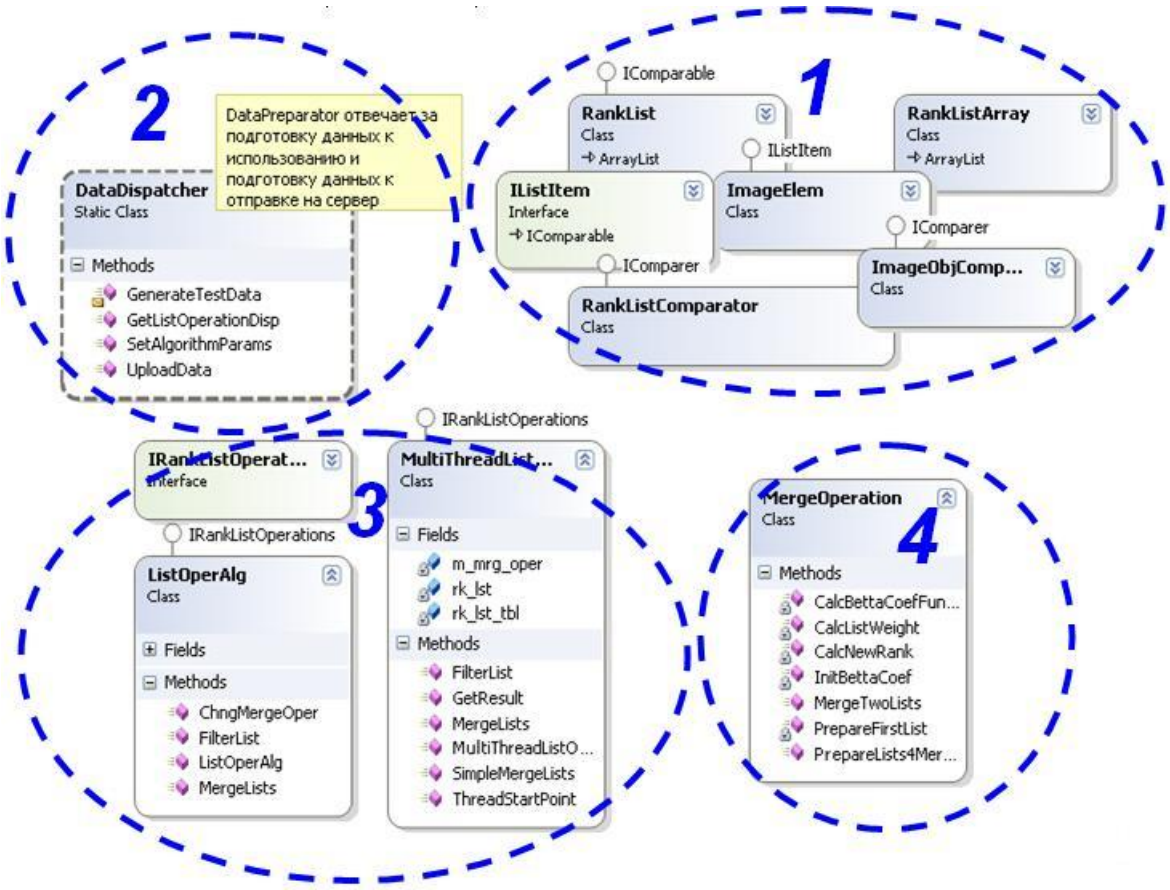


Рис. 12: Схема основных классов MergeAlg

- Пользовательский интерфейс (описан на 12 под цифрами 1,2)
  Пользователю доступны следующие классы и интерфейсы:

  1. Класс DataDispatcher (рисунок 12 цифра 2)
     Класс диспетчера данных. В его основные задачи входит подготовка данных к использованию (в том числе и подготовка данных для тестирования), выбор и создание необходимого для обработки этих данных алгоритмов (в соответствии с конфигурационной информацией).
  2. Класс RankListArray (рисунок 12 цифра 1)
     Класс наследник ArrayList. В качестве элементов этого массива – содержит ранжированные списки – RankList. Для выдачи сортированного по

весам массива ранжированных списков следует создать объект obj класса RankListComparator и вызвать метод
RankListArray.Sort(obj)

3. Класс RankList (рисунок 12 цифра 1)
Класс наследник ArrayList. Каждому списку присваивается уникальный идентификационный номер. Существует также возможность получения индекса списка с данным id. В качестве элементов этого массива – содержит элементы с рангом, которые должны быть объектами классов, реализующих интерфейс IobjListElem. Реализован также интерфейс Icomparable для сравнения списков между собой на равенство (и их сортировки по id).
4. Интерфейс IListItem (рисунок 12 цифра 1)
Интерфейс, который должны поддерживать объекты, добавляемые в список RankList Этот интерфейс поддерживает также интерфейс Icomparable. Однако, ранг и сравнение возможны в границах только одного ранжированного списка. В целях облегчения кода и уменьшения размера данных (а также облегчения работы с ранжированными списками) я не стал добавлять некорректных случаев сравнения, однако, это стоит иметь ввиду. В случае необходимости можно добавить ссылку на родителя и бросание исключения в таких ситуациях.
5. Интерфейс IRankListOperations (рисунок 12 цифра 3)
Интерфейс, который должны поддерживать все классы, которые описывают алгоритмы для работы со списками

- Рабочие классы (алгоритм)
  - Основной алгоритм. MergeOperation (рисунок 12 цифра 4).
    Основной алгоритм использует естественные условия, описанные в документе
    "TeXDetails.pdf" Таким образом к качестве элементарной операции мы можем рассматривать слияние двух списков. Эта операция реализована в классе MergeOperation. При вызове метода MergeTwoList происходит слияние двух списков по следующему алгоритму:
    Этап 1. Просчет весов списков На данный момент веса списков остаются неизменными. А вес результирующего списка полагается равным

$$\omega_3 = (\omega_1^2 + \omega_2^2)^{\frac{1}{2}}.$$

Этап 2. Подготовка списков к слиянию

Шаг 1. Проверяем, совпадают ли наборы объектов в двух списках, в противном случае, данные элементы добавляются в список с весами равными 0.

Шаг 2. Для объектов списков, у которых коэффициент $\beta = 0$ (то есть либо элемент только был добавлен в список, либо этот элемент ещё не был проинициализирован раньше) нужно его проинициализировать по формуле:

$$\beta = \omega^2 \left(rank + \frac{1}{12} \right)^4$$

Эта формула была получена экспериментальным способом (одна из формул на которой выполняются граничные условия, указанные в "TeXDetails.pdf") *Этап 3.* Слияние На данном этапе происходит слияние списков. Ранг пересчитывается по формуле:

$$rank = \frac{\beta_1 rank_1 + \beta_2 rank_2}{\omega_1 + \omega_2}$$

Коэффициент β пересчитывается по формуле:

$$\beta_3 = \beta_1 + \beta_2.$$

- Описание существующих реализаций и их настроек. В настоящее время, реализован алгоритм смешивания списков используя Гауссовы кривые (функция конуса). Этот алгоритм работает в двух вариантах: простом случае и многопоточное смешивание (что, скорее всего, должно отразиться на производительности методов). Простой случай слияния – описан в классе ListOperAlg "Многонитевый" метод слияния – реализован как иерархический вызов нитей. Данный метод реализован в классе MultiThreadListOperAlg (рисунок 12 цифра 3) Предполагается, что списки будут приблизительно одинакового размера, вследствие чего все нити должны выполняться за приблизительно одинаковое время. Сейчас в одной нити происходит слияние максимум 2 списков (если ему требуется слить больше чем 2 списка – он делит на 2 множества и вызывает их в разных нитях). Возможно, слегка преобразовав код сливать в одной нити не 2, а K списков. В общем, в экспериментах следует поиграть с этими параметрами.

- Конфигурирование

Для конфигурирования предусмотрены конфигурационные файлы, которые доступны как в случае SQLServer'а (инструкцию по установке см выше), так и в случае обычного проекта. На данный момент известно пока 2 метода работы: слияние в одной нити, слияние с использованием нескольких нитей. Ближе к опытам в конфигурационные файлы будет добавлена дополнительная информация для настройки методов слияния.

1.5. Анализ результатов работы (тестовый случай)

В приведённой ниже таблице показана специфика работы метода смешивания на реальном примере.
Смешиваются 3 объекта (X,Y,Z), которые появлялись или нет в 3–ёх списках.

Как можно видеть из таблицы все свойства сооблюдается и результат соотвествует логике смешивания списков.

Номер списка	Вес списка	X	Y	Z
1	0.9	0.7	0.3	0.2
2	0.7	0.0	0.9	0.4
3	0.5	0.0	0.7	0.5
Результат	0.9895	0.69968	0.843	0.389

2. Порядок установки *RosaGallery*

2.1. Инсталляция

Требования к программному обеспечению:

Для работы с системой RosaGallery необходимо выполнение следующих требований к программному обеспечению.

Для выполнения

1. .Net Framework 2.0
2. SQL Server 2005 Enterprise Edition (с SQL Server Configuration Manager)
3. Installed full–text search services

Для редактирования и разработки

1. SQL Server 2005 Developer Edition

2. Visual Studio 2005

Запуск скриптов для настройки БД Перед запуском скриптов проверьте наличие БД xcavator. Необходимо также проверить есть ли у вас разрешения политики безопасности для запуска скриптов. Далее необходимо выполнить следующие действия

1. Построить индекс

2. Создать необходимые хранимые процедуры

Построение индекса для полнотекстового поиска осуществляется путём запуска скрипта InitImageFullSearchIndexConstruct.sql. Возможно, потребуется в настройках БД явно указать о том, что будет использоваться полнотекстовый поиск.

Можно проверить установлены ли необходимые сервисы и разрешения полнотекстового поиска для БД путём выполнения следующих предложений:
–SELECT FULLTEXTSERVICEPROPERTY('IsFulltextInstalled')
–SELECT SERVERPROPERTY('IsFulltextInstalled')
–SELECT DATABASEPROPERTY('xcavator','IsFulltextEnabled').

Хранимые процедуры устанавливаются путём вызова InitStoredProcedures.sql. Все скрипты размещены в папке DataBaseScripts.

Установка клиентского приложения на IIS Для установки клиентского приложения необходимо

1. Создайте папку в которой будут храниться данные необходимые для функционирования нашего сайта(по умолчанию это
Рабочий_Диск[$C:\backslash$]$\backslash Inetpub\backslash wwwroot\backslash$ [НАЗВАНИЕ САЙТА]).

2. Скопируйте в созданную папку содержимое папки PrecompiledWeb

3. Откройте IIS Manager (Настройка -> Панель управления -> Администрирование -> Internet Information Services)

4. В дереве обозревателя выбрерите вкладку веб-узлы->веб-узел по умолчанию

5. Щелкните правой кнопкой мыши и выберите из меню "создать->виртуальный каталог"(см рисунок 13)

6. Укажите название сайта и выберите месторасположение страниц (сайт будет располагаться в директории, которую мы создали на шаге (1))

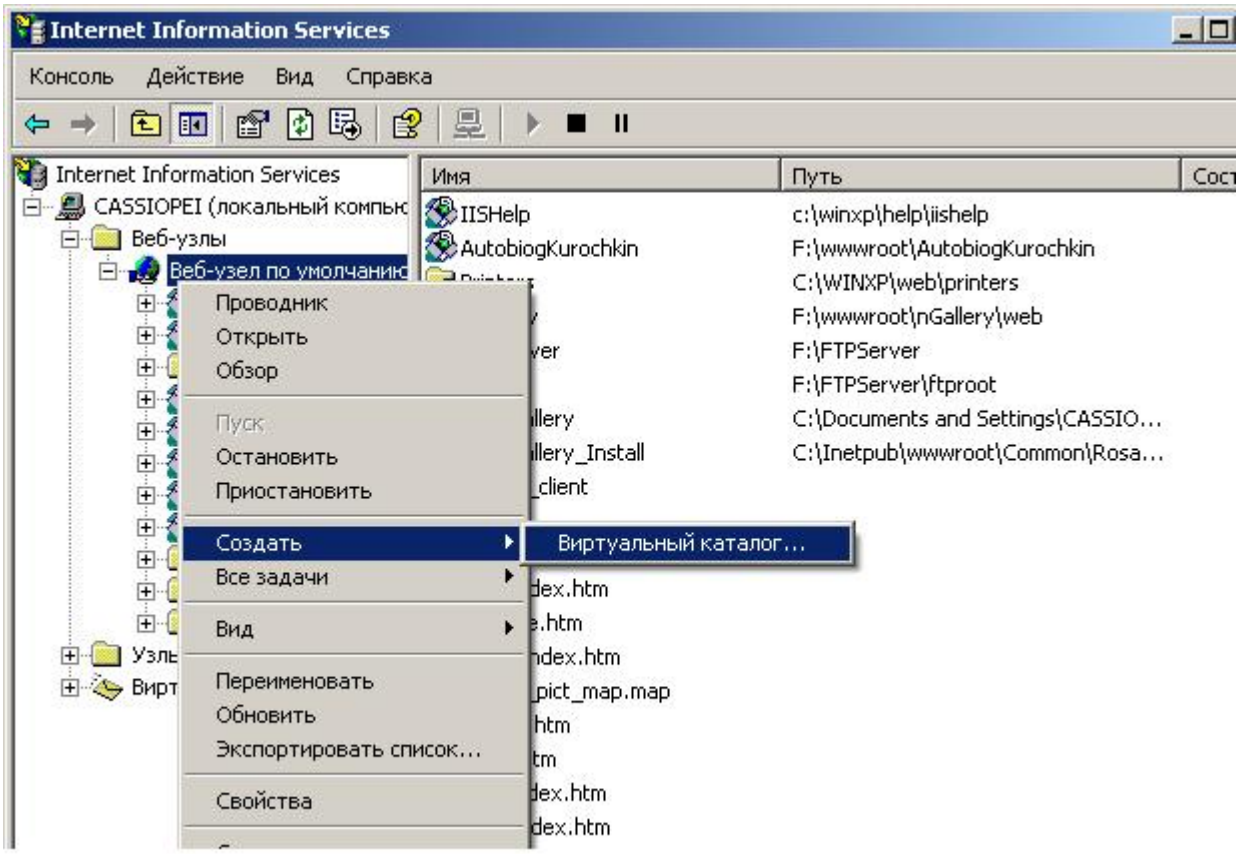


Рис. 13: Создание виртуального каталога в IIS

7. В созданном каталоге (свойства) разрешить выполнение скриптов, установить использование ASP.NET 2.0, в качестве порядка загрузки страниц установить SearchByText.aspx, далее Default.aspx.

   Следующие рисунки (14) иллюстрируют эти действия:

8. Можно запускать приложение
   ($http:\backslash\backslash localhost\backslash$ [Название_Созданного_Каталога]).

2.2. Основные классы

На рисунке 15, приведённом ниже, изображены два основных класса для создания и выполнения запросов а также чтения результатов.

Класс SearchImageByTextQuery позволяет создавать запрос (каждый запрос – новый объект данного класса) и их выполнять путём вызова метода Execute.

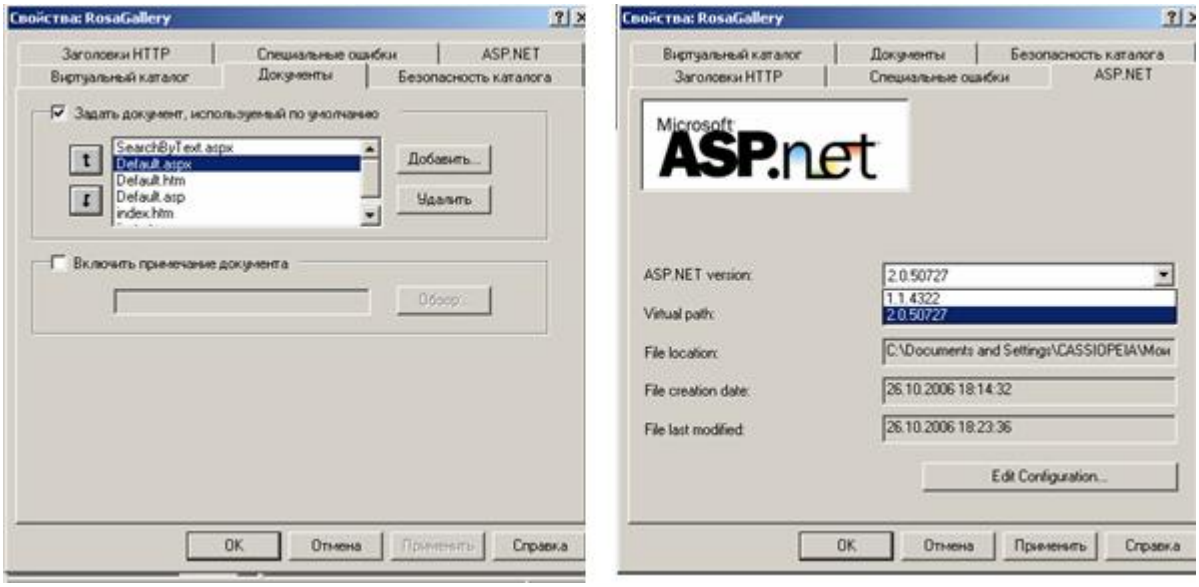


Рис. 14: Настройка сайта *RosaGallery* в IIS

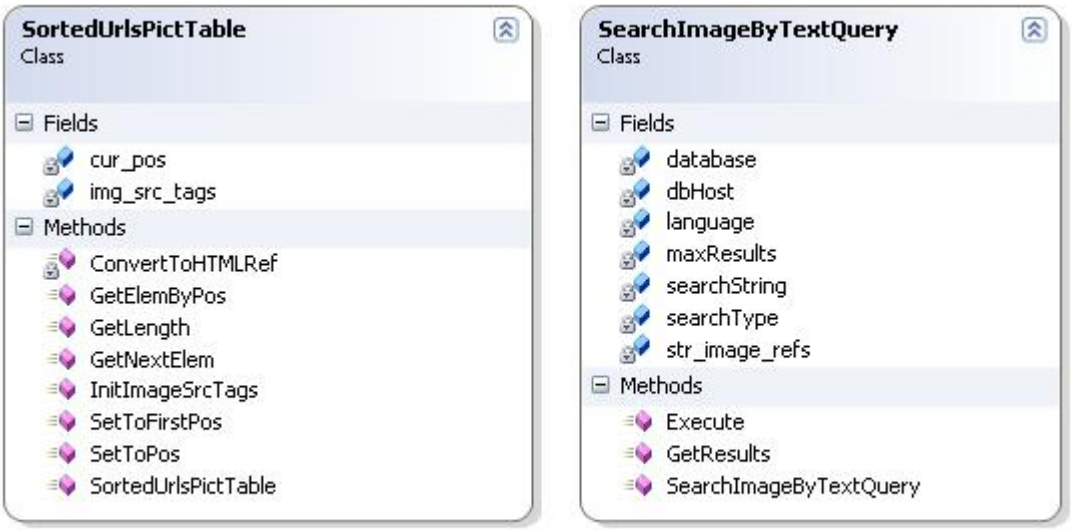


Рис. 15: Основные классы *RosaGallery*

Результат можно получить посредством вызова метода GetResults, который возвращает результат в виде объекта класса SortedUrlsPictTable, который хранит ссылки на найденные картинки в порядке убывания их релевантности. На сего-

дняшний день этих двух внешних классов для работы с UI достаточно. Возможно, имеет смысл использовать не классы, а интерфейсы.

Внутри себя они могут содержать сложные условия для поиска и сортировки ответа в зависимости от его релевантности. Планируется сортировку таблиц организовывать на уровне приложения сервера. Так как нам нужно будет слить несколько в общем независимых результатов обработки запроса по притяжению картинок в один результат.

2.3. Настройка клиентского сайта

Настройки вынесены в отдельный файл для конфигурирования веб приложений (web.config).

Его можно редактировать либо вручную (как текстовый документ), либо используя средства Visual Studio 2005 (Главное меню-> Website -> ASP.NET configuration), либо используя IIS manager (во вкладке свойства->ASP.NET (рисунок выше приведён) есть редактирование конфигурации)

Открытие проекта в Visual Studio 2005 Либо создать новый проект website. Либо использовать мой файл проекта, в таком случае необходимо установить переменную среды RosaGalleryProject равную пути до RosaGallery.sln.

3. Экспериментальная среда "RosaGallery"

Для проведения экспериментов была создана специальная система.

Для работы данной системы необходимо следующее окружение.

Сводная таблица объектов базы данных с описаниями приведена ниже в таблице 3.

Название	**Тип объекта**	**Описание**
Image	таблица	Основная таблица с данными (картинка и её атрибуты)
Child2Parent, TreeNode,	таблицы	Таблицы, необходимые для функционирования индекса
Border, Config GetImagesByTextQuery_XML,	Хранимые процедуры	Процедуры для осуществления поиска в БД. Результат возвращается в виде XML данных
GetImagesRefsFullAnnotationSearch_XML MergeRankLists	Библиотечная хранимая процедура	При помощи этой процедуры происходит смешивание результатов
GetNN	Функция	Используется для поиска ближайших соседей
GetRandomImage, BuildVP_Tree, BuildVP_TreeLevel, GetBorders, GetRandomImageSet, GetSecondMoment, GetVP	Процедуры	Эти процедуры необходимы для построения индекса
ColorDistance	Скалярная функция	Вычисляет расстояние между изображениями
GetBorder, GetNNRec	Функции	Вспомогательные функции необходимых для работы индекса

Таблица 3: Сводная таблица объектов базы данных с описаниями

Для проведения экспериментов, сбора статистики, хранение таблиц запросов, хранения информации об участниках эксперимента и анализа данных была создана отдельная база данных (TestDB) которая имеет структуру согласно нижеприведенной таблице сущностей 4.

Название	**Объект**	**Описание**
ExperimentsResultsTbl	Таблица	Служит для сохранения результатов работы методов в виде XML
MethodsTbl	Таблица	В таблице хранятся зарегистрированные методы для тестирования (название метода и его уникальный номер)
QueryDataTbl	Таблица	Служит для сохранения (кеширования) результатов работы метода над конкретным запросом. Если для какого-нибудь другого пользователя или для какого-нибудь другого потребуется значение работы метода на данном запросе, то он не будет вычисляться заново, а возьмет соответствующее значение из таблицы В этой таблице регистрируются все источники данных запросов для всех БД (это необходимо для присвоения запросу уникального идентификатора)
<Таблицы запросов>	Таблицы	Таблицы, хранящие запросы для экспериментов
UserTbl	Таблица	Содержит данные участников эксперимента
AddExperimentResult	Процедура	Добавление результатов эксперимента в таблицу результатов
AddNewQueries	Процедура	Служит для пополнения таблицы запросов
ConvertXml2FullResultTbl, ConvertXml2ResultTbl	Процедуры	Преобразуют вариант с XML к табличному виду
ExecuteQuery	Процедура	Исполняет запрос в рамках какого-либо эксперимента
GenerateNewQueryId	Процедура	Генерация нового id запроса (которое не было использовано)
GetQueryNameById	Процедура	По id запроса получение его имени
GetQuerySignatureByQueryId	Процедура	По id запроса получение его сигнатуры
<Хранимые процедуры обертки для запросов>	Процедуры	Такие процедуры создаются для тестирования – они должны быть зарегистрированы в таблице методов

Таблица 4: Сущности базы данных для тестирования

Таким образом, в нашей базе данных присутствуют следующие таблицы, изображенные на рисунке 16

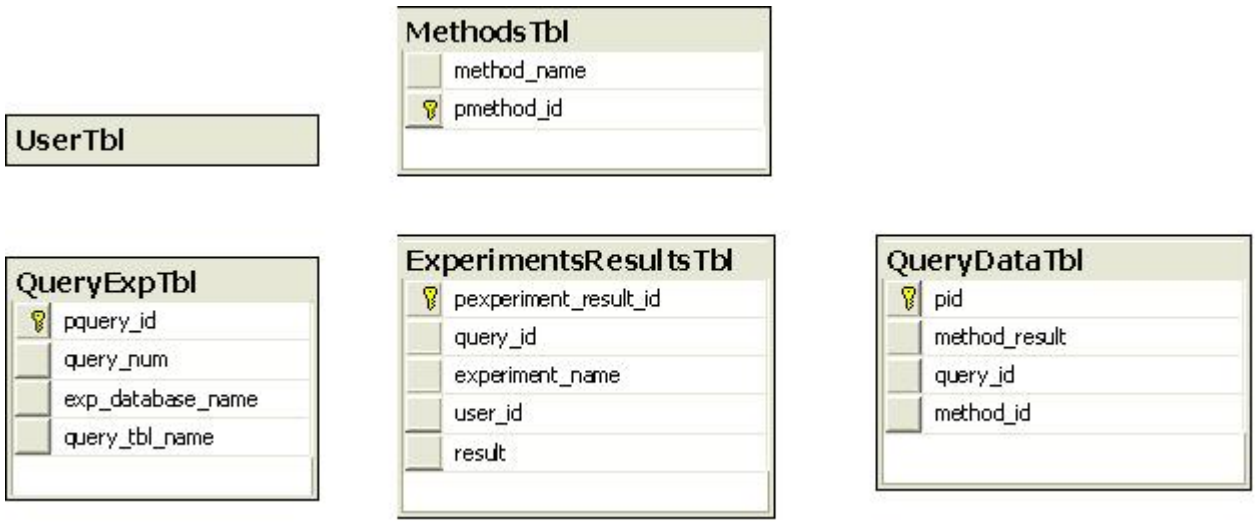


Рис. 16: Таблицы в TestDB

4. Пример файла конфигурирования.

При конфигурировании среды происходит считывание данных из конфигурационного файла экспериментов (местоположение данного файла указывается в web.config в разделе appSettings).

Ниже приведён пример файла конфигурирования.

<TestConfig>
<!- - Указывается строка для подключения к СУБД. В текущей версии не поддерживается. - ->
<connectionString key=”TESTCONNECTION” />
<!- - Регистрация методов, над которыми необходимо ставить эксперименты. В текущей версии не поддерживается. - ->
<method name=”First_alg” function=”test_function”>
<param pos=”1” value=”2”/>
</method>
<!- -Регистрация данных для запроса, над которыми необходимо ставить эксперименты. В текущей версии не поддерживается. - ->
<queryMap/>
<!- -Регистрация эксперимента. Необходимо указать название эксперимента и данные с которыми он будет работать - ->
<ExperimentCase name=”NonrelRel_case” exp_data=”Query_1”>
<HelpInfo>
<!- -Сюда прописывается справочная информация, которая будет выводится в помощь экспериментатору- -> This is instruction </HelpInfo>
<votingList>
<!–Добавляем критерии оценки и значение, которое будет выносится в результат–>
<rb pos=”1” text=”*” value=”1”/>
<rb pos=”2” text=”**” value=”2”/>
<rb pos=”3” text=”***” value=”3”/>
</votingList>
<ExperimentSet mapScheme=”MergeDataSources” order=”mixed”
queryTable=”SimpleQuery” queryDataBase=”xcavator”>
<!- -Собственно сюда записываем как ставить эксперимент

- mapScheme=”MergeDataSources” – означает, что объекты, полученные от dataSources будут слиты воедино и для каждого объекта можно будет произвести оценку
- order=”mixed” – означает, что объекты будут перемешаны

- queryTable=”SimpleQuery” – указание, какую таблицу запросов использовать
- queryDataBase=”xcavator” – экспериментальная БД

- ->
<dataSource method=”SimpleFullTextSearch” name=”ds1”>
<filter startpos=”1”finishpos=”5” />
</dataSource>
<!- - Собственно сюда записываем как ставить эксперимент Описание dataSources. Необходимо указать какой из методов следует использовать, указать его имя (при схеме order=”mixed” это значение не используется) и, если необходимо, фильтр - ->
<dataSource method=”SimpleFullTextSearch” name=”ds2” >
<filter startpos=”3”finishpos=”6” />
</dataSource>
</ExperimentSet>
</ExperimentCase>
<ExperimentCase name=”CompetitionMethod” exp_data=”Query_1”>
<HelpInfo> This is instruction </HelpInfo>
<votingList>
<rb pos=”1” text=”*” value=”1”/>
<rb pos=”2” text=”**” value=”2”/>
</votingList>
<ExperimentSet mapScheme=”useDataSourceAsOneElement” order=”mixed” queryTable=”SimpleQuery” queryDataBase=”xcavator”>
<dataSource method=”SimpleFullTextSearch” name=”ds1”> <filter startpos=”1” finishpos=”5” />
</dataSource>
<!- -Собственно сюда записываем как ставить эксперимент. Демонстрация другого вида схемы.- ->
<dataSource method=”SimpleImagesByTextQuery” name=”ds2” >
<filter startpos=”1”finishpos=”5” />
</dataSource>
</ExperimentSet>
</ExperimentCase>
</TestConfig>