САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Математико-механический факультет

Кафедра системного программирования

Тверьянович Мария Андреевна

Система хранения данных с группировкой по категории KAPLARO

Магистерская диссертация

Допущена к защите.

Зав. кафедрой:

д. ф.-м.н., профессор Терехов А.Н.

Научный руководитель:

старший преподаватель Луцив Д.В.

Рецензент:

доцент Графеева Н.Г.

Санкт-Петербург
2012

Оглавление

Введение

В современных файловых системах человек работает с информацией посредством таких абстракций, как файл и папка [8], что определяет основную структуру хранения в них информации. Ведь совокупность всех вложенных папок и файлов файловой системы образует определенную иерархию. Такой способ организации файлов называется иерархической.

Парадигма иерархического хранения файлов удобна для компьютера, но не для человека. Она удобна для компьютера, поскольку является единообразной и хорошо масштабируемой. Однако человек мыслит иначе, чем компьютер, человек мыслит ассоциациями. Пользователь описывает один и тот же объект целым набором определений, названий, схожих понятий и категорий и наоборот, одно и то же понятие может относиться к целому набору объектов, что в итоге соответствует бинарной связи «многие-ко-многим», а не «один-ко-многим» в случае иерархии. Собственно, поэтому человек и определяет сам объект посредством множества ассоциаций. Таким образом, традиционные файловые системы сильно ограничивают возможность описания файла для человека. Они, по сути, заставляют пользователя сужать, отображать свои знания об объекте и его свойствах в единую иерархическую структуру, задаваемую в основном только двумя характеристиками - названием и местом в иерархии папок. Причем из всех равнозначных вариантов, к какой категории или к чему отнести файл, приходится выбрать только один и соответствующую с этим вариантом иерархию.

Ограничение в описании данных неизменно влечет сложности при их поиске. Компьютер сохраняет только малую долю информации о файле, определяемой иерархией, а ведь человеческая память не безгранична. Современные пользователи имеют в своем распоряжении уже сотни гигабайт и тысячи уникальных файлов, как например, видео, изображения и документы. Причем, если некоторая часть информации не используется даже пусть и небольшой промежуток времени, пользователь легко забывает, где,

под каким названием и как она хранилась. Он может только предположить, как бы он мог назвать файл или в какую директорию, папку он мог бы его сохранить. По сути, при поиске пользователь оперирует лишь своим неточным и неполным описанием файла. Однако единственные легко задаваемые параметры, по которым ведется поиск в современных файловых системах, – это названия всех папок в пути до файла и название файла. Таким образом, если пользователь сделал неправильное предположение, то соответственно сразу он файл не найдет и потратить время на проверку всех своих предположений о месте нахождения файла. Итак, описание нужных пользователю файлов может далеко не полностью соответствовать параметрам, по которым можно осуществлять их поиск в файловых системах.

Вследствие вышесказанного важно увеличить эффективность организации информации, чтобы полнее описывать файлы, и соответственно эффективность поиска необходимых файлов. Именно существование большего числа ключевых слов, у хранимого файла может способствовать тому, что человек быстрее и легче найдет необходимую ему информацию в большом массиве данных. Поскольку ключевые слова являются как бы краткими описаниями содержимого файла и позволяют группировать файлы по категориям, соответствующим наличию этих ключевых слов у файла. Таким образом, возникли идеи создания систем, в которых файл будет определяться не только его местонахождением и названием, но и пользовательским описанием его содержимого. Примерами таких систем являются Windows Media Player, Adobe Photoshop Lightroom [15], различные системы версионного контроля, различные интернет ресурсы и другие. Однако все они предназначены лишь для работы с определенными форматами данных, что оправдано для людей, которые большую часть времени работают в конкретной предметной области. Тем не менее, не для всех пользователей это приемлемо. Например, при поиске совокупности файлов совершенно разных форматов придется использовать несколько инструментов и соответственно потратить время на «ручное» сопоставление

результатов их работы. Для конечного пользователя было бы полезнее иметь систему, которая помогала бы ему хранить файлы с ключевыми словами вне зависимости от их формата и отдельной узкоспециализированной предметной области.

И наряду со специализированными приложениями, примеры которых были приведены выше, производители операционных систем добавили функциональность по определению файлов с помощью дополнительной информации о содержимом файла[19,23].

Так или иначе, существующие решения строятся по принципу добавления к обыкновенной файловой системе дополнительной информации и усиления поисковых алгоритмов и средств. Это повышает удобство работы конечного пользователя в текущей файловой системе, но по-прежнему оставляет ее иерархическую структуру данных первичной. Фактически существующие решения ориентированы лишь на исправление недостатков традиционных файловой системы с сохранением совместимости, а не на устранение проблемы в корне.

Целесообразно же создание альтернативного подхода к группировке файлов, изначально структурирующего файлы удобным для пользователя способом, а именно реализующим связь «многие-ко-многим» на основе пользовательской информацией о содержимом файлов, и не требующего трудоемких средств для поиска файлов. Таким образом, человек избегает необходимости отображения и сужения своих знаний о природе и свойствах данных в иерархическую структуру.

Представленная в данной работе система KAPLARO (KAtegoria PLurmedia ARĥivO, эсперанто) ставит своей целью хранение, поиск и полуавтоматическую классификацию данных, где наравне с самими файлами должна храниться и дополнительная информация об их содержимом в виде ключевых слов. При этом в отличие от решений, примеры которых приведены выше, предполагается изначально отталкиваться от группировки данных по категории, а не от традиционной иерархической организации.

В данной работе предполагается продолжение работы по данному направлению, а именно:

- Доработка прототипа системы, включающей уже ранее реализованные компоненты. [4]
- Добавление функциональности по полуавтоматической классификации файлов. Поскольку пользователь не всегда в полном объеме может описать файлы с помощью ключевых слов, меток. Существует необходимость в том, чтобы система сама предложила новые метки на выбор, либо автоматически добавила.
- Обеспечение более удобных интерфейсов для работы с системой. В качестве более удобного интерфейса для приложений, не работающих изначально с KAPLARO, рассматривается виртуальная файловая система, эмулирующая традиционную файловую систему с древовидной организацией файлов. При этом ключевые слова будут отображаться в каталоги виртуальной файловой системы.

Постановка задачи

В результате данной работы предполагается:

- рассмотреть существующие модели хранения данных мультимедиа;
- разработать модульную архитектуру системы хранения данных, структурирующей файлы по метаданным;
- реализовать прототип системы, обладающей следующими свойствами:
    - Система должна обеспечивать возможность поиска файлов: по метаданным, по поисковому запросу
    - Система должна производить полуавтоматическую классификацию файлов для улучшения их организации и облегчения поиска
    - Должны быть спроектированы интерфейсы: API, ВФС

Обзор

Анализ подходов к группировке файлов

Исторический обзор

Идея предоставить пользователю более дружественную, чем файловая система, среду для хранения данных имеет те же источники что и идея «канцелярских» метафор, таких как «рабочий стол», «папка», «корзина». Источником данных «канцелярских» метафор была компания Xerox, применившая их в системах Xerox Alto и Xerox Star. Именно эти системы стали прототипами систем компании Apple, Inc., о которой пойдёт речь дальше.

Закладывая и укрепляя традиции метафоры «офисного рабочего пространства», соблюдаемые и до сих пор, система Apple Lisa отображала данные файловой системы в виде, привычных для человека, канцелярских объектов. Каталоги показывались в виде папок, содержимое которых можно было просмотреть в окне. Файлы, хранящие документы различной природы (тексты, таблицы, изображения) отображались при помощи разных иконок. На том же рабочем пространстве находились и метафоры инструментов: текстовых процессоров, календарей, часов.

Некоторые метафоры могли показаться по современным меркам слишком буквальными: так, например, был специальный объект «пачка бумаги», из которого можно было извлечь листы для написания нового документа. Обрабатываемые в данный момент документы помещались на рабочий стол. Для того чтобы сохранить и закрыть документ, необходимо было перетащить его иконку обратно в папку, из которой документ был взят или в другую папку. [6][14]

Одна из интересных и важных в контексте данной работы особенностей метафоры рабочего пространства - возможность задания одинаковых имён для разных документов Apple Lisa [19]. Уже в начале 1980-х годов создатели системы осознавали, что оператору компьютера имя

документа, само по себе, может говорить мало, и позволили для отображения на рабочем столе использовать т.н. виртуальные имена, которые не были уникальны. Так, например, созданная копия документа называлась, так же как и оригинал, хотя при редактировании её и оригинала свойства начинали отличаться. В дальнейшем пользователь различал документы по их свойствам (дате, размеру) и содержимому. При сохранении на диск данные документов записывались в файлы, физические имена которых были уникальны и соответствовали накладываемым файловой системой ограничениям. Разумеется, пользователь мог изменить и логическое имя документ, если считал нужным. В современных ОС семейств Windows и в UNIX-подобных логические имена файлов не применяются.

Microsoft уделила должное внимание дружественному именованию документов в офисных и домашних версиях своих систем (расширив возможности по именованию файлов файловой системы FAT при помощи надстройки VFAT) более чем на 10 лет позже, в 1995 году при создании Windows 95 [12].

В Windows NT, ориентированной изначально на корпоративный сектор, возможности именования файлов сразу были достаточно богатыми. Часть из них файловой системой NTFS была унаследована от файловой системы HPFS, использованной в операционной системе OS/2.

Таким образом, уже в середине 1990-х гг. Microsoft пошла по традиционному пути, предоставив пользователю традиционную файловую систему с достаточно широкими возможностями по сравнению с семейством DOS, например. Эта тенденция сохранялась ещё более 10 лет вплоть до попытки реализации проекта WinFS[24][26], а затем и выпуска Windows 7[22].

Современные операционные системы

Из-за быстрого роста объемов хранимой информации возникла потребность в новых принципах систематизации данных. Что в свою очередь привело к появлению идей и исследований, связанных с добавлением к

файлам дополнительной информации о содержимом файла. Определенные разработки на данную тему появились и в последних версиях операционных систем, таких как Windows и Mac OS. Что говорит о действительной значимости данного вопроса.

В них были введены возможности для поиска файлов не только по пути, названию, метаданным, но и по меткам (комментариям в Spotlight Mac OS [20]), добавленными пользователями к файлам.

Рассмотрим подробнее Mac OS X. С появлением в OS X новой функции Spotlight, поиск на Mac стал как никогда быстрым и удобным. Spotlight упростил поиск файлов по меткам (ключевым словам), не требуя от пользователя знания точного названия файлов или их содержимого.

Тем не менее, присвоить метку к документу для быстрого поиска не так-то просто. Ключевые слова можно включать в название документа или открывать свойства файла и прописывать теги в поле “Комментарии Spotlight”. Для более простого добавления меток в Mac, разработаны дополнительные приложения, такие как Tags и Punakea. А для поиска файлов разработан Finder, к которому в последней версии Lion добавлено немного особенностей. Таких как возможность сортировки файлов по более широкому спектру параметров, так и поиск по поисковому запросу, где по каждому слову из запроса Lion предлагает некие соответствия, например, с известными типами файлов, то есть пытается угадать, что же пользователь хочет найти в зависимости от вида и области поиска [21].

В Windows начиная с XP и до появления 7й версии, хотя и существовала возможность добавления меток, использовать ее было не только неудобно, но и мало кто знал о данной возможности. Так как чтобы добраться до этого, нужно было зайти в «Свойства», а затем на вкладку «Сводка» и так уже выставить дополнительную информацию. Если представить, что данные манипуляции нужно проделать с каждым из тысячи файлов, то задача представляется весьма сложной. Однако специалисты из Microsoft осознали недостатки данной системы и необходимость в удобной

возможности добавлять дополнительную информацию в виде ключевых слов к файлам для последующего поиска. Таким образом, уже в Windows 7 появились удобный интерфейс для добавления ключевых слов без необходимости заходить в свойства файла, а также модифицированные возможности по каталогизации и категоризации файлов в специальной библиотеке по этим ключевым словам. Под библиотекой в Windows 7 понимается определенный пользователем ассоциированный с ней набор папок и их специфическое содержимое, например изображения, которые могут отображаться отсортировано вне зависимости от названия файлов в них и места их хранения, но единым списком. Например, можно отсортировать файлы в конкретной библиотеке по меткам. Кроме того в Windows 7 был добавлен интерфейс поисковой строки как в меню «Пуск» так и интерфейс папок. В поисковую строку можно вводить любые данные, по которым хочется произвести поиск. Так же в поисковой строке можно задавать фильтры поиска (например «датасъемки:давно») для ограничения области будущего поиска. [22]

Узкоспециализированные приложения

Десктопные приложения

Многие небольшие компании, применяя принцип организации данных с помощью ключевых слов, создали узкоспециализированные приложения – хранилища, которые являются сугубо коммерческими проектами, скрытыми от посторонних глаз. Поэтому сложно рассказать, как решились технические вопросы, связанные с созданием этих приложений. Кроме того, данные хранилища ориентированы на определенные файловые системы и работают только с определенными типами файлов, такими как: видео, изображения или реже тексты. В них есть возможность добавления ключевых слов и соответственно поиска по ним.

Например, Adobe Photoshop Lightroom [15] использует каталог для отслеживания местоположения файлов фотографий и запоминает о них информацию. Каталог, как база данных, содержит записи по этим файлам.

Эта запись, хранящаяся в каталоге, содержит такие данные, как ссылки, указывающие, где находятся фотографии на компьютере, метаданные, описывающие фотографии. При оценивании фотографии можно добавлять метаданные и ключевые слова, организовывать фотографии в коллекции или удалять фотографии из каталога. Поиск производится посредством специальной панели задающей фильтр с тремя режимами: текст, атрибут, метаданные, - которые можно смешивать или использовать отдельно. Режим текста как раз позволяет производить поиск по любой индексированной строке метаинформации о фотографии, как имя, подпись и ключевые слова.

В свою очередь проигрыватель Windows Media Player 11 предоставляет широкий спектр возможностей хранения и использования цифрового мультимедиа. Он значительно упрощает доступ ко всем видеозаписям, изображениям и записанным телепередачам на компьютере. В этой версии не только музыка, но и другие виды мультимедиа, такие как: видео, изображения и записанные телепередачи - выделены в отдельную категорию на панели «Библиотека». На этой панели можно отсортировать музыку, кроме обычных способов и по ключевым словам.

В следующей версии своей версии, Windows Media Player 12 не пошел по пути улучшения организации файлов. Разработанный специально под Windows 7, он полностью основывается на библиотеке Windows 7 и ее особенностях в структурировании данных и поиске.

Облачные системы хранения данных

Многие современные системы удалённого хранения данных (так называемые облачные хранилища), не ограниченны необходимостью обратной совместимости с иерархической структурой файловой системы, тем самым предоставляют достаточно удобные способы организации данных. Это обстоятельство делает их интересными объектами для анализа в контексте данной работы.

Не смотря на то, что большинство сервисов облачного хранения данных появились в течение последнего десятилетия, некоторые идеи,

заложенные в них, были не только сформулированы, но и реализованы в 80-е годы XX века.

История

Ранние системы удалённого хранения данных можно датировать 70-ми и 80-ми годами XX века. Фактически они организовывали данные иерархически, но позволяли снабжать данные *метаинформацией*, которая могла быть использована для повышения эффективности их поиска.

Протоколы FTP[1] и NFS[2] позволяли получить доступ к удалённому каталогу. FTP рассчитан на использование в глобальной сети, NFS — в локальной. Пользуясь хорошей пропускной способностью и низкой латентностью локальных сетей, ресурсы NFS обычно монтируют, «включая» их в иерархию локальной файловой системы.

Одним из существенных различий в традициях их использования является норма этикета, предписывающая снабжать каталоги FTP с не очевидным содержимым файлами "descript.ion", "files.bbs" и так далее. Ведь связываться с хозяином ресурса для уточнения того, что же на нём расположено, в случае FTP неуместно, так что описание данных становится тут особенно ценным. Фактически, файлы "descript.ion" являются средством хранения метаинформации, используемым на практике, но исходно не подержанным технологически.

Система Gopher, основанная на одноимённом протоколе[3], предназначена для организации каталогов с информацией и простейших гипертекстовых меню с возможностью ввода текстовых запросов для поиска.

Сейчас система Gopher практически полностью вытеснена WWW, в основном, из-за более богатых оформительских и мультимедиа-возможностей последнего. Отдельные серверы поддерживаются

---

[1] RFC http://www.ietf.org/rfc/rfc959.txt
[2] RFC http://tools.ietf.org/rfc/rfc2623.txt
[3] RFC http://tools.ietf.org/html/rfc1436

энтузиастами и, традиционно, некоторыми образовательными и исследовательскими учреждениями.

Система Gopher предназначена для публикации данных (поиск и доступ на чтение) в глобальном масштабе. Информация структурируется иерархически, но гипертекстовые меню используются фактически для хранения метаинформации. Для индексирования этих меню существуют несколько программных систем, самая популярная из них — Veronica[4]. Важным шагом вперёд по сравнению с FTP является выделение метаинформации для её отдельного хранения и для поиска с её использованием.

Современные системы

Из современных систем можно рассмотреть универсальные файловые хранилища и хранилища, чувствительные к типу содержимого.

Универсальные файловые хранилища, такие как DropBox, SugarSync, Vuala, Яндекс.Диск в большинстве случаев служат для синхронизации выбранных каталогов на нескольких «подключённых» к ним устройствах. Они обычно не предоставляют развитых дополнительных средств поиска и работы с метаинформацией сверх стандартных средств, типичных для распространённых файловых систем.

Интереснее хранилища, чувствительные к типу содержимого. Так как его представители используют метаинформацию для организации данных. Характерно, что при этом пользователю предоставляются возможности снабжать файлы одной или несколькими метками, пользуясь которыми впоследствии можно достаточно эффективно выбирать содержимое.

Ниже приведём примеры нескольких сервисов.

Хранилище фотографий PicasaWeb позволяет группировать фотографии по альбомам и искать по метаинформации, например такой, как

[4] http://en.wikipedia.org/wiki/Veronica_%28search_engine%29

время и место съёмки, если они заданы. Текстовые описания фотографий также можно использовать при поиске.

Хранилище документов Google Drive (ранее система Google Docs) позволяет хранить и предоставляет доступ на чтение к документам очень многих офисных форматов. Для документов собственного формата реализует в веб-браузере редакторы: редакторы векторной графики и презентаций, текстовый процессор и система электронных таблиц. Позволяет ставить документам в соответствие несколько меток, причём сами метки можно организовывать иерархически.

Хранилище заметок EverNote позволяет параллельно организовывать заметки иерархически (в блокнотах) и по меткам, при поиске позволяет ограничивать его набором меток, блокнотом, задавать искомые метаданные (время, автор) и слова из текста заметки. Для присоединённых изображений поддерживает распознавание текста, в том числе, в удачных ситуациях, и рукописного. Успешно распознанные слова позволяет использовать при поиске, что является уникальной особенностью EverNote.

Сервисы хранения веб-закладок, как и многие другие аналогичные сервисы, организуют закладки в основном при помощи меток, а также предоставляют средства для графической визуализации отношения метка-статья, обычно в виде графа, позволяют разделять метки на личные и общедоступные. Они ориентированы на «социальное» присваивание меток, в частности, предлагают присвоить ресурсу метки, уже присвоенные другими пользователями. Многие сервисы обладают развитой расширенной функциональностью в некоторых более узких, нежели просто организация ссылок, областях. Так, сервисы BibSonomy.org (Университет г. Касселя, ФРГ) и Connotea.org (Nature Publishing Group) следует выделить в виду их «академической ориентированности»: они содержит развитую функциональность для хранения публикаций и автоматической генерации библиографии.

Интеграция в локальную пользовательскую среду

Для удобства пользователя (в частности, для повышения производительности) некоторые из перечисленных систем могут интегрироваться в локальную среду при помощи установки специализированных приложений. Так, например, универсальные хранилища синхронизируются с локальным каталогом, иногда предоставляя виртуальную файловую систему по протоколу WebDav или аналогичному.

Что касается PicasaWeb, то локальное пользовательское приложение работает, в целом, отдельно от сервиса, но может обмениваться с ним данными: синхронизировать изображения в отдельно выделенных каталогах.

EverNote помимо веб-приложения предоставляет и локальное приложение с той же функциональностью, которое обращается к данным веб-сервиса, но с более удобным интерфейсом и возможностью кеширования данных.

Google Drive синхронизирует структуру папок с каталогом локальной файловой системы. Локальные копии документов, которые поддерживаются веб-редакторами Google, представляют собой ссылки, ассоциированные с веб-браузером, при открытии которых начинается редактирование документа. Локальное приложение играет в данном случае лишь роль системы интеллектуальной организации ссылок. Локальные копии не редактируемых (например, PDF) документов — действительно настоящие файлы. При попытке копировать их более, чем в одну папку, система начинает трактовать их, как два разных документа, что очевидно выбивается из общей картины. Для редактирования некоторых видов документов (Microsoft Office, LibreOffice/OpenOffice) могут быть использованы локальные офисные приложения, снабжённые специальными подключаемыми модулями для доступа к хранилищу.

И большинство сервисов хранения веб-закладок предлагают установить в веб-браузеры дополнительные модули для повышения комфорта при работе с ними.

Выводы

Подведем итоги. Возможность, удобно структурировать пользовательские данные и снабжать их метаинформацией, актуальна на протяжении нескольких десятков лет. Большое количество современных систем хранения данных, ориентированных на конкретные виды содержимого, предоставляют удобные средства для их структурирования и поиска. В контексте данной работы это делает подобные системы крайне интересным объектом для анализа.

Выясняется, что глубокая интеграция таких систем с локальным пользовательским окружением (в частности, на уровне файловой системы) часто либо не производится, либо производится с определёнными неочевидными ограничениями, как в случае с Google Drive.

Предложенный в данной работе подход позволяет избавиться от подобных ограничений, применив виртуальную файловую систему, адекватно отображающую структуру каталогов, отличную от иерархической.

Обзор подходов к классификации данных по метаданным

Байесовские классификаторы

Байесовский классификатор — это вероятностный классификатор, основанный на применении теоремы Байеса[2]. По сути, он схож с перцептроном [1], поскольку представляет собой линейный классификатор. Как и все алгоритмы обучения с учителем, байесовский классификатор обучается на заданных примерах. Пример – это список признаков или свойств образца и, к какому классу его отнесли. Классификатор фиксирует все встретившиеся признаки, а также вычисляет вероятности того, что признак ассоциирован с конкретной классификацией.

Применимо к задаче классификации документов. Образцом считается документ, признаками - слова в нем встречающиеся, без учета слов не

несущих смысловой нагрузки (такие как союзы и местоимения). Примеры предъявляются классификатору последовательно. После каждого примера классификатор обновляет свои данные, вычисляя вероятность того, что документ из указанного класса содержит то или иное слово.

Обученный классификатор – это список признаков с ассоциированными вероятностями. При этом совсем не нужно хранить исходные данные, на которых проводилось обучение. После обучения байесовский классификатор можно применять для автоматической классификации новых образцов.

В данной работе будет рассматриваться наивный байесовский классификатор - упрощенная версия байесовского классификатора с предположением о независимости признаков. Не смотря на то, что, по сути, весьма сильное упрощение и не всегда верное, наивные байесовские классификаторы часто работают намного лучше во многих производственных задачах.

Классификатор вычисляет совокупную вероятность последующей формуле:

P(Категория|Документ) = P(Документ|Категория) * P(Категория),

где

P(Документ|Категория) = P(Слово$_1$|Категория) * ... * P(Слово$_n$|Категория)

P(Слово$_j$|Категория) – значения, взятые из таблицы обученного классификатора, Слово$_j$ – слово, встретившееся в документе. Вероятность P(Категория) равна частоте встречаемости данной категории среди всех документов.

Результатом считается та категория, для которой P(Категория|Документ) принимает максимальное значение.

Самое существенное преимущество наивных байесовских классификаторов по сравнению с другими методами заключается в том, что их можно обучать и затем опрашивать на больших наборах данных. Это

особенно важно, при инкрементальном обучении, когда каждый новый предъявленный образец можно использовать для обновления вероятностей и, как говорилось выше, без использования старых обучающих данных.

Основной же недостаток наивных байесовских классификаторов – их неспособность учитывать зависимость результата от сочетания признаков.

Латентно-семантический анализ

Латентно-семантический анализ или ЛСА - это метод обработки информации на естественном языке, анализирующий взаимосвязь между набором документов и словами в них встречающимися. ЛСА отображает документы и слова в так называемое «семантическое пространство», в котором и производятся все дальнейшие сравнения слов и документов. [9]

Обычно при этом документ рассматривается как набор слов или как вектор по векторной модели (Vector space model) [11]. В векторной модели документ описывается вектором $d_j = (w_{1j}, w_{2j}, \ldots, w_{nj})$, где w_{1j} - вес i-ого слова в этом j-м документе, отражающий важность данного слова для данного документа. Причем вектор строится относительно всего множества различных слов встречающихся во всех документах. В результате семантическое значение документа определяется набором слов, в нем встречающимся.

Как работает данный метод. Для начала составляется частотная матрица встречаемости индексируемых слов в документах. В этой матрице строки соответствуют индексированным словам, а столбцы - документам. В каждой ячейке матрицы указано, какое количество раз, слово встречается в соответствующем документе.

Далее над частотной матрицей проводится сингулярное разложение. Сингулярное разложение - это разложение любой прямоугольной матрицы на три составляющих:

$M = U \Sigma V^T$, где матрицы U и V– ортогональные, а Σ – диагональная матрица, значения на диагонали которой называются сингулярными значениями матрицы.

Основное преимущество сингулярного разложения состоит в том, что оно выделяет ключевые составляющие матрицы, позволяя игнорировать шумы. Как это происходит. Согласно правилам произведения матриц, становится понятно, что столбцы U и строки V^T соответствующие меньшим сингулярным значениям дают наименьший вклад в итоговое произведение. Так, мы можем отбросить последние столбцы матрицы U (полученную матрицу обозначим U_k) и последние строки матрицы V^T (обозначим V^T_k), соответствующие наименьшим сингулярным значениям. Иными словами, оставив только k сингулярных чисел. И исходная матрица $M \approx U_k \Sigma_k V^T_k$ - при этом гарантируется, что полученное произведение оптимально. Сравнивая между собой строки U_k, мы получаем схожесть слов относительно общего семантического пространства, а по столбцам V^T_k получаем сравнение документов.

Данный метод хорошо выявляет скрытые или так называемые латентные зависимости внутри множества документов. Кроме того к достоинствам этого метода можно отнести, что в задачах кластеризации он работает без учителя.

Существенным недостатком метода считается значительное снижение скорости вычисления при увеличении объема входных данных. Ведь SVD-преобразование (сингулярное разложение) может быть достаточно трудоемким, оперируя над достаточно большой матрицей. Однако в последнее время были созданные быстрые, требующие мало памяти и работающие с большими матрицами SVD алгоритмы. Так что пункт перестает быть столь большим минусом ЛСА.

KAPLARO

Общая структура системы KAPLARO отражено на схеме. Разработка основы системы производилась в квалификационной работе бакалавра.[4]

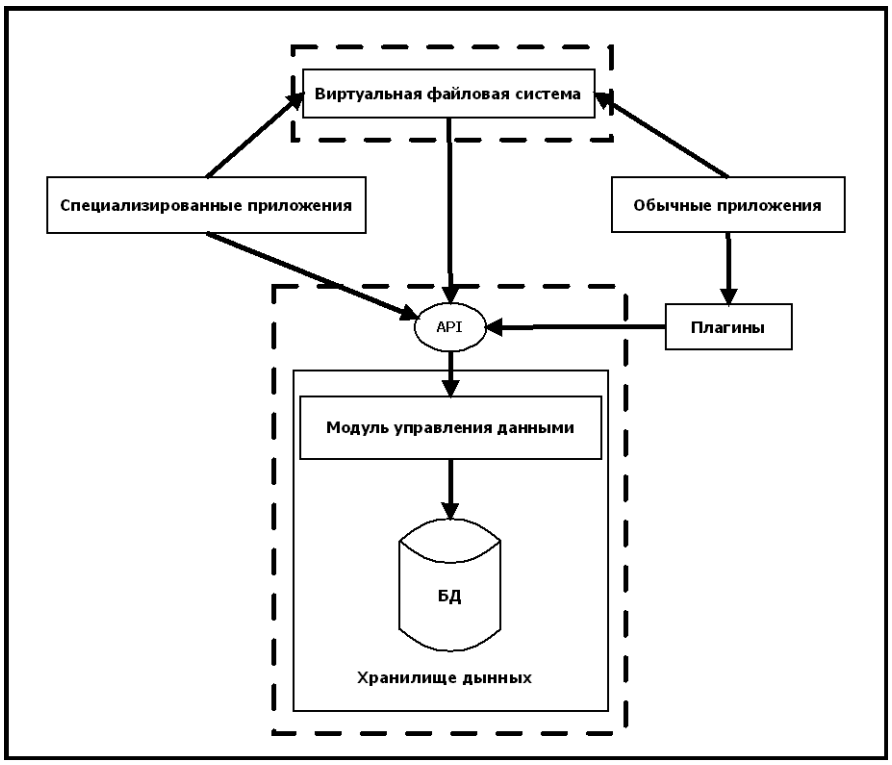

Схема 1. Архитектура прототипа системы KAPLARO выпускной квалификационной работы бакалавра

Основная идея системы заключается в хранении мультимедиа в неиерархической структуре. Для этого все файлы дополняются пользовательскими метками – словами или словосочетаниями, описывающими, по мнению пользователя, содержимое файла или ассоциирующимися с файлом. Поскольку добавляется взгляд пользователя на файл, далее по меткам легче найти необходимые данные.

Поиск может вестись по набору метаданных по мощности близкому к набору, называемому «дублинским ядром». [18]

Все данные о файлах хранятся в базе данных, а сами файлы в отдельно отведенной области памяти. Из системы можно запросить любые файлы, множество выдаваемых файлов определяется запросом к базе данных.

Для работы с системой было создано прототип специализированного приложения – интерфейс. В нем были реализованы следующие функции по работе с системой: добавление, открытие, удаление файла и добавление меток к файлу.

Также в системе был создан специализированный язык запросов для формирования критериев поиска в один поисковой запрос к хранилищу

данных. Так что пользователь может искать необходимые ему данные по поисковому запросу, представляющего из себя простое предложение содержащие параметры поиска. Для построения синтаксического анализатора по этой грамматике использовался проект Irony.

В рамках системы предполагалось создание в будущем виртуальной файловой системы, чтобы обычные приложения, не знающие о KAPLARO, могли работать с файлами в KAPLARO.

Хранилище данных

Теперь подробнее о хранилище данных KAPLARO. Хранилище данных является основной компонентой системы KAPLARO. Поэтому в начале и был реализован прототип хранилища данных. Оно состоит из следующих компонент:

- база данных не иерархической структуры
- модуль управления данными и их метаданными
- API

В качестве модели хранения данных была выбрана реляционная [7], хотя применительно к данной задаче можно было рассматривать и прочие модели. Но на этапе создания прототипа это наиболее удобный и главное простой способ сохранения необходимой информации о файлах в неиерархическом виде, то есть без особых сложностей можно реализовать отношение «многие ко многим» - многие файлы можно описать многими соответствующими ключевыми словами.

Модуль управления данными и их метаданными, как основная компонента хранилища, отвечает за управление и редактирование файлов и данных о них, а также за взаимодействие с базой данных, сохранение целостности связей и актуальности информации.

Архитектура

Общая структура системы

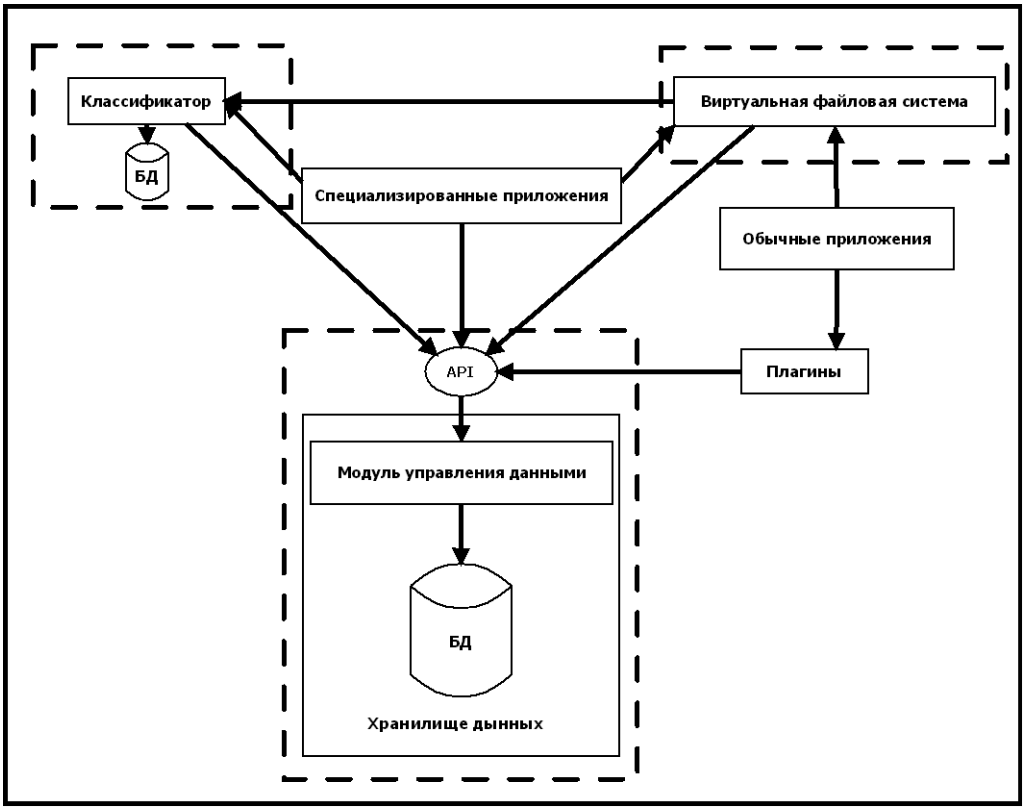

Схема 2. Архитектура прототипа системы KAPLARO магистерской диссертации

Вся система состоит из следующих компонент:

- Хранилище данных
    - Модуль управления данными
    - База Данных
    - API
- Классификатор
    - Сам классификатор
    - База Данных Классификатора
    - API
- Виртуальная файловая система
- Специализированный интерфейс для работы с системой

Что касается Хранилища данных и специализированного интерфейса, то основные особенности были описаны в предыдущей работе о KAPLARO

[4]. В данной же работе речь пойдет о изменениях в системе, некоторые из которых были объявлены как возможности в предыдущей работе, так и обуждены на 2-й межвузовской научной конференции по проблемам информатики СПИСОК-2011 [3].

Изменения в исходной архитектуре

Расширение функциональности хранилища

В связи с появлением новых компонент, которые требуют более точных данных из системы, которые не ограниченны поиском файлов или полным списком некоторых параметров, как списков расширений и категорий. Встала необходимость расширения списка методов API хранилища данных и соответственно их реализация.

Были добавлены следующие методы:

- Получение всех идентификаторов файлов
- Получение всех идентификаторов меток
- Получение идентификаторов меток по списку их имен.
- Получение таблицы содержащей связь «файл – метка»
- Получение по комбинации меток, все метки, которые вместе с этой комбинацией, присвоены хотя бы одному файлу.

Добавление новых компонент

В целях модификации системы были добавлены новые компоненты: Виртуальная файловая система (ВФС) и Классификатор.

Хотя виртуальная файловая система и предполагалась ранее, вопрос ее создания возник в именно данной работе, поэтому она считается новой компонентой.

Задача Классификатора состоит в том, что пользователь не всегда может добавить к файлу весь спектр подходящих меток, так как это требует времени и размышлений. Следовательно, необходимо предоставлять автоматически подобранные метки, как варианты того что следует добавить к уже текущему набору.

Компоненты системы

Виртуальная файловая система (ВФС)

Виртуальная файловая система необходима для того, чтобы отображать файлы, хранящиеся в Хранилище данных, в стандартной древовидной иерархии для внешних программ не знакомых с KAPLARO.

Особый интерес представляет, как именно отобразить структуру хранения данных «много-ко-многим» (в KAPLARO) в структуру «один-ко-многим» (иерархическая структура), поскольку возможны многие вариаций и все равно простое отображение будет сложнее, чем хотелось бы для пользователя. Подробнее эта тема будет обсуждена в Особенностях реализации.

С точностью до особенностей реализации, ВФС работает, используя интерфейс Хранилища данных KAPLARO для получения необходимой информации для отображения виртуальных каталогов и виртуальных файлов, а также проведения операций над ними. Любая внешняя пользовательская программа, умеющая обращаться с иерархиями каталогов, должна спокойно проходить по виртуальным каталогам в поиске нужного файла. Также ВФС может обращаться к классификатору при изменении меток у файлов и форсировать проведение классификации на новых или измененных файлах.

Классификатор

В качестве основных методов для классификации были выбраны Латентно-семантический анализ (сокращенно ЛСА) и наивная классификация Байеса. Оба метода используются для классификации документов на основе слов в них содержащихся и имеют свои достоинства и недостатки, которые были рассмотрены в обзоре.

Однако применительно к задаче добавления подходящих меток, данные методы, использованы в рамках работы в не совсем стандартном виде. Основная проблема возникает, так как KAPLARO предполагает хранить наряду с текстовыми файлами и нетекстовые. А нетекстовые файлы

особенны тем, что не содержат в себе информацию о содержимом в форме, легко позволяющей её анализировать. Таким образом, вся информация о них ограниченна данными, полученными от пользователя. Следовательно, классический ЛСА и классификация Байеса не применима. Но если рассмотреть в качестве признаков файла набор меток с ним связанный и описывающий, по сути, его категорию и, что он собой представляет, то на основе этого набора можно сделать выводы какие метки целесообразно добавить еще к файлу.

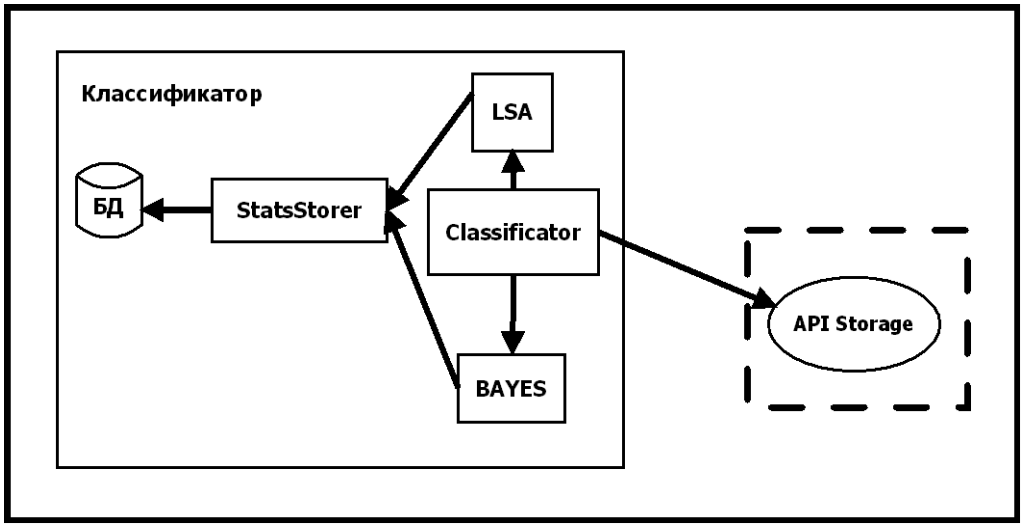

Схема 3. Структура классификатора

Классификатор – состоит из следующих модулей:

- База Данных
- StatsStorer – хранитель статистики
- Classificator - классификатор
- LSA
- BAYES

Теперь подробнее о каждом модуле.

Классификатор – модуль, содержащий в себе основные методы классификации и выдачи результатов. Во-первых, заполнение и пересчет статистики для классификации, во-вторых, получение по файлу с определенным количеством меток вариантов наилучших меток для добавления к этому файлу.

Хранитель статистики, по сути, общается с локальной базой данных, сохраняя в ней данные и выдавая их из нее методам классификации.

Как было оговорено ранее, в качестве основных методов классификации используются ЛСА и наивная классификация Байеса. Соответственно, модули LSA и BAYES, в каждом из которых реализованы сами методы применительно к задаче добавления меток. О них пойдет речь далее.

Классификация нетекстовых данных

В задаче выбора меток, подходящих к нетекстовому файлу наилучшим образом, мы можем оперировать лишь метками, уже связанными с файлами и метками добавлены к рассматриваемому файлу. Общий алгоритм решения этой задачи использует оба рассмотренных выше метода классификации: ЛСА и наивная классификация Байеса, а следовательно и оба модуля LSA и BAYES.

Рассмотрим методы применительно к задаче. Для классификации Байеса недостаточно материала для обучения и, результат может быть не таким четким при небольшом количестве меток у большинства файлов, кроме того он не несет семантической нагрузки, которая показывала бы смысловую связь между метками. ЛСА может дать такую семантическую связь, так как он выявляет скрытые зависимости между документами и между словами из документов в классическом своем применении.

В рамках данной задачи, ЛСА используется не совсем стандартно: в роли документа для ЛСА выступает файл, а как его слова — набор меток, уже ему присвоенный. Но при этом ЛСА, хотя и может использоваться для выдачи готового ответа на вопрос к каким меткам отнести данный файл[5], использовать его именно для получения такого прямого ответа не очень перспективно. Обработка данных может занять несколько больше времени,

[5] Если уже проставленные метки рассмотреть как отдельный файл и провести сравнение по ЛСА с другими файлами (аналогично классическому применению ЛСА к классификации документов).

чем хотелось бы, ведь ЛСА должен оперировать частотной матрицей размера количество всех документов на количество всех слов. С другой стороны, хотя ЛСА и оперирует весьма большой частотной матрицей, над которой производится SVD разложение. В случае меток, данная матрица будет содержать только 1 и 0, соответственно, поставлена ли метка у файла или нет. Кроме того дальнейшие сравнения между векторами меток и файлов, полученные из SVD разложения, могут проводиться по уменьшенному количеству сингулярных векторов. Ведь основной семантический смысл будет заложен в сингулярных векторах соответствующим наибольшим сингулярным значениям. Так что вычисление скалярных произведений векторов, описывающих метки и файлы в семантическом пространстве, упростится при правильном подборе порога для выбора наибольших сингулярных значений.

Таким образом, ЛСА дает хорошее сравнение слов в семантическом пространстве и не слишком дорого, если использовать правильный упрощения, но для получения быстрого готового ответа какие метки нужно добавить, он слишком медленный даже с упрощениями, поскольку требует полноценного пересчета.

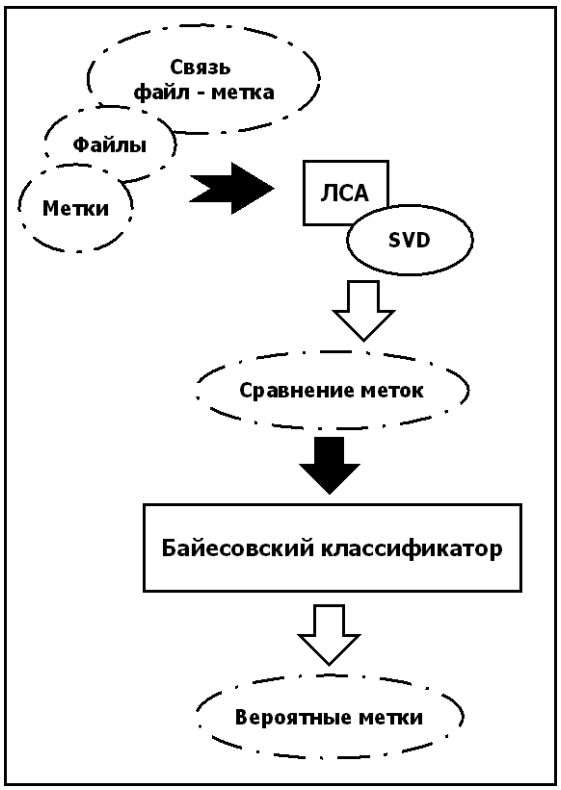

Схема 4. Алгоритм классификации нетекстовых данных

Обученный же классификатор Байеса может давать быстрый ответ в виде наиболее вероятных меток, как говорилось выше, без ущерба производительности и, не оперируя большими объемами данных. Но чтобы в задаче выбора наиболее вероятных меток для нетекстового файла в Байесовский классификатор привнести семантические связи между метками, в основу его обучения ложится

результат ЛСА. А именно: из корреляции векторов, описывающих метки в семантическом пространстве, полученном из ЛСА, высчитывается им соответствующим вероятность выставления данной метки при присутствии некоторой другой метки - **P(метка | метка).** Данная вероятность необходима для подсчета полной вероятности в методе наивной классификации данных.

Так же хотелось добавить, что классификаторы Байеса обладают возможностью усиливать адекватность результата за счет взвешенной вероятности. Такая вероятность позволяет избежать недооценивания редко встречающихся или еще не встретившихся меток и переоценивание часто используемых меток. Подобный подход улучшает результирующую вероятность и дает более адекватный желаемому ответ.

Классификация текстовых данных

Для классификации текстовых данных предполагается использование только наивной Байесовской классификации. Наивный Байесовский классификатор на текстовых файлах дает хороший результат и не столь трудоемок как ЛСА. Ведь ЛСА в данном случае в классическом виде оперирует при классификации документов матрицей частоты размера MxN, где M – количество документов, N – количество слов, чье множество намного разнообразнее, чем множество меток.

Кроме того слова текстов, не так зависят от текущего пользовательского взгляда. Когда как метки сильно зависят от взгляда пользователя и при его изменении пришлось бы переучивать классификатор Байеса, в том случае, если бы он оперировался только на прошлый опыт. Подобное при использовании текстовых данных маловероятно.

Используемые технологии

Microsoft .NET

В качестве основной технологии в данной работе используется .NET Framework, разработанный компанией Microsoft. Данная платформа была выбрана по следующим причинам:

- компоненты технологических решений на её базе хорошо интегрированы друг с другом;
- возможности, которые предоставляют её встроенные и независимые компоненты, хорошо подходят для реализации на их базе прототипа, создание которого входит в цели данной работы;
- компоненты и средства разработки платформы .NET позволяют, с одной стороны, создавать программное обеспечение высокого качества, а с другой - делать это достаточно быстро и легко, что важно для создания прототипов программного обеспечения в целом.

Языком программирования для реализации хранилища мультимедиа был выбран C#. А для разработки базы данных используется Microsoft SQL Server. Поскольку он обладает следующими важными с точки зрения данной работы возможностями: поддерживает платформу .NET; позволяет определять и использовать хранимые процедуры и функции.

Проект Irony

Проект Irony представляет собой средство разработки и реализации языков на .NET [16]. Он содержит платформу для реализации языков, LALR и NLALR [13] синтаксический анализатор.

Посредством Irony можно описать свои собственные грамматики, на основе которых строку ввода разбирается в дерево с помощью синтаксического анализатора. Кроме того данный проект написан полностью на C#, используя большую гибкость и мощь языка C# и .NET, реализует новую и простую методику построения компиляторов.

Проект Irony использован для построения синтаксического анализатора языка запросов к хранилищу, упомянутого выше.

Проект Dokan

Библиотека Dokan позволяет создавать собственную файловую систему в пользовательском режиме под Windows [17]. Например, требуется усовершенствовать классическую архитектуру файловой системы FAT. Для самостоятельного решения подобной задачи требуется написать драйвер файловой системы, который должен работать в режиме ядра Windows. Самостоятельно разработать такой драйвер весьма трудоемко и сложно.

Dokan содержит в себе готовый драйвер файловой системы и библиотеку для разработки собственного кода файловой системы. Основная библиотека написана на языке C, но также предоставляются обёртки для написания файловых систем на Ruby и на .NET (C#, VB, C++CLI).

По сути Dokan похож на FUSE, который представляет модуль для разработки собственных файловых систем, но в UNIX-подобных операционных системах. Аналогично FUSE, Dokan выступает в качестве промежуточного звена между кодом файловой системы, запускаемом в пользовательском пространстве и написанным пользователем (File System Application), и ядром Windows.

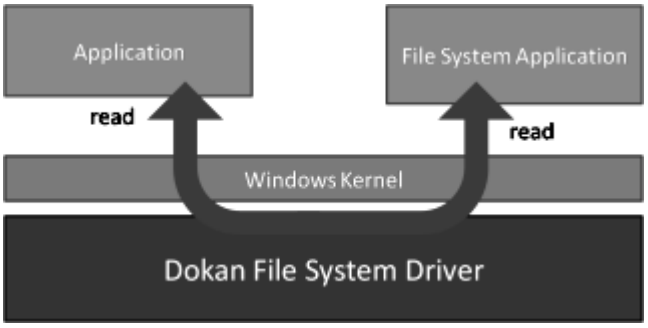

Схема 5. Схема работы проекта Dokan

Данный проект особенно полезен для написания виртуальных файловых систем, которые не хранят файлы непосредственно, как традиционные файловые системы, а выступают как представление из существующей файловой системы.

Проект ALGLIB

ALGLIB является кросс-платформенной библиотекой численного анализа. Она поддерживает следующие языки программирования: C++, C#, Pascal, VBA - и несколько операционных систем: Windows, Linux, Solaris. ALGLIB - весьма удобная библиотека с высокопроизводительными реализациями известных и часто используемых методов численного анализа, легко переносима и проста в использовании, распространяется с исходным кодом. Библиотека постоянно обновляется и улучшается.

ALGLIB включают в себя:

- Линейную алгебру
- Решение линейных и нелинейных систем уравнений
- Интерполяцию
- Оптимизацию
- Численное интегрирование
- Линейную и нелинейную аппроксимацию по МНК
- Решение обыкновенных ДУ
- Вычисление специальных функций
- Статистику (описательную статистику и проверку гипотез)
- Анализ данных - классификацию, регрессию, в т.ч. с использованием нейронных сетей
- Реализацию алгоритмов линейной алгебры, интерполяции и т.д. в арифметике высокой точности

В данной работе ALGLIB используется для подсчета SVD преобразования матрицы.

Особенности реализации

Взаимное отображение поискового запроса и пути в ВФС

Как уже говорилось, задача преобразования неиерархической структуры хранения данных «много-ко-многим» (в KAPLARO) в структуру иерархическую «один-ко-многим» достаточно сложна и неоднозначна.

В KAPLARO файл определяется названием, метаданными, расширением, общей категорией и, конечно, метками, дающими наиболее полную картину о содержимом файла. Так что за основу построения иерархической структуры взят именно набор меток, связанный с файлом. То есть каталогами (узлами) будут являться метки. Сразу встает несколько вопросов: какие узлы являются дочерними к каталогу и, что должно находиться в каталоге, определяемом одной меткой.

Определим, что дочерним узлом к каталогу являются узел, определяемый меткой, встречающейся вместе с меткой самого каталога, хотя бы у одного файла.

В каталоге, определяемом одной меткой, пусть находятся все файлы, имеющие данную метку и все дочерние каталоги.

Опишем, что произойдет при заходе в дочерний каталог. Текущий путь **Метка$_1$ -> Метка$_2$,** где **Метка$_1$** вместе с **Метка$_2$** встречается хотя бы у одного файла. Следовательно по этому пути получаем, те файлы, которые связаны и с **Метка$_1$** и с **Метка$_2$**. Также расширим понятие дочерний каталог, в данном случае у нас есть путь дающий каталог **Метка$_1$ & Метка$_2$**. Значит для него дочерним узлом будет узел, определяемый меткой, встречающейся в наборе меток вместе с **Метка$_1$** и с **Метка$_2$**, хотя бы у одного файла. И аналогично далее вглубь иерархии, пока существуют файлы, у которых есть комбинация меток, определяемая путем, с еще другими нерассмотренными метками.

Таким образом, получается иерархическая структура по меткам. Но за превращение широкой связи «много-ко-многим» в узкую «один-ко-многим», приходится платить уменьшением в удобстве поиска и усложнением.

Основной проблемой данного подхода является, что на верхнем уровне должны отображаться все каталоги, определяющиеся каждой меткой. А их столько же, сколько всего существует меток. То есть обзор всех каталогов занимает пользовательское время.

Кроме того, существует много неоднозначностей в выборе подкаталога (дочернего каталога). Поскольку неизбежно возникали синонимичные по смыслу метки, и выбор нужной метки обременен возможностью ошибки, а именно выбором похожей метки, а не метки, которая на самом деле проставлена у файла. А, следовательно, возникает необходимость просмотра всего списка подкаталога и, при обнаружении ошибки возвращение назад для выбора правильного пути.

Рассмотрим далее, каким образом отражаются на состоянии хранилища действия над файлами, происходящие в виртуальной файловой системе.

Под обычными действиями понимаются: добавление, переименование, копирование, перемещение, удаление, открытие файлов или каталогов.

Самое очевидная операция — открытие. Файл должен быть запущен в соответствующей программе и, возможно, после закрытия, его можно было бы сохранить.

Добавление файла по некоторому пути из внешней файловой системы должно вызывать добавление данного файла в хранилище со всеми метками пути, по которому кладется новый файл. Добавление каталога из описанных правил является неуместным. Ведь каталог получается, только если существует файл с меткой этого каталога.

Переименование файла не представляет собой ничего особенного, то есть простая замена имени у файла в хранилище. А вот переименование каталога может быть опасным. Поскольку, по сути, должна быть переименована метка, связанная со многим количеством файлов, лежащими

возможно и вне текущего пути, где производится переименование. Ввиду неочевидной семантики переименования, разумно исключить такую возможность для каталогов ВФС.

Копирование файла в другой путь можно проинтерпретировать как добавление к его меткам еще и меток из нового пути, не связанных пока с файлом. А копирование каталога, аналогично переименованию каталога является нежелательным, так как по логике определения иерархии должны быть добавлены новые метки ко всем файлам с этой меткой.

Семантика перемещения файла традиционно соответствует помещению файла по новому пути и удалению из старого. Поэтому для каталогов данную операцию не рассматриваем, а для файла должно означать удаление всех меток по тому пути, в котором он лежал, и добавление всех меток пути, по которому его переместили.

Определимся с семантикой удаления файла из ВФС. С одной стороны можно удалить из набора меток файла все метки этого пути, а можно лишь метку каталога, в котором он находился при удалении, тем самым переместив его как бы на уровень наверх, что сродни перемещению. Как стало понятно, исходя и определения, можем выбрать любой, но для начала стоит выбрать второй вариант, ибо он не так сильно влияет на набор меток файла и не приводит к значительному сокращению объема его метаинформации.

Что же касается удаления каталога, то, как и в предыдущих пунктах, данное действие слишком опасно для всей структуры. Хотя при дальнейшем или альтернативном развитии системы можно рассмотреть и вариант, что удаление каталога будет означать удаление этой метки у всех файлов по текущему пути с этой меткой.

Таким образом, были определены правила отображения из неиерархической структуры хранилища данных KAPLARO в обычную иерархическую структуру, были обозначены возможные функции над файлами и каталогами и описано, что они должны собой представлять

относительно файла и его меток. Опишем теперь реализацию ВФС на основе этих правил и возможных функций.

Реализация ВФС на основе библиотеки Dokan

Для разработки собственной виртуальной файловой системы (ВФС) может быть успешно использована библиотека Dokan.

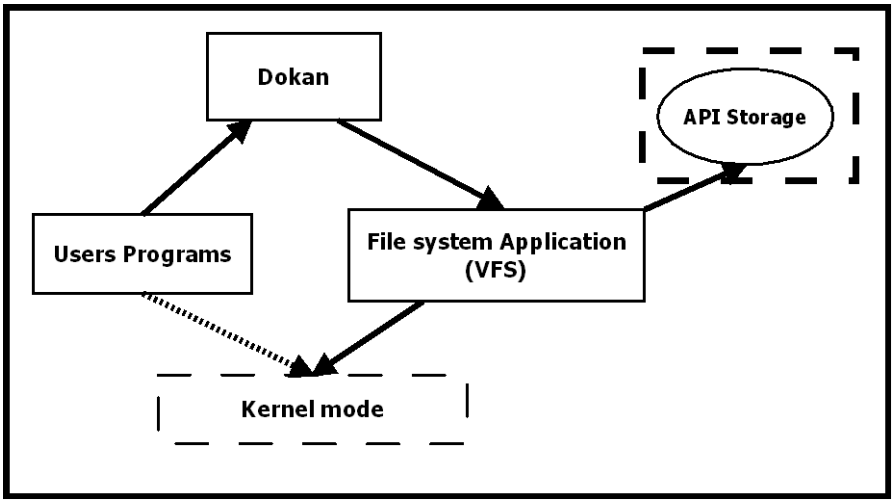

Схема 6. Архитектура виртуальной файловой системы на основе проекта Dokan.

Для создания ВФС, во-первых, требуется установить драйвер dokan.sys, поставляемым создателем Dokan, который перехватывает обращения к режиму ядра пользовательскими приложениями и передает на приложение файловой системы (сокращенно VFS на схеме от virtual file system – поскольку является основой для описания свойств и создания отдельной ВФС). Во-вторых, написать такое приложение файловой системы, в котором нужно описать callback функции на запросы к режиму ядра. В этих функциях ВФС KAPLARO обращается к интерфейсу хранилища данных, и получать необходимые данные для отображения виртуальных каталогов и файлов, а также вызывать методы производящие необходимые операции над файлами в хранилище.

Испытания предложенных методов работы классификатора

Отметим, что применительно к поставленным задачам от описанных в работе подсистем классификации не требуются результаты очень высокой

точности: на основании результатов классификации KAPLARO лишь предлагает пользователю 5 вариантов меток, которые наиболее вероятны для добавления к файлу с уже имеющимися метками. Вероятность определяется на основе специального метода в случае нетекстовых данных, когда невозможно оперировать внутренним содержимым.

ЛСА в связке с Байесовским классификатором

Как было описано выше для анализа нетекстовых данных, а именно пространства меток – файлов и их семантических, не явных связей за основу взят метод латентно-семантического анализа (ЛСА), измененный в соответствии с этой задачей.

Для получения быстрого ответа в виде нескольких наиболее вероятных меток используется наивный Байесовский классификатор, который основные вероятности в данной задаче считает не сам, а берет из результатов работы ЛСА.

Поскольку подобная связка нигде более не встречалась, необходимо было произвести проверку работоспособности совокупного метода.

Для проверки работоспособности классификатора был применён следующий метод:

1. В хранилище были импортированы 150 различных файлов, причём метками им были назначены фрагменты исходных (до импорта в KAPLARO) путей к ним (по названиям каталогов);
2. Далее на основе уже использованных путей были составлены частичные наборы меток (1-2 метки были удалены);
3. На основе частичных наборов Классификатор предложил по 5 вариантов наиболее вероятных исходя из метода не хватающих меток.

Сводные результаты тестирования представлены в приложении 1, детальные - в приложении 2.

В результате Классификатор показал себя весьма оптимистично. В 90% частичных наборов, он предложил все предполагаемые метки (убранные). В

4.26% случаях, либо не предложил одну из двух меток (когда были убраны ровно две метки), либо предложил предполагаемую метку, но не как наиболее вероятную. И в 1.42% классификатор не предложил предполагаемых меток для восстановления полного пути.

В результате работы выяснилось, что метод действительно чувствителен к порогам выбора сингулярных значений и стартовых вероятностей для подсчета взвешенной вероятностей, что потребовало дополнительной настройки.

Наивная байесовская классификация для текстовых данных

Байесовские классификаторы для текстовых данных применяются уже не один десяток лет и, не смотря на простоту, хорошо себя зарекомендовали во многих областях, начиная от систем борьбы со СПАМом, и заканчивая системами автоматизации документооборота. [10]

Поскольку байесовский классификатор в данной работе не занимает центральной позиции и никакие его доработки не производились, предполагается, что он будет выдавать информацию об ожидаемых метках файлов с приемлемым качеством.

Заключение

При выполнении данной работы были получены следующие результаты:

- Проведен обзор подходов к структуризации данных, на примере:
  - современных операционных систем, а именно Windows 7 и Mac OS X,
  - узкоспециализированных приложений, как:
    - десктопные приложения, как Adobe Photoshop Lightroomи и Windows Media Player,
    - облачные системы хранения данных и наиболее интересных из них: чувствительных к типу контента, как Picasa, WebGoogle, DriveEverNote, так и некоторые других.
- Проведен обзор подходов к классификации данных, на примере:
  - Латентно-семантического анализа,
  - наивной классификации Байеса.
- Подход и архитектура системы были представлены на 2-й межвузовской научной конференции по проблемам информатики СПИСОК-2011, а также опубликованы в материалах конференции
- Разработан прототип системы хранения данных на платформе Microsoft .NET на языке C# с использованием Microsoft SQL Server
- Спроектированы и разработаны компоненты системы: классификатор, виртуальная файловая система
- На базе Байесовской классификации, латентно семантического анализа создан механизм полуавтоматического добавления меток к файлам в системе
- Реализована функциональность по поиску файлов: по метаданным, по поисковому запросу
- Функции системы проверены на тестовом наборе данных - набор мультимедийных файлов, описанных небольшим количеством меток

Список используемых источников

[1] Галушкин А. И. Нейронные сети: основы теории. М.:Горячая линия-Телеком, 2010. 496 с.

[2] Сегаран Т. Программируем коллективный разум. Пер. с англ. СПб.: Символ-плюс, 2008. 368 с.

[3] Тверьянович М. А., Луцив Д. В. Система хранения данных с группировкой по категории // Материалы 2-й межвузовской научной конференции по проблемам информатики СПИСОК-2011. СПб.: ВВМ. 2011 С. 422-425.
http://spisok.math.spbu.ru/txt/SPISOK-2011.pdf

[4] Тверьянович М. А. Хранилище мультимедиа с группировкой данных по категории. Выпускная квалификационная работа. СПбГУ, 2010. 33 с.

[5] Chen P. The Entity-Relationship Model-Toward a Unified View of Data. // ACM Transactions on Database Systems (TODS). Vol. 1. No. 1. ACM. New York, USA. 1976. P. 9-36.

[6] Craig D. Lisa 1 Owner Guide, Apple Computer Inc. 1983. 418 p.

[7] Date C. J. An Introduction to Database Systems, 8th edition. Addison-Wesley. 2003. 1024 p.

[8] Giampaolo D. B. Practical File System Design with the Be File System. San Francisco, California: Morgan Kaufmann Publishers. 1998. 256 p.

[9] Landauer T.K., Foltz P.W., Laham D. Introduction to Latent Semantic Analysis // Discourse Processes. Vol. 25. No.2–3. 1998. P. 259–284.

[10] Manning C.D., Raghavan P., Schütze H., Introduction to Information Retrieval, Cambridge University Press. 2008.

[11] Salton G., Wong A., Yang C. S. A Vector Space Model for Automatic Indexing //Communications of the ACM. Vol. 18. No. 11. USA. 1975. P. 613–620.

[12] Stan M. Inside the Windows 95 file system. O'Reilly. 1997. 360 p.

[13] Schmitz *S.* Noncanonical LALR(1) Parsing // 10th international conference Developments in language theory (DLT). Santa Barbara, CA, USA. June, 2006. P. 95-108.

[14] Williams G. Journal Byte, BYTE Publications Inc. February, 1983. P. 37-50.

[15] Adobe. The Library module: Basic workflow, 2012. http://help.adobe.com/en_US/lightroom/using/WS31C90D9B-2D4C-490f-B72F-EDD9D8DF60B6.html

[16] Codeplex. Irony - .NET Language Implementation Kit, March, 2012. http://irony.codeplex.com/

[17] Dokan-dev. Dokan - user mode file system for windows, 2012. http://dokan-dev.net/en/

[18] Dublincore. The Dublin Core® Metadata Initiative, 2012. http://dublincore.org/

[19] LisaFAQ. What is the Apple Lisa? 2006. http://lisafaq.sunder.net/single.html#lisafaq-hs_about_lisa

[20] Macworld. Organize files with Spotlight comments, May, 2007. http://www.macworld.com/article/58012/2007/05/spotcomments.html

[21] Macworld. What's new in Lion: the Finder and files, July, 2011 http://www.macworld.com/article/1161170/lion_finder_spotlight_quick_look.html

[22] Microsoft. Поиск необходимой информации. Упорядочение данных в Windows 7, 2012. http://windows.microsoft.com/ru-RU/windows7/help/find-what-you-are-looking-for-staying-organized-in-windows-7

[23] MSDN. A Developer's Perspective on WinFS: Part 1, March, 2004. http://msdn.microsoft.com/en-us/library/ms996622.aspx

[24] MSDN. WinFS 101: Introducing the New Windows File System, March, 2004. http://msdn.microsoft.com/en-us/library/aa480687.aspx

[25] MSDN. Создание нового поколения файловой системы для Windows: ReFS, Январь 2012 http://blogs.msdn.com/b/b8_ru/archive/2012/01/20/windows-refs.aspx

[26] Udell J. Interview with Quentin Clark. Where is WinFS now? 05.15.2008. http://channel9.msdn.com/posts/JonUdell/Where-is-WinFS-now/

Приложение 1

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ	
ЛСА + Байесовский классификатор на не полных наборах меток, аналогичных наборам меток файлов, сохраненных в базу хранилища	
150	файлов в базе
58	меток в базе
626	связей файл – метка
141	не полный набор меток для тестирования
127	**Зеленый - найдены все метки**
6	**Черный - найдена метка, с вероятностью меньше максимальной**
6	**Голубой - найдена одна из двух меток**
2	**Красный - метка не найдена**
Найдено	**90,07%**
Найдено с не максимальным значением	**4,26%**
Найдено один з двух	**4,26%**
Не найдено	**1,42%**

ПРИЛОЖЕНИЕ 2

Полный путь имеющегося файла	Укороченный путь	Недостающие метки	РЕЗУЛЬТАТ наиболее вероятные метки по порядку				
MY, фотки, фильм	фотки, фильм	MY	**MY = 0,213115**	фото = 0,047130	ModelData = 0,019330	котенок = 0,018755	03.12 = 0,011866
MY, фотки, фильм	MY, фильм	фотки	**фотки = 0,139647**	фото = 0,034898	учеба = 0,025263	ModelData = 0,015400	котенок = 0,013498
MY, фотки, фильм	MY, фотки	фильм	**фото = 0,113362**	ДеньРожд2012 = 0,016969	ДеньГорода09 = 0,016969	Таллин = 0,016708	Природа = 0,012180
MY, фотки, ModelData	MY, ModelData	фотки	**фотки = 0,138269**	фото = 0,029862	учеба = 0,021039	фильм = 0,015264	котенок = 0,014098
MY, фотки, ModelData	фотки, ModelData	MY	**MY = 0,207772**	фото = 0,040329	котенок = 0,019588	фильм = 0,018977	03.12 = 0,012394
MY, фотки, ModelData	ModelData	MY,фотки	**MY = 0,283490**	фотки = 0,187845	фото = 0,047846	котенок = 0,041426	учеба = 0,031324
MY, фотки, фото	MY, фото	фотки	**фотки = 0,190334**	Таллин = 0,019279	ДеньРожд2012 = 0,019057	ДеньГорода09 = 0,019057	Природа = 0,014908
MY, фотки, фото, Таллин	фотки, Таллин	MY,фото	**MY = 0,213482**	фото = 0,141521	ДеньРожд2012 = 0,027215	ДеньГорода09 = 0,027215	Природа = 0,024350
MY, фотки, фото, Таллин	MY, фотки, Таллин	фото	**фото = 0,088327**	ДеньРожд2012 = 0,015468	ДеньГорода09 = 0,015468	Природа = 0,011017	Кот = 0,007268
MY, фотки, фото, Таллин	MY, фото, Таллин	фотки	**фотки = 0,129242**	ДеньРожд2012 = 0,017371	ДеньГорода09 = 0,017371	Природа = 0,013485	Кот = 0,009012
MY, фотки, фото, Таллин	фотки, фото, Таллин	MY	**MY = 0,132372**	ДеньГорода09 = 0,023464	ДеньРожд2012 = 0,023464	Природа = 0,018314	Кот = 0,012262
MY, фотки, фото, Таллин	фото, Таллин	MY,фотки	**MY = 0,180613**	фотки = 0,175581	ДеньГорода09 = 0,030564	ДеньРожд2012 = 0,030564	Природа = 0,029803
MY, фотки, фото, Природа	фотки, Природа	MY,фото	**MY = 0,185292**	фото = 0,129138	Таллин = 0,028480	ДеньРожд2012 = 0,026615	ДеньГорода09 = 0,026615
MY, фотки, фото, Природа, Бабочка	MY, Природа, Бабочка	фотки, фото	**фотки = 0,070955**	фото = 0,060358	Таллин = 0,016598	ДеньРожд2012 = 0,015327	ДеньГорода09 = 0,015327
MY, фотки, фото, Природа, Бабочка	MY, фотки, Бабочка	фото,Природа	**фото = 0,071557**	ДеньРожд2012 = 0,013199	ДеньГорода09 = 0,013199	Таллин = 0,013189	Природа = 0,009741
MY, фотки, фото, Природа, Кот	фотки, фото, Кот	MY, Природа	**MY = 0,109531**	Таллин = 0,020614	ДеньГорода09 = 0,020023	ДеньРожд2012 = 0,020023	Природа = 0,016193
MY, фотки, фото, Природа, Кот	фотки, Природа, Кот	MY,фото	**фото = 0,081515**	MY = 0,078693	Таллин = 0,022482	ДеньГорода09 = 0,020703	ДеньРожд2012 = 0,020703
MY, фотки, фото, Природа	MY, фото, Природа	фотки	**фотки = 0,111873**	Таллин = 0,017437	ДеньРожд2012 = 0,016989	ДеньГорода09 = 0,016989	Кот = 0,009145
MY, фотки, фото, Разное, Творчество	MY, фотки, фото, Разное	Творчество	**Творчество = 0,003854**	figu = 0,002158	ДеньГорода09 = 0,001895	ДеньРожд2012 = 0,001895	выставка фигурок = 0,001359
MY, фотки, фото, Разное, Творчество	MY, фото, Творчество	фотки,Разное	**фотки = 0,094738**	Разное = 0,011118	figu = 0,004767	ДеньГорода09 = 0,004524	ДеньРожд2012 = 0,004524
MY, фотки, фото, Разное, Творчество	фото, Разное, Творчество	MY,фотки	**фотки = 0,060045**	MY = 0,048905	figu = 0,015039	выставка фигурок = 0,011080	ДеньГорода09 = 0,001031
MY, фотки, фото, Разное, figu	MY, фотки, фото, figu	Разное	**Разное = 0,005548**	Творчество = 0,003630	Природа = 0,002007	Таллин = 0,001554	Кот = 0,001412
MY, фотки, фото, Разное, figu	MY, фото, Разное, figu	фотки	**фотки = 0,035774**	Творчество = 0,006485	выставка фигурок = 0,0032	Природа = 0,000401	Бабочка = 0,000341
MY, фотки, фото, Разное, figu	Разное, фотки, figu	MY, фото	**фото = 0,047860**	MY = 0,046252	Творчество = 0,013979	выставка фигурок = 0,009758	Природа = 0,000725
MY, фотки, фото, Разное, figu, выставка фигурок	MY, фотки, Разное, выставка фигурок	фото,figu	**фото = 0,027457**	Творчество = 0,004832	figu = 0,003526	Природа = 0,000420	Бабочка = 0,000342
MY, фотки, фото, Разное, figu, выставка фигурок	фотки, фото, figu, выставка фигурок	MY, Разное	**MY = 0,025606**	Разное = 0,012609	Творчество = 0,007294	Природа = 0,001245	Бабочка = 0,000955
MY, фотки, фото, Разное	MY, Разное	фотки,фото	**фотки = 0,105835**	фото = 0,074330	Творчество = 0,011664	figu = 0,010935	выставка фигурок = 0,008024
MY, фотки, фото, Разное	фото, Разное	MY, фотки	**MY = 0,124425**	фотки = 0,120634	Творчество = 0,018689	figu = 0,017925	выставка фигурок = 0,013295
MY, фотки, котенок	котенок	MY, фотки	**MY = 0,197023**	фотки = 0,143344	ModelData = 0,030647	06.11 = 0,029281	03.12 = 0,028777
MY, фотки, котенок	фотки, котенок	MY	**MY = 0,144400**	ModelData = 0,020117	фильм = 0,018871	фото = 0,017740	03.12 = 0,013603
MY, фотки, котенок, 06.11	фотки, 06.11	MY, котенок	**MY = 0,141389**	фото = 0,028659	котенок = 0,019755	ModelData = 0,018396	фильм = 0,017253
MY, фотки, котенок, 06.11	MY, котенок	фотки, 06.11	**фотки = 0,105513**	ModelData = 0,016028	фильм = 0,015178	фото = 0,013135	учеба = 0,010907
MY, фотки, котенок, 06.11	котенок, 06.11	MY, фотки	**MY = 0,066975**	фотки = 0,063115	ModelData = 0,024776	03.12 = 0,024452	фильм = 0,022141
MY, фотки, фото, ДеньРожд2012	MY, фотки, ДеньРожд2012	фото	**фото = 0,090331**	ДеньГорода09 = 0,015272	Таллин = 0,014993	Природа = 0,010692	Кот = 0,007039
MY, фотки, фото, ДеньРожд2012	фотки, фото, ДеньРожд2012	MY	**MY = 0,139769**	Таллин = 0,023433	ДеньГорода09 = 0,023167	Природа = 0,017774	Кот = 0,011876
MY, фотки, фото, ДеньРожд2012	фотки, ДеньРожд2012	MY, фото	**MY = 0,225411**	фото = 0,144732	Таллин = 0,028255	ДеньГорода09 = 0,026870	Природа = 0,023632

Полный путь имеющегося файла	Укороченный путь	Недостающие метки	РЕЗУЛЬТАТ наиболее вероятные метки по порядку				
МУ, фотки, фото, ДеньРожд2012	фото, ДеньРожд2012	МУ, фотки	**МУ = 0,190705**	фотки = 0,184354	Таллин = 0,032603	ДеньГорода09 = 0,030176	Природа = 0,028925
МУ, фотки, фото, ДеньРожд2012	МУ, ДеньРожд2012	фотки, фото	**фотки = 0,161738**	фото = 0,107168	Таллин = 0,020859	ДеньГорода09 = 0,019893	Природа = 0,017400
МУ, фотки, фото, ДеньГорода09	МУ, фотки, ДеньГорода09	фото	**фото = 0,090331**	ДеньРожд2012 = 0,015272	Таллин = 0,014993	Природа = 0,010692	Кот = 0,007039
МУ, фотки, фото, ДеньГорода09	фотки, фото, ДеньГорода09	МУ	**МУ = 0,139769**	Таллин = 0,023433	ДеньРожд2012 = 0,023167	Природа = 0,017774	Кот = 0,011876
МУ, фотки, фото, ДеньГорода09	МУ, фото, ДеньГорода09	фотки	**фотки = 0,135700**	Таллин = 0,017299	ДеньРожд2012 = 0,017151	Природа = 0,013087	Кот = 0,008728
МУ, фотки, фото, ДеньГорода09	фото, ДеньГорода09	МУ,фотки	**МУ = 0,190705**	фотки = 0,184354	Таллин = 0,032603	ДеньРожд2012 = 0,030176	Природа = 0,028925
МУ, фотки, фото, ДеньГорода09	фотки, ДеньГорода09	МУ, фото	**МУ = 0,225411**	фото = 0,144732	Таллин = 0,028255	ДеньРожд2012 = 0,026870	Природа = 0,023632
МУ, учеба, 2011	МУ, 2011	учеба	**учеба = 0,080530**	2010 = 0,013621	фотки = 0,013318	Философия = 0,008666	фото = 0,008664
МУ, учеба, 2011	учеба, 2011	МУ	**МУ = 0,126457**	2010 = 0,018883	exam = 0,012834	Философия = 0,012586	old = 0,010161
МУ, учеба, 2011	2011	МУ, учеба	**МУ = 0,189325**	учеба = 0,119898	2010 = 0,031299	exam = 0,028428	Философия = 0,019384
МУ, учеба, 2011	учеба	МУ, 2011	**МУ = 0,379061**	old = 0,163978	ЧМВ = 0,036367	тРАНСЛЯЦИИ = 0,036367	АВС = 0,036367
МУ, учеба, 2011, exam	2011, exam	МУ, учеба	**МУ = 0,062758**	учеба = 0,052179	2010 = 0,024633	Философия = 0,007446	наука и просвещение = 0,004
МУ, учеба, 2011, exam	учеба, exam	МУ,2011	**МУ = 0,125653**	old = 0,024785	2011 = 0,016796	2010 = 0,016460	Философия = 0,013673
МУ, учеба, 2010	учеба, 2010	МУ	**МУ = 0,162420**	old = 0,033736	2011 = 0,018261	тРАНСЛЯЦИИ = 0,012422	АВС = 0,012422
МУ, учеба, 2010	2010	МУ, учеба	**МУ = 0,243167**	учеба = 0,144632	old = 0,038907	2011 = 0,038085	фотки = 0,033441
МУ, учеба, 2010	МУ, 2010	учеба	**учеба = 0,097142**	фотки = 0,024615	old = 0,022860	2011 = 0,012318	фото = 0,010783
МУ, учеба, old, Философия	МУ, учеба, old	Философия	**ЧМВ = 0,022041**	тРАНСЛЯЦИИ = 0,022041	АВС = 0,022041	вопросы = 0,019904	Философия = 0,014610
МУ, учеба, old, Философия	учеба, Философия	МУ, old	**МУ = 0,167671**	old = 0,143727	АВС = 0,029398	ЧМВ = 0,029398	тРАНСЛЯЦИИ = 0,029398
МУ, учеба, old, Философия, наука и просвещение	учеба, Философия, наука и просвещение	МУ, old	**old = 0,114138**	МУ = 0,071192	АВС = 0,021555	ЧМВ = 0,021555	тРАНСЛЯЦИИ = 0,021555
МУ, учеба, old, Философия, наука и просвещение	МУ, old, Философия, наука и просвещение	учеба	**учеба = 0,053370**	АВС = 0,013970	ЧМВ = 0,013970	тРАНСЛЯЦИИ = 0,013970	вопросы = 0,012283
МУ, учеба, old, Философия, наука и просвещение	учеба, old, наука и просвещение	МУ, Философия	**МУ = 0,093901**	Философия = 0,027767	АВС = 0,025728	ЧМВ = 0,025728	тРАНСЛЯЦИИ = 0,025728
МУ, учеба, old, Философия, счастье	МУ, учеба, old, счастье	Философия	**ЧМВ = 0,016161**	тРАНСЛЯЦИИ = 0,016161	АВС = 0,016161	вопросы = 0,014446	Философия = 0,012414
МУ, учеба, old, Философия, счастье	МУ, Философия, счастье	учеба, old	**old = 0,077342**	учеба = 0,061835	АВС = 0,014478	ЧМВ = 0,014478	тРАНСЛЯЦИИ = 0,014478
МУ, учеба, old, Философия, счастье	МУ, учеба, счастье	old, Философия	**old = 0,076513**	ЧМВ = 0,016749	тРАНСЛЯЦИИ = 0,016749	АВС = 0,016749	вопросы = 0,015023
МУ, учеба, old, АВС	МУ, учеба, АВС	old	**old = 0,085456**	ЧМВ = 0,020559	тРАНСЛЯЦИИ = 0,020559	вопросы = 0,018626	Философия = 0,012456
МУ, учеба, old, АВС	учеба, old, АВС	МУ	**МУ = 0,130996**	ЧМВ = 0,031580	тРАНСЛЯЦИИ = 0,031580	вопросы = 0,028222	Философия = 0,025583
МУ, учеба, old, АВС	МУ, old, АВС	учеба	**учеба = 0,126829**	ЧМВ = 0,021212	тРАНСЛЯЦИИ = 0,021212	вопросы = 0,019037	Философия = 0,017615
МУ, учеба, old, АВС	old, АВС	МУ, учеба	**МУ = 0,196120**	учеба = 0,188832	Философия = 0,039401	ЧМВ = 0,033770	тРАНСЛЯЦИИ = 0,033770
МУ, учеба, old, АВС	учеба, АВС	МУ, old	**МУ = 0,224530**	old = 0,145441	ЧМВ = 0,032730	тРАНСЛЯЦИИ = 0,032730	вопросы = 0,029349
МУ, учеба, old, ЧМВ	МУ, учеба, ЧМВ	old	**old = 0,085456**	тРАНСЛЯЦИИ = 0,020559	АВС = 0,020559	вопросы = 0,018626	Философия = 0,012456
МУ, учеба, old, ЧМВ	МУ, old, ЧМВ	учеба	**учеба = 0,126829**	тРАНСЛЯЦИИ = 0,021212	АВС = 0,021212	вопросы = 0,019037	Философия = 0,017615
МУ, учеба, old, ЧМВ	учеба, old, ЧМВ	МУ	**МУ = 0,130996**	тРАНСЛЯЦИИ = 0,031580	АВС = 0,031580	вопросы = 0,028222	Философия = 0,025583
МУ, учеба, old, ЧМВ	МУ, ЧМВ	учеба, old	**учеба = 0,146944**	old = 0,098554	тРАНСЛЯЦИИ = 0,021985	АВС = 0,021985	вопросы = 0,019797
МУ, учеба, old, ЧМВ	old, ЧМВ	МУ, учеба	**МУ = 0,196120**	учеба = 0,188832	Философия = 0,039401	тРАНСЛЯЦИИ = 0,033770	АВС = 0,033770
МУ, учеба, old, вопросы	МУ, учеба, вопросы	old	**old = 0,083320**	ЧМВ = 0,020099	тРАНСЛЯЦИИ = 0,020099	АВС = 0,020099	Философия = 0,012092
МУ, учеба, old, вопросы	old, вопросы	МУ, учеба	**МУ = 0,195312**	учеба = 0,185830	Философия = 0,038249	ЧМВ = 0,033014	тРАНСЛЯЦИИ = 0,033014
МУ, учеба, old, вопросы	учеба, old, вопросы	МУ	**МУ = 0,130456**	ЧМВ = 0,030873	тРАНСЛЯЦИИ = 0,030873	АВС = 0,030873	Философия = 0,024835

Полный путь имеющегося файла	Укороченный путь	Недостающие метки	РЕЗУЛЬТАТ наиболее вероятные метки по порядку				
MY, учеба, old, вопросы	MY, old, вопросы	учеба	**учеба = 0,124812**	ЧМВ = 0,020737	тРАНСЛЯЦИИ = 0,020737	ABC = 0,020737	Философия = 0,017100
MY, учеба, old, тРАНСЛЯЦИИ	MY, учеба, тРАНСЛЯЦИИ	old	**old = 0,085456**	ЧМВ = 0,020559	ABC = 0,020559	вопросы = 0,018626	Философия = 0,012456
MY, учеба, old, тРАНСЛЯЦИИ	MY, old, тРАНСЛЯЦИИ	учеба	**учеба = 0,126829**	ЧМВ = 0,021212	ABC = 0,021212	вопросы = 0,019037	Философия = 0,017615
MY, учеба, old, тРАНСЛЯЦИИ	учеба, old, тРАНСЛЯЦИИ	MY	**MY = 0,130996**	ЧМВ = 0,031580	ABC = 0,031580	вопросы = 0,028222	Философия = 0,025583
MY, учеба, old, тРАНСЛЯЦИИ	учеба, тРАНСЛЯЦИИ	MY, old	**MY = 0,224530**	old = 0,145441	ЧМВ = 0,032730	ABC = 0,032730	вопросы = 0,029349
MY, учеба, old, тРАНСЛЯЦИИ	old, тРАНСЛЯЦИИ	MY, учеба	**MY = 0,196120**	учеба = 0,188832	Философия = 0,039401	ЧМВ = 0,033770	ABC = 0,033770
MY, учеба, old	учеба, old	MY	**MY = 0,221153**	ЧМВ = 0,035089	тРАНСЛЯЦИИ = 0,035089	ABC = 0,035089	Философия = 0,032679
MY, учеба, old	MY, учеба	old	**old = 0,096348**	ЧМВ = 0,022844	тРАНСЛЯЦИИ = 0,022844	ABC = 0,022844	вопросы = 0,020699
Games, World, Errors	Games, World	Errors	**Interface = 0,091595**	Cinematics = 0,025685	Data = 0,025685	AddOns = 0,020916	Errors = 0,018972
Games, World, Errors	Games, Errors	World	**World = 0,100136**	Interface = 0,038256	Logs = 0,026467	The Settlers II = 0,018426	Dancing Craze = 0,016097
Games, World, Errors	World, Errors	Games	**Games = 0,173797**	Interface = 0,047943	Logs = 0,019244	AddOns = 0,017947	Lib = 0,010473
Games, World, Errors	Errors	Games, World	**Games = 0,194838**	World = 0,111768	Interface = 0,051052	MY = 0,045400	The Settlers II = 0,035010
Games, World, Data, Interface, Cinematics	Games, World, Data, Interface	Cinematics	**Cinematics = 0,017850**	Errors = 0,001054	Logs = 0,001054	AddOns = 0,000735	Lib = 0,000510
Games, World, Data, Interface, Cinematics	Games, World, Data, Cinematics	Interface	**Interface = 0,047365**	Logs = 0,000383	Errors = 0,000383	AddOns = 6,449798	Recount = 5,713365
Games, World, Data, Interface, Cinematics	Games, Data, Interface, Cinematics	World	**World = 0,064717**	Errors = 0,000237	Logs = 0,000237	AddOns = 6,090943	Recount = 5,408154
Games, World, Data, Interface, Cinematics	World, Data, Interface, Cinematics	Games	**Games = 0,046471**	Errors = 0,000172	Logs = 0,000172	AddOns = 7,951848	Recount = 7,149765
Games, World, Data, Interface, Cinematics	Data, Interface, Cinematics	Games, World	**World = 0,072235**	Games = 0,052098	Errors = 0,000274	Logs = 0,000274	AddOns = 0,000118
Games, World, Data, Interface, Cinematics	Games, Data, Interface	World, Cinematics	**World = 0,101043**	Cinematics = 0,027310	Errors = 0,001671	Logs = 0,001671	AddOns = 0,001096
Games, World, Data, Interface, Cinematics	World, Interface, Cinematics	Games, Data	**Games = 0,087404**	Data = 0,033058	AddOns = 0,001431	Errors = 0,001215	Logs = 0,001215
Games, World, Data, Interface, Cinematics	World, Data, Cinematics	Games, Interface	**Interface = 0,063207**	Games = 0,062674	Logs = 0,000442	Errors = 0,000442	AddOns = 0,000125
Games, World, Data, Interface, Cinematics	Games, Data, Cinematics	World, Interface	**World = 0,069504**	Interface = 0,050437	Logs = 0,000608	Errors = 0,000608	The Settlers II = 0,000143
Games, World, Data, Interface, Cinematics	Games, World, Interface, Cinematics	Data	**Data = 0,017850**	Errors = 0,001054	Logs = 0,001054	AddOns = 0,000735	Lib = 0,000510
Games, World, Data, Interface, Cinematics	Games, World, Interface	Data, Cinematics	**Cinematics = 0,018790**	Data = 0,018790	AddOns = 0,013252	NugMiniPet = 0,007922	Lib = 0,007558
Games, World, Logs	Games	World, Logs	**World = 0,169429**	Interface = 0,097535	Cinematics = 0,039298	Data = 0,039298	AddOns = 0,031176
Games, World, Logs	World	Games, Logs	**Games = 0,221706**	Interface = 0,122231	Cinematics = 0,047568	Data = 0,047568	AddOns = 0,040701
Games, World, Logs	World, Logs	Games	**Games = 0,173797**	Interface = 0,047943	Errors = 0,019244	AddOns = 0,017947	Lib = 0,010473
Games, World, Logs	Games, Logs	World	**World = 0,100136**	Interface = 0,038256	Errors = 0,026467	The Settlers II = 0,018426	Dancing Craze = 0,016097
Games, World, Interface, AddOns, SlideBar	Games, World, Interface, SlideBar	AddOns	**AddOns = 0,011264**	NugMiniPet = 0,006727	Lib = 0,006423	Recount = 0,005956	Errors = 0,003255
Games, World, Interface, AddOns, SlideBar	World, AddOns, SlideBar, Lib	Games, Interface	**Games = 0,024722**	Interface = 0,022954	Recount = 0,012714	NugMiniPet = 0,012166	Errors = 0,001801
Games, World, Interface, AddOns, SlideBar, Lib	Games, World, Interface, AddOns, Lib	SlideBar	**NugMiniPet = 0,005200**	Recount = 0,004611	SlideBar = 0,004574	Errors = 0,001390	Logs = 0,001390
Games, World, Interface, AddOns, Recount	Interface, AddOns, Recount	Games, World	**World = 0,070278**	Games = 0,044601	Lib = 0,018921	NugMiniPet = 0,016918	SlideBar = 0,016612

Полный путь имеющегося файла	Укороченный путь	Недостающие метки	РЕЗУЛЬТАТ наиболее вероятные метки по порядку				
Games, World, Interface, AddOns, Recount	Games, Interface, Recount	World, AddOns	**World = 0,096537**	AddOns = 0,017378	Lib = 0,010082	NugMiniPet = 0,009918	SlideBar = 0,009103
Games, World, Interface, AddOns, Recount	World, AddOns, Recount	Games, Interface	**Games = 0,053655**	Interface = 0,044435	Lib = 0,020006	NugMiniPet = 0,017996	SlideBar = 0,017595
Games, World, Interface, AddOns, Recount	World, Interface, AddOns	Games, Recount	**Games = 0,083051**	Lib = 0,014185	NugMiniPet = 0,013513	Recount = 0,013274	SlideBar = 0,012686
Games, World, Interface, AddOns, NugMiniPet	World, Interface, NugMiniPet	Games, AddOns	**Games = 0,082687**	AddOns = 0,020612	Lib = 0,012054	Recount = 0,011278	SlideBar = 0,010788
Games, World, Interface, AddOns, NugMiniPet	Games, World, NugMiniPet	Interface, AddOns	**Interface = 0,053270**	AddOns = 0,016718	Lib = 0,009675	Recount = 0,009012	SlideBar = 0,008760
The Settlers II	Games, The Settlers II	doc	**World = 0,027427**	Logs = 0,022296	Errors = 0,022296	Dancing Craze = 0,018469	Alawar.ru = 0,018469
Games, The Settlers II, doc	The Settlers II, doc	Games	**Games = 0,062495**	Alawar.ru = 0,024399	Dancing Craze = 0,024399	Logs = 0,018014	Errors = 0,018014
Games, The Settlers II, doc	Games, doc	The Settlers II	**World = 0,028145**	The Settlers II = 0,022765	Logs = 0,021081	Errors = 0,021081	Interface = 0,020409
Games, The Settlers II, doc	The Settlers II	Games, doc	**Games = 0,127608**	doc = 0,032059	World = 0,030613	MY = 0,028375	Alawar.ru = 0,028296
Games, Alawar.ru, Dancing Craze	Dancing Craze	Games, Alawar.ru	**Games = 0,142396**	MY = 0,075667	World = 0,061486	фотки = 0,041092	The Settlers II = 0,038753
Games, Alawar.ru, Dancing Craze	Alawar.ru, Dancing Craze	Games	**Games = 0,081581**	The Settlers II = 0,030545	doc = 0,021376	World = 0,019991	Logs = 0,018334
Мои документы, study, магистратура	Мои документы, study	магистратура	**магистратура = 0,161938**	БД = 0,066488	Oracle = 0,041096	OpenEdge = 0,032990	полезная инф = 0,018660
Мои документы, study, магистратура	магистратура	Мои документы, study	**study = 0,165618**	Мои документы = 0,165618	БД = 0,083280	Oracle = 0,056969	OpenEdge = 0,033818
Мои документы, study, магистратура,полезная инф	Мои документы, магистратура, полезная инф	study	**study = 0,106757**	БД = 0,021482	OpenEdge = 0,020784	Viruses = 0,010913	Oracle = 0,009897
Мои документы, study, магистратура,полезная инф	Мои документы, study, полезная инф	магистратура	**магистратура = 0,106757**	БД = 0,021482	OpenEdge = 0,020784	Viruses = 0,010913	Oracle = 0,009897
Мои документы, study, магистратура, БД, OpenEdge	Мои документы, study, магистратура, БД	OpenEdge	**Oracle = 0,028375**	OpenEdge = 0,026549	ТЕКСТЫ заданию APEX = 0	pdf = 0,011154	полезная инф = 0,004211
Мои документы, study, магистратура, БД, OpenEdge	Мои документы, магистратура, OpenEdge	study, БД	**study = 0,142202**	БД = 0,050612	Oracle = 0,028880	полезная инф = 0,012614	ТЕКСТЫ заданию APEX = 0,011
Мои документы, study, магистратура, БД, OpenEdge	study, магистратура, OpenEdge	Мои документы, БД	**Мои документы = 0,142202**	БД = 0,050612	Oracle = 0,028880	полезная инф = 0,012614	ТЕКСТЫ заданию APEX = 0,011
Мои документы, study, магистратура, БД, OpenEdge	Мои документы, study, БД, OpenEdge	магистратура	**магистратура = 0,112248**	Oracle = 0,027643	ТЕКСТЫ заданию APEX = 0	pdf = 0,011033	полезная инф = 0,003676
Мои документы, study, магистратура, БД, Oracle	study, БД, Oracle	Мои документы, магистратура	**Мои документы = 0,092565**	магистратура = 0,092565	pdf = 0,023513	ТЕКСТЫ заданию APEX = 0,021056	OpenEdge = 0,020905

Полный путь имеющегося файла	Укороченный путь	Недостающие метки	РЕЗУЛЬТАТ наиболее вероятные метки по порядку				
Мои документы, study, магистратура, БД, Oracle	Мои документы, БД, Oracle	study, магистратура	**study = 0,092565**	магистратура = 0,092565	pdf = 0,023513	ТЕКСТЫ заданию APEX = 0,021056	OpenEdge = 0,020905
Мои документы, study, магистратура, БД, Oracle	Мои документы, study, Oracle	магистратура, БД	**магистратура = 0,114661**	БД = 0,062928	OpenEdge = 0,024721	pdf = 0,016698	ТЕКСТЫ заданию APEX = 0,015987
Мои документы, study, магистратура, БД, Oracle	study, магистратура, БД, Oracle	Мои документы	**Мои документы = 0,090508**	OpenEdge = 0,020394	pdf = 0,015796	ТЕКСТЫ заданию APEX = 0,015319	полезная инф = 0,000976
Мои документы, study, магистратура, БД, Oracle, ТЕКСТЫ заданию APEX	Мои документы, study, магистратура, БД, ТЕКСТЫ заданию APEX	Oracle	**Oracle = 0,024118**	OpenEdge = 0,018273	pdf = 0,009452	полезная инф = 0,000960	utils = 5,759978
Мои документы, study, магистратура, БД, Oracle, ТЕКСТЫ заданию APEX	Мои документы, БД, Oracle, ТЕКСТЫ заданию APEX	study, магистратура	**study = 0,060639**	магистратура = 0,060639	pdf = 0,019926	OpenEdge = 0,014389	полезная инф = 0,000287
Мои документы, study, магистратура, БД, Oracle, ТЕКСТЫ заданию APEX	study, магистратура, ТЕКСТЫ заданию APEX	Мои документы, БД, Oracle	**Мои документы = 0,106085**	БД = 0,056173	Oracle = 0,034930	OpenEdge = 0,022706	pdf = 0,014873
Мои документы, study, магистратура, БД, Oracle, pdf	Мои документы, study, магистратура, pdf	БД, Oracle	**БД = 0,046895**	Oracle = 0,026615	OpenEdge = 0,021740	ТЕКСТЫ заданию APEX = 0,010940	полезная инф = 0,002034
Мои документы, study, магистратура, БД, Oracle, pdf	Мои документы, БД, Oracle, pdf	study, магистратура	**study = 0,058799**	магистратура = 0,058799	ТЕКСТЫ заданию APEX = 0,018891	OpenEdge = 0,014122	полезная инф = 0,000177
Мои документы, study, магистратура, БД, Oracle, pdf	Мои документы, магистратура, БД, pdf	study, Oracle	**study = 0,081197**	Oracle = 0,035314	OpenEdge = 0,018385	ТЕКСТЫ заданию APEX = 0,014410	полезная инф = 0,000765
Мои документы, study, магистратура, БД, Oracle, pdf	магистратура, БД, Oracle, pdf	Мои документы, study	**study = 0,058799**	Мои документы = 0,058799	ТЕКСТЫ заданию APEX = 0,018891	OpenEdge = 0,014122	полезная инф = 0,000177
Мои документы, study, магистратура, БД, Oracle, pdf	study, БД, Oracle, pdf	Мои документы, магистратура	**Мои документы = 0,058799**	магистратура = 0,058799	ТЕКСТЫ заданию APEX = 0,018891	OpenEdge = 0,014122	полезная инф = 0,000177
Мои документы, study, магистратура, Viruses, books	Мои документы, Viruses, books	study, магистратура	**магистратура = 0,047434**	study = 0,047434	полезная инф = 0,016319	utils = 0,012749	OpenEdge = 0,008177
Мои документы, study, магистратура, Viruses, books	Мои документы, магистратура,books	study, Viruses	**study = 0,084337**	OpenEdge = 0,015928	полезная инф = 0,014413	Viruses = 0,011324	Oracle = 0,008944
Мои документы, study, магистратура, Viruses	Мои документы,магистратура,	study	**study = 0,089055**	OpenEdge = 0,016522	полезная инф = 0,016363	books = 0,009387	utils = 0,008780
Мои документы, study, магистратура, Viruses, utils	магистратура, Viruses, utils	Мои документы, study	**Мои документы = 0,047126**	study = 0,047126	полезная инф = 0,017161	books = 0,013884	OpenEdge = 0,008025
Мои документы, study, магистратура, Viruses, utils	Мои документы, study, Viruses, utils	магистратура	**магистратура = 0,046079**	полезная инф = 0,013290	books = 0,007976	OpenEdge = 0,007829	Oracle = 0,001535
Мои документы, study, магистратура, Viruses, utils	study, магистратура, utils	Мои документы, Viruses	**Мои документы = 0,083791**	OpenEdge = 0,015632	полезная инф = 0,015156	Viruses = 0,012030	Oracle = 0,008921