1. Обзор предметной области

1.1 Раскрытие проблемы

Обнаружение источников инсайдерской информации — актуальная на данный момент проблема. Инсайдерская информация — конфиденциальная информация организации, переданная третьим лицам. В дальнейшем используются термины «конфиденциальная информация» в значении, равнозначном термину «инсайдерская информация». Обусловлена необходимость в системе, способной на анализ базы текстов и поиск в ней упоминаний конкретных слов, в том числе, искаженных, запросами бизнеса, нуждающегося в средствах для определения факта и возможного источника утечки информации.

Постановка задачи была следующая: необходимо определить факт и источник «утечки» инсайдерской информации на основе снимков экрана, делавшихся на компьютерах предприятия в течение некоторого времени. Было решено разработать два приложения: первое распознавало окна систем обмена мгновенными сообщениями на снимках экрана и распознавала текст, содержащихся в них. Распознав информацию, приложение составляло XML-файл, содержащий в себе информацию о сообщениях и текст сообщений, найденных на снимках экрана. Это стороннее приложение, не рассматриваемое в данной работе.

Второе приложение, создание которого было целью данной работы, принимает в качестве входных данных XML-файл, составленный первым приложением и производит поиск по сообщениям, хранящимся в данном файле. Для ориентации системы на выявление инсайдерской информации, кроме информационного поиска, были признаны нужными следующие функции: определение семантической связи между сообщениями (обусловлено особенностями систем обмена мгновенными сообщениями, при общении в которых обсуждение вопроса зачастую размывается на несколько сообщений), возможность выявления намеренного или случайного искажения слов (включая случайные опечатки, свойственные для общения в сети «интернет» и намеренные искажения слов, призванные скрыть факт распространения конфиденциальной информации).

Таким образом, приложение для обнаружения вероятных источников инсайдерской информации представляет собой локальную поисковую систему, способную вести поиск в заданном банке данных, учитывая возможность искажения слов, а также строить связи между сообщениями.

* 1. Существующие решения

Существует целый класс систем, решающий схожую задачу, это различные поисковые системы. Среди них находятся веб-решения, производящие поиск по всемирной сети «Интернет» и локальные решения, производящие поиск по локальным банкам документов.

Возможности поисковых машин по сети «Интернет» лежат в другой сфере и потому эти приложения не способны на решение подобной задачи. Поисковые машины не способны вести поиск по информации, расположенной локально, а также функционал большинства из них не соответствует требованиям поставленной задачи.

Локальные решения включают в себя, в первую очередь, различные юридически-правовые системы, такие как «Консультант Плюс», «Гарант», «Кодекс». Вышеперечисленные системы предоставляют функционал поиска по заданной разработчиками базе документов. Присутствует возможность формирования сложных поисковых запросов, включающих в себя несколько терминов, ранжирование результатов поиска по релевантности. Точно так же, отсутствует возможность изменять заданный банк данных, по которому ведётся поиск, а также функционал в сфере определения связей между документами и выявления искаженных слов не превосходит возможности обычной поисковой системы, что не является достаточным для решения поставленной задачи.

Кроме вышеперечисленных готовых систем, существуют библиотеки, реализующие некоторые алгоритмы информационного поиска, в том числе, Apache Lucene. Библиотека Apache Lucene представляет собой набор методов и структур данных, реализующих задачу индексации, токенизации, запроса и других этапов информационного поиска. Библиотеки не являются готовым приложением, способным на выполнение задачи, а приложения, построенные на них, плохо поддаются глобальным изменениям требований и запросов.

В нижеприведённой таблице сгруппированы существующие решения и возможности, которыми они обладают, критичные для решения поставленной задачи, в сравнении с разрабатываемым приложением:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ФункционалПриложение | Возможность задавать банк документов | Возможность использования связей между документами | Возможность поиска намеренно искаженных слов | Готовое приложение |
| Интернет-поисковики | — | + | — | + |
| Локальные поисковые системы | — | — | — | + |
| Библиотеки для информационного поиска | + | +/— | +/— | — |
| Разрабатываемое приложение | + | + | + | + |

 Таблица 1. Сравнение функционала существующих систем и разрабатываемого приложения.

* 1. Информационный поиск

«Информационный поиск — процесс поиска неструктурированной документальной информации, удовлетворяющей информационные потребности и наука об этом поиске.» [1].

Решить задачу по поиску вероятных источников инсайдерской информации способна система информационного поиска с возможностью определения искаженной информации и построения семантических связей между документами. При построении приложения для информационного поиска, как правило, выделяют две части приложения: отвечающую за индексацию и за выполнение запросов.

Индексация — процесс построения матрицы «термин»-«документ», позволяющий сократить объем хранимой информации и ускорить последующий процесс поиска. Существуют различные подходы к составлению индекса, в число которых входит инвертированный индекс, представляющий собой построение индекса, состоящего из списка «термин»-«вхождения термина в документы» и позволяющий значительно сократить объем информации, хранимой в памяти.

Процесс индексации включает в себя несколько ступеней: токенизация, обработка терминов (слов) и, собственно, построение индекса.

Токенизация — процесс разделения текста на лексические единицы (токены) [1]. Как правило, производится разбиение текста по пробелам или другим символам, в зависимости от постановки задачи.

Морфологическая обработка терминов представляет собой приведение слова к первоначальной словоформе, что позволяет находить различные формы слова, не включая их в индекс, что способствует сокращению размеров индекса.

Построение индекса — процесс объединения словоформ в индекс, содержащий термины и их вхождения в документы. Существуют различные технологии составления и хранения индекса, например, блочное упорядоченное индексирование (blocked sort-based indexing), сочетающая в себе возможности быстрого доступа к индексу (словарь хранится в оперативной памяти) и возможность хранения большой коллекции документов (сами вхождения терминов в документы хранятся на жестком диске).

Реализация вышеперечисленных техник лежит в основе построения практически любой информационной поисковой системы.

* 1. Особенности реализуемой системы

Как было описано в разделе 1.2, существующие решения не подходят для решения проблемы по ряду причин. Поисковые машины для сети «Интернет» ведут поиск лишь по сайтам и не приспособлены для поиска по локальным данным.

Существующие локальные системы поиска, как правило, ориентируются на заранее заданный банк документов и не имеют возможности ручного добавления документов в пространство поиска.

Библиотеки для информационного поиска не в состоянии сами по себе заниматься поиском, требуют надстроек и доработок.

Также одно из важных условия создаваемой системы — учитывать возможные искажения слов в документах, в том числе, намеренные.

Таким образом, кроме реализации приложения для информационного поиска в общем смысле, создаваемое приложение должно обладать ещё двумя возможностями — находить связи между документами для определения семантической схожести, а также при индексации учитывать возможность написания слов с искажением.

Определение семантической схожести документов необходимо, исходя из постановки задачи — поиск по сообщениям пользователей в сетях мгновенного обмена сообщениями (IM). Зачастую информация, затронутая в одном сообщении, раскрыта в другом.

1. Разработка проекта решения

2.1 Формулировка задачи

Задача создаваемого приложения — оказание помощи аналитику путём поиска по текстовым сообщениям пользователей. Включает в себя создание системы для поиска, учитывающую специфику поискового пространства — наличие просторечий, разрыв текста послания на несколько сообщений, наличие преднамеренных и непреднамеренных ошибок в написании слов.

2.2 Построение системы информационного поиска

Как было указано выше, система информационного поиска строится на двух частях: индексация и поиск.

При разработке приложения применяется инвертированный индекс, представляющий из себя набор значений «слово»-«вхождения слова в документы».

Используется усовершенствованная технология индексирования BSBI (blocked sort-based indexing), когда составляется индекс до тех пор, пока он помещается в оперативной памяти, после чего «вхождения слова в документы» пишутся на диск, а словарь остаётся в памяти. Затем индексируется новая порция документов. После чего словарь остаётся в памяти, что ускоряет поиск, а нужные вхождения считываются с диска, что позволяет хранить объемы информации, превышающие объем оперативной памяти. Таким образом, индекс хранится целиком в оперативной памяти, пока это позволяет объем памяти, после чего начинается блочная запись на диск. При запросе же нужный блок считывается с диска в оперативную память. Данный подход является компромиссом между хранением индекса в оперативной памяти и хранением его на жёстком диске.

Словарь представляет собой бинарное дерево поиска, что ускоряет поиск по нему и позволяет использовать методы поиска вроде расстояния Левенштейна. Словарь всегда хранится в оперативной памяти и, изначально, представляет собой структуру вида «слово»-«вхождения», а при записи индекса на диск видоизменяется к форме «слово»-«ссылка на блок индекса».

Перед добавлением слова в индекс необходимо произвести разбивку текста на слова, иначе говоря, токенизацию. В силу постановки задачи, процесс токенизации начинается с разбивки текста на слова по пробелам. После чего происходит анализ получившихся слов, убираются символы вроде «!», «.», «,» и прочие. Если слово содержит кириллические и латинские буквы, то происходит замена схожих букв («а» и «а») на буквы языка, букв которого в слове большинство. Таким образом, обходится одна из методик искажения слов.

После токенизации происходит морфологическая обработка слова. Для того, чтобы не включать все формы слова в индекс (существительное, глагол, различные окончания), необходимо произвести предварительную обработку слова. В первую очередь применяется стемминг — процесс усечения окончания и суффикса слова для приведения его к нормализованной словоформе. Существует несколько алгоритмов стемминга, в их числе Snowball stemmer, стеммер Портера, Light и Aggressive стеммеры, разработанные в Université de Neuchâtel. Последние показывают лучшие результаты на русскоязычных коллекциях по показателям MAP (Mean Average Percision) [2]. При построении приложения применяется Light стеммер, усекающий лишь окончания, так как на выборке из просторечий Aggressive стеммер зачастую искажает слово, написанное не по правилам русского языка.

Кроме стемминга, обработка слова включает в себя определение части речи слова, используется алгоритм, разработанный командой сайта aot.ru, позволяющее определить часть речи слова. Используется морфологический анализ при определении части речи слова из запроса, а затем слова из найденного документа. Таким образом, можно определить релевантность найденного документа запросу, располагая сначала документы, где часть речи найденного слова совпадает с частью речи слова в запросе.

После разбиения и обработки слова происходит добавление слова в индекс. Если данная словоформа уже присутствует в индексе — то добавляется вхождение слова в документ к уже существующему слову в словаре, если словоформа в индексе отсутствует — добавляется новое слово в словарь.

Архитектура индексирующей части приложения изображена на рисунке ниже.



Рис. 1. Индексирующая часть приложения

На вход приложению подаётся документ в формате XML, включающий в себя несколько сообщений, который преобразуется в набор экземпляров java-класса, являющегося абстракцией сообщения, используя SAX-парсер. Каждый документ обрабатывается токенизатором, стеммером, после чего слова заносятся в индекс, который строится и, при необходимости, записывается на диск.

Схожим образом устроена архитектура части приложения, занимающегося поиском, изображенная на рисунке ниже.



Рис. 2. Поисковая часть приложения

 Запрос, вводимый пользователем, подвергается обработке, аналогичной обработке исходного текста, что является обязательным условием при построении IR-системы.[1]

Запрос подвергается токенизации, стеммингу, производится поиск по индексу, по запросу пользователя происходит поиск схожих результатов, результаты поиска выводятся на дисплей с возможностью сохранения для последующего анализа.

2.2 Поиск ошибок в сообщениях

В интернет-переписке пользователи зачастую искажают слова, намеренно и ненамеренно. [3]

Для поиска ошибок, намеренных и ненамеренных, используются несколько техник. Во-первых, используется расстояние Дамерау-Левенштейна, которое является мерой различия двух строк и представляет из себя подсчёт операций добавления буквы, удаления буквы, замены буквы и перестановки буквы. Выводя, например, слова с расстоянием меньше двух, можно избежать большинства ошибок написания при индексации.

Следующий способ — фонетические алгоритмы. Принцип работы фонетических алгоритмов основан на кодировании сочетаний символов, одинаково произносящихся, одним символом. Используется алгоритм Mataphone, модифицированный для русского языка.

Кроме того, существуют преднамеренные ошибки, вносимые в слова, часть из которых устраняется описанными выше подходами, часть необходимо учитывать отдельно.

Разрабатываемое приложение учитывает возможности написания слова в обратном порядке, разделение букв слова символами. Пропуски и искажения букв в словах исправляются фонетическими алгоритмами.

2.3 Определение связанных сообщений

2.3.1 Временные метки

При переписке пользователи иногда ссылаются на другие сообщения через временные метки (вчера, завтра, в пятницу и т.д.). В программе реализована возможность отследить эти временные метки, по которым строится граф связи сообщений, позволяющий определить, какие именно из сообщений пользователя, не содержащих ключевых слов, несут в себе искомую информацию.

2.3.2 Латентно-семантическое индексирование

Латентно-семантический анализ представляет собой инструмент для сравнения текстов путём сравнения матриц «термин»-«документ». [4] Матрицы вхождения терминов в документы раскладываются сингулярным разложением, которые затем сравниваются. Выводятся документы с наименьшим расстоянием до найденного. Латентно-семантическое индексирование подразумевает анализ слов сообщения, находя в данном случае слова, наиболее часто встречающиеся в сообщениях с искомыми словами.

Временные метки и латентно-семантическое индексирование используются для поиска сообщений, схожих с найденным по семантическим или иным признакам.

1. Особенности реализации

3.1 Выбор среды разработки

Для реализации приложения выбран язык разработки Java, среда разработки NetBeans.

3.2 Структура приложения

В приложении 6 основных классов:

1. Message.java

Класс, представляющий собой сообщение, содержащий в себе атрибуты и методы, необходимые для обработки и хранения сообщений. Поддаётся сериализации, записывается на диск.

1. SAXPars.java

Класс представляет из себя парсер XML-файла, на выходе генерирует экземпляр класса Message.

1. MessageShow.java

Класс, отрисовывающий форму для отображения сообщения, выведенного в результате поиска.

1. SearchResult.java

Форма для отображения результатов поиска

1. LeakFinder.java

Основной класс, реализующий логику программы, в том числе составление индекса, обработку сообщений, построение связей.

1. MainForm.java

Стартовый экран приложения, принимает информацию о расположении XML-файла, из которого читаются сообщения, а также принимает поисковой запрос пользователя.

Используемая литература

1. Manning, C. & Raghavan, P. & Schütze, H. (2009). *An Introduction to Information Retrieval.* Cambridge: Cambridge University Press.
2. Dolamic, L. & Savoy J. (2009). Indexing and Searching Strategies for the Russian Language. *Journal of the American Society for Information Science and Technology,* 60(12), 2540-2547.
3. Sproat, R. (1992). *Morphology and computation*. Cambridge, MA: MIT Press.
4. Aho, A. & Ullman, J. (1994). *Foundation of Computer Science: C edition.* New York: Computer Science Press
5. Manning, C. (1999). *Foundations of Statistical Natural Language Processing.* Cambridge, MA: MIT Press.
6. Venkata, C. C. & Umamaheswaran, S. (2012) An Information Retrieval Method for Digitally Communicated Datasets to Address High Recall Requirements. *International Journal of Scientific and Research Publications, Volume 2, Issue 1.*