ПРИМЕНЕНИЕ МЕТОДОВ ИНТЕЛЛЕКТУАЛЬНОЙ ОБРАБОТКИ В ЗАДАЧАХ ОЧИСТКИ ХРАНИЛИЩА ДАННЫХ

А.Ю. Гула, А.П. Игнатенко, И.А. Перечинский
СП ООО "Эр-Джи-Дейта Украина"
e-mail: alexg@rgdata.com.ua

Введение. Хранилище данных как важнейший инструмент управления и развития бизнеса приковывает к себе все большее внимание. Развитие банковского и финансового секторов экономики требует создания и ведения больших интегрированных хранилищ, предназначенных для хранения жизненно важной для функционирования компании информации. Поэтому задачи сбора, унификации и согласования разнородных данных, корректного наполнения ими хранилища представляют значительный интерес.

Хранилище можно определить как предметно-ориентированный, интегрированный, зависимый от времени набор данных, предназначенный для поддержки принятия решений различными группами пользователей. Так как хранилище носит предметно-ориентированный характер, его организация нацелена на содержательный анализ информации, а не на автоматизацию бизнес-процессов. Это свойство определяет архитектуру построения хранилища и принципы проектирования модели данных, отличные от тех, что применяются в оперативных системах. Интегрированность означает, что, например, данные о клиентах, подразделениях и банковских продуктах, полученные из различных источников, хранятся согласованно и централизованно.

Хранилище содержит исторические данные, или зависимый от времени набор данных. Иными словами, если в оперативных источниках представлены самые последние значения (например, текущее наименование клиента или его физический адрес), то хранилище данных будет содержать в себе всю их предысторию с указанием периода, когда те или иные данные были актуальны. Хранилище данных предназначено для поддержки принятия решений, и его пользователи — это высший и средний менеджмент банка, аналитики, представители подразделений финансового анализа и маркетинга.

Наиболее уязвимым местом использования хранилища, с точки зрения анализа и последующего принятия решения, является корректность загруженных в него данных. К сожалению, при создании хранилищ данных, как правило, очень мало внимания уделяется очистке поступающей в него информации. Такая практика приводит к засорению хранилища несогласованными, ошибочными данными. Очистка данных необходима, поскольку информация разнородна и собирается из различных источников. Именно наличие множества точек сбора информации делает процесс очистки особенно актуальным.

В любой сложной системе сбора информации возникают рассогласования, сбои или ошибки ввода. Возможно, иногда есть резон смириться с ними, чем тратить деньги и время на избавление от них. Но, в общем случае, нужно стараться любым способом снизить количество ошибок до приемлемого уровня. Рассмотрим основные типы ошибок данных:

- противоречивость информации;
- пропуски в данных;
- аномальные значения;
- ошибки ввода данных.

Для решения каждой из этих проблем существуют отработанные методы. Конечно, ошибки можно править и вручную, но при больших объемах данных это становится просто невозможно. Поэтому приемлемым решением этих задач является использование интеллектуальных алгоритмов обработки данных в автономном режиме при минимальном участии человека.

Следует отметить, что «грязные» данные представляют собой очень большую проблему. Фактически они могут свести на нет все усилия по созданию хранилища данных. Причем, речь идет не о разовой операции, а о постоянной работе в этом направлении. Идеальным вариантом было бы создание шлюза, через который проходят все данные, попадающие в хранилище. Механизмы фильтрации должны стать таким же неотъемлемым атрибутом хранилищ данных как, например, OLAP.

Задача идентификации. В данной работе мы остановимся на одной из характерных проблем, возникающих при ведении хранилища. Это проблема работы с информацией, описывающей деятельность физических и юридических лиц. Первый вопрос, возникающий при заполнении хранилища новой информацией (что происходит регулярно) по некоторому субъекту деятельности – данное физическое или юридическое лицо уже присутствует в базе или является новой сущностью? Задача существенно усложняется наличием следующих факторов:

- Независимость источников – у каждой информационной системы существуют собственные схемы хранения и правила ввода данных. Кроме того, список хранимых атрибутов и полнота описания может существенно отличаться. В результате чего одни и те же сущности описаны разным набором атрибутов с различным составом и полнотой.
- Отсутствие или недостаточная проработанность процедур контроля при вводе и перегрузке данных. В итоге в хранилище происходит умножение экземпляров одного физического или юридического лица.
- Ошибки ввода – как показывает практика, всегда сопутствуют вводу данных в информационную систему.
- Наличие в хранилище данных, представление которых допустимо различным образом (формат, язык и т.п.). К примеру, фамилия физического лица может быть введена на русском, украинском или английском.

Решение этой задачи является достаточно нетривиальным и включает в себя как математические алгоритмы сравнения строк, так и создание вспомогательных структур хранилища, поддерживающих ведение эталонов и версий.

Предлагается решать задачу в рамках создания системы сопоставления записей, которая должна обеспечивать загрузку новых данных, их анализ, идентификацию и запись в хранилище. Различные аспекты создания такой системы описаны в работе [3]. Ее ядром является подсистема идентификации лиц, реализующая математические алгоритмы [4] сравнения записей рабочего набора – множества не идентифицированных записей с эталонным набором.

Методы нечеткого сравнения. Исходя из описанного принципа работы подсистемы, легко заметить, что основной вклад в эффективность ее работы вносит именно стадия интеллектуальной обработки данных. Информация, описывающая физическое или юридическое лицо, как правило, содержит следующие поля (табл. 1.)

Таблица 1.

	Физические лица	**Юридические лица**
1	Идентификационный код	Код ЕДРПОУ
2	Фамилия	Полное название
3	Имя	Короткое название
4	Отчество	-
	Дата рождения	Дата регистрации
6	Паспортные данные	Регистрационные данные

Для оценки степени идентичности двух записей проще всего использовать деревья решений [4]. Дерево решений – это структурированное множество правил, которое позволяет классифицировать данные по определенным критериям. Дерево решений является плоским графом. Каждый узел дерева соответствует некоторому варианту сравнения пары полей – совпадает, не совпадает, отсутствие данных. Прохождение по дереву тем или иным путем соответствует степени отдаленности рабочей записи от эталонной. Таким образом, возникает задача сравнения «близости» двух строк. Проиллюстрируем некоторые наиболее характерные ошибки на примере данных ФИО (табл. 2.)

Таблица 2.

	1	2	3	4	5
.	Иванов	Іванов	Ивнов	!ВАНОВ	ИВАНОВ А.П.
2	Александр	Олександр	Аликсандр Петрович	АЛЕКСАНРД	-
3	Петрович	Петровч	-	ПЕТРОВИЧ	-

Метод Q-грам. Это один из методов сравнения текстовых строк [5], который показывает хорошие результаты для таких видов данных, как фамилии или названия фирм. Суть метода заключается в том, что сравниваемые строки режутся на подстроки длины Q (Q-грамы), далее осуществляется сравнение наборов подстрок и, исходя из количества совпавших элементов, делается вывод об их степени схожести.

Критериями, в данном случае, могут служить следующие величины:

1. Количество совпавших подстрок с учетом длин сравниваемых строк;
2. Количество совпавших подстрок с учетом их позиций в сравниваемых строках;
3. Количество совпавших подстрок с учетом их позиций в сравниваемых строках и длин сравниваемых строк;
4. Количество совпавших подстрок с учетом допустимого интервала отклонения их позиций в сравниваемых строках и допустимого отклонения длин сравниваемых строк.

Метод расстояния редактирования. Расстояние Левенштейна (также дистанция Левенштейна или редактирования) в теории информации — это мера разницы двух последовательностей символов (строк) относительно минимального количества операций вставки, удаления и замены, необходимых для перевода одной строки в другую. Метод разработан в 1965 году советским математиком Владимиром Иосифовичем Левенштейном и назван его именем.

Расстояние Левенштейна активно применяется при поиске и обработке текстов:

- в поисковых системах для нахождения объектов или записей по имени
- в базах данных при поиске с неполно-заданным или неточно-заданным именем
- для коррекции ошибок при вводе текста
- для коррекции ошибок в результате автоматического распознавания отсканированного текста или речи
- в других приложениях, связанных с автоматической обработкой текстов

Расстояние редактирования равно минимальному числу элементарных операций редактирования, необходимых для преобразования одной строки в другую. Таким образом, для любой цепочки преобразований одного слова в другое можно определить ее стоимость как количество элементарных операций редактирования данного преобразования.

Реализация. Для реализации приложения интеллектуальной обработки данных была выбрана трёхуровневая архитектура. Одним из недостатков архитектуры клиент-

сервер является то, что любые изменения в реализации функциональности, вызывают изменения в клиентской части приложения, что вызывает необходимость повторного разворачивания приложения. Поэтому уровень пользователя представлен веб-клиентом, чем достигается необходимая гибкость переконфигурирования и развёртывания. Уровень хранения данных представлен реляционным хранилищем данных, реализованным в СУБД Oracle. В качестве технологии для разработки бизнес-логики приложения была выбрана переносимая объектно-ориентированная среда программирования Java.

В результате было создано приложение, схема которого представлена на рисунке 1. Приложение включает систему идентификации, реализованную на базе программного обеспечения Blaze Advisor 5.5 компании Fair Isaac, настройку и редактирование системы правил которой осуществляет специально подготовленный пользователь – эксперт.

Пользователи-аналитики работают с программным комплексом с помощью интерфейса *АРМ ведения реестра лиц*. После загрузки данных из внешних источников проводится предварительная обработка данных, что включает в себя нормализацию и очистку данных. Далее проводится идентификация лиц с применением системы правил и методов нечёткого сравнения строк, описанных выше. После этого производится загрузка очищенных данных в автоматическом либо полуавтоматическом режиме.

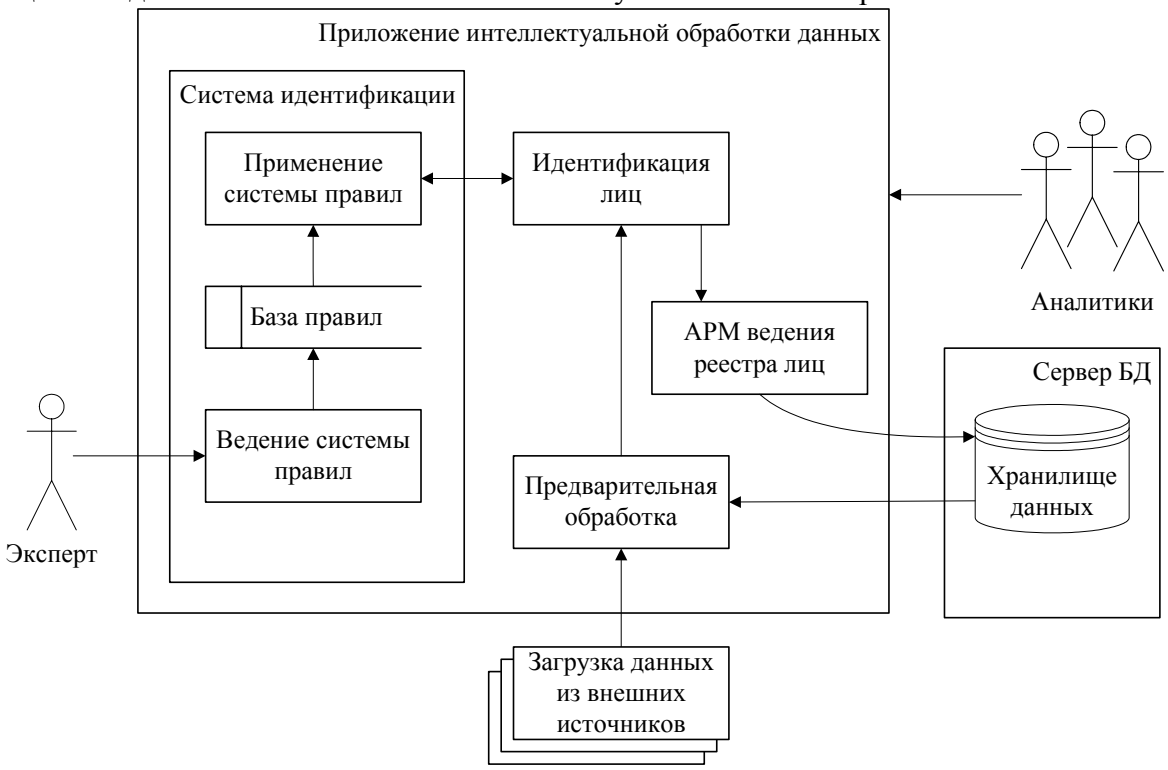

Рис. 1.Архитектура приложения

Литература

1. Э. Спирли, Корпоративные хранилища данных. Планирование, разработка и реализация. т. 1: Пер. с англ. – М.: Издательский дом «Вильямс», 2001. – 400 с.

2. И.Т. Кадощук, Е.А. Липчинский, Обзор технологии хранилищ данных. http://www.olap.ru/basic/genstore.asp

3. Д. Орлов, Подсистема сопоставления записей в хранилище данных, http://www.olap.ru/basic/CompareLog_dw.asp

4. Mueller J.-A., Lemke F. Self-Organizing Data Mining. Berlin, Dresden 1999.

5. Graham A. Stephen, String Search, Technical Report TR-92-gas-01 School of Electronic Engineering Science University College of North Wales Dean Street, Bangor, Gwynedd, UK LL57 1UT, October 1992, русский перевод:
http://www.3ka.mipt.ru/vlib/books/Programming/ComputerScience/StryngAnalysis/index.html