РАЗРАБОТКА ИНТЕРНЕТ-СЕРВИСОВ ДЛЯ ОЧИСТКИ ПЕРСОНАЛЬНЫХ ДАННЫХ

Скачков Д.М., Тамплон А.В.
Алтайский государственный технический университет им. И.И. Ползунова
(г. Барнаул)

Современные информационные технологии позволяют хранить, обрабатывать и анализировать большие объемы данных. Основная масса данных о клиентах, партнерах, поставщиках современных организаций хранятся в электронном виде, и зачастую эти данные вводятся вручную, или копируются из источников, созданных людьми.

Но все люди могут ошибаться, иметь разную подготовку и степень информированности о данных, а также иметь различные социальные и культурные особенности, которые в результате приводят к тому, что большое количество данных имеет низкое качество, или вообще непригодны для дальнейшего использования. В процессе ручного ввода данных возможно появление опечаток, неправдоподобности данных, отсутствия единообразия, внутренних противоречий, что может обернуться для организации большими потерями.

Например, в письме будет неправильно указан адрес, в результате чего оно не дойдет до адресата. Либо будут перепутаны фамилия и отчество клиента, указан неправильный пол.

Использование различных форматов значительно усложняет обработку хранящейся информации. Разные формы одних и тех же данных будут восприняты системой как данные, описывающие различные сущности, что снизит эффективность анализа информации.

Невозможно даже оценить величину возможных неприятностей, связанных с некорректными данными. Все эти причины значительно снижают отдачу инвестиций в информационные технологии и препятствуют их внедрению. Более того, низкое качество данных стоит денег в том смысле, что ведет к снижению производительности и принятию неправильных бизнес-решений.

В процессе работы были проанализированы существующие на Российском рынке решения, в результате чего выяснилось, что существующие системы являются проприетарными и комплексными, что затрудняет их интеграцию и усложняет их использование для проверки отдельных видов данных. Также не заявлено о возможности разработки сторонних модулей проверки для данных систем.

Поэтому появилась необходимость создать программный комплекс для проверки и очистки персональных данных, обладающий необходимой расширяемостью и позволяющий использовать модель SaaS (программное обеспечение как услуга) для распространения системы.

Одним из архитектурных требований к будущей системе явилась модульная структура, позволяющая легко изменять характер проверяемых системой данных. Было необходимо предоставление API для легкого написания дополнительных модулей и подключения собственных источников данных. Возможность их простого подключения и отключения, без необходимости перекомпиляции.

Взаимодействие с системой должно осуществляться посредством удаленного вызова функций Web-сервиса, что обеспечит возможность легкой интеграции системы в любое приложение практически на любом языке программирования, а также возможность использовать модель SaaS для распространения системы.

Система также должна выдавать не только одиночный результат, но и группу возможных результатов для предоставления возможности выбора оператору. Модулям должна быть предоставлена возможность проверки как одиночных элементов, так и наборов комплексных данных.

Важной частью также является механизм сбора статистики из модулей и реализации программного интерфейса для проведения обучаемости системы.

В рамках работы также необходимо было разработать модули для системы, обеспечивающие проверку Личных и Адресных данных на некорректность и неполноту, а также приведение данных к единому формату.

Так как проверка данных по словарю является одной из самых востребованных, был

разработан специальный алгоритм проверки, абстракция которого вошла в ядро системы.

Алгоритм состоит из четырёх основных этапов:

- Подготовка
- Поиск по точному совпадению
- Быстрый поиск
- Медленный поиск

Подготовка данных заключается в расшифровке пакета, пришедшего на обработку, а также предварительной обработке данных при необходимости.

Под предварительной обработкой [3] понимается приведение строки к общему регистру, удаление незначащих знаков препинания, а также транслитерация и исправление неверной раскладки при необходимости.

Поиск по точному совпадению осуществляется в словаре. Предполагается что поиск по точному совпадению производится быстро.

Быстрый поиск производится на основе обработанных данных. Например, в реализованных модулях проверки используется алгоритм фонетического кодирования MetaphoneRu [1], результаты которого вычислены заранее, за счет чего достигается высокая скорость поиска.

Медленный поиск производится, если он разрешен в конфигурации системы. Предполагается, что он занимает время значительно большее, чем все предыдущие виды поиска. Например, в реализованных модулях для медленного поиска используется вычисление редакционного расстояния Левенштейна-Дамерау [2].

Т. к. базовый алгоритм [4] нахождения редакционного расстояния Левенштейна-Дамерау является достаточно медленным, он был оптимизирован посредством отсечения незначимых результатов, а также были введены эвристики сужения выборки, что позволило уменьшить количество вычислений расстояния в 2–6 раз.

Отсечение незначимых результатов основывается на том, что можно завершить вычисление расстояния, если оно уже превысило некоторый заданный порог.

Эвристика длины основана на идее, что искать соответствие необходимо в словах, длина которых мало отличается от длины исходного слова. Поэтому можно не принимать во внимание слова, длина которых сильно отличается от длины исходного слова.

Эвристика первой буквы основана на идее, что вероятность неверного написания первой буквы в названии мала. Эвристика позволяет значительно сузить область поиска, но достаточно сильно снижает эффективность.

Для словаря улиц (~ 800 000 записей) было принято решение использовать индексное дерево расстояний редактирования на основе дерева Бёрхард-Келлера для увеличения скорости поиска.

Проверка комплексных данных (состоящих из нескольких элементов, которые могут быть проверены по отдельности) производится по принципу определения наиболее подходящего шаблона [5]:

- определяется предположительное множество шаблонов, которым могут соответствовать исходные данные;
- определяется сумма рейтингов полученных результатов для каждого шаблона;
- выбирается шаблон, с наибольшей суммой рейтингов.

Для обеспечения обучаемости системы была использована схема с обратной связью: клиент выполняет специальный вызов, в результате чего либо будет увеличен рейтинг данного элемента при последующих проверках, либо новый элемент будет добавлен в словарь.

В качестве хранилищ данных были выбраны:

СУБД PostgreSQL, как самая развитая из бесплатных СУБД, для хранения четко структурированных данных, таких как словари населенных пунктов, улиц, фамилий и т. д. Обращение к хранилищу производится посредством ORM NHibernate, что обеспечивает гибкость и не привязывает к конкретному серверу баз данных;

MongoDB для хранения неструктурированных данных выбрана по причине её бесплатности и наличию библиотеки для связи с NET Framework. Данное хранилище используется внутри системы для хранения статистических данных и словарей анализатора языка, а также используется модулем проверки адресов для хранения индексного дерева.

Для удобной загрузки данных в словари системы была разработана утилита с возможностью работы из командной строки.

Для первоначального заполнения словарей были собраны данные из Интернета, а также использован справочник КЛАДР.

Было принято решение использовать язык C# и платформу .NET Framework для разработки системы в среде Visual Studio 2008. Выбор средств обусловлен их удобством для разработки web-сервисов. Модульную архитектуру было решено построить на базе IoC контейнера Spring.NET. Использование данной технологии позволило добавлять новые модули обработки к системе посредством простого редактирования текстовых файлов, а также упростило механизм сборки статистики с помощью аспектно-ориентированного подхода.

На рисунке 1 представлена связь между компонентами системы.

Система состоит из ядра, управляющего потоком поступающих запросов и перенаправлением пакетов данных конкретным модулям проверки.

Модули проверки осуществляют проверку данных.

Библиотека алгоритмов используется модулями проверки при обработке данных.

Анализатор языка используется модулями обработки на первичных этапах для подготовки данных.

Хранилище данных служит для хранения данных, используемых на стадии проверки.

Рисунок 1 – Взаимосвязь модулей системы

Проверка данных производится по следующему сценарию:

Клиентское приложение отправляет специальным образом сформированный запрос на проверку данных. Получив запрос, ядро системы по значению специального маркера в запросе переадресует полученные данные соответствующему модулю проверки, в случае если модуль проверки для данного маркера зарегистрирован. При отсутствии соответствующего модуля система возвращает пустой результат. Модуль проверки, получив адресованные ему данные, расшифровывает пакет и выполняет предварительную обработку данных с помощью анализатора языка и модуля предварительной обработки. Затем обработчик данных выполняет этапы проверки, по мере необходимости используя алгоритмы и хранилище данных. После завершения всех этапов алгоритма обработчика, результат передается клиентскому приложению.

Следует упомянуть, что каждому элементу результата присваивается рейтинг, т. е. число, показывающее его большую или меньшую достоверность по сравнению с другим результатом проверки.

Для взаимодействия с системой используется протокол, основанный на передаче пар ключ-значение, что позволяет передавать на обработку любые строковые данные в необходимом для модуля проверки виде.

Для написания собственного подключаемого модуля достаточно реализовать интерфейс *IChecker*, добавить описание модуля в контекст приложения и добавить возможность передачи результирующего объекта посредством XML.

Существует возможность написания модулей на базе класса *BaseChecker* если необходим доступ к настройкам системы, или на базе *BaseDictionaryChecker* если предполагается использование словарной проверки (рисунок 2).

Разработанные модули осуществляют проверку личных данных (фамилия, имя, отчество) и адресных данных (населенный пункт, улица, индекс) как по отдельности, так и полностью.

В демонстрационных целях было разработано приложение, использующее систему соответственно схеме с проверкой информации перед записью в базу данных.

Для просмотра и анализа статистики было разработано приложение визуализации собранных статистических данных.

Разработаны также Unit-тесты для проверки правильности работы системы при изменении конфигурации и добавлении новой функциональности.

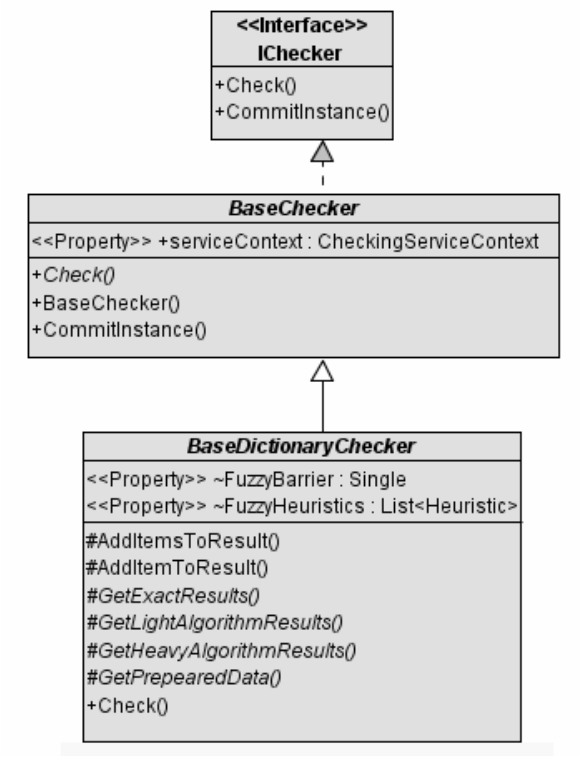

Рисунок 2 – Иерархия абстракций проверки

СПИСОК ЛИТЕРАТУРЫ

1. Каньковски, П. "Как ваша фамилия", или Русский MetaPhone [Электронный ресурс]: статья / П. Каньковски. – М., 2005. – Режим доступа: http://kankowski.narod.ru/dev/metaphoneru.htm

2. Википедия - Расстояние Левенштейна [Электронный ресурс]/Режим доступа: http://ru.wikipedia.org/wiki/Расстояние_Левенштейна

3. Карпов, В. Э. Об одной задаче очистки и синхронизации данных [Электронный ресурс]: статья / В. Э. Карпов, И. П. Карпова. – М. : Московский Государственный институт электроники и математики, 2002. – Режим доступа: http://www.raai.org/about/persons/karpov/pages/dscrubb/dscrubb.html.

4. Вычисление функции похожести [Электронный ресурс] / Режим доступа: http://itman.narod.ru/ir/ faq/fzfaq_calc.html.

5. Бегтин, И. Систематизация распознавания пола и этноса по ФИО [Электронный ресурс]: статья / И. Бегтин. – М., [2009]. – Режим доступа: http://ivan.begtin.name/2010/05/04/систематизация-распознавания-пола-и/.