РОЗРОБКА ПРОГРАМНИХ КОМПОНЕНТІВ ІНФОРМАЦІЙНОЇ СИСТЕМИ ЕКСТРАКЦІЇ ФАКТОГРАФІЧНИХ ДАНИХ З ВЕБ-РЕСУРСІВ

УДК 004.9:510.635

ДОРОШЕНКО Анна Юріївна
аспірант кафедри програмної інженерії ХНУРЕ,
Наукові інтереси: інтелектуальні системи, комп'ютерна лінгвістика, автоматизована обробка текстової інформації.
e-mail: yova.tanya@gmail.com.

ШАРОНОВА Наталія Валеріївна
професор, доктор технічних наук, завідувач кафедри інтелектуальних комп'ютерних систем
Національного технічного університету «ХПІ»,
Наукові інтереси: інтелектуальні системи, комп'ютерна лінгвістика, автоматизована
обробка текстової інформації, математичне моделювання.
e-mail: sharonovanv@gmail.com.

ЄНА Богдан Олександрович
студент кафедри програмної інженерії та інформаційних технологій управління Національного технічного університету «ХПІ»,
Наукові інтереси: програмна інженерія, видобування даних, інтелектуальні системи.
e-mail: enafortest@gmail.com.

ЯНГОЛЕНКО Ольга Василівна
кандидат технічних наук, асистент кафедри програмної інженерії
та інформаційних технологій управління Національного технічного університету «ХПІ»,
Наукові інтереси: інформаційні технології, веб-моніторинг, багатоагентні системи.
e-mail: olga.yan26@gmail.com.

ВСТУП

На сучасному етапі особливого значення набуває завдання збору інформації для підтримки процесів прийняття рішень, що передбачає надання фактографічної інформації. Фактографічна продукція стала першим комерційним продуктом, який почали пропонувати на інформаційному ринку. За останні роки значно виріс об'єм фактографічних запитів та їх розмаїття. Факт розглядається як знання у формі твердження, достовірність якого строго встановлена. На практиці, в сфері інформаційних технологій, фактографічну інформацію зазвичай трактують дещо інакше – як конкретні відомості або дані незалежно від того, чи є вони фактичними або прогнозованими. Головне, що ці відомості повідомляють про якусь предметну область, а не про документи, присвячені цій області.

Велика частина вилучення знань здійснюється в спеціальних програмних системах, розроблених з нуля. Зазвичай ці рішення включають збір, аналіз, перенесення і зберігання даних. Процес проводиться розробниками з певним досвідом програмування. Розробники повинні мати справу з двома різними проблемами: технічною складністю аналізу даного документа і розумінням семантики інформації, що міститься в цьому документі. Таким чином виникла потреба в мові, яка дозволяє відображати дані при збереженні певного макета. HTML стала такою мовою Інтернету і, отже, самим прийнятим рішенням для багатьох розробників.

Проте, HTML не надає за умовчанням ніякого механізму, який полегшує автоматичний аналіз існуючих документів. Це обмеження не дозволяє відрізнити контент від макета і семантики даних.

Ми можемо ідентифікувати трьох основних учасників в будь-якій проблемі вилучення знань. Перший – джерело даних, яке містить відповідну інформацію. Другий – база даних, призначена для зберігання даних. Третій – експерт, який може визначити, як перетворити дані з джерела в базу даних. Незалежно від рівня автоматизації, роль експерта потрібна для того, щоб додати семантику щодо даних до початку вилучення. Крім того, експерт відповідає за визначення правильності та коректності вилучення даних.

При роботі з будь-яким проектом вилучення вебзнань зазвичай ігноруються такі питання, як:

1. Кількість джерел даних. Більшість підходів враховують тільки джерела даних із аналогічним дизайном (наприклад, на сторінках продуктів Amazon, або текстів на сторінках Wikipedia). Система повинна враховувати, що джерела даних не мають узагальненого дизайну, а також дані міститься в декількох джерелах інформації.

2. Багатомовність. Зазвичай існуючі підходи розглядають тільки джерела даних, написані однією мовою. Багатомовний набір джерел даних або навіть джерел даних, що використовують одночасно кілька мов не враховується в більшості підходів.

3. Різний формат даних. HTML є найбільш поширеним форматом даних, однак можуть бути присутніми інші формати, такі як XML, DOC або PDF. Це ускладнює автоматичне вилучення даних. У деяких випадках файли архівуються в різні формати (наприклад, ZIP або RAR). Також можуть виникати складнощі із застарілими форматами даних. Крім того, кодування даних також є проблемою. Не зважаючи на те, що UTF-8 є широко використовуваним стандартом, в деяких випадках сервери використовують інше кодування.

4. Оновлення та багаторазове використання. У системному проектуванні необхідно враховувати, що дані, отримані з відповідних джерел даних, розвиваються в часі. Це означає, що можна додавати більше документів, але також процесу вилучення даних може бути змінений через структурні зміни документу. При цьому більшість операцій, які виконуються в процесі вилучення, можна повторно використовувати в доменах і джерелах даних.

Таким чином, метою дослідження є розробка програмних компонентів для екстракції фактографічних даних з веб ресурсів певного типу. Це потребує аналізу алгоритмів екстракції, сучасних програмних рішень, які реалізовані для задач видобування даних, та вибору методології і технологій вирішення проблеми.

ОГЛЯД ПРОБЛЕМИ ЕКСТРАКЦІЇ ДАНИХ

Зазвичай обробка даних, які містяться на будь-якому веб-ресурсі, передбачає певний ступінь взаємодії з людиною (заповнення форми пошуку, взаємодія зі сценарієм і т.п.). При вирішенні завдання видобування знань більшість підходів пропонує комплекс задач, пов’язаних з пошуком та визначенням необхідних даних, аналізом та зберіганням. У випадку з декількома джерелами даних складність проблеми суттєво зростає.

Кілька стандартів, таких як RDF, RDFS і OWL, були розроблені для забезпечення загального синтаксису для визначення моделей даних. Ці рішення дозволяють розробляти онтології, які підтримують запити. Ці технології зазвичай не розуміються розробниками, які спочатку ігнорують процес семантичної анотації при розробці HTML-сторінок. Щоб вирішити цю проблему, відомий підхід Schema [1], визначає словник понять, таких як люди, місця, події та продукти, які дозволяють анотувати дані, що містяться в документі HTML. Це дозволяє додати семантику в документи, які містяться в Інтернет.

Дизайн веб-сторінок також може приховувати дані з існуючих пошукових систем. Використання динамічного контенту, CAPTCHA, приватних веб-сторінок, сценаріїв або незв'язаного контенту серед інших призводить до створення Deep Web [2]. Простим прикладом є використання веб-сторінок, які виконують пошукові запити по базі даних. Інформація, що міститься в базі даних, не може бути проіндексована пошуковою системою, оскільки для цього потрібно, щоб програмний двигун взаємодіяв з формою пошуку, задавав параметри пошуку і розумів семантику даних. Комерційні пошукові системи, такі як Google, розробляють свої інструменти з чітким акцентом на індексування так званої поверхневої мережі [3].

Екстракція знань – це створення знань з структурованих або неструктурованих джерел даних. Прямим рішенням є розробка парсеру, який обробляє документи і отримує дані, які використовуються для заповнення відповідної моделі даних. Такий підхід може бути достатнім для невеликих обсягів даних з використанням відомих структур даних. Найпростішим рішенням, прийнятим багатьма проектами, є використання XQuery [4] або регулярних виразів для вилучення точного шляху до цільового елементу. Цей підхід не дуже стійкий до структурних змін шаблону документа. Іншим популярним підходом є використання розширених перетворень мови таблиць стилів (XSLT) [5], який забезпечує уніфікований синтаксис для запису правил перетворення між сумісними з XML мовами. У базовій формі HTML в основному сумісний з XML, тому цей підхід може бути застосований до HTML. Цей підхід більш стійкий, ніж XQuery до структурних змін, але його зазвичай дуже складно налагоджувати. Ще одна проста методика, дуже практична в невеликих проектах, – це спрощена версія синтаксису HTML [6].

Для більш складних рішень використовуються спеціальні доменні мови. Рішення, подібні до тих, які представлені в [7, 8], використовують мову видобування декларативної інформації для визначення потрібних для видобування даних. Аналогічним чином [9] використовує набори правил для визначення моделі видобування знань. У цих рішеннях якість екстракції особливо залежить від навичок експертів. Друге сімейство рішень досліджує використання методів машинного навчання для поліпшення видобування інформації. Огляд існуючих методів інтелектуального аналізу вебконтенту з упором на автоматизовані і машинні методи навчання розглянуті у [10].

Автоматизований ітеративний процес для створення формального опису серії документів з використанням єдиного шаблону, який може автоматично виявляти повторювані структури і створювати структури сторінки, розглянуто в [11, 12]. Підхід IEPAD [13] використовує подібну ідею, групуючи теги елемента HTML в різні категорії, що знижує складність здачі. Результатом IEPAD є дерево, яке наближається до єдиного, однорідного, шаблонного документа. Автор [14] розширює цей підхід, порівнюючи подібність дерева, отримане шляхом побудови декількох дерев елементів HTML.

Одним з популярних фреймворків, що уніфікує методи машинного навчання і засновані на словниках, є GATE [15]. Цей інструментарій надає повну структуру для анотації, створення словників для іменованих об'єктів і різних методів обробки природної мови та машинного навчання, що дуже корисно для створення контрольованих підходів в Data Mining. Також останнім часом підходи DeepDive [16] привернули велику увагу дослідницького співтовариства. DeepDive використовує набір визначених правил для встановлення відношень між об'єктами. Остаточне створення бази даних – це ітеративний цикл, в якому оператор може керувати процесом машинного навчання, ідентифікуючи помилки, допущені системою. Подібним чином Google Knowledge Vault будує відношення з використанням триплетів RDF. Цей підхід показує деяку схожість з DeepDive з явним акцентом на масштабованість даних.

Таким чином, не зважаючи на різноманіття існуючих підходів до видобування знань питання розробки уніфікованих методів, здатних до масштабування та перенесення, працюючих в режимі реального часу та з можливістю адаптації до веб ресурсів, які постійно змінюються, все ще залишаються відкритими.

ПОСТАНОВКА ЗАВДАННЯ

Сканування веб-сторінок є першим кроком в процесі збору даних. Елементи, що підлягають обробці, можуть відрізнятися в залежності від джерела даних. У спрощеному сценарії можна припустити, що даний продукт націлений на сканування веб-сторінок. Як сканувати ці веб-сторінки залежить від структури джерела даних. У деяких випадках сторінки легко доступні через одну URL-адресу або можуть бути отримані після виконання форми пошуку. Можна виділити наступні сценарії:

– Ідентифікація по загальнодоступному ідентифікатору. У цьому випадку кожен елемент, що підлягає обробці, визначається URL-адресою, яка містить унікальний ідентифікатор. Якщо генерація ідентифікаторів відома, можна статично генерувати список можливих URL-адрес для запиту.

– Ідентифікація по невідомому ідентифікатору. Як і в попередньому сценарії. Однак, як генеруються ідентифікатори, невідомо. В цьому випадку ідентифікатори повинні бути спочатку витягнуті з самого веб-сайту, а

потім використані для створення кінцевої URL-адреси, що підлягає обробці.

– Динамічні URL. Багато платформ розподіляють контент за динамічними URL-адресами. Це робить неможливим статично генерувати список адрес для вивчення. Цей сценарій передбачає початкову навігацію, яка запитує веб-платформу URL-адреси, а потім генерує унікальні URL-адреси, які можуть ідентифікувати ці елементи, щоб гарантувати їх унікальність в архівній системі.

Як було показано раніше, складно розробити загальне рішення для сканування, яке може бути використано в усіх сценаріях. Це пов'язано з тим, що на багатьох платформах знаходиться велика кількість коду Javascript у поєднанні з обміном повідомленнями AJAX.

Можна узагальнити деякі аспекти, які необхідно враховувати під час розробки алгоритмів обробки вебсторінок.

– Багато веб-порталів відображають зміст, отриманий після запиту бази даних. Це означає, що не весь існуючий зміст відображається відразу, а лише невелика частина. Щоб переглянути весь існуючий контент (або посилання, які роблять його доступним), натискаємо кнопку, поки не буде відображений результат запитів. У цій операції один потік може досліджувати кожну сторінку зі списком посилань, в той час як інші можуть переміщатися по існуючим посиланням.

– Результати, які відображаються в деяких системах, є просто результатом запиту бази даних. У деяких випадках результуючий набір розбивається на сторінки, які необхідно переключати, в інших випадках в результуючому наборі є межа, яка унеможливлює повне відображення існуючих результатів. У цих випадках може бути неможливо виконати сканування всіх існуючих результатів, оскільки система не розкриває цю інформацію.

– У багатьох випадках використання Javascriptрішень сильно ускладнює сканування цих систем. Найпростіше рішення – емулювати поведінку користувачів за допомогою двигунів Javascript. Проте, це накладає штрафні санкції на стороні сканування через надмірне використання ресурсів деяких браузерів.

– Щоб уникнути атак типу «відмова в обслуговуванні» (DDoS) багато платформ відстежують запити. Перевищення певної кількості запитів може привести до тимчасового припинення послуги для користувача. Кожна платформа відрізняється, і тільки метод пробної помилки може розкривати, які заходи використовують ці платформи.

– Багато систем використовують файли cookie для зберігання ідентифікаторів сеансу, які дозволяють серверу ідентифікувати параметри запиту, які використовуються користувачем. Однак ці файли cookie мають дату закінчення терміну дії.

– Сервери можуть виходити з ладу. Багаторазове сканування цих серверів буде просто повертати код HTTP-помилки. Однак багато систем повертають вебсторінку, що інформує про недоступність служби.

– Код типу контенту, що повертається HTTPзаголовком, особливо важливий при обході вебсторінок, які не закодовані з використанням стандартних ASCII або UTF8 систем кодування.

Враховуючи ці особливості, завданням даної роботи є аналіз та розробка моделей та програмних компонентів для екстракції даних, які надають інформацію щодо певного факту (події або об'єкту) з веб ресурсів певного типу. У якості прикладу будемо розглядати процес збору даних з веб-сторінок, які містять опис пропозицій мобільних телефонів. Така інформація розташована на сторінках торгівельних платформ, має однакову семантичну складову, але представлена різними структурами. Для успішного виконання поставленої задачі програмне рішення повинно отримувати вхідні дані для екстрактора у вигляді HTML сторінок. Додаток повинен проаналізувати та обробити ці дані для найбільш точного визначення важливості різних блоків. Далі необхідно провести екстракцію даних на основі методу обробки DOM-дерева та повернути результат, який відповідає висунутим вимогам щодо формальної моделі представлення фактографічної інформації.

ВИРІШЕННЯ ПОСТАВЛЕНОГО ЗАВДАННЯ. ОСОБЛИВОСТІ ПРОЕКТУВАННЯ

Мета даного проекту полягає в тому, щоб автоматизувати процес вилучення точних і конкретних даних, що містяться на різноманітних веб-сторінках. Приклад загальної взаємодії системи зображено на рисунку 1.

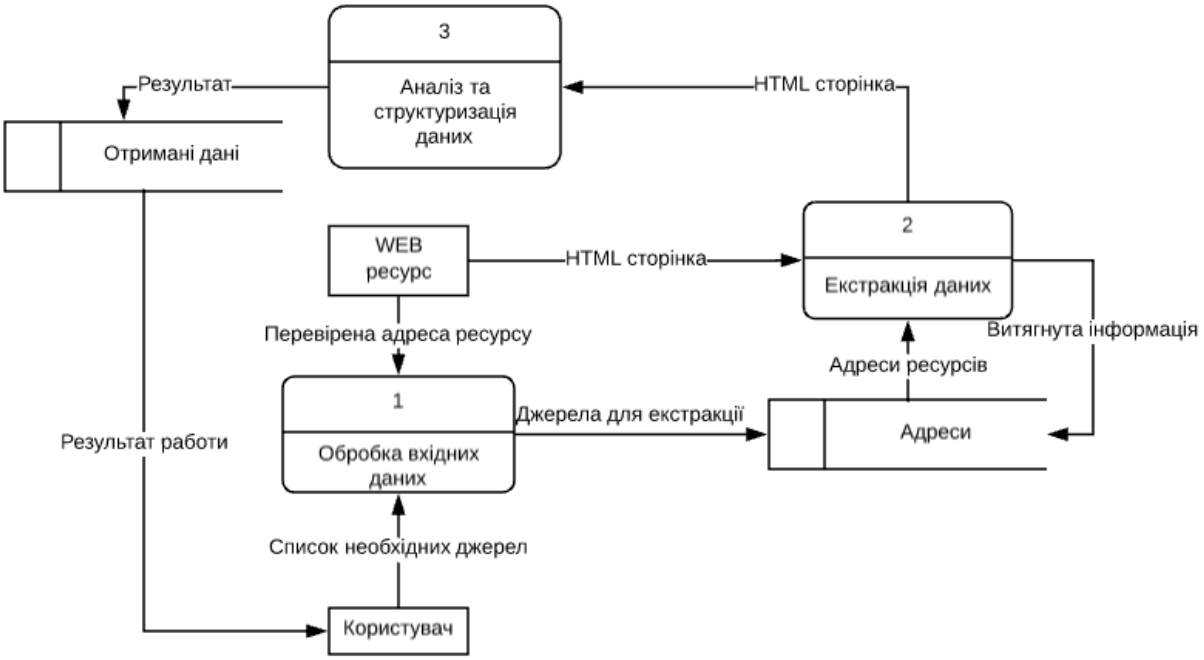

Рисунок 1 – Діаграма DFD в системі екстракції даних

Даний компонент є складним програмно-технічним комплексом. Для повноцінного його функціонування програмне забезпечення розробляється з урахуванням можливості обробки великої кількості запитів від користувачів продукту, а також його надійності, відкритості для подальших доробок, безпеки і правильно виробленого режиму його подальшого супроводу.

На рисунках 2 і 3 зображені діаграма компонентів і діаграма прецедентів відповідно.

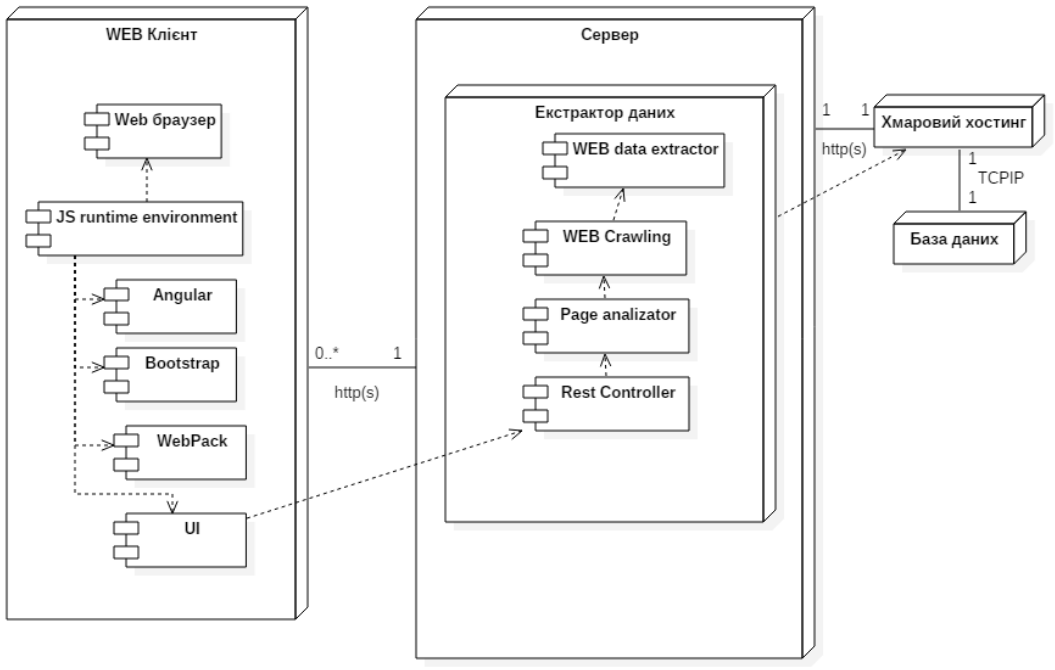

Рисунок 3.2 – Діаграма компонентів

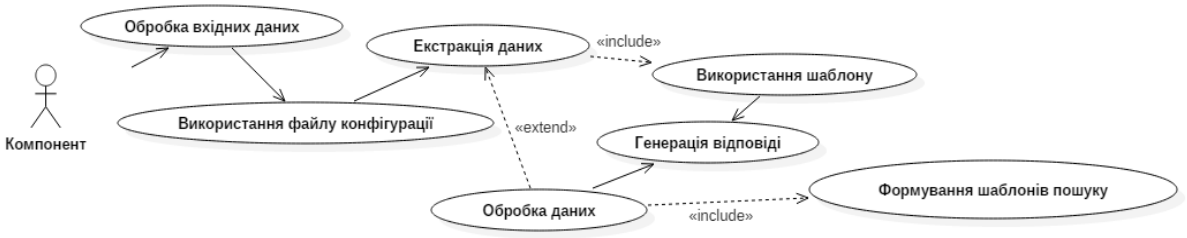

Рисунок 3 – Діаграма прецедентів

Визначимо наступні вимоги та обмеження щодо проектування програмних компонентів. Система буде отримувати адреси веб-сторінок на вході, та повертати користувачу результат роботи у тому форматі, який він заздалегідь вибере. Зв'язок між інтерфейсом та сервером буде використовувати JSON, як формат серіалізації. Компоненти системи не повинні залежати від платформи. Даний атрибут якості реалізується через використання хмарних сховищ. Завдяки цьому, не важливо на якому саме сервері буде зберігатися компонент. Якщо з'явиться необхідність перенести його з одного серверу на другий, витрати повинні бути мінімальними. Звертаючи увагу на те, що система може зростати до дуже великих розмірів, складність у конфігуруванні нових ресурсів не повинна зростати. Завдяки конфігурації екстрактора за допомогою YAML файлів, розроблюємий компонент може легко адаптуватися до екстракції з нових ресурсів без втрати великої кількості часу і сил.

Основним критерієм до вибору платформи послугувала можливість кросплатформності у використанні технології пошуку графічних елементів. Оскільки дана методологія дозволяє абстрагуватися від властивостей і технологій використаних при створенні графічного інтерфейсу. Тож серед усіх відомих мов програмування вибір падає саме на Java. Віртуальна машина Java забезпечує кросплатформність, що поширюється на усі існуючі на сьогодні найбільш популярні платформи – Windows, Linux, Mac. Це дозволяє повноцінно користуватися перевагами графічного підходу до автоматизації GUI. Окрім кросплатформності сильними сторонами мови Java є також висока надійність роботи, розвиненість мови.

РЕЗУЛЬТАТИ

Реалізацію запропонованої архітектури наведено у вигляді діаграми класів на рисунку 4. Дана діаграма презентує основні класи, які розроблено в даному проекті, такі як: парсер, екстрактор та конфігуратор. Для досягнення необхідного результату у компонент подається URL необхідного джерела, яка оброблюється у UrlUtils. Якщо джерело є коректним, то далі компонент завантажує всю HTML сторінку джерела. За дану дію відповідає клас Parser. Після отримання компонентом веб-сторінки, він починає екстрацію даних. За екстракію даних відповідають класи Extractor та Configuration. Екстрактор використовує екземпляр класу Configuration для своїх налаштунків.

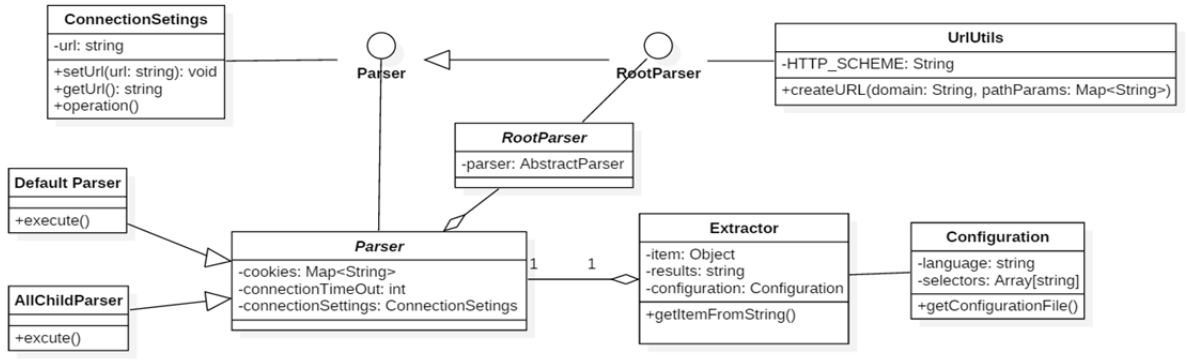

Рисунок 4 – Діаграма класів

Реалізація програмного додатку потребує тестування та валідації пропонованих рішень. Тестування – перевірка відповідності реальної поведінки програми очікуваній, що здійснюється на скінченному наборі тестів, який був обраний певним чином. У більш широкому сенсі, тестування — це одна з технік контролю якості, що включає в себе активності з планування робіт, проектування тестів, виконання тестування і аналізу отриманих результатів.

Верифікація – це процес оцінки системи або її компонентів з метою визначення чи задовольняють результати поточного етапу розробки умовам, що були сформовані на початку цього етапу. Тобто чи виконуються наші цілі, терміни, завдання по розробці проекту, визначені на початку поточної фази. Валідація – це визначення відповідності програмного забезпечення, що розроблюється, очікуванням і потребам користувача, вимогам до системи.

На першому етапі тестування необхідно провести модульне тестування усіх компонентів системи, які можуть бути протестовані окремо від інших у штучному середовищі тестування. Unit-тестування проведено за допомогою засобу автоматизації тестування Junit. Такий вибір зумовлений простотою інтеграції Junit з Java. Junit надає великий об'єм валідаційних методів, завдяки яким можна легко та ефективно тестувати як модулі обробки даних та взаємодії з базами даних, так і модулі вводу та виводу інформації.

Крім цього, враховуючи особливості реалізації трьохрівневої архітектури, необхідно провести тестування взаємодії клієнтської та серверної частин. Для цього пропонується використання сервісу Postman. Основне призначення програми Postman - створення колекцій з запитами до API. Postman дозволяє проектувати дизайн API і створювати на його основі Mock-сервер. Реалізацію сервера і клієнта можна запустити одночасно. Тестувальники можуть писати тести і виробляти автоматизоване тестування прямо з Postman. "Postman Sandbox" – це середовище виконання JavaScript, доступного при написанні "Pre-request Script" і "Tests" скриптів. "Pre-request Script" використовується для проведення необхідних операцій перед запитом, наприклад, можна зробити запит до іншої системи і використовувати результат його виконання в основному запиті. "Tests" використовується для написання тестів, перевірки результатів, і при необхідності їх збереження в змінні.

Проведене тестування запитів за допомогою тестів, реалізованих у Postman Sandbox, показало стійку роботу розроблених компонентів.

Тестування навантаженням – підвид тестування продуктивності, збір показників і визначення продуктивності і часу відгуку програмно-технічної системи або пристрою у відповідь на зовнішній запит з метою встановлення відповідності вимогам, що пред'являються до даної системи. Для дослідження часу відгуку системи на високих або пікових навантаженнях проводиться стрес-тестування, при якому створене на систему навантаження перевищує нормальні сценарії її використання.

Apache JMeter — інструмент для проведення навантажувального тестування, що розробляється Apache Software Foundation, підпроекту Jakarta. Хоча спочатку JMeter розроблявся як засіб тестування веб-застосунків, натепер він здатний проводити навантажувальні тести для JDBC-з'єднань, FTP, LDAP, SOAP, JMS, POP3, IMAP, HTTP і TCP. У програмі реалізовані механізми авторизації віртуальних користувачів, підтримуються користувацькі сеанси. Організовано логування результатів тесту і різноманітна візуалізація результатів у вигляді діаграм, таблиць тощо (рисунок 5).

У результаті проведених тестів отримані такі дані:

- середній час відгуку: 1135 мс;
- середня швидкість обробки: 1498 запита/хв;
- медіана часу відгуку: 455.

Таким чином, можна зробити висновок, що відповідно до отриманих даних розроблені компоненти відповідають висунутим вимогам.

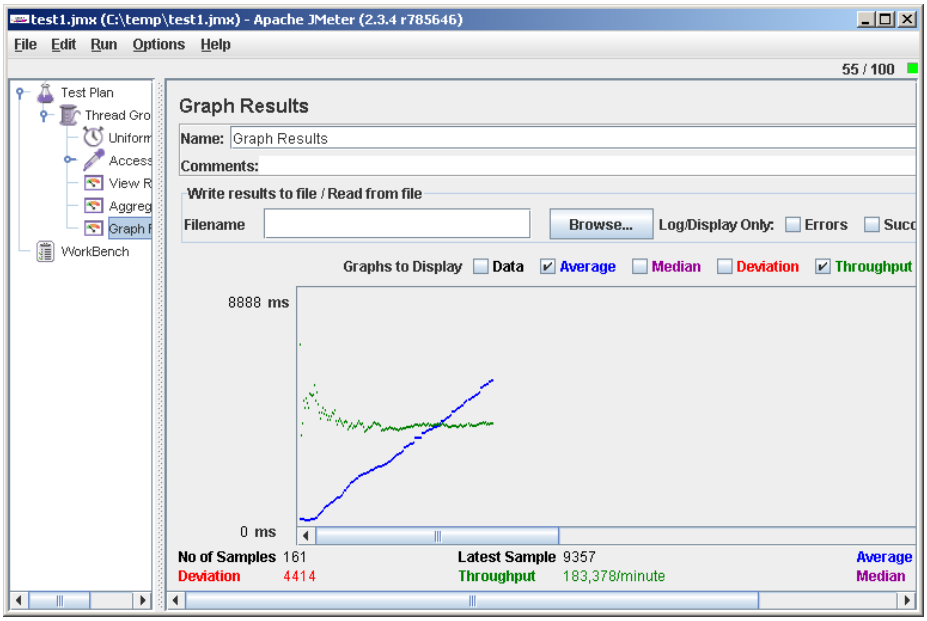

Рисунок 5 – Приклад роботи Apache Jmeter

Проведений аналіз компонента, що розроблюється з навантаженням в 100 користувачів одночасно зображений на рисунку 6.

Рисунок 6 – Результати Apache Jmeter для компонента екстракції даних

ВИСНОВКИ

В результаті дослідження проведено аналіз проблеми та особливості практичної реалізації вирішення задачі екстракції фактографічних даних. Проаналізовані підходи та інформаційні технології вирішення задач парсингу на базі існуючих інформаційних систем.

Створена специфікація вимог до програмного забезпечення. Це дозволить у подальшій роботі над проектом чітко розуміти вимоги та обмеження до реалізації. Розроблено еталонну архітектуру та запропоновано варіант розгортання програмної системи.

Розроблено програмні компоненти серверної частини програмної системи, що дозволяє проводити

екстракцію даних з торгівельних площадок на основі використання гнучкого конфігурування та предикатної моделі видобування даних. Розроблено та імплементовано програмні компоненти для збору даних на прикладі збору характеристик моделей мобільних телефонів. Проведено тестування розроблених компонентів та доведено їх працездатність для збору даних з трьох різних торгівельних площадок. Слід зазначити, що збір даних налаштовано тільки для англійської мови, але проведене тестування компонентів дозволяє стверджувати, що отримані результати можуть бути базисом створення програмної системи для видобування фактографічної інформації. Запропоновано архітектурне рішення може бути розвинене за рахунок імплементації моделі екстракції необхідної інформації. Додаткові джерела даних можуть бути додані в систему за допомогою YAML конфігурації.

ЛІТЕРАТУРА

1. R. V. Guha, Dan Brickley, and Steve Macbeth. Schema.org: Evolution of structured data on the web. Commun. ACM, 59(2):44–51, January 2008.
2. Sriram Raghavan and Hector Garcia-Molina. Crawling the hidden web. In Proceedings of the 27th International Conference on Very Large Data Bases, VLDB '01, pages 129–138, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.
3. Memex (Domain-Specific Search) // URL: www.darpa.mil/program/memex, 02.11.2017.
4. W3C XML Query (XQuery) // URL: https://www.w3.org/XML/Query/, 04.11.2017.
5. XSL Transformations (XSLT) Version 3.0 // URL: https://www.w3.org/TR/xslt, .11.2017.
6. Apache Nutch™ // URL: http://nutch.apache.org/, 18.11.2017.
7. Warren Shen, AnHai Doan, Jeffrey F. Naughton, and Raghu Ramakrishnan. Declarative information extraction using datalog with embedded extraction predicates. In Proceedings of the 33rd International Conference on Very Large Data Bases, VLDB '07, pages 1033–1044. VLDB Endowment, 2007.
8. Scrapy | A Fast and Powerful Scraping and Web Crawling Framework // URL: http://scrapy.org/, 25.11.2017.
9. Ndapandula Nakashole, Martin Theobald, and Gerhard Weikum. Scalable knowledge harvesting with high precision and high recall. In Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, WSDM '11, pages 227–236, New York, NY, USA, 2011. ACM.
10. Xin Luna Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Kevin Murphy, Shaohua Sun, and Wei Zhang. From data fusion to knowledge fusion. Proc. VLDB Endow., 7(10):881–892, June 2014.
11. Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Web-scale information extraction in knowitall: (preliminary results). In Proceedings of the 13th International Conference on World Wide Web, WWW '04, pages 100–110, New York, NY, USA, 2004. ACM.
12. Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In AAAI. AAAI Press, 2010.
13. Bing Liu and Kevin Chen-Chuan-Chang. Editorial: special issue on web content mining. Acm Sigkdd explorations newsletter, 6(2):1–4, 2004.
14. B Anantha Barathi. Structured information extraction system from web pages. MiddleEast Journal of Scientific Research, 19(6):817–820, 2014.
15. Arvind Arasu and Hector Garcia-Molina. Extracting structured data from web pages. In Proceedings of the 2003 ACM SIGMOD international conference on Management of data, pages 337–348. ACM, 2003.
16. Chia-Hui Chang and Shao-Chen Lui. Iepad: information extraction based on pattern discovery. In Proceedings of the 10th international conference on World Wide Web, pages 681–688. ACM, 2001.
17. Postman | API Development Environment // URL: www.getpostman.com, 23.03.2018.

Рецензент: д.т.н., проф. Марасанов В. В.
Херсонський національний технічний університет