# ЗМІСТ

[Вступ 7](#_Toc451063709)

[1. Задача розробки високопродуктивної довідково-інформаційної системи на основі Wikipedia 9](#_Toc451063710)

[2. Опис предметної області та огляд існуючих програмних рішень 11](#_Toc451063711)

[2.1. Онлайн-енциклопедії 11](#_Toc451063712)

[2.2. Принципи функціонування вікі-технології 14](#_Toc451063712)

[2.2. Рекомендаційні системи 20](#_Toc451063713)

[2.3. Огляд існуючих програмних рішень 25](#_Toc451063714)

[2.3.1. Система автоматизованої побудови дидактичної онтології на основі Wikipedia](#_Toc420669025) 25

[2.3.2. Система аналізу онлайн-енциклопедії для генерації дидактичних рекомендацій](#_Toc420669025) 27

[3. Засоби та методи розробки програмного продукту](#_Toc451063715) 30

[3.1. Засоби розробки](#_Toc451063716) 30

[3.1.1. Мова програмування Elixir](#_Toc420669025) 30

[3.1.2. Віртуальна машина Erlang](#_Toc420669025) 32

[3.1.3. Пошукова платформа Elasticsearch](#_Toc420669025) 37

[3.2. Модель нечіткого виведення на основі фактора впевненості](#_Toc451063717) 39

[3.3. Оптимізація процесу завантаження та обробки даних](#_Toc451063718) 43

[3.4. Формування множини зв’язаних понять](#_Toc451063719) 47

[3.5. Факти та правила дидактичного впорядкування 49](#_Toc451063720)

[3.6. Граф дидактичних зв’язків та результуюча послідовність понять 51](#_Toc451063721)

[3.7. Усунення конфліктних ситуацій](#_Toc451063722) 53

[4. Опис програмної реалізації 5](#_Toc451063723)5

[4.1. Структура сховища даних 5](#_Toc451063724)5

[4.2. Архітектура системи 5](#_Toc451063725)8

[4.3. Етапи роботи програмної системи](#_Toc451063726) 60

[5. Методика роботи користувача з програмною системою](#_Toc451063727) 62

[5.1. Інсталяція та системні вимоги](#_Toc451063728) 62

[5.2 Сценарії роботи користувача з системою](#_Toc451063729) 63

[6. Економіко-організаційний розрахунок](#_Toc451063727) 65

[6.1. Розрахунок трудомісткості розробки та впровадження програмного продукту](#_Toc451063728) 65

[6.2. Кошторис витрат на розробку та впровадження програмного продукту](#_Toc451063728) 70

[6.3. Визначення економічного ефекту від застосування програмного продукту](#_Toc451063728) 76

[6.4. Визначення ціни розробки](#_Toc451063728) 78

[6.5. Техніко-економічне обґрунтування розробки програмного продукту на основі функціонально вартісного аналізу](#_Toc451063728) 79

[Висновки](#_Toc451063730) 89

[Список використаних джерел](#_Toc451063731) 90

[Додаток 1](#_Toc451063731) 94

[Додаток 2](#_Toc451063731) 96

[Додаток 3](#_Toc451063731) 100

# ВСТУП

Онлайн-енциклопедії на основі вікі-технологій, зокрема Wikipedia, є, можливо, найпотужнішим засобом збереження знань людства з тих, що були коли-небудь створені за всю історію. Сучасні технології можуть забезпечити для них зберігання практично необмеженої кількості даних та високу надійність; їх політика вільного редагування створює цікаву альтернативу класичним енциклопедіям; а будь-хто з можливістю виходу до мережі Інтернет може отримати доступ до цієї інформації і використати її для самоосвіти та вивчення будь-якої тематики.

Wikipedia є найбільшою у світі вікі-енциклопедією, яка містить більш ніж 40 мільйонів статей на 299-ти різних мовах [1] та має 18 мільярдів відвідувань сторінок щомісяця, серед яких 500 мільйонів – унікальні відвідувачі [2].

Одним з недоліків вікі-енциклопедій, який є предметом дослідження даної роботі, є те, що структура їх статей не оптимізована для легкого для послідовного вивчення якого-небудь поняття, і користувачам інколи може бути складно зорієнтуватися у величезній кількості зв’язаної інформації, з якою необхідно ознайомитися для розуміння обраної області, особливо у незнайомій тематиці. У даній роботі роздивляється можливість впорядкування понять онлайн-енциклопедій за допомогою побудови між ними дидактичних зв’язків, що є перспективним способом вирішити вищеописану проблему. Базуючись на результатах попередніх досліджень [3], дана робота фокусується на питаннях застосування розроблених раніше підходів на великих масивах даних, індексування цих масивів даних за найменш можливий час, та загальної швидкодії системи, що розробляється.

Для реалізаціх поставлених цілей при розробці програмного продукту було використано такі технології як мова програмування Elixir та система Elasticsearch у якості сховища даних.

Короткий зміст розділів, які пропонується розглянути у даній пояснювальній записці:

* у першому розділі детально описано постановку задачі програмного забезпечення, яке розробляється;
* у другому розділі наведено опис предметної області та огляд аналогічних програмних систем;
* у третьому розділі наведено методи реалізації системи та їх обґрунтування;
* четвертий розділ є документальним супроводом програмного забезпечення. В ньому описано складові модулі розробленої системи та їх взаємодію;
* у п’ятому розділі описується методика роботи користувача з програмною системою.

# 1. ЗАДАЧА РОЗРОБКИ ВИСОКОПРОДУКТИВНОЇ ДОВІДКОВО-ІНФОРМАЦІЙНОЇ СИСТЕМИ НА ОСНОВІ WIKIPEDIA

Метою цієї роботи є створення системи, яка, отримавши від користувача назву статті онлай-енциклопедії, що описує деяке поняття, яке його цікавить, має подати у якості результату список зв’язаних з ним понять у послідовності вивчення, таким чином полегшуючи процес дослідження обраної користувачем тематики. Така функціоанльність повинна бути досягнута за допомогою побудови дидактичних зв’язків між статтями онлайн-енциклопедії на основі автоматичного аналізу їх тексту [4]. При цьому особлива увага повинна бути приділена високій продуктивності та змозі програмної системи, що розробляється, працювати с великими масивами даних. У якості демонстрації таких можливостей, необхідно за прийнятний час виконати повну індексацію англомовної частини Wikipedia як найбільшого у світі вікі-проекту, що на момент написання даної роботи має більш ніж 5,5 мільйонів статей.

Задачі, які повинна виконувати вищеописана система:

* здійснити завантаження статей з даної вікі-енциклопедії з максимально високою швидкістю та зберегти результати до локального сховища даних в оптимальному форматі;
* для обраної користувачем статті знайти зв’язані статті (тематично та через посилання між ними);
* аналізуючи текст вибраних статей, побудувати дидактичні зв’язки між ними на основі автоматичного збору фактів, що свідчать на користь їх наявності та напрямку;
* оцінити фактор впевненості у тому чи іншому зв’язку між статтями;
* здійснити побудову зв’язків максимально швидко;
* представити результати аналізу у зручній формі для кінцевого користувача;

Таку систему можна розділити на три компоненти:

* підсистема, що виконує завантаження даних з онлайн-енциклопедії, зберігаючи результуючу інформацію до локального сховища даних;
* модуль формування вибірки статей та побудови дидактичних зв’язків між ними.
* інтерфейс системи, що слугує для форматування та відображення результатів.

Вхідними даними для інтерфейсу системи є назва статті, яка цікавить користувача. У відповідь на запит користувачеві пропонується список зв’язаних статей, які система радить йому дослідити перед вхідною, та список тих статей, з якими йому слід ознайомитися після вхідної.

Потенційними користувачами запропонованої системи є люди, що зацікавленні у використанні онлайн-енциклопедій у цілях самоосвіти або в ефективному отриманні необхідної їм інформації стосовно тої чи іншої тематики.

# 2. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

Далі наведено опис онлайн-енциклопедій, базові принципи функціонування вікі-технології та розглянуто поняття рекомендаційних систем, які є спорідненою тематикою. У кінці розділу наведено огляд існуючих програмних рішень.

## 2.1. Онлайн-енциклопедії

Електронні або онлайн-енциклопедії — енциклопедії, які існують у мережі Інтернет. Стали популярними довідковими ресурсами в умовах швидкого збільшення обсягу інформації. Віртуальна форма подання енциклопедичної інформації відкрила нові перспективи в розвитку видань даного жанру, дозволила зміну формату (пошук по запиту без гортати сторінки, пошук за зображеннями, більший обсяг тексту, посилання) та збільшила доступність для широкого загалу, як в частині вільного доступу до інформації, так і в частині участі загалу у формуванні контенту. В електронну енциклопедію, як і в традиційну, входять: статті-огляди, статті-довідки, статті-тлумачення і статті-відсилання (адресують до іншого терміну). Перші два типи статей відрізняються лише об'ємом, до їх складу входить інформація по суті питання і відомості про шляхи отримання більш глибоких і повних знань (за посиланнями на літературу в кінці статті або в спеціальному бібліографічному блоці енциклопедії).

Онлайн-енциклопедії диференціюються наступним чином:

* за структурою: алфавітні, систематичні;
* за змістом інформації: універсальні (з усіх предметів), предметні (загально-предметні, спеціалізовані або тематичні);
* за формами представлення інформації: електронні копії традиційних енциклопедичних видань, мультимедійні енциклопедичні видання;
* за виконанням: як різновид локальних інформаційних ресурсів (на носіях інформації), як різновид ресурсів мережі Інтернет;
* за ступенем активності віртуального середовища: пасивні (орієнтовані тільки на пред'явлення інформації); інтерактивні (спираються на використання “зворотного зв’язку”).

Віртуальне середовище дозволяє принципово змінити подання інформації. До способів її подання в електронній енциклопедії відносяться:

* гіпертекст як засіб нелінійної архітектури викладу навчального матеріалу;
* анімації — динамічні малюнки, графіки, таблиці, схеми, діаграми;
* демонстраційні і маніпулятивні динамічні моделі об'єктів і процесів;
* аудіоінформація;
* відеосюжети;
* елементи «віртуальної реальності».

При цьому у віртуальному енциклопедичному середовищі збережені і традиційні статичні способи представлення предмета навчання (тексти, символіка, малюнки, фотознімки тощо).

Спосіб подачі інформації може задаватися самим користувачем, ним же за рахунок просування по гіпертекстовому інформаційному “дереву” варіюється її повнота і складність. У віртуальній енциклопедії може бути організований самостійний вибір виду та методу роботи з інформацією:

* пасивне сприйняття інформації,
* цільовий пошук і обробка необхідної інформації;
* редагування відібраних блоків інформації у формі тематичних презентацій, тощо.

Велике значення в енциклопедії надається його бібліографічному оснащенню. В електронній енциклопедії бібліографічні посилання можуть включати інтернет-адреси відповідних електронних видань і бібліотек.

Значне місце займають ілюстрації: фотознімки, малюнки, карти, плани, схеми, креслення. Вони можуть носити анімаційний характер і включати елементи інтерактивності і гіперграфіки.

Принципово важливим для електронної енциклопедії є апарат орієнтування. Це можуть бути традиційні способи орієнтування користувача в енциклопедичному матеріалі: тематичний план (покажчик розподіл статей за типами і видами), словник (повний перелік термінів, яким присвячені статті). При цьому ясно, що можливості ЕОМ для відображення структури інформації в енциклопедії, системи зв'язків між її основними елементами непорівнянні з можливостями традиційної книги. Електронна енциклопедія може включати різні види «навігаторів». У неї може бути закладена ціла система покажчиків (алфавітних, систематичних, бібліографічних), яка може бути доповнена пошуковою системою на конкретний термін, на ключове слово та ін.

Особливістю навчальної електронної енциклопедії є наявність у ній яскраво вираженого апарату засвоєння навчального матеріалу. Даний апарат представлений різноманітними способами систематизації навчальної інформації і особливо способами візуального відображення структури наукового знання (схемами, таблицями, діаграмами, класифікаціями, опорними сигналами та ін.).

Перша спроба створити якщо не інтернет-енциклопедію, то хоча б мережеве співтовариство по створенню енциклопедії відноситься до 1991 року, коли учасники групи новин Usenet “alt.fan.douglas-adams” почали проект зі створення реальної версії Путівника для подорожуючих автостопом по Галактиці, вигаданої енциклопедії, використовуваної в роботах Дугласа Адамса [5]. Він став відомий як Project Guide Galactic. Незважаючи на те, що спочатку метою його було утримувати тільки реальні, фактичні статті, політика була змінена, щоб дозволити і заохочувати також напів-реальні і нереальні статті. Project Galactic Guide містить понад 1700 статей, однак з 2000 року додавання нових статей припинилося; ймовірно, частково через підстави h2g2, більш офіційного проекту серед подібних ліній.

До 1993 року відноситься перша спроба побудувати інтернет-енциклопедію, яка дозволяла будь-кому писати статті і додавати їх в центральний каталог всіх сторінок. Даний проект, що отримав назву Interpedia, так в кінцевому рахунку і не був реалізований [6].

У січні 1995 року в рамках проекту “Гутенберг” почалася робота по публікації ASCII-тексту 11 видання Енциклопедії Британіка 1911, але спір, пов’язаний з методами перервав роботу над першим томом. Через товарні знаки ця енциклопедія була опублікована як Gutenberg Encyclopedia. У 2002 році текст всіх 28 томів був опублікований інше джерело на сторінці під назвою “Classic encyclopedia”.

У січні 2001 року з’явилася Вікіпедія, заснов ана технології вікі, що з’явилася ще в 1995 році [7]. Ця технологія змогла представити методи, які добре підходять для створення проектів енциклопедичного характеру, а також надала інструментарій, що сприяло швидкому розвитку енциклопедичних проектів. Вікіпедія є повністю відкритою мережевою енциклопедією, розробкою і наповненням якої займаються самі відвідувачі [8]. Уже в 2003 році почався успішний розвиток паралельних проектів фонду Вікімедія. З’явилися Вікіцитати, Віківиди, Вікісховище, Вікітека, ВікіНовини і інші “Вікі” [9].

Завдяки успіху і бурхливому розвитку Вікіпедії в 2000-ні роки почався бурхливий розвиток подібних мережевих інтернет-енциклопедій. Багато з них створювалися учасниками Вікіпедії, які мають альтернативну точку зору на те, якою має бути вільна інтернет-енциклопедія [10]. В даний час на технології вікі побудовано більшість інтернет-енциклопедій, як “загальних”, так і вузькоспеціалізованих, що описують будь-які конкретні області. З них, мабуть, тільки китайська Енциклопедія Байду може змагатися з Вікіпедією за розміром і відвідуваності.

## 2.1. Принципи функціонування вікі-технології

Вікі (англ. Wiki) — веб-сайт, структуру і вміст якого користувачі можуть самостійно змінювати за допомогою інструментів, що надаються самим сайтом. Форматування тексту і вставка різних об'єктів в текст проводиться з використанням вікі-розмітки. Термін “вікі” також стосується спільного програмного забезпечення (collaborative software), яке створюється для створення такого сайту. На базі цієї технології побудована Вікіпедія та інші проекти Фонду Вікімедіа [11].

Першим вікі став Portland Pattern Repository [12], створений Уордом Канінгемом (Ward Cunningham) у 1995 році. Канінгем вигадав і назву “вікі” [13], і концепцію — і сам створив перше втілення системи вікі. Дехто дотримується думки, що тільки перший вікі слід називати Вікі (з великої літери) або ж ВікіВікіВеб (WikiWikiWeb). Вікі Уорда Канінгема залишається одним з найпопулярніших вікі-сайтів. Канінгем узяв термін вікі від “вікі вікі”, тобто “хутких” автобусиків, що функціонували в аеропорту Гонолулу. Вікі вікі — це було перше гавайське слово, яке Канінгем вивчив під час першого відвідання островів, коли працівник аеропорту сказав йому їхати з одного терміналу в інший автобусом вікі-вікі. За словами самого Канінгема, “Я обрав термін ‘вікі-вікі’, щоб замінити надто хутку думку назвати цю штуку хуткою мережею”.

Наприкінці 20 століття вікі-сайти все більше визнаються як перспективний шлях розвитку приватних і публічних баз знань, і саме цей потенціал надихнув засновників енциклопедичного проекту Nupedia — Джимбо Вейлза (Jimbo Wales) та Ларрі Сенгера (Larry Sanger) — до використання технології вікі як основи електронної енциклопедії: так, у січні 2001 року було створено енциклопедію 21 століття — “Wikipedia”. Спочатку вона працювала на базі програмного забезпечення UseMod, але потім перейшла на власну відкриту базу кодів MediaWiki, яку тепер перебрали багато інших вікі.

Сьогодні Вікіпедія, а також окремо її англійська частина — найбільший вікі-портал у світі. Друге місце займає німецька версія. А от четвертим завбільшки свого часу була шведська вікі Susning.nu (закрита в 2009 році), що використовує програмне забезпечення UseMod. Всеохопна природа Вікіпедії стала значущим чинником її росту, тоді як багато інших вікі є високоспеціалізованими. Дехто пов’язує швидке зростання Вікіпедії з рішенням не використовувати CamelCase у формуванні посилань на сторінки. У будь-якому разі, те, що вона є найбільшим вікі, призвела до того, що часом на неї посилаються як на Материнську вікі на менших вікі, спеціалізованих за темами.

Уорд Каннінгем і його співавтор Бо Леуф (Bo Leuf) у своїй книзі [14] описали сутність концепції вікі наступним чином:

* вікі пропонує всім користувачам редагувати будь-яку сторінку або створювати нові сторінки на вікі-сайті, використовуючи звичайний веб-браузер без будь-яких його розширень;
* вікі підтримує зв’язки між різними сторінками за рахунок майже інтуїтивно зрозумілого створення посилань на інші сторінки і відображення того, існують дані сторінки чи ні;
* вікі не є ретельно виготовленим сайтом для випадкових відвідувачів. Навпаки, вікі прагне залучити відвідувачів до безперервного процесу створення і співробітництва, який постійно змінює вигляд сайту.

Вікі характеризується такими ознаками:

* можливість багаторазово правити текст за допомогою самого вікі-середовища (сайту), без застосування спеціальних пристосувань на стороні редактора;
* особлива мова розмітки — так звана вікі-розмітка, яка дозволяє легко і швидко розмічати в тексті структурні елементи і гіперпосилання, форматувати і оформляти окремі елементи;
* облік змін (версій) сторінок: можливість порівняння редакцій та відновлення більш ранніх;
* прояв змін відразу після їх внесення;
* розділення вмісту на іменовані сторінки;
* гіпертекстовість: зв’язок сторінок і підрозділів сайту через контекстні гіперпосилання;
* безліч авторів. Деякі вікі можуть правити всі відвідувачі сайту.

Для створення вікі-середовища необхідно особливе ПЗ — програмний рушій вікі. Це приватний вид систем управління сайтом, досить простий в своєму пристрої та функціональності, оскільки майже всі дії щодо структурування та обробки вмісту робляться користувачами вручну.

Робота Вікіпедії та інших сайтів Фонду Вікімедіа заснована на програмному забезпеченні MediaWiki (рисунок 2.1).

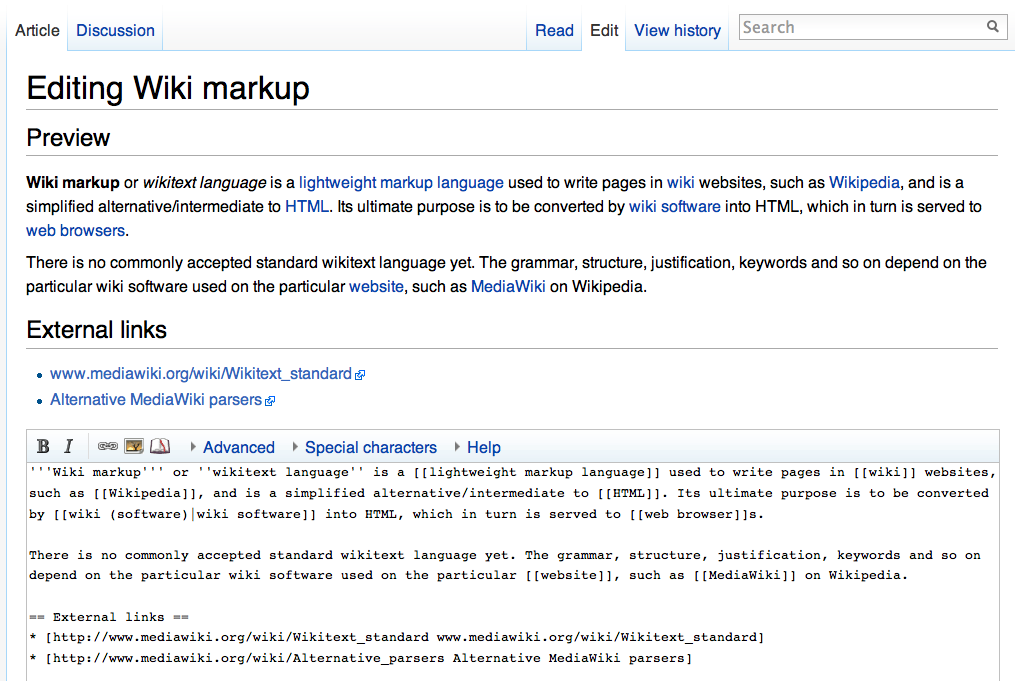


Рисунок 2.1 — Редагування вікі-тексту в MediaWiki

Фірма Kaltura спільно з фондом Вікімедіа розробляє розширення, яке дозволить користувачам правити мультимедійні дані так само легко як зараз правлять тексти. Цю технологію називають “відеовікі”.

Мова вікі підтримує гіперпосилання для створення посилань між вікі-сторінками і є більш наочним, ніж HTML, і більш безпечним, оскільки використання JavaScript і каскадних таблиць стилів обмежено.

Деякі вікі дозволяють змінювати їх вміст всім бажаючим, а не тільки зареєстрованим користувачам. Подібно до того, як стіни будівель і паркани обписують непристойними написами і прикрашають малюнками графіті, в таких вікі іноді псують вміст або додають щось недоречне [15]. Але, на відміну від стін і парканів, в вікі легко повернути вміст до ранньої версії: виправляти легше, ніж псувати. Якщо ж хто-небудь наполегливо і навмисно прагне нашкодити користувачам вікі-сайту, можна закрити йому можливість вносити правки.

Більшість публічних вікі уникають обов’язкових реєстраційних процедур. Однак багато найбільших вікі-систем (включаючи MediaWiki, MoinMoin, UseModWiki та TWiki) мають певні методи з обмеження доступу до написання тексту. Деякі вікі-системи дають можливість забороняти редагування певним індивідуальним користувачам, чого можна досягти, блокуючи конкретні IP-адреси або імена користувачів, якщо вони відомі. Проте, це може викликати певні технічні проблеми.

Загальним способом захисту від настирливих “вандалів” є просто дозволити їм зіпсувати стільки сторінок, скільки ті бажають, знаючи, що ці сторінки легко відстежити та змінити назад після того, як вандал піде. Однак ця стратегія швидко може стати непрактичною, оскільки злість або почуття власної неповноцінності таких осіб можуть змусити їх систематично псувати чужі статті.

У разі надзвичайних ситуацій, деякі вікі дозволяють перемикати бази даних в режим, коли вони доступні тільки для читання. Інші застосовують політику, що дозволяє продовжувати редагування тільки давнім користувачам, які зареєструвалися до якоїсь довільно обраної дати. Однак загалом кажучи, будь-яку шкоду, завдану вандалом, можна швидко та легко виправити. Більш проблематичними є непомітні помилки, які вставляють у сторінки. Приклад: зміни в датах випусків альбомів співаків, їхньої дискографії.

У крайніх випадках, багато вікі дають змогу захистити певні сторінки від редагування. Наприклад, захищені сторінки Вікіпедії можуть редагувати тільки адміністратори, які можуть також знімати такий захист. Але зазвичай вважають, що така практика суперечить основній філософії вікі, а тому її, як правило, уникають. Наприклад, англійська Вікіпедія водночас має, щонайбільш, кілька десятків захищених сторінок — із понад п’яти мільйонів.

В останні роки все частіше піднімається питання про застосування нових інформаційних технологій в освітній діяльності [16, 17]. Її модернізація передбачає заміну використання традиційних засобів передачі та відтворення інформації на більш сучасні методи, які обов’язково повинні бути орієнтовані на веб-технології.

Сьогодні вікі-проекти можна використовувати в навчально-виховному процесі [18], що дає змогу вирішувати наступні питання :

* створювати єдину платформу для надання енциклопедичних відомостей з певної галузі знань;
* активізувати використання та створення освітніх веб-ресурсів;
* організовувати індивідуальну або групову роботу;
* глибше вивчити потрібну галузь знань;
* скоротити час навчання і підвищити рівень підготовки;
* підвищити ефективність навчання.

Технології вікі дуже зручні для студентів і викладачів ще й тому що, вивчивши нескладну мову вікі-розмітки, викладачі можуть:

* розміщувати у відкритому доступі навчальні матеріали;
* організовувати обговорення з різних питань;
* залучати учнів до самостійної роботи з розробки нових інтернет–ресурсів.

Вікі-сайти створюють дуже зручні можливості для спільної розробки проектів, що припускають створення електронних матеріалів, їх розміщення і обговорення в мережі Інтернет.

Вікі-мережі мають ряд переваг над іншими веб-ресурсами. Можливість багатократно правити текст дозволяє залучати різних людей до створення статей та дає можливість постійно оновлювати і вдосконалювати їх. При цьому корисною буде історія обліку змін, що були внесені до змісту сторінки, та можливість повернутися до попередньої версії. Сторінка обговорень до кожної статті, де відвідувач може залишити свої коментарі, допоможе врахувати думки інших користувачів щодо представленої теми.

Цікавим є досвід професора Інституту технологій Рочестеру Елізабет Лейн Лоулі, яка використала технологію вікі в одній зі своїх університетських груп для формування бази питань проміжного іспиту. Оцінка студента на 10% залежала від його активності у підготовці питань, необхідно було придумати питання не менше ніж на 10 балів і відкоригувати питання інших студентів. В цілому якість питань, запропонованих студентами, не була відмінною, але це дало змогу отримати достатню кількість питань, щоб створити повний іспит [19]. У результаті було розроблено 60 запитань, викладач скоротив свої трудовитрати на розробку тесту, студенти взяли активну участь у їх підготовці, спільно працювали над навчальними матеріалами.

У всесвітній мережі Інтернет існує велика кількість вікі-ресурсів, що містять навчальну інформацію та дають змогу педагогам вирішувати найрізноманітніші освітні завдання.

## 2.2. Рекомендаційні системи

Рекомендаційні системи — програми, які намагаються передбачити, які об’єкти (фільми, музика, книги, новини, веб-сайти) будуть цікаві користувачеві, маючи певну інформацію про його профіль. Найчастіше реалізуються на алгоритмі колаборативної фільтрації.

Рекомендаційні системи з’явилися в інтернеті досить давно, близько 20 років тому. Однак справжній підйом в цій області стався приблизно 5-10 років тому, коли відбулося змагання Netflix Prize. Компанія Netflix тоді давала в прокат не цифрові копії, а розсилала VHS-касети і DVD. Для них було дуже важливо підвищити якість рекомендацій. Чим краще Netflix рекомендує своїм користувачам фільми, тим більше фільмів вони беруть в прокат. Відповідно, зростає і прибуток компанії. У 2006 році вони запустили змагання Netflix Prize. Вони виклали у відкритий доступ зібрані дані: близько 100 мільйонів оцінок за п’ятибальною шкалою з зазначенням ID користувачів, які їх виставили. Учасники змагання повинні були якомога краще передбачати, яку оцінку поставить певному фільму той чи інший користувач. Якість передбачення вимірювалося за допомогою метрики RMSE (середньоквадратичне відхилення). У Netflix вже був алгоритм, який передбачав оцінки з якістю 0.9514 за метрикою RMSE. Завдання було поліпшити прогноз хоча б на 10% — до 0.8563. Переможцю був обіцяний приз в $ 1 000 000. Змагання тривало приблизно три роки. За перший рік якість поліпшили на 7%, далі все трохи сповільнилося. Але в кінці дві команди з різницею в 20 хвилин надіслали свої рішення, кожне з яких проходило поріг в 10%, якість у них була однакова з точністю до четвертого знака. У задачі, над якою безліч команд билося три роки, все вирішили якихось двадцять хвилин. Команда, що запізнилася (як і багато інших, які брали участь в конкурсі) залишилася ні з чим, однак сам конкурс дуже сильно прискорив розвиток в цій галузі.

Рекомендаційні системи можуть використовуватись для рекомендацій практично будь-яких сутностей: новини, фільми, книги, статті, готелі, веб-сторінки, музика, готелі, ресторани. Можна рекомендувати нові (фільми, книги, готелі) або раніше оцінені (музика, продукти харчування).

При створенні рекомендацій може враховуватись думка всіх користувачів, експертів, близьких за інтересами користувачів. Рекомендації можуть бути однакові для всіх користувачів, поділені по групах (вік, стать, проживання, тощо) та персоналізовані. Оцінка користувачем продукту може бути пряма (рейтинг, відгук, лайк) чи непряма (купив, переглянув, клікнув).

Рекомендаційні сервіси збирають різну інформацію про людину, використовуючи кілька методів, за якими і поділяють всі системи. Отже, перший тип - явний збір даних. Як можна було здогадатися з назви, користувач сам надає необхідні для роботи матеріали. Наприклад, коли рекомендаційні системи Яндекс або інших пошукачів просять людини дати оцінки різних елементів, скласти список фаворитів певної сфери або ж відповісти на кілька запитань.

Основні алгоритми:

* Неперсоналізована система: загальна статистика (найкраще продається, найбільш бажане, популярне, разом з Х купують Y)
* Фільтрація по контенту (Content Based Filtering) [20]: користувачі та продукти представляються вектором фіч (розмірність простору = кількості всіх фіч). Обчислюємо кут між векторами - чим менший кут - тим краще продукт підходить користувачу.
* Колаборативна фільтрація (Collaborative filtration) [21]. Містить в собі надлишкову інформацію про смаки кожного користувача та категорії кожного продукту. Зменшення розмірності матриці рейтингів, дозволяє виявити ці смаки. Для зменшення розмірності використовують SVD або його поєданння з методом спадного градієнту FuncSVD або SVD++.

Колаборативна фільтрація, у свою чергу, поділяється на наступні типи:

* Модель "користувач - користувач" (User-User). Обчислюємо подібність між користувачами використовючи коефіцієнт кореляції. Для рекомендацій конкретному користувачу вибираємо його "найближчих сусідів" (достатньо вибрати 30-50) та використвуємо їхні оцінки (хороший підхід: зважене середнє, де рейтинг "ближчих" сусідів має бульшу вагу).
* Модель "продукт - продукт" (Item-Item). Подібність між продуктами змінюється не так часто, як між користувачами, тому можна порахувати подібність між продуктами і при покупці одного продукту, рекомендувати подібні на нього.
* Зменшення розмірності (Dimensionality Reduction) . Матриця Користувач - Продукт - Рейтинг:

При фільтрації по контенту створюються профілі користувачів і об’єктів.

* профілі користувачів можуть включати демографічну інформацію або відповіді на певний набір питань;
* профілі об’єктів можуть включати назви жанрів, імена акторів, імена виконавців і т.п. — в залежності від типу об’єкта.

Цей підхід використовується в проекті Music Genome Project: музичний аналітик оцінює кожну композицію по сотням різних музичних характеристик,

які можуть використовуватися для виявлення музичних уподобань користувача.

При колаборативній фільтрації використовується інформація про поведінку користувачів в минулому, наприклад, інформація про покупки або оцінки. В цьому випадку не має значення, з якими типами об’єктів ведеться робота, але при цьому можуть враховуватися неявні характеристики, які складно було б врахувати при створенні профілю. Основна проблема цього типу рекомендаційних систем — “холодний старт”: відсутність даних про користувачів або об'єктів, які нещодавно з’явилися в системі.

Наведемо приклад, щоб інформація сприймалася простіше. Створюється сайт, де аудиторії будуть рекомендуватися музичні твори. Як в даному випадку будуть працювати сервіси рекомендації на основі коллаборатівной методики? За таким принципом: за основу візьмуть одну спільноту, де учасники додають в плей-лист однакові по жанрової приналежності треки. Далі, визначаються найпопулярніші з усіх музичних творів і рекомендуються одному користувачеві з групи, який поки ще не слухав цю мелодію.

Виділяють також змішані підходи, відповідно до яких здійснюється розробка рекомендаційної системи. Змішаний підхід – це поєднання коллаборатівной і контентної фільтрації. Як відомо, більше – краще, тому змішання цих двох методик збільшують ефективність систем рекомендації, а саме значно підвищують точність прогнозів для конкретних людей.

Нижче ми розглянемо алгоритми рекомендаційних систем, які використовуються для отримання коректних результатів.

Кореляція Пірсона: цей алгоритм дозволяє виділити загальні характеристики між декількома користувачами. Яким чином? За допомогою простої математики, а саме визначенням лінійної залежності між двома елементами. Важливий момент – така методика не підходить для спільноти людей.

Кластеризація: цей принцип роботи рекомендаційних систем ґрунтується на виділенні подібності між елементами (користувачами) шляхом обчислення їх близькості один одному в так званому просторі ознак. Ознаками виступають ті елементи, за якими сходяться інтереси певних учасників процесу (для музичних ресурсів це треки, для кіно-порталів – фільми). Схожі по характеристиках користувачі об’єднуються в так звані кластери.

Алгоритм спільної фільтрації: жорстку кластеризацию можна замінити і іншим алгоритмом, який працює за досить складною формулою, і також як і всі попередні, ґрунтується на поведінці користувачів з його групи. Однак в цій методиці є кілька досить суттєвих мінусів. По-перше, новим або нетиповим користувачам (що не об’єднуються в групи) складно знайти рекомендації. По-друге, так званий “холодний старт”, коли нові об’єкти не потрапляють в рекомендаційні системи.

Алгоритм фільтрації вмісту. Алгоритм, симетричний до попереднього, але якщо в першому випадку ми відштовхувалися від припущення, що об’єкт сподобається користувачеві, тому що він подобається його “одногрупникам”, то тут ми будемо рекомендувати на основі схожих об’єктів, які він вже зазначив для себе. І тут уже традиційно можна виділити кілька проблем. Той же “холодний старт” і те, що рекомендації часто є повсякденними.

В процесі роботи рекомендаційні системи збирають дані про користувачів, використовуючи поєднання явних і неявних методів.

Приклади явного збору даних:

* запит у користувача оцінки об’єкта за диференційованою шкалою;
* запит у користувача ранжування групи об’єктів від найкращого до найгіршого;
* пред’явлення користувачеві двох об’єктів з питанням про те, який з них краще;
* пропозиція власноруч створити список об’єктів, які подобаються користувачеві.

Приклади неявного збору даних:

* спостереження за тим, що оглядає користувач в інтернет-магазинах або базах даних іншого типу;
* ведення записів про поведінку користувача онлайн;
* відстеження вмісту комп’ютера користувача.

Рекомендаційні системи порівнюють однотипні дані від різних людей і обчислюють список рекомендацій для конкретного користувача. Деякі приклади їх комерційного та некомерційного використання наведені в статті. Для обчислення рекомендацій використовується граф інтересів [22]. Рекомендаційні системи — зручна альтернатива пошуковим алгоритмам, так як дозволяють виявити об’єкти, які не можуть бути знайдені останніми. Цікаво, що рекомендаційні системи часто використовують пошукові машини для індексації незвичайних даних.

Конкретна рекомендаційна система не обов’язково використовує один алгоритм – можна застосовувати поєдання або переключення між різними, в залежності від обставин.

Для оцінки ефективності рекомендаційних систем і/або готових алгоритмів можна застосовувати різні набори метрик:

* метрики точності прогнозу (MAE, MSE, RMSE);
* метрики прийняття рішень (ROC, Precision/recall);
* метрики оцінки ранжування результатів (Spearman Rank Coefficient, Fraction of concordant pairs);
* бізнес-метрики (збільшення продаж, конверсії).

Інструменти для розробки рекомендаційних систем:

* Apache Mahout;
* LensKit;
* MyMediaLight;
* GraphLab;
* PredictionIO;
* EasyRec;
* Crab;
* Recommenderlab.

## 2.3. Огляд існуючих програмних рішень

Далі пропонується огляд розроблених раніше програмних систем, які намагаються вирішити аналогічну задачу, та приводяться їх недоліки.

**2.3.1. Система автоматизованої побудови дидактичної онтології на основі Wikipedia.** У роботі проведено аналіз особливостей структури статей Вікіпедії та перехресних посилань щодо визначення дидактичних відношень між поняттями онлайн-енциклопедії. На основі аналізу запропоновано набір правил нечіткого виведення для визначення дидактичного порядку статей Вікіпедії, а також додаткова обробка результату роботи апарату нечіткого виведення. Результат у вигляді графу дидактичної онтології на базі відношень дидактичного порядку дозволяє будувати адекватні послідовності вивчення понять цільової предметної області, що спрощує процес самонавчання. Розроблений програмний комплекс реалізує запропонований формальний апарат [23]. Приклад результату роботи системи наведено на рисунку 2.2.

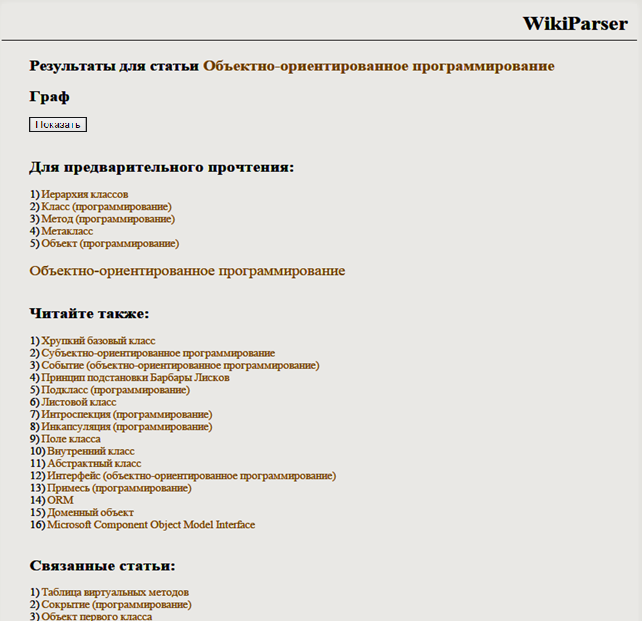


Рисунок 2.2 — Результат роботи системи автоматизованої побудови дидактичної онтології на основі Wikipedia

Хоча дана робота є значним вдосконаленням роботи, що розглядалася попередньо (наприклад, модернізує формальний апарат та впроваджує деякі механізми кешування даних на проміжних та кінцевому етапах роботи системи), вона також має очевидні недоліки:

* швидкість роботи системи не є задовільною для можливості реального практичного застосування;
* обмежена кількість рівнів вкладеності статей під час аналізу;
* деякі питання щодо формального апарату потребують додаткових досліджень;
* зазначений механізм кешування не дає можливості роботи з реальними об’ємами даних та не вирішує деяких важливих проблем, що перешкоджають його більш ефективному використанню та вдосконаленню (наприклад, проблема HTTP-перенаправлень, що виникає під час зміни назви статей онлайн-енциклопедії);
* можлива робота тільки з Вікіпедією.

**2.3.2. Система аналізу онлайн-енциклопедії для генерації дидактичних рекомендацій.** У ході виконання цієї роботи [3] було розроблено систему, що виконує побудову дидактичних зв’язків між статтями онлайн-енциклопедії на основі автоматичного аналізу їх тексту для полегшення процесу вивчення обраної користувачем тематики.

Робота має значні переваги над попередньо розробленими системами на цю тематику:

* у роботі застосовано та удосконалено формальний апарат, запропонований попередніми дослідниками даної теми;
* розроблено ефективні засоби кешування проміжних результатів, що позитивно вплває на продуктивність системи;
* використовується API онлайн-енциклопедії, що забезпечує мінімальні витрати часу на збір даних;
* розглянуто та вирішено проблему HTTP-перенаправлень, що виникають при зміні назв статей енциклопедії;
* було модифіковано старі та запропоновано нові типи фактів, що має безпосередній вплив на якість результатів роботи системи;
* розроблена система має переваги також в тому, що здатна працювати не

лише з Вікіпедією, а й з будь-якою онлайн-енциклопедію, API якої сумісний з API WikiMedia.

Приклад роботи системи можна побачити на рисунку 2.3.

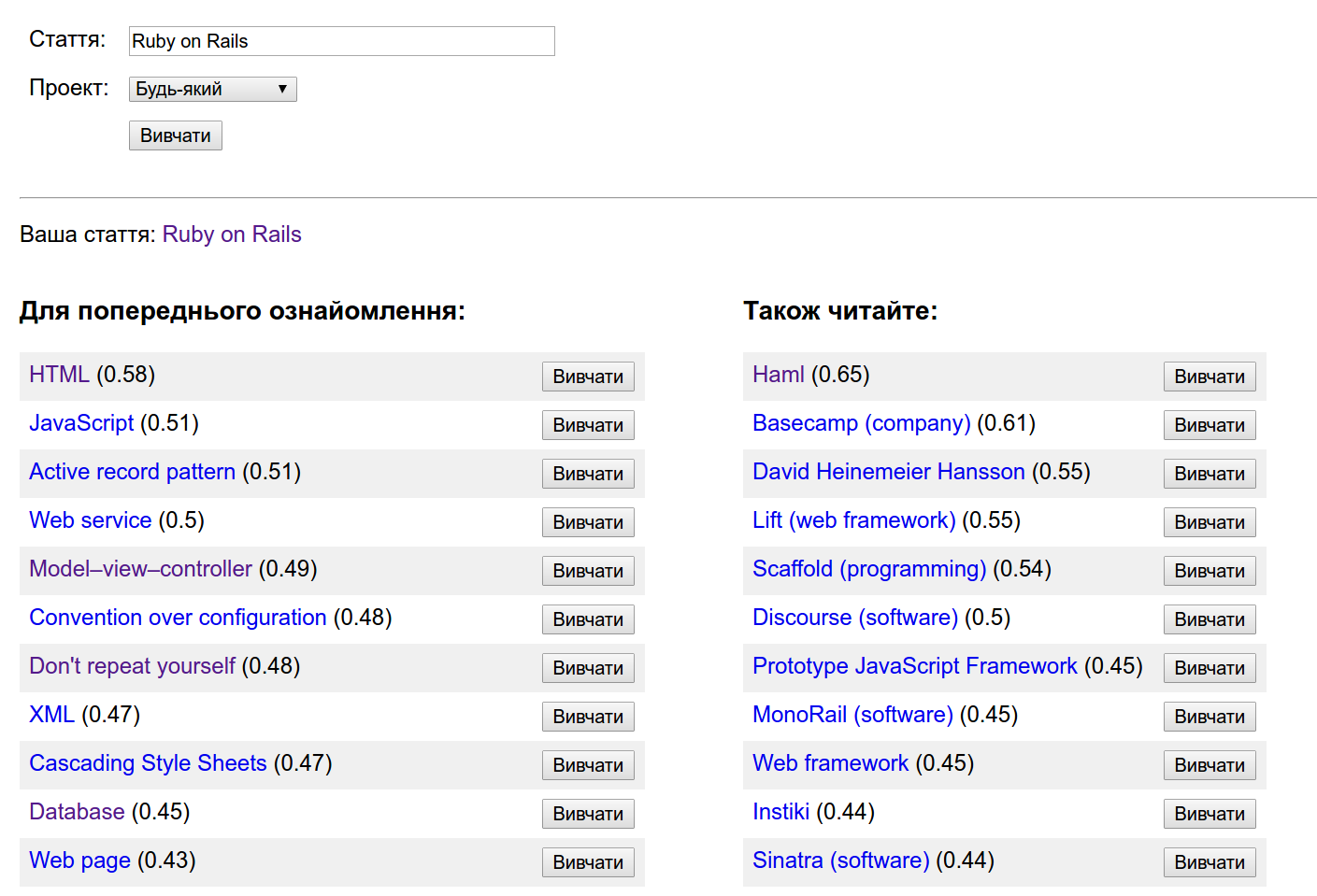


Рисунок 2.3 — Результат роботи системи аналізу онлайн-енциклопедії для генерації дидактичних рекомендацій

У висновках цієї роботи було вказано наступні удосконалення, які можуть бути зроблені у майбутньому:

* впровадження нових типів фактів для покращення якості побудови дидактичних зв’язків;
* використання паралельного аналізу даних для забезпечення більшої швидкості процесу;
* дослідження можливості використання нереляційних баз даних як більш ефективного інструменту для вирішення даної задачі;
* запуск системи на потужному апаратному забезпеченні;
* розроблення більш точного механізму відсіювання статей з результуючої вибірки, які не є важливими для вивчення тематики, до якої належить поняття, що цікавить користувача.

Отже, бачимо, що робота у цій галузі на сто відстоків не закінчена, і ми маємо низку задач для продовження досліджень.

Дана робота фокусується на вирішенні проблем високопродуктивності для того, щоб виштовхнути систему, що розробляється, на радикально новий рівень та зробити можливим її використання великою кількістю користувачів на реальних об’ємах даних. У наступному розділі детально розглядаються методи вирішення цих проблем, що були використані у виконання цієї роботи.

# 3. ЗАСОБИ ТА МЕТОДИ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ

Для написання програмного продукту використано наступні методи та засоби розробки.

## 3.1. Засоби розробки

**3.1.1. Мова програмування Elixir.** Elixir – функційна мова програмування загального призначення з сильною (strongly typed) динамічною типізацією. Програма транслюється у байт-код, який виконується на віртуальній машині Erlang (BEAM).

Мова орієнтована на розробку надійних систем які забезпечують паралельну обробку запитів та невибагливі до апаратних ресурсів у порівнянні з подібними системами написаними на інших мовах програмування. Код програми виконується в легких потоках (не плутати з процесами, які виділяються операційною системою для програми). Потоки ізольовані і не мають спільного стану. Передача інформації іншим потокам можлива тільки через синхронний або асинхронний обмін повідомленнями.

Побудована поверх Erlang, що забезпечує розподіленість, відмовостійкість, виконання в режимі м’якого реального часу, метапрограмування з макросами і поліморфізмом, реалізованим через протоколи. Використовує Erlang / OTP для роботи з деревами процесів. Створений Жозе Валімом (José Valim), який раніше був одним з основних розробників фреймворка Ruby on Rails і співзасновником компанії Plataformatec. Його метою було включити більш високу розширюваність і продуктивність в Erlang VM, зберігаючи сумісність з інструментами і екосистемою Erlang. Програми компілюються в байт-код для віртуальної машини Erlang (BEAM). Кожен елемент програми є виразом, функції мови Erlang можуть бути викликані без впливу на час виконання через компіляції байт-коду в Erlang і навпаки [24].

Метапрограмування дає можливість прямого маніпулювання абстрактним синтаксичним деревом (АСД). Поліморфізм реалізований через механізм протоколів. Як і в Clojure, протоколи забезпечують механізм диспетчеризації (не варто плутати з множинною диспетчеризацією). Паралельне програмування – без поділу ресурсів, через передачу повідомлень (модель акторів). У практиці застосування роблять акцент на рекурсії і функціях вищого порядку замість зациклення, заснованого на побічних ефектах. Для реалізації простого паралелізму використані механізми Erlang зі спрощеним синтаксисом (наприклад, Task). Реалізовано ледачі обчислення і асинхронні колекції з потоками, зіставлення зі зразком. Мова підтримує Unicode та UTF-8-рядки. Реалізована підтримка документування коду в стилі рядків-документації з мови Python (Markdown) [24].

На низькому рівні Elixir використовує примітивні типи, доступні в віртуальній машині Erlang. Так, список – всього лише упорядкований набір значень. Крім того, відповідні модулі і синтаксичний цукор дозволяють використовувати більш високорівневі абстракції. Наприклад, тип Keyword є модулем Elixir, а його реалізація заснована на списку кортежів, і, звичайно, зі значеннями типу Keyword можна працювати і як зі списками.

Elixir має наступні вбудовані типи даних:

* цілі довільної розрядності;
* числа з плаваючою комою;
* атоми;
* інтервальні типи (англ. Ranges);
* регулярні вирази;
* ідентифікатори процесів (PID) і портів;
* посилання;
* кортежі;
* списки;
* відображення (англ. Maps);
* бінарні дані (англ. Binaries);
* функції.

На основі цих примітивних типів в Elixir зокрема побудовані рядки і структури [26].

Elixir був задуманий як поліпшення Erlang, зокрема, значне спрощення синтаксису. Одною з основних відмінностей є можливість повторного привласнення значень змінних. У Elixir не потрібно завершувати кожну команду точкою (за прикладом Прологу), так як вираження поділяються переносом рядка і крапкою з комою (;). У Elixir не потрібно експортувати функції модуля, тоді як в Erlang за замовчуванням всі функції недоступні з інших модулів, якщо вони не згадані в директиві -export. Тим самим, синтаксис Elixir більше схожий на синтаксис Ruby.

**3.1.2. Віртуальна машина Erlang.** Erlang — мова функційного програмування з динамічною типізацією, призначена для розробки програм для різного роду розподілених і багатониткових систем. Розроблена і підтримується компанією Ericsson. Мова містить в собі засоби породження паралельних процесів та їхньої взаємодії за допомогою посилання асинхроних повідомлень, без використання блокувань.

Мова орієнтована для розробки розподілених відмовостійких застосунків, які забезпечують паралельну обробку запитів в режимі реального часу. Мова набула поширення в таких областях, як телекомунікації, банківські системи, електронна комерція, комп’ютерна телефонія і організація миттєвого обміну повідомленнями.

Програма транслюється в байт-код, що виконується віртуальною машиною, що забезпечує переносність. Одночасно розробниками випускається OTP (Open Telecom Platform) — супутній набір бібліотек і компонентів для розробки розподілених систем на мові Erlang. Код проекту поширюється під модифікованою вільною ліцензією MPL.

Свій синтаксис і деякі концепції Erlang успадкував від мови логічного програмування Пролог. Мова підтримує багато типів даних, умовні конструкції, зіставлення зі зразком, обробку винятків, спискові включення і вирази бітових рядків, функції (анонімні функції, функції вищого порядку, рекурсивні визначення функцій, оптимізацію хвостовій рекурсії), модулі, прийом і відправку повідомлень між процесами. Препроцесор підтримує роботу з макросами і включення заголовних файлів.

Популярність Erlang почала зростати в зв’язку з розширенням його області застосування (телекомунікаційні системи) на високонавантажені паралельні розподілені системи, які обслуговують мільйони користувачів WWW, такі як чати, системи управління вмістом, веб-сервери і розподілені, що вимагають масштабування, бази даних. Erlang застосовується в декількох NoSQL-базах даних високої доступності.

В середині 1980-х у комп’ютерній лабораторії компанії Ericsson досліджували можливість застосування існуючих на той момент мов програмування для програмного забезпечення телекомунікаційних систем. Джо Армстронг (Joe Armstrong), Роберт Вірдінг (Robert Virding) і Майк Вільямс (Mike Williams) під керівництвом Бьярне Деккера (Bjarne Däcker), написавши прототипи програм на різних мовах, прийшли до висновку, що жодна з цих мов не мала повного набору можливостей, необхідних в області телекомунікаційних систем. В результаті була створена нова мова програмування – Erlang. Свою назву мову, ймовірно, отримала на честь датського математика та інженера Агнер Ерланга, засновника наукового напрямку з вивчення мережевого трафіку в телекомунікаційних системах. За іншою версією, назва мови спочатку було скороченням від «ericsson language». Вплив на Erlang надали ML, Міранда, Ада, Модула-2, CHILL, Пролог. Крім того, на спосіб оновлення програмного забезпечення вплинув Smalltalk і використані Ericsson пропрієтарні мови EriPascal і PLEX.

Знадобилося чотири роки розвитку мови і прототипування з використанням віртуальної машини Прологу, після чого в 1991 році Майк Вільямс переписав віртуальну машину для Erlang на Сі. У 1992 році Erlang був вперше використаний в комерційному проекті. У 1995 році вийшов новий реліз Erlang, що увібрав накопичений до того моменту досвід використання мови. Мова була визнана досить розвиненою для використання в інших продуктах компанії (рішення для широкосмугового зв’язку, GPRS, ATM).

У грудні 1995 року сталася подія, яку Джо Армстронг вважає вирішальним для Erlang: проект AXE-N в Ellemtel по створенню нового маршрутизатора (як обладнання, так і системного програмного забезпечення на C ++) зазнав невдачі. В результаті реорганізації проекту вдалося, використавши розроблене обладнання та мову програмування Erlang, почати роботи над ATM-маршрутизаторами серії AXD. Ресурсів лабораторії для такого проекту виявилося недостатньо, тому для робіт по Erlang було створено виробничий підрозділ під назвою OTP (Open Telecom Platform). У 1996 році світ побачив однойменний фреймворк OTP.

Несподівано, в 1998 році топ-менеджмент Ericsson вирішив не брати на себе зобов’язань по розробці і підтримці власної мови програмування, зосередившись натомість на Java. Використання Erlang було заборонено в нових проектах Ericsson Radio AB у зв’язку з реалізацією плану по аутсорсингу програмної технології компанії Rational Inc. Це рішення дуже сильно вплинуло на майбутнє Erlang: воно призвело до відкриття коду Erlang під відкритою ліцензією EPL (аналог Mozilla Public License), а також послужило головною причиною початку поширення мови за межами створившої її компанії. Основним запереченням проти відкриття сирцевого коду було вирішення питань, що стосуються патентів, але ці труднощі були подолані. Незабаром багато хто з основних розробників покинули Ericsson, щоб організувати власний бізнес – Bluetail AB.

На початку 2000-х вчені кола стали проявляти інтерес до Erlang. З 2002 року став проводитися щорічний Erlang Workshop. Ericsson продовжував спонсорування проекту HiPE (від англ. High-Performance Erlang - високопродуктивний Erlang) Уппсальского університету. Проект HiPE займався ефективною реалізацією мови і інструментами для перевірки типів, а з 2001 року створений в групі проекту компілятор в машинний код входить в поставку версії Erlang/OTP, що вільно розповсюджується. Роботи, пов’язані з Erlang, ведуть і інші вищі навчальні заклади. Інструменти для рефакторінга створені в Кентском університеті у Великій Британії та університеті Лоранда Етвеша в Угорщині, інструменти для різних видів тестування – в Мадридському політехнічному університеті, технічному університеті Чалмерса і Гетеборзькому університеті.

Коли системи із симетричною багатопроцесорною технологією тільки починали завойовувати ринок серверів і настільних комп’ютерів, кидаючи виклик розробникам програмного забезпечення, вже в 2006 році перша версія Erlang з підтримкою SMP була випущена спільними зусиллями команди OTP з Ericsson і команди HiPE. Незабаром після цього вийшла перша майже за десятиліття велика монографія по Erlang: “Programming Erlang” Джо Армстронга, після чого багато розробників відкрили для себе Erlang/OTP, і мова стала набирати популярність.

Процес розвитку мови включає в себе розгляд пропозицій щодо розвитку – EEP (англ. Erlang Enhancement Proposal). Через ці пропозиції Erlang-спільнота вносить зміни в стандартну поставку Erlang [27].

За свідченням Майка Вільямса, Erlang замислювався для вирішення трьох проблем розробки розподілених систем м’якого реального часу з високим ступенем паралелізму: можливості швидкої і ефективної розробки ПЗ; отримання системи, стійкої до програмних і апаратних збоїв, і можливості оновлення системи “на льоту”, без простою обладнання. За словами Вільямса, філософія, якої дотримувалися розробники Erlang, підходить і для розробки програмного забезпечення на цій мові [28].

Більшість мов, створених раніше Erlang, були розроблені без попереднього знаходження своєї області застосування, тоді як Erlang був розроблений спеціально на основі вимог до розподіленим, відмовостійким, паралельним системам реального часу. З розвитком мережі Інтернет виявилося, що багато програм мають аналогічні вимоги, чим і пояснюється зростання інтересу до мови.

Висока відмовостійкість криється в застосуванні ізольованих один від одного легких процесів, пов’язаних лише механізмом обміну повідомленнями та сигналами виходу.

Erlang є декларативною мовою програмування, яка швидше використовується для опису того, що повинно бути обчислено ніж як. Наприклад, визначення функції, яка використовує зіставлення зі зразком для вибору одного з варіантів обчислення або вилучення елемента даних з складової структури, нагадує рівняння. Зіставлення зі зразком поширене навіть на бітові рядки, що спрощує реалізацію телекомунікаційних протоколі.

Функції є об’єктами першого класу в Erlang. У мові також широко застосовуються характерні для функціональної парадигми програмування спискові включення (генератори списків).

Відмінною особливістю мови є застосування легких процесів відповідно до моделі акторів. Такий підхід дозволяє виконувати одночасно сотні тисяч і навіть мільйони таких процесів, кожен з яких може мати скромні вимоги по пам’яті. Процеси ізольовані один від одного і не мають загального стану, але між ними можна встановити зв’язок і отримувати повідомлення про їх стан. Для взаємодії процесів використовується асинхронний обмін повідомленнями. Кожен процес має свою чергу повідомлень, обробка якої використовує зіставлення зі зразком. Процес, який відправив повідомлення, не отримує повідомлення про доставку, навіть якщо ідентифікатор процесу-одержувача недійсний або одержувач ігнорує повідомлення. Таким чином, відповідальність за правильно організовану взаємодію між процесами лежить на розробнику [29].

Наприклад, при реалізації на Erlang мережевого чату структура програми може безпосередньо відображати одночасність дій користувачів з обміну повідомленнями шляхом запуску нових процесів. Ефективність передачі повідомлень зберігається і при збільшенні числа процесів, а вимоги до пам’яті мінімізуються за рахунок того, що легкими процесами управляє віртуальна машина, а не операційна система.

Erlang з самого початку проектувався для розподілених обчислень і масштабованості. Розподіл обчислень вбудовано в синтаксис і семантику мови, тому побудову системи можна вести, абстрагуючись від конкретного місця обчислень. У стандартному постачанні Erlang може налагодити зв’язок процесів по протоколу TCP / IP незалежно від підтримуваних їм нижчих платформ (операційних систем).

Працюючий екземпляр середовища виконання Erlang (англ. Erlang runtime system) називається вузлом (англ. Node). Програми, написані на Erlang, здатні працювати на декількох вузлах. Вузлами можуть бути процесори, декілька ядер одного процесора, і навіть цілий кластер машин. Вузол має ім’я і “знає” про існування інших вузлів на даній машині або в мережі. Створення та взаємодія процесів різних вузлів не відрізняється від організації взаємодії процесів всередині вузла. Для створення процесу на іншому вузлі процесу досить знати його ім’я і, без особливих на те підстав, він може не цікавитися фізичним розташуванням процесу, який з ним взаємодіє. Синтаксис відправки повідомлення процесу на своєму вузлі і віддаленому – один і той же .

Завдяки вбудованим в мову можливостям розподілених обчислень об’єднання в кластер, балансування навантаження, додавання вузлів і серверів, підвищення надійності викликають лише невеликі витрати на додатковий код. За замовчуванням вузли спроектовані для роботи всередині відокремленого сегмента мережі (DMZ), але, якщо необхідно, комунікація між вузлами може відбуватися із застосуванням захищеного криптографічними методами протоколу SSL [30].

Програми на високорівневій мові Erlang можуть бути використані в системах м’якого реального часу (що іноді переводять як “псевдореальність” або “квазіреальність”). Автоматизоване управління пам’яттю і збірка сміття діють в рамках одного процесу, що дає можливість створювати системи з мілісекундним часом відгуку (навіть незважаючи на необхідність збору сміття), які не відчувають погіршення пропускної здатності при високому навантаженні.

Для систем, які не можуть бути зупинені для поновлення коду, Erlang пропонує гарячу заміну коду (англ. Hot code upgrade). При цьому в додатку можуть одночасно працювати стара і нова версії коду. Таким чином, програмне забезпечення на Erlang може бути модернізовано без простоїв, а виявлені помилки виправлені.

**3.1.3. Пошукова платформа Elasticsearch.** Elasticsearch — вільне програмне забезпечення, пошуковий сервер, розроблений на базі Lucene. Надає розподілений, мультиарендний повнотекстовий пошуковий рушій з HTTP веб-інтерфейсом і підтримкою безсхемних JSON документів.[31]

Є найбільш популярним пошуковим рушієм, випереджаючи Apache Solr. Розробляється на Java і випускається під ліцензією Apache License.[32]

Elasticsearch є продовженням проекту Compass, значну частину якого було переписано задля підтримки масштабовності (Compass — пошуковий рушій на базі Lucene, розроблявся Шейєм Беноном в 2004—2009 роках). Перший випуск Elasticsearch (версія 0.4) відбувся у лютому 2010.

Elasticsearch може використовуватись для індексування та пошуку будь-яких типів документів. Він надає масштабовний пошук, має пошук близький до реального часу і підтримку мультиарендності.

Elasticsearch має можливість розподілення, індекси можуть бути розділені по шардах, при чому кожен шард може мати нуль чи більше реплік. Кожен вузол містить один чи більше шардів і діє як координатор делегування операцій на потрібний шард. Балансування та маршрутизація виконується автоматично.

Приклади використання:

* реалізація пошуку по веб-сайту, наприклад, пошук товарів в інтернет-магазині. В цьому випадку Elasticsearch індексує каталоги товарів, та надає можливості пошуку та припущення щодо автозаповнення;
* зберігання журналів подій чи транзакцій, аналізування і добування даних для отримання тенденцій, статистик, висновків, аномалій. В даному випадку можна використовувати Logstash для збору, об’єднання, аналізу даних, і потім перенаправляти ці дані в Elasticsearch для подальшого опрацювання;
* для розробки, наприклад, платформи по ціновому оповіщенню, що дозволяє досвіченим в цій сфері користувачам визначати правила типу “Я зацікавлений в придбанні електронного ґаджету і я хочу бути оповіщеним якщо ціна у будь-якого постачальника впаде нижче певного значення протягом наступного місяця. В даному випадку можна збирати ціни, індексувати їх в Elasticsearch і використовувати функцію зворотнього пошуку: зіставляти коливання цін з запитом користувача і при відповідності до запиту надсилати сповіщення;
* для впровадження аналітики/бізнес-аналітики в проект, коли треба швидко досліджувати, аналізувати, візуалізувати надзвичайно великі об’єми даних (мільйони чи мільярди записів). В даному випадку доцільно використання Elasticsearch для збереження даних і Kibana для побудови користувацьких панелей відображення і візуалізації необхідних аспектів. До того ж, можна використовувати агрегаційні функції Elasticsearch для здійснення комплексної бізнес-аналітики даних.[33]

## 3.2. Модель нечіткого виведення на основі фактора впевненості

Стенфордська теорія фактора впевненості ґрунтується на ряді спостережень. По-перше в традиційній теорії ймовірностей сума ймовірностей відношення і його заперечення повинна дорівнювати одиниці. Однак нерідкі ситуації, при яких людина-експерт, оцінивши ймовірність (достовірність) деякого відношення значенням 0.7, вважає що відношення істинне, і зовсім не враховує, що воно може бути помилковим. Ще одне припущення, підкріплююче теорію фактора впевненості, полягає в тому, що знання самих правил набагато важливіше, ніж знання алгебри для обчислення їх достовірності. Міра впевненості (або довіри) — це неформальна оцінка, яку людина-експерт додає до висновку, наприклад: “ймовірно, це так”, “майже напевно, це так” або “це абсолютно неймовірно”.

Стенфордська теорія фактора впевненості вводить деякі прості припущення про міру довіри і пропонує правила для об’єднання свідоцтв при виведенні висновків. Першим допущенням є поділ міри довіри і недовіри (“за” і “проти”) для кожного відношення.

Нехай — міра впевненості у гіпотезі при заданому свідоцтві .

Позначимо через міру недостовірності гіпотези при заданому свідоцтві .

Тоді , якщо , або , якщо .

Ці дві міри накладають обмеження одна на одну, так як заданою вважається частина свідчення на користь даної гіпотези, або проти неї. У цьому полягає важлива відмінність між алгеброю впевненості і теорією ймовірності. Якщо зв’язок між заходами довіри і недостовірності встановлено, їх можна знову об’єднати за формулою (3.1).

. (3.1)

З наближенням фактора впевненості (certainty factor) до 1 посилюється довіра до гіпотези, а з наближенням до 0 — її заперечення. Близькість значення до 0 означає, що доказів на користь гіпотези і проти неї занадто мало, або ці свідоцтва збалансовані.

Коли експерти формують базу правил, вони зіставляють з кожним правилом певне значення . Фактор відображає впевненість у надійності правила. Міри впевненості дозволяють регулювати продуктивність системи, хоча слабкі варіації міри довіри зазвичай мало впливають на загальну результатність. Ця друга роль міри довіри підтверджує тезу про те, що “знання — сила”. Іншими словами, найкращою гарантією коректності діагностики є цілісність самих знань.

Передумова кожного правила складається з ряду факторів, пов’язаних операціями кон’юнкції і диз’юнкції. При використанні продукціонного правила враховуються фактори довіри, пов’язані з кожною умовою передумови. Їх поєднання визначає міру довіри всій передумові для передумов і за формулами (3.2) та (3.3).

, (3.2)

. (3.3)

Для отримання фактора впевненості в заключенні правила об’єднаний фактор впевненості в передумовах , отриманий за допомогою наведених вище правил множиться на самого правила.

Розглянемо, наприклад, таке правило бази знань:

, де , , — передумови, а , — укладення правила з фактором довіри , рівним 0.7 і 0.3 відповідно. Ці числа додаються до правила при його розробці і являють впевненість експерта у висновку, якщо всі передумови відомі з повною визначеністю. Якщо в процесі виконання програми для , , отримані значення , рівні 0.6, 0.4 і 0.2 відповідно, то в даному випадку і слід враховувати з факторами довіри , що рівні 0.28 і 0.12 відповідно. Проведемо обчислення для цього прикладу:

,

.

Значення для в описі правила одно 0.7, так що додається до множини конкретних знань про дану ситуацію зі значенням = 04 \* 07 = 0.28.

Значення для в загальному правилі 0.3, тому додається до множини знань про дану ситуацію зі значенням = 0.3 \* 0.4 = 0.12.

Потрібно визначити ще одну метрику. Як об’єднати кілька значень , якщо два або більше правил приводять до одного і того ж результату ? Правило для цього випадку відображає аналогію алгебри достовірності з теорією ймовірності. Міри довіри при об’єднанні незалежних свідчень перемножуються. Багаторазово використовуючи це правило, можна об’єднувати результати будь-якої кількості правил, використовуваних для визначення результату . Якщо представляє фактор довіри результату , а раніше не використане правило призводить до результату (знову) зі значенням , то нове значення результату для додатніх і обчислюється за формулою (3.4), для від’ємних — за формулою (3.5), а у всіх інших випадках — за формулою (3.6).

(3.4)

(3.5)

. (3.6)

Крім легкості обчислень ці комбінаційні рівняння мають інші корисні властивості. По-перше, значення фактора , що обчислене згідно з цим правилом, завжди буде лежати між 1 і -1. По-друге, в результаті об’єднання протилежні значення скорочуються, що теж є позитивним моментом. І, нарешті, комбінована міра є монотонно зростаючою (спадною) функцією, що в якійсь мірі і слід було очікувати для узагальненого свідоцтва.

Отже, міра впевненості стенфордської алгебри описує людську (суб’єктивну) оцінку причинної ймовірності міри. Як вказувалося в традиційному баєсовському підході, якщо , і впливають на , то при міркуванні про необхідно виділити і скомбінувати все апріорні і апостеріорні ймовірності, включаючи , , , , . Підхід, заснований на стенфордському факторі впевненості, дозволяє спеціалісту по знанням описати всі ці взаємозв’язки одним фактором довіри правилу, тобто . Ця проста алгебра краще відображає спосіб мислення людини-експерта.

Теорія фактора впевненості може бути піддана критиці як необґрунтована. Незважаючи на те, що вона визначається в рамках формальної алгебри, значення мір довіри не так строго обґрунтовано, як в теорії ймовірностей. Проте теорія фактора впевненості не намагається будувати алгебру для “коректного” міркування. Вона забезпечує компроміс, що дозволяє експертній системі об’єднувати свідоцтва по мірі вирішення задачі. Ці заходи є евристичними в тому сенсі, що впевненість експерта в результатах є неповною, евристичною та неформальною. У системі MYCIN фактори використовуються при евристичному пошуку для установки пріоритетів цілей і визначення точки відсікання, після якої мета не повинна більше розглядатися. Але, незважаючи на використання факторів для підтримки виконання програми і збору інформації, продуктивність програми визначається якістю правил, які повинен визначити експерт [34].

## 3.3 Оптимізація процесу завантаження та обробки даних

У роботі [3] було використано наступну послідовність дій під час виконання поставленої задачі системою: на вхід система отримувала назву статті онлайн-енциклопедії, (цільова стаття); наступним етапом був збір статей, що зв’язані з цільовою; далі вібдувалася побудова фактів і дидактичних зв’язків. Важливим є те, що усі етапи роботи системи відбувалися послідовно, і, хоча було використано декілька технік задля оптимізації цього процесу (таких як постійне збереження даних до локального сховища для того, щоб уникнути зайвих запитів до енциклопедії), у ході виконання даної роботи на основі приблизної оцінки було зроблено висновок, що даний підхід не є задовільним с точки зору ефективності. Дана робота ставить перед собою ціль обробити англійську Вікіпедію у повному її обсязі, і якщо припустити, що такий алгоритм буде використано для послідовного аналізу всіх статей (що призведе до того, що результати будуть збережені до локального сховища даних і зможуть бути відображені для користувача миттєво після запиту), то на цей процес піде дуже багато часу. Тому у даній роботі було запропоновано розділити систему на дві складові. Перша складова буде виконувати завантаження та обробку усіх даних. Друга буде використовувати ці дані для побудови фактів та дидактичних зв’язків у режимі реального часу у якості відповіді на запит користувача.

У даному підрозділі зосередемося на процесі збору даних. Він включає у себе декілька дій:

* завантаження вмісту статей онлайн енциклопедії;
* завантаження інформації щодо усіх існуючих HTTP-перенаправлень для кожної статті;
* оборобка даних та збереження іх у форматі, що забезпечить максимальну ефективність наступного етапу пошуку фактів да зв’язків;

Взагалі, етап завантаження даних є найбільш слабким місцем з точки зору ефективності роботи системи, адже на даному етапі ми залежимо як мінімум від передачі даних по мережі, що завжди буде призводити до втрат часу. Також є неможливим (або дуже обмеженим) створення паралельних HTTP-запитів для отримання інформації, оскільки це може бути причиною занадто великого навантаження на онлайн-енциклопедію. Таке обмеження зазвичай регулюється правилами та етикетом використання API енциклопедії та призводить до блокування зі сторони модераторів проекту у разі їх порушення.

Що стосується методу завантаження даних з онлайн-ециклопедії, існує два варіанти: виконувати HTTP-запити напряму до адрес статей або скористатися API (прикладний програмний інтерфейс), що існує у WikiMedia. Слід зазначити, що для виконання поставлених цілей є більш простим аналіз HTML-коду сторінок, аніж їх відповідної вікі-розмітки, адже HTML-код є більш детальним, а спрощений вікі-код є більш зручним для редагування сторінок користувачами (але більш складним для автоматичної обробки). Проте можливість отримання HTML-коду через API не є задовільною з точки зору ефективності через природу самого API, тож останній може бути використан лише для отримання вікі-розмітки. Також, оскільки На жаль, як буде більш детально описано у наступному підрозділі, використання API є обов’язковим також з точки зору ефективності. Та, не дивлячись на те, це створює додаткові складнощі при аналізі отриманих даних, використання API для отримання вікі-розмітки є обовз’язковим, оскільки створення паралельних запитів не є допустимим, а API дозволяє отримувати більш ніж одну статтю за один запит. Для Вікіпедії їх кількість може становити до 50 для звичайних користувачів та до 500 для користувачів зі спеціальними привілеями “бота” (отримання таких привілеїв може значно прискорити швидкість збору інформації; проте, у даній роботі використоваються привілеї звичайного користувача, що виявилось цілком достатнім).

Далі роздивимось проблему HTTP-перенаправлень. На етапі пошуку фактів система працює з посиланнями уже завантажених статей, тобто необхідна можливість кожному такому посиланню однозначно поставити у відповідність деяку статтю з локального сховища. Рішенням є збереження разом з кожною статтею не тільки її реальної назви, а й усіх альтернативних назв, посилання з використанням яких ведуть до цієї статті через перенаправлення. Таким чином, зустрівши ці посилання під час пошуку фактів ми завжди зможемо знайти відповідну статтю. У роботі [3] збір таких альтернативних назв виконувався системою під час аналізу тексту, але існує спосіб зробити запит до API для отримання цих даних. Такий спосіб використувуються у даній роботі. Хоча це і потребує додаткових запитів до API, це позитивно впливає на ефективніть процесу у цілому.

Тепер, роздивившись детально існуючи проблеми та вирішивши використовувати API ециклопедії для їх вирішення, необхідно визначити найефективніший спосіб звернення до API. Остаточна комбінація параметрів HTTP-запиту, що використовується у цій роботі та дозволить отримувати усі необхідні дані максимально швидко, наведена у таблиці 3.1.

Таблиця 3.1 — GET-параметри запиту до API енциклопедії.

|  |  |
| --- | --- |
| Параметр | Значення |
| action | query |
| format | json |
| formatversion | 2 |
| generator | allpages |
| gapnamespace | 0 |
| gapfilterredir | nonredirects |
| gaplimit | max |
| prop | revisions|redirects |
| rvprop | content |
| rdprop | titlte |
| rdnamespace | 0 |
| rdlimit | max |

Наведемо призначення цих параметрів:

* параметр “action” зі значенням “query” дозволяє отримати інформацію про енциклопедію та дані, що у ній зберігаються, такі як вікі-розмітка сторінок, їх посилання та категорії;
* параметр “format” зі значенням “json” дозволяє отримувати результати у форматі JSON;
* параметр “formatversion” зі значенням “2” вказує, що ми бажаємо отримувати дані використовуючи більш сучасну версію формату, з якою у нашому випадку працювати легше;
* параметр “generator” зі значенням “allpages” дозволяє послідовно отримувати інформацію про усі статті енциклопедії замість того, щоб вказувати назви чи ідентифікатори статей вручну;
* параметр “gapnamespace” зі значенням “0” дозволяє отримувати інформацію лише про сторінки, що належать до основної категорії (відсіюючи службові сторінки, зображення тощо);
* параметр “gapfilterdir” зі значенням “nonredirects” відсіює сторінки, що слугують лише перенаправленнями на інші сторінки, а для отримання цієї інформації використовується інша стратегія за допомою параметрів, які будуть розглянуті далі;
* параметр “gaplimit” зі значенням “max” вказує, що ми бажаємо отримувати максимально можливу кількість сторінок за один запит (у нашому випадку – до 50, в залежності від характеру сторінок);
* параметр “prop” зі значенням “revisions|redirects” вказує, що ми бажаємо отримати інформацію про версії вмісту сторінок, а також список усіх перенаправлень на неї;
* параметр “rvprop” зі значенням “content” вказує, що з інформації про версії вмісту сторінок ми зацікавлені тільки у самому вмісті (при комбінації параметрів, що використувуються, ми будемо отримувати лише вміст останньої версії);
* параметр “rdprop” зі значенням “title” вказує, що ми бажаємо отримувати лише назву сторінки з інформації про перенаправлення;
* параметр “rdnamespace” зі значенням “0” вказує, що ми бажаємо отримувати лише перенаправлення, що належать до до основної категорії сторінок;
* параметр “rdlimit” зі значенням “max” вказує, що ми бажаємо отримувати

максимально можливу кількість перенаправлень за один запит (у нашому випадку –  
не більше 500);

Також дуже важливим є робити HTTP-запити з заголовком “Accept-Encoding” встановленим у значення “gzip”, що дозволить використовувати стиснення даних, та тримати HTTP-з’єднання відкритими.

Після завантаженням даних потрібно перед збереженням їх до локального сховища даних виконати обробку, привевши їх до структури, яка дозволить максимально легко та ефективно працювати з ними під час побудови зв’язків. Ця задача також потребує досить великої кількості часу. Проте, ми можемо дуже просто зменшити цей час до нуля. Справа в тому, що немає сенсу чекати повного завершення завантаження даних. З того моменту як система починає робити черговий запит до енциклопедії, до моменту, коли ми отримаємо відповідь , ресурси апаратного забезпечення, на якому працює система, є вільними. Отже, маємо час для того, щоб обробити та зберегти результати попередньго запиту. На практиці, використувуючи при цьому можливості віртуальної машини Erlang для паралельного обчислення, цього часу цілком достатньо щоб обробити їх повністю. Таким чином, після завершення процесу завантаження даних, ми вже маємо їх оброблені та збережні до нашого сховища.

Використовуючи наведені вище методи та зауваження, програмна система, що розробляється, успішно здійснила завантаження та обробку усіх статей англомовної Вікіпедії, кількість яких на момент написання цієї записки становить понад   
5 500 000, за приблизно 70 годин, що є дуже задовільним результатом.

## 3.4. Формування множини зв’язаних понять

Задачею системи є обробка статей онлайн-енциклопедії для побудови для кожної з них дидактичних зв’язків між цільовою та зв’язаними статтями. Отже, першим кроком після отримання на вхід цільової статті є формування множини зв’язаних статей над якою будуть проводитися усі подальші дії. Пошук зв’язаних статей виконується на основі перехресних гіперпосилань. Проте для статей онлайн-енциклопедій може бути характерна наявність великої кількості посилань на статті з інших предметних областей, що не є важливими для вивчення деякої теми, тож виникає потреба відсіювання зайвих посилань. У даній роботі, як і у роботі [3] не існує обмеження на максимальний рівень віддаленості зв’язаної статті від цільової через посилання, а пропонується вибирати статті до того рівню, на якому кількість статей, що мають спільні категорії с цільовою, буде дорівнювати нулю. Нехай:

* — цільова стаття;
* множина таких статей *с*, що існує шлях зі статті в статтю *с*;
* множина таких статей *с*, що та *с* мають спільні категорії;
* множина таких статей *с*, що посилається на *с*

Тоді результуюча множина статей може бути визначена як (3.7).

) (3.7)

У даній роботі використовується досить цікавий спосіб формування цієї множини. Замість того щоб вибирати статті зі сховища даних та збирати дані на основі посилань та категорій програмними засобами, цей процес виконується на рівні Elasticsearch, сховища даних. Дані вже знаходяться у сховищі в підготовленому виді. Система виконує ряд послідовних запитів до Elasticsearch та у результаті отримує необхідну вибірку статей.

Слід зазначити, що ефективність такого підхіду сильно залежить від потужності апаратного забезпечення. Elasticsearch є дуже чутливим в основному до таких факторів:

* швидкість диску. Тому для ефективної роботи системи необхідно використовувати швидкі SSD-диски;
* об’єм оперативної пам’яті. Найкращою буде ситуація, якщо об’єм пам’яті буде більшим, ніж об’єм усіх даних у сховищі (близько 26-ти гігабайтів інформації), але це не є обов’язковим.

При використанні апаратного забеспечення, що відповідає цим умовам, дозволить системі виконувати цей етап, по приблизним оцінкам, від сотні мілісекнуд до секунди в залежності від статті.

## 3.5. Факти та правила дидактичного впорядкування

Після формування вибірки статей постає задача побудови дидактичних зв’язків між цими статтями. Зв’язки формуються автоматично на основі аналізу тексту статей з використанням нечіткої логіки Б’юкенена [34] на основі прикладу використання даного апарату для дидактичних задач, запропонованих в роботі [35]. Логіка визначення дидактичного порядку ґрунтується на правилах, кожному з яких відповідає певний фактор впевненості .

Першою частиною етапу формування зв’язків є пошук фактів, кожен з яких свідчить у користь наявності зв’язку між двома статтями та його напрямку, для кожної пари з обраних статей. Результуючу множину фактів можна представити у вигляді графу (рисунок 3.1), вершинами якого є статті онлайн-енциклопедії, а ребрами — факти.

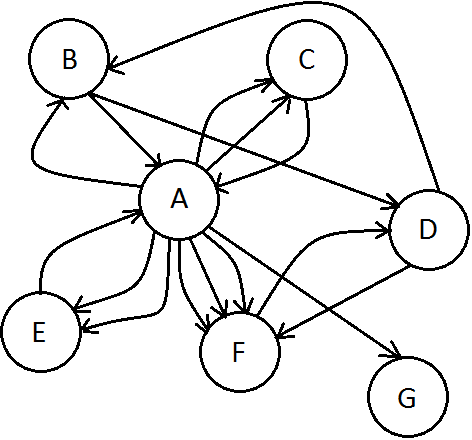


Рисунок 3.1 — Ілюстрація графу знайдених фактів

Кожному типу факту відповідає певне правило, згідно до якого для кожного знайденого факту розраховується відповідний фактор впевненості. Далі описано типи фактів, що були застосовані у даній роботі та подано відповідні їм правила:

1. Якщо у вступі статті знайдено посилання на статтю , то це означає, що стаття дидактично передує статті (3.8).

, (3.8)

де — множина пар таких статей , що у вступі статті існує посилання на статтю ;

— значення, що конфігурується. За замовченням — 0.4.

1. Якщо в основному тексті статті знайдено посилання на статтю , то це буде означати, що стаття дидактично передує статті з фактором впевненості, що залежить від віддаленості позиції посилання від початку основного тексту статті (3.9).

(3.9)

де — множина пар таких статей , що в основному тексті статті існує посилання на статтю ;

— значення, що конфігурується. За замовченням — 0.2;

— позиція посилання на статтю в основному тексті статті ;

— довжина основного тексту статті .

1. Якщо назва статті містить у собі назву статті , то це означає, що стаття дидактично передує статті (3.10).

(3.10)

де — множина пар таких статей , що назва статті містить у собі назву статті ;

— значення, що конфігурується. За замовченням — 0.9.

1. Якщо основний текст статті містить у собі назву статті , то це означає, що стаття дидактично передує статті з фактором впевненості, що залежить від віддаленості позиції назви від початку основного тексту статті (3.11).

(3.11)

де — множина пар таких статей , що в основному тексті статті міститься назва статті ;

— значення, що конфігурується. За замовченням — 0.1;

— позиція назви статті в основному тексті статті ;

— довжина основного тексту статті .

У ході виконання роботи [3] були зафіксовані ідеї нових типів фактів, що можуть брати участь у побудові дидактичних зв’язків, але потребують додаткового дослідження.

Загалом, етап побудови фактів є найбільш важливим для якості результатів роботи системи, на що безпосередньо впливає кількість типів фактів та точність їх відповідних правил.

## 3.6. Граф дидактичних зв’язків та результуюча послідовність понять

На фінальному етапі роботи системи зібрані факти для кожної пари статей трансформуються у зв’язок, що має напрямок та якому відповідає фактор впевненості. Останній розраховується наступним методом: фактори впевненості усіх фактів для кожного типу окремо (якщо їх може бути більше одного) сумуються відповідно до формул Б’юкенена (3.12) [34]. Остаточне значення отримується сумуванням факторів кожного типу за допомогою того ж способу.

(3.12)

Ця операція виконується двічі для перевірки двох гіпотез: стаття дидактично передує статті та навпаки. Якщо отримується ненульовий фактор впевненості для обох гіпотез, така ситуація вважається конфліктною та потребує вирішення на користь однієї з гіпотез (процес детально оглянуто у наступному підрозділі). Нульовий фактор вказує на відсутність зв’язку.

Кінцевим результатом є зважений орієнтований граф, вершинам якого відповідають статті, а ребрам — дидактичні зв’язки (рисунок 3.2).

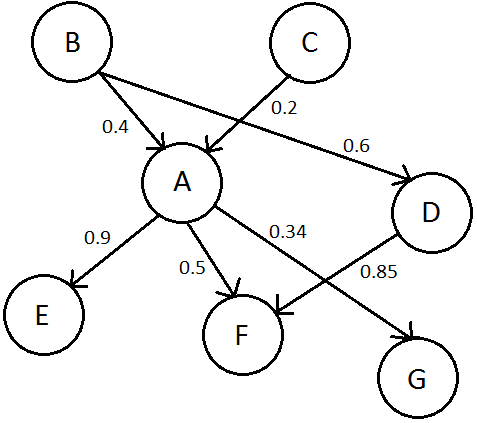


Рисунок 3.2 — Ілюстрація графу дидактичних зв’язків

Для кінцевого користувача граф буде перетворено у лінійну послідовність статей для детального вивчення цільового поняття. Для цього разом з цільовою статтею вибираються статті, що дидактично слідують безпосередньо перед і після неї, і сортуються за фактором впевненості.

## 3.7. Усунення конфліктних ситуацій

При перевірці протилежних гіпотез під час визначення дидактичного відношення між двома статтями можлива ситуація, коли одночасно існують свідчення на користь обох гіпотез. Формально така ситуація виражається у наявності двох протилежних ребер між двома вершинами графа дидактичних зв’язків (рисунок 3.3).

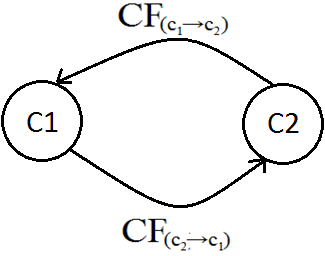


Рисунок 3.3 — Ілюстрація конфлікту

Отже, для статей та маємо наступну ситуацію:

* ;
* .

Таким чином, існує фактор впевненості , що показує достовірність того, що стаття дидактично передує статті і також існує фактор впевненості у тому, що стаття дидактично передує статті . У такому випадку остаточному зв’язку буде відповідати відношення, що має більший фактор впевненості. Сам фактор раніше було запропоновано перераховувати за формулою (3.13) [18].

(3.13)

Але у результаті досліджень, що були проведені у ході виконання даної роботи, було виявлено, що більш точним є інший підхід. Тому у даній роботі результуючий фактор впевненості під час вирішення конфліктної ситуації перераховується за більш простою формулою (3.14).

(3.14)

# 4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

В цьому розділі описано структуру сховища даних, яке використовує система, розглянуто загальну архітектуру та наведено список послідовності етапів роботи системи.

## 4.1. Структура сховища даних

У роботі було використано пошукову платформу Elasticsearch як сховище даних. Elasticsearch є нереляційною системою та дозволяє працювати с даними і спілкуватися з нею використовуючи формат JSON.

Далі роздивимось структуру, що використовується для збереження інформації про статті онлайн-енциклопедії (що також описується за допомогою формату JSON). Ці дані зберігаються в індексі “wiki\_assistant” під типом “page” (рисунок 4.1). Інформація про кожну конкретну статтю поділяється на такі властивості:

* властивість “title” використовується для збережння основної назви статті;
* властивість “redirects” використовується для збереження усіх альтернативних назв статті (що слугують в енциклопедії як перенаправлення до основної назви);
* властивість “categories” використовується для збережння усіх категорій статті;
* властивість “sections” використовується для вмісту статті, який поділяється на декілька секцій.

Властивості “title”, “redirects” та “categories” мають тип даних “keyword”, що призводить до індексації даних, які вони зберігають, таким чином, що вони можуть бути використані для лише для пошуку по точному збігу. Завдяки цьому дані будуть зберігатися більш оптимально, а також це дозволяє нам використати для них опцію “eager\_global\_ordinals”, що призведе при можливості до завантежння усієї інформації с цих властивостей до оперативної пам’яті та, як наслідок, прискорення пошуку даних с використанням цих властивостей.



Рисунок 4.1 — Скорочена структура індексу, що зберігає статті

Окремо слід розглянути також структуру властивості “sections” (рисунок 4.2). Під час етапу завантаження та обробки, вміст статей приводиться до найбільш вигідного с точки зору ефективності виконання етапу побудови фактів та зв’язквів виду. По-перше, вміст розділяється на дві частини: “intro” і “body”, які зберігають вступну частину та основну частину тексту статті відповідно. Крім того, кожна з частин поділяється на такі властивості:

* властивість “text”, що зберігає текст статті. Опція “index” зі значенням “false” забороняє його індексування для пошуку, оскільки це не використовується системою;
* властивість “links”, що зберігає список посилань на інші статті, які були знайдені у тексті даної статті під час його аналізу. Кожне посилання, у свою чергу, має властивості “position” та “target”. Властивість “position” зберігає позицію посилання у тексті статті як ціле число. Властивість “target” зберігає назву статті, на яку вказує дане посилання.

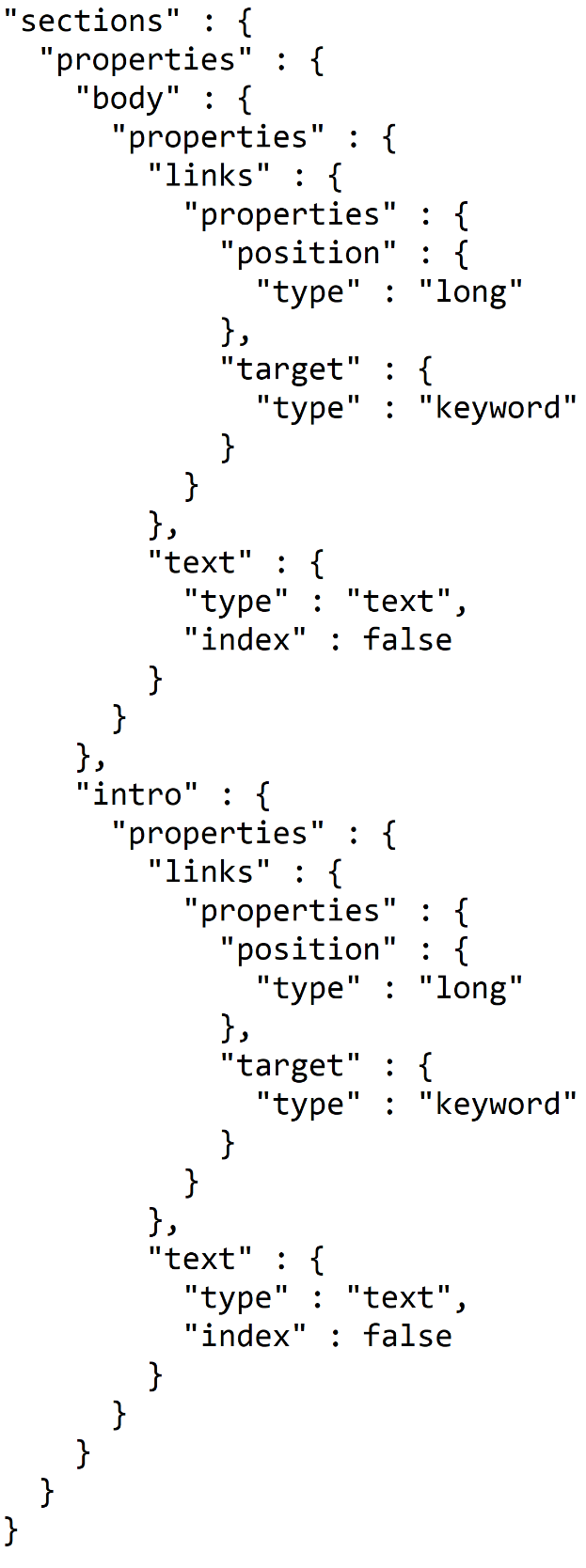


Рисунок 4.2 — Детальна структура властивості “sections”

Також система має окремий індекс у сховищі для зберігання кешованої інформації (рисунок 4.3). Кешування відбувається безпосередньо після побудови зв’язків коли користувач робить запит через веб-інтерфейс.



Рисунок 4.3 —Структура індексу, що використовується для кешування

Структура кешованих даних досить проста. Вона зберігає назву статті під властивістю “title” та список дидактичний зв’язків під властивістю “connections”, кожен з яких має інформацію про назву статті, на яку вказує цей зв’язок, напрямок зв’язку та фактор впевненості під властивостями “page”, “type” та “certainty\_factor” відповідно.

## 4.2. Архітектура системи

Система складається з двох основних компонентів: компонент, що виконує завантаження та аналіз даних з енциклопедії, та веб-інтерфейс, який запускає процес побудови дидактичних зв’язків та демонструє результати користувачу, а також включає в себе декілька допоміжних модулів (рисунок 4.4).

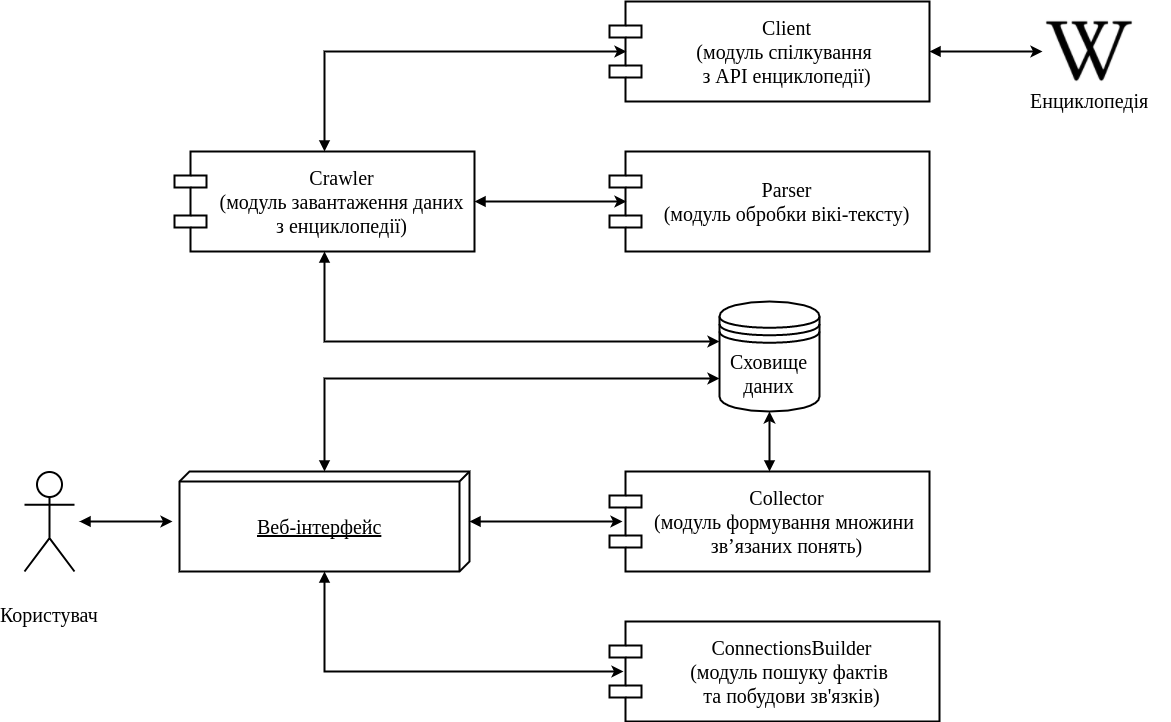


Рисунок 4.4 — Архітектура системи

Роздивимось більш детально призначення модулів системи та те, як вони взаємодіють між собою.

* “Crawler” — основний модуль компоненту, який виконує завантаження даних. Призначений для запуску в окремому процесі. Отримавши на вхід адресу онлайн-енциклопедії, виконує завантаження та аналіз усіх її статей, зберігаючи результати до сховища даних. Делегує виконання деякої роботи модулям “Client” та “Parser”;
* “Client” — модуль, що слугує абстракцією над API WikiMedia та забезпечує спілкування з ним;
* “Parser” — модуль, що відповідає за аналіз вікі-тексту статті та приведення його до потрібної форми, яка була обговорена у попередньому розділі, для подальшого збереження до сховища даних;
* Веб-інтерфейс отримує від користувача назву статті, що його цікавить та, делегуючи виконання цих задач іншим модулям, знаходить зв’язані статті та виконує між ними побудову дидактичних зв’язків. Також, має механізм кешування , який спілкується зі сховищем даних напряму. Деталі внутрішньої реалізації не розглядаються, оскільки це є тривіальною задачею.
* “Collector” — модуль формування множини зв’язаних статей. На цьому етапі вже не відбувається ніяких запитів до енциклопедії, задача виконується з використанням можливостей Elasticsearch.
* “ConnectionsBuilder” — модуль, який, отримуючи на вхід множину статей, виконує пошук фактів та побудову дидактичних зв’язків.

## 4.3. Етапи роботи програмної системи

Робота компоненту, що відповідає за завантаження та аналіз даних, відбувається у такій послідовності:

1. В окремому процеси, використовуючи засоби екосистеми Erlang та Elixir, а також інструменти операційної системи на вибір адміністратора, запускається модуль “Crawler”, якому на вхід подається адреса онлайн-енциклопедії (за замовчуванням використовується англомовна Вікіпедія).
2. Використовуючи функціонал модулей “Client” та “Parser” виконується завантаження та аналіз даних (з допомою паралельних обчислень). Приблизна тривалість процесу складає 70 годин з моменту запуску. Результати завантажуються по частинам до 50-ти статей, після обробки кожної частини результати зберігаються до сховища даних.

Після заверешення цього процесу система готова до роботи. Може бути активізований інтерфейс користувача. При необхідності цей процес може бути виконаний повторно (одноразово або регулярно для підтримання актуальності даних). Робота веб-інтерфейсу при цьому (що є незалежним компонентом) може не перериватися.

Етапи роботи інтерфейсу системи:

1. Кінцевим користувачем через HTML-форму на вхід подається назва статті, що його цікавить.
2. Система перевіряє наявність вже готових результатів для даної статті та, якщо вони вже були закешовані раніше, результати виводяться користувачу.
3. Якщо результати не знайдено, система запускає модуль “Collector”, який формуэ множину зв’язаних з цільовою статей.
4. Далі викликається модуль “ConnectionsBuilder”, який виконує пошук фактів та побудову дидактичних зв’язків.
5. Отримана інформація перетворюється у відсортовану лінійну послідовність.
6. Результат зберігється до локального сховища для кешування.
7. Система формує HTML-документ, що містить візуальне представлення результату.

# 5. МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

В цьому розділі наведені системні вимоги та інструкція по інсталяції програмної системи, також наведені сценарії роботи користувача з програмною системою.

## 5.1. Інсталяція та системні вимоги

Робота система протестована під керуванням UNIX-подібних операційних систем. Для запуску необхідно попередньо встановленовити таке програмного забезпечення:

* Erlang/OTP версії 20 або вище;
* Elixir версії 1.5.2;
* пошукова платформа Elasticsearch версії 5.6.4 або вище.

Детальний процес інсталяції перерахованого програмного забезпечення залежить від операційної системи.

Для інсталяції самої системи необхідно:

* виконати команду “mix local.hex” для отримання Hex – менеджера пакетів Elixir;
* знаходячись у кореневому каталозі коду системи виконати послідовно команди “mix deps.get” та “mix deps.compile” для завантаження та компіляції залежностей проекту;
* для ручного тестування системи можна, викливши інтерактивну консоль за допомогою команди “iex –S mix”, виконати рядок коду “WikiAssistant.Crawler.start()”, який розпочне завантаження даних з англомовної Вікіпедії. Потім, скориставшись командою “mix phx.server”, можно запустити локальний HTTP-сервер та за адресою “http://localhost:4000” (за замовчуванням) протестувати інтерфейс системи.

Для запуску системи в production-середовищі, сервер та процес завантаження даних може бути активізований найбільш зручним способом на вибір адміністратора.

## 5.2 Сценарії роботи користувача з системою

Інтерфейс користувача (рисунок 5.1) представляє собою HTML-сторінку, що містить форму для взаємодії с системою.

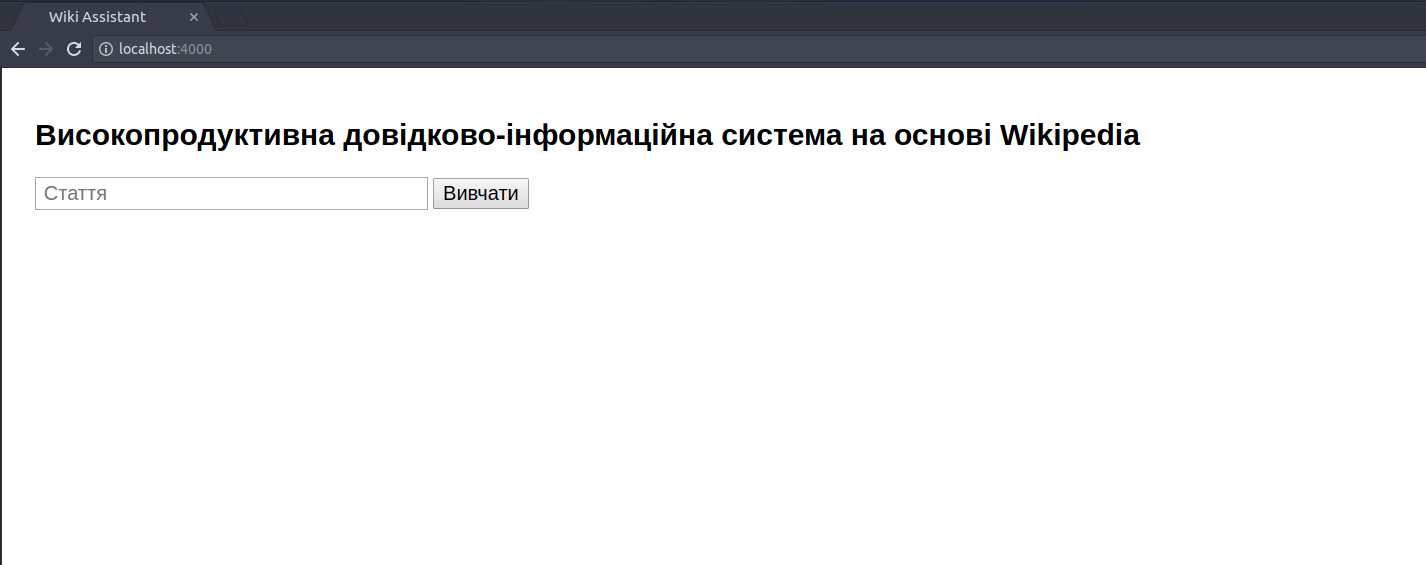


Рисунок 5.1 — Початковий стан інтерфейсу

Користувачеві пропонується вказати назву статті, яку він прагне вивчати. Після натиснення кнопки “Вивчати” користувач отримує результати (рисунок 5.2), що складаються с таких елементів:

* назва статті, яка була запитана користувачем, та посилання на неї;
* лінійна послідовність посилань на статті, які слід вивчати до тієї статті, яка цікавить користувача (зліва);
* лінійна послідовність посилань на статті, які пропонується прочитати після цього (справа);
* перша п’ятірка понять у кожній колонці візуально виділена, як найбільш важливі (з найбільшим фактором впевненості);
* обидві послідовності відсортовано за фактором впевненості у зв’язку, а сам фактор впевненості вказано у дужках після посилання на відповідну йому статтю;
* кнопка “Вивчати” після кожного посилання, що виконує аналогічні дії показуючи дидактичні зв’язки для відповідної статті.

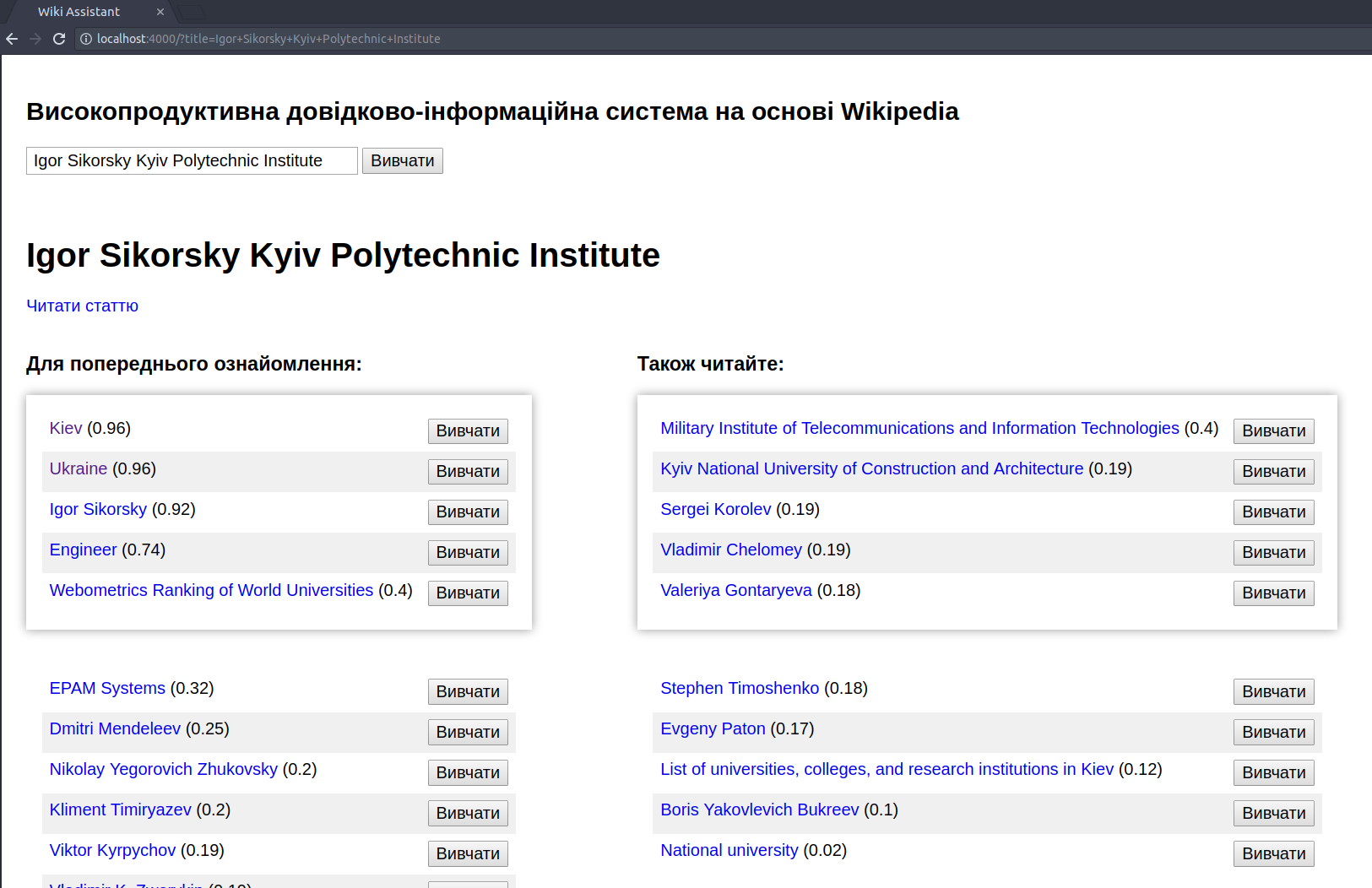


Рисунок 5.2 — Результати

**6. ЕКОНОМІКО-ОРГАНІЗАЦІЙНИЙ РОЗРАХУНОК**

## 6.1 Розрахунок трудомісткості розробки та впровадження програмного продукту

Трудомісткість розробки та впровадження програмного продукту (ПП) на тему “Високопродуктивна довідково-інформаційна система на основі Wikipedia визначається для таких стадій розробки:

технічне завдання (ТЗ);

ескізний проект (ЕП);

технічний проект (ТП**);**

робочий проект (РП);

впровадження (Вп).

Всі ці стадії мають місце тільки при розробці дуже великих і складних програмних продуктів, у переважній більшості випадків деякі стадії можуть бути відсутні. Наприклад, може бути відсутньою стадія ЕП, тоді трудомісткість цієї стадії (ТЕП) враховується в трудомісткості ТП (Т’ТП) (формула 6.1).

(6.1)

Т’ТП = ТЕП + ТТП

Стадії ТП і РП можуть об’єднуватися в техноробочий проект (ТРП), тоді його трудомісткість складає розраховується за формулою 6.2.

(6.2)

ТТРП = 0,85 ТТП + ТРП

Трудомісткість розробки програмного продукту розраховується на основі типових норм часу на програмування.

На стадіях ТЗ та ЕП трудомісткість в людино-годинах визначається залежно від типу завдання і ступеню новизни.

Вхідні дані для економіко-організаційного розрахунку приведені в таблиці 6.1

Таблиця 6.1 — Вхідні дані.

|  |  |
| --- | --- |
| Назва | Значення |
| Кількість макетів (наборів даних) вхідної інформації (банк даних) | 2 |
| Кількість різновидів форм вихідної інформації | 3 |
| Ступінь новизни групи завдань | «Б» |
| Складність алгоритму | 1 |
| Складність організації контролю вхідної і початкової інформації, яка характеризується такими групами | 12/22 |
| Мова програмування | Elixir |
| Використання стандартних модулів | 25% |
| Програмний продукт | стандартний |
| Запланований термін розробки | 0,5 року |

Для економічних завдань на стадіях “Технічний проект”, “Робочий проект” і “Впровадження” трудомісткість може бути розрахована залежно від кількості різноманітних форм вхідної і початкової інформації за формулою 6.3.

*Tр = a\*kb\*lc,*  (6.3)

де k – кількість макетів вхідної інформації;

l – кількість різноманітних форм вихідної інформації;

а, b, с – коефіцієнти.

При використанні інформації різних видів розраховується поправочний коефіцієнт (формула 6.4).

*,*  (6.4)

де – поправочні коефіцієнти;

m – кількість наборів даних НДІ.

Далі наведено значення поправочних коефіцієнтів для технічного і робочого проектів, враховуючи, що ступінь новизни – “Б” та група складності 1.

Значення Кп для стадії “Технічний проект”:

Кп = =3,12

Розрахунок Кп для стадії “Робочий проект”:

Кп = = 0,81.

Поправковий коефіцієнт, який враховує складність контролю вхідної і вихідної інформації: Kск  = 1.

При використанні мов програмування низького рівня, норми часу для стадії “Робочий проект” потрібно скоригувати з урахуванням коефіцієнта Км. У нашому випадку Км = 1.

Коли при розробці ПП використовуються стандартні модулі і (або) пакети прикладних програм (ППП), типові програми, норми часу корегують за допомогою коефіцієнта Кст, значення якого залежить від процентного відношення використаних пакетів і програм, і для стадій РП і ВП приведено у таблиці 6.2.

Таблиця 6.2 – Поправочні коефіцієнти при використанні типових проектних рішень, типових програм і стандартних модулів на стадіях “Робочий проект” і “Впровадження”.

|  |  |
| --- | --- |
| Ступінь використання ППП, типових програм, стандартних модулів | Кст |
| 60% і вище | 0,5 |
| 40...60% | 0,6 |
| 25...40% | 0,7 |
| 20...25% | 0,8 |

У нашому випадку поправочний коефіцієнт Кст = 0,8 при використанні типових програм і стандартних модулів із ступенем використання ПП 25% на стадіях “Робочий проект”, “Впровадження”.

При розробці стандартного ПП норму часу слід коректувати за допомогою коефіцієнта Кст.п рівного 1,2.

Коефіцієнти для розрахунку трудомісткості розробки ПП на стадіях “Технічний проект”, “Робочий проект”, “Впровадження” приведені в   
таблиці 6.3.

Таблиця 6.3 – Коефіцієнти для розрахунку трудомісткості розробки ПП на стадіях “Технічний проект”, “Робочий проект”, “Впровадження”.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Комплекс завдань | Стадія розробки | Для розробника постановки задачі | | | Для розробника програмного забезпечення | | |
| a | b | c | a | b | c |
| Управління науково-технічною інформацією | ТП | 20.13 | 0.41 | 0.32 | 7.08 | 0.57 | 0.18 |
| РП | 7.36 | 0.43 | 0.44 | 37.9 | 0.37 | 0.38 |
| Вп | 4.90 | 0.46 | 0.37 | 7.12 | 0.40 | 0,48 |

Тр(ТП)= 20,13 \* 20,41 \* 30,32 + 7,08 \* 20,57 \* 30,18 = 50,82 люд./дні.

Тр(РП)= 7,36 \* 20,43 \* 30,44 + 37,9 \* 20,37 \* 30,38 = 90,44 люд./дні.

Тр(Вп)= 4,90\*20,46 \* 30,37 + 7,12 \* 20,40 \* 30,48 = 26,04 люд./дні.

Загальна трудомісткість програмування завдань визначається за формулою 6.5:

*Тз = Тр \* Кn \* Кск \* Км \*Кст \* Кст.п* (6.5)

Тз (ТЗ) = 20 \* 1,2 = 24 люд./дні;

Тз (ЕП) = 25 \* 1,2 = 30 люд./дні;

Тз (ТП) = 50,82 \* 3,12 \* 1 \* 1,2 =190,27 люд./дні;

Тз (РП) = 90,44 \* 0,81 \* 1 \* 1 \* 0.8 \* 1,2 = 70,33 люд./дні;

Тз (Вп) = 26,04 \* 1 \* 0,8 \* 1,2 = 24,99 люд./дні;

Тз = 24 + 30 + 190,27 + 70,33 + 24,99 = 339,59 люд./дні.

ТТРП = (ТЕП + ТТП) \* 0,85 + ТРП = (30 + 190,27) \* 0,85 + 70,33 = 257,56 люд./дні.

Кореговане значення Тз:

Тз = Тз(ТЗ) + ТТРП + Тз(Вп) = 24 + 257,56 + 24,99 = 306.55 люд./дні.

Кількість людей, які беруть участь в розробці, розраховується за формулою 6.6:

, (6.6)

де R – кількість людей;

Тз – загальна трудомісткість (людино-дня);

n – кількість робочих днів в одному році, n = 228;

T – період розробки, в роках, Т = 0,5.

R = 306.55 / (228 \* 0,5) = 3 (людини).

Це керівник роботи, програміст та інженер-впровадник.

У таблиці 6.4 приведений розрахунок трудомісткості розробки ПП групи завдань “Планування і контроль виконання робіт”.

Таблиця 6.4 – Розрахунок трудомісткості розробки ПП для всіх стадій.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Найменування | Стадії розробки | | | | | | | | | |  | |
| ТЗ | ЕП | ТП | | РП | | Вп | | | Всього | |
| Постановка завдання | Розробка програм | Постановка завдання | Розробка програм | Постановка завдання | | Розробка програм |
| Трудомісткість, людино-дни | 20 | 25 | 50,82 | | 90.44 | | 26.04 | | | 212,3 | |
| в т.ч.  постановка задачі | 5 | 3 | 4.42 | - | 5.7 | - | 8,14 | | - | - | |
| Розробка програм | 15 | 22 | - | 46.4 | - | 84.74 | - | | 17.9 | - | |
| Поправочні коефіцієнти на:  Види інформації Кп | - | - | 3.12 | | 0,81 | | - | | | - | |
| Складність контролю інформації Кск | - | - | 1 | | 1 | | 1 | | | - | |
| Мова програмування Км. | - | - | - | | 1 | | - | | | - | |
| Використання стандартних модулів Кст | - | - | - | | 0,8 | | 0,8 | | | - | |
| Розробку стандартних ПП Кст.п | 1,2 | 1,2 | 1,2 | | 1,2 | | 1,2 | | | - | |
| Скоригована трудомісткість | 24 | 30 | 190,27 | | 70,33 | | 24,99 | | | 339,59 | |
| Трудомісткість з урахуванням об’єднання стадій розробки | 24 | 257,56 | | | | | | 24,99 | | | 306,55 | |
| Кількість працівників | 1 | 2 | | | | | | 1 | | | 3 | |
| Тривалість розробки, років | 0,08 | 0,35 | | | | | | 0,07 | | | 0,5 | |

## 6.2 Кошторис витрат на розробку та впровадження програмного продукту

Кошторис витрат розробляється виконавцем робіт на основі нормативів трудомісткості розробки і впровадження програмного продукту і затверджується замовником робіт або органом, який забезпечує фінансування робіт. Витрати, які включаються в собівартість ПП, групуються відповідно до їх економічного змісту за такими статтями:

* витрати на оплату праці;
* відрахування на соціальні заходи;
* витрати на спеціальне устаткування;
* витрати на службові відрядження;

Проведемо розрахунок Кошторису витрат за вищезазначеними статтями.

Витрати на оплату праці. До цієї статті витрат належать витрати на виплату основної і додаткової заробітної плати виконавців, обчислені згідно системам оплати праці, які прийняті в організації, включаючи всі види матеріальних і грошових доплат.

Основна заробітна плата розраховується на основі даних про трудомісткість робіт, і посадових окладів основних виконавців. Інформацію про трудомісткість окремих стадій знаходиться у таблиці 6.5

Таблиця 6.5 – Трудомісткість виконання робіт.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Стадія | Трудомісткість люд/дні | | | Всього, люд/дні |
| Керівник роботи | Програміст | Інженер- впровадник |  |
| ТЗ | 24 | - | - | 24 |
| ТРП | 28 | 230 | - | 258 |
| Вн | - | - | 25 | 25 |
| Всього | 52 | 230 | 25 | 307 |

Заробітну плату визначають, виходячи з місячних окладів, враховуючи тривалість умовного місяця (21.1 – при 5-денному робочому тижні). Результати розрахунків основної заробітної плати виконавців знаходиться в таблиці 6.6.

Таблиця 6.6 – Основна заробітна плата виконавців.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Посада | Місячна ставка, грн | Денна зар. плата, грн | Трудомісткість, люд-дні | Основна зар. плата, грн |
| Керівник роботи | 15000 | 710,90 | 52 | 36966,8 |
| Програміст | 12000 | 568,72 | 230 | 130805,6 |
| Інженер-впровадник | 9000 | 426,54 | 25 | 10663,5 |
| Всього | | | 307 | 376492,64 |

Основна заробітна плата = 376492,64 грн.

Додаткова заробітна плата (оплата відпусток, премії, одноразові заохочення і тому подібне) розраховується згідно нормативу, який встановлює підприємство і який складає в нашому випадку = 20%.

Додаткова заробітна плата = 0,2 \* 376492,64 = 75298,53 грн.

Сума основної і додаткової заробітної плати складає витрати за статтею “Заробітна плата” або фонд оплати праці.

Фонд оплати праці = 376492,64 + 75298,53 = 451791,17 грн.

Відрахування на соціальні заходи. До цієї статті належать витрати, здійснювані в порядку і розмірах, передбачених законодавством України за 2017 рік:

* на обов’язкове державне пенсійне страхування – 32%;
* на обов’язкове соціальне страхування – 4%;
* на страхування на випадок безробіття – 1,5%.

Всього страхування – 37,5% від фонду оплати праці.

Пенсійне страхування = 0,32 \* 451791,17 = 144573,18 грн.

Соціальне страхування = 0,04 \* 451791,17 = 18071,65 грн.

На випадок безробіття = 0,015 \* 451791,17 = 6776,87 грн.

Всього страхування = 0,375 \* 451791,17 = 169421,69 грн.

Матеріальні витрати. До цієї статті належать витрати на папір, канцелярське приладдя, картриджі і інші витратні матеріали. Ці витрати в середньому складають 3% від основної заробітної плати.

Матеріальні витрати = 0,03 \* 376492,64 = 11294,78 грн.

Витрати на спеціальне устаткування та на службові відрядження не передбачені.

Експериментально-виробничі витрати. До експериментально виробничих витрат відносять оплату машинного часу, пов’язаного з підготовкою і налагодженням програм. Витрати розраховуються, виходячи з кількості годинника машинного часу, необхідного для виконання потрібного об'єму обчислюваних робіт по темі і вартості однієї машинної години.

Кількість годин машинного часу для економічних завдань розраховується за формулою 6.7.

*Тр = a \* kb \* lc*, (6.7)

де k – кількість макетів вхідної інформації;

l - кількість різноманітних форм вихідної інформації;

а, b, с – коефіцієнти.

k = 2; l = 3; а = 10,69; b = 0,42; с = 0,55.

Тр = 10,69 \* 2 0,42 \* 3 0,55 = 26,18 годин.

Результати розрахунків приведені в таблиці 6.7.

Таблиця 6.7 – Витрати на оплату машинного часу.

|  |  |  |  |
| --- | --- | --- | --- |
| Роботи, які виконуються на ЕОМ | Тривалість виконання робіт, години | Вартість однієї машино-години, грн. | Сума витрат, грн. |
| Написання програмного коду | 23,18 | 4,8 | 111,26 |
| Тестування програми | 3 | 4,8 | 14,4 |
| Разом | | | 125,66 |

Накладні витрати. Витрати на цією статтею охоплюють витрати на оплату праці управління персоналу з нарахуваннями, оплату службових відряджень, консультаційно-інформаційних витрат, ремонт і техобслуговування інших основних фондів, окрім ПК, оренду приміщення і ін. Ці витрати розраховуються за нормативом, встановленим на підприємстві у відсотках до заробітної плати, які можуть представляти від 50 до 100% (для КПІ – 67%), або 20% від статті 11 “Повна вартість роботи, виконаної власними силами”.

У нашому випадку витрати по цій статті не передбачені.

Прибуток. Прибуток визначається у відсотках від суми витрат. Прибуток складає 10% від суми витрат (формула 6.8).

*Св = (Фзп + Свсего + Ззмат + Зекс)* (6.8)

Св = (451791,17 + 169421,69 +11294,78 +125,66) = 632633,3 грн.

Прибуток=0,1 \* Св = 0,1 \* 632633,3 = 63263,33 грн.

Податок на прибуток (30% від прибутку):

Податок на прибуток = 0,3 \* 63263,33 = 18979 грн.

Загальні витрати. Загальні витрати обчислюються як сума витрат і прибутку (формула 6.9).

*З = Пр + Под + Св* (6.9)

З = 632633,3 +63263,33 +18979 = 714875,63 грн.

Податок на додану вартість (ПДВ). ПДВ обчислюється у розмірі 20% від загальних витрат (формула 6.10).

*ПДВ = З \* 0.2* (6.10)

ПДВ = 0,2 \* 714875,63 = 142975,13 грн.

Повна вартість роботи, виконаної власними силами. Повна вартість роботи обчислюється як сума загальних витрат і ПДВ (формула 6.11).

*Сума загальних витрат = ПДВ + З* (6.11)

Сума загальних витрат = 714875,63+ 142975,13 = 857850,76 грн.

Загальні підсумки витрат зводяться в Кошторис вартості роботи і приведені в таблиці 6.8.

Таблиця 6.8 – Кошторис вартості роботи.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(вигляд, тема роботи і номер реєстрації)

Джерело фінансування\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Замовник\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Співвиконавці\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Термін виконання робіт: почало\_\_\_\_\_\_\_\_\_\_, закінчення\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Стаття витрат | Норматив | | Сума  грн | | Питома вага статті %% |
| 1.Заробітна плата, всього  зокрема:  основна  додаткова |  | | 451791,17 | | 52,67 |
| 376492,64 | | 43,89 |
| 75298,53 | | 8,78 |
| 2.Відрахування на соц. заходи  зокрема:  на обов’язкове державне пенсійне страхування  на обов’язкове соціальне страхування  на страхування на випадок безробіття | 37.5% | | 169421,69 | | 19,75 |
| 32,0% | | 144573,18 | | 16,86 |
| 4,0% | | 18071,65 | | 2,11 |
| 1,5% | | 6776,87 | | 0,79 |
| 3. Матеріали |  | | 11294,78 | | 1,32 |
| 4. Спеціальне устаткування | Не передбачені | | | | |
| 5. Відрядження | Не передбачені | | | | |
| 6. Експериментально-виробничі витрати |  | 125,66 | | 0,02 | |
| 7. Накладні витрати | Не передбачені | | | | |
| 7а. Сума витрат |  | | 632633,3 | | 73,75 |
| 8. Прибуток  зокрема податок на прибуток | 10% від п.7а | | 63263,33 | | 7,38 |
| 30% від п.8 | | 18979 | | 2,22 |
| 9. Загальні витрати |  | | 714875,63 | | 83,34 |
| 10. ПДВ | 20% від п.9 | | 142975,13 | | 16,67 |
| 11. Повна вартість роботи, виконаної власними силами |  | | 857850,76 | | 100 |
| 12. Договірна ціна |  | | 857850,76 | | 100 |

Дата складання Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

кошторису Економіст\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 6.3 Визначення економічного ефекту від застосування програмного продукту

Річний економічний ефект від використання ПП розраховується за формулою 6.12.

*ЕР = Е - ЕН \* К*,(6.12)

де ЕР – річна економія від функціонування ПП;

ЕН – нормативний коефіцієнт ефективності капіталовкладень (для програмного забезпечення може складати 0,33);

К – об’єм інвестицій, пов’язаних із створенням і впровадженням ПП.

Об’єм інвестицій складається з витрат на розробку і впровадження ПП і вартості придбання і монтажу технічних засобів, необхідних для використання ПП.

Економія витрат праці на рішення завдань із застосуванням ПП розраховується за формулою 6.13.

, (6.13)

де m – кількість типів завдань, які вирішує даний ПП;

C1i і С2i – витрати на рішення і-того типу завдань без ПП і із застосуванням ПП;

Qi – кількість завдань і-го типу, які розв’язуються протягом року.

У загальному випадку значення C1i і С2i складаються з витрат на оплату праці користувачів ПП і на оплату машинного часу (формула 6.14).

|  |  |
| --- | --- |
| *Сji =[Tji \*cji\*(1+нд/100)\*(1+нвід/100)\*(1+ннв)] + +Тмчji\*cмч]\*(1+нПДВ/100),* | (6.14) |

де Tji – витрати часу на вирішення завдань без ПП (j=1) і із застосуванням ПП (j=2), годин;

cji – погодинна заробітна плата фахівця, який вирішує задачу і-го типу без ПП (j=1) і з ПП (j=2), залежно від складності завдання і кваліфікації;

Нд, Нвід, Ннв, Нпдв – нормативи відповідно додаткової зарплати, відрахувань на соціальні заходи. Накладних витрат і нормативу ПДВ;

Нд=30%;

Нвід=37,5%;

Ннв=67%;

Нпдв=20%;

Tмчji – витрати машинного часу на вирішення завдань без ПП (j=1) і з ПП (j=2), годин;

Смч – вартість одної годинни роботи ЕОМ.

У таблиці 6.9 приведені розрахунки економічного ефекту (розрахунок ведеться 5-денним робочим тижнем з 8-годинним робочим день).

При розрахунках використані такі дані: 21.1 робочих днів в місяць, 8 годинний робочий день. Вартість однієї години роботи на ЕОМ = 4,8 грн.

Даний програмний продукт передбачений для побудови дидактичних зв’зяків між статтями онлайн-енциклопедії, тобто вирішує лише одну задачу.

Таблиця 6.9 – Вхідні дані для розрахунку економічного ефекту.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Час на розвиток задачі | | Витрати машинного часу | | Кількість виконавців | | Заробітна плата | | Кількістьзадач в рік |
| без ПП | з ПП | без ПП | з ПП | без ПП | з ПП | без ПП | з ПП |
| 40 | 12 | 40 | 12 | 3 | 1 | 53,3 | 53,3 | 20 |

Розрахунок за формулою 6.14:

С11 = (((40 \* 3 \* 53,3 \* (1 + 0,3) \* (1 + 0,375) \* (1 + 0,67)) + 40 \* 4,8) \* (1 + 0,2)= = (19092,85 + 192) \* 1,2= 23141,82 грн/година.

С12 = (((12 \* 1 \* 53,3 \* (1 + 0,3) \* (1 + 0,375) \* (1 + 0,67)) + 12 \* 4,8) \* (1 + 0,2) = = (1909,28 + 57,6) \* 1,2= 2360,25 грн/година.

Економія витрат за формулою 6.13:

Е = (23141,82 - 2360,25) \* 20 = 415631,28грн/рік.

К = 857850,76 грн.

Річний економічний ефект рівний за формулою 6.12:

Ер= 415631,28 - 0,33 \* 857850,76 = 132540,53 грн.

## 6.4 Визначення ціни розробки

Серед різних методів ціноутворення на ранніх стадіях проектування ПП достатньо поширений метод лімітних цін. При цьому визначаються нижня і верхня межа ціни.

Нижня межа ціни. Захищає інтереси розробника ПП і передбачає, що ціна повинна покрити витрати розробника, пов'язані з розробкою і впровадженням ПП, і забезпечити прийнятний рівень рентабельності, не нижче за те, що він має при впровадженні вже розроблених ПП .

Нижня межа ціни розраховується за формулою 6.15.

*ЦНМ = Сп\*(1 + рн/100)\*(1 +нПДВ/100),* (6.15)

де Сп – повна собівартість розробки ПП;

Нпдв - норматив ПДВ, 20%;

рн - нормативний рівень рентабельності, 5%.

Цнм = 857850,76 \* (1+0,05) \* (1+0,2) = 1080891,96 грн.

Верхня межа ціни. Захищає інтереси споживача і визначається тією ціною, яку споживач згоден заплатити за продукцію з кращою для нього якістю. Верхня межа може бути розрахований, виходячи з рівня якості ПП (Кяк) або на базі економічного ефекту від впровадження нового ПП.

Верхня межа ціни розраховується за формулою 6.16.

*ЦВМ = Сп + Kе \* Е \*Т,* (6.16)

де Сп – повна собівартість розробки та впровадження ПП, грн;

Ке – доля економічного ефекту, яка залишається в розпорядженні розробника (найчастіше – 0,3);

Е – очікуваний економічний ефект від застосування ПП;

Т – термін застосування програмного продукту, роки.

Цвм = 857850,76 + 0,3 \* 132540,53 \* 5= 1056661,56 грн.

## 6.5 Техніко-економічне обґрунтування розробки програмного продукту на основі функціонально вартісного аналізу

Вихідні положення. Формулюються задачі вдосконалення ПП (в процесі його модернізації) або вимоги, яким повинен відповідати новий ПП. Визначається ціль ФВА відносно цього ПП. Якщо ПП, що досліджується, є частиною складної системи, то необхідно привести структурну модель системи і визначити місце і значення елемента, який буде досліджуватися, відносно системи. Якщо ПП, що досліджується, є складною системою, то для проведення ФВА можна взяти більш простий елемент (модуль) системи. При цьому необхідно в роботі привести структурну модель всієї системи і обґрунтувати вибір даного елемента як об’єкта дослідження.

В основі ФВА лежить функціональний підхід, згідно з яким об’єктом аналізу є не сам ПП, а функції, які він виконує. ФВА проводиться в два етапи:

* функціональний аналіз;
* вартісний аналіз.

Обґрунтування функцій об’єкту. На основі вивчення організаційно-економічної суті об’єкту, досліджується (в нашому випадку програмний продукт), обґрунтовуються основні функції, які буде реалізовувати об’єкт (наприклад обробка інформації, зберігання інформації та інше). При цьому будують функціональну модель об’єкту (рисунок 6.1).

**Головна функція**

**F0** – Високопродуктивна побудова дидактичних зв’язків

**Основні функції**

**F1** – Завантаження даних вікі-енциклопедії

**F2** – Аналіз даних

**F3** –Вибірка множини зв’язаних с цільовим понять

**F4** – Пошук фактів та побудова зв’язків

Рисунок 6.1. — Функціональна модель

Головна функція реалізує ціль розробки. Основні функції – ті, заради яких об’єкт створюється. Кожна з основних функцій може мати декілька варіантів реалізації. Функції і їх варіанти повинні бути чітко і повно описані. Вони використовуються для опису морфологічної карти (рис. 6.2)

На основі цієї карти виконують якісну оцінку варіантів. Для обмеження кількості варіантів, що аналізуються, будують позитивно-негативну матрицю, в якій оцінюють переваги і недоліки варіантів рішень. Результати аналізу наведено у таблиці 6.10.

За даними таблиці 6.10 проводиться порівняльний аналіз всіх можливих варіантів реалізації функцій ПП. Варіанти, які мають суттєві недоліки, не відповідають з різних причин умовам технічного завдання, виключаються з подальшого розгляду. А з варіантами, які залишаються, буде проводитись технічна і економічна порівняльна оцінка.

**Функції**

F1

F2

F3

F4

**Варіанти реалізації**

**а)** Повне завантаже-ння як окремий етап

**б)** Заванта-ження частинами як окремий етап

**в)** Завантаження у режимі реального часу

**а)** Паралельний аналіз даних під час завантаження

**а)** Засобами сховища даних

**б)** Програмними засобами

**а)** У режимі реального часу

**б)** Як окремий етап

Рисунок 6.2. Морфологічна карта

Таблиця 6.10 – Позитивно-негативна матриця варіантів реалізації функцій.

|  |  |  |  |
| --- | --- | --- | --- |
| Основна функція | Варіанти реалізації | Переваги | Недоліки |
| F1 | а | Найоптимальніший спосіб | Найбільш складний у реалізації |
| б | Може базуватися на попередніх розробках | Недостатня швидкість |
| в | Легко реалізувати | Недостатня швидкість |
| F2 | а | Забезпечує високу швидкість виконання | Потребує потужне апаратне забезпечення |
| F3 | a | Найшвидше виконання | Потребує велику кількість оперативної пам’яті |
| б | Більш легко реалізувати | Більш низька продуктивність |
| F4 | a | Більш легко реалізувати | Дещо нижча продуктивність |
| б | Дещо вища продуктивність | Необходність виконати весь процес заздалегідь, що потребує багато часу |

Обґрунтування системи параметрів ПП і оцінка його рівня якості. Основні положення з вибору і обґрунтування системи параметрів ПП, які будуть використовуватися для розрахунків рівня якості виробу і сама методика проведення розрахунків рівня якості викладена.

Для характеристики ПП, який розробляється використовуються наступні параметри:

* X1 – об’єм пам’яті на жорсткому диску, який займає встановлений программний продукт, Кб;
* X2 – потреби в об'ємі оперативної пам’яті, який необхідний для роботи програми, Мб;
* X3 – час, що витрачається на завантаження аналіз статті статті, с;
* X4 – час, що витрачається на побудову зв’язків, с;
* X5 – результат побудови, доля одиниці;
* X6 – можливість розширення, доля одиниці.

Коефіцієнт вагомості варіанта реалізації функції обчислюється за формулою 6.17.

, (6.17)

де *bі -* вагомість і-го параметра за результатами оцінок експертів, і обчислюється за формулою 6.18.

, (6.18)

де *аij* - коефіцєнти переваги, дані усіма експертами по і-му параметру.

Результати розрахунку вагомості параметрів наведені у таблиці 6.11.

Таблиця 6.11 – Вагомість параметрів.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Х1 | Х2 | Х3 | Х4 | Х5 | Х6 | bi | φі |
| Х1 | 1 | 1,5 | 0 | 0,5 | 0,5 | 0 | 3,5 | 0,127 |
| Х2 | 0,5 | 1 | 0,5 | 1 | 0,5 | 0 | 3,5 | 0,127 |
| Х3 | 0 | 1 | 1 | 0 | 0,5 | 0,5 | 3 | 0,109 |
| Х4 | 1,5 | 1 | 0 | 1 | 1,5 | 1,5 | 6,5 | 0,236 |
| Х5 | 1,5 | 1,5 | 1,5 | 0,5 | 1 | 1,5 | 7,5 | 0,273 |
| Х6 | 0 | 0 | 1,5 | 0,5 | 0,5 | 1 | 3,5 | 0,127 |
| Сума |  | | | | | | 27,5 | 1 |

Результати розрахунку показників технічного рівня варіантів виконання функцій наведені у таблиці 6.12.

Таблиця 6.12 – Показники технічного рівня варіантів виконання функцій.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Основ-на функ-ція | Варіант реалізації функції | Параметри, які приймають участь в реалізації | Абсолютне значення параметра | Оцінка параметра в балах, bi | Коефіцієнт вагомості. φі | Показник технічного рівня, Ктр (Fi) |
| F1 | а | Х1 | 6,75 | 7 | 0,127 | 0,889 |
| Х2 | 1,75 | 2 | 0,236 | 0,472 |
| б | Х1 | 9,67 | 10 | 0,127 | 1,270 |
| Х2 | 2,52 | 3 | 0,236 | 0,708 |
| в | Х1 | 11,23 | 11 | 0,127 | 1,397 |
| Х2 | 0,89 | 1 | 0,236 | 0,236 |
| F2 | а | Х3 | 8,78 | 7 | 0,127 | 0,889 |
| Х4 | 1,64 | 2 | 0,236 | 0,472 |
| F3 | а | Х4 | 2,03 | 2 | 0,273 | 0,546 |
| Х5 | 0,57 | 1 | 0,127 | 0,127 |
| б | Х4 | 1,57 | 2 | 0,273 | 0,546 |
| Х5 | 0,65 | 1 | 0,127 | 0,127 |
| F4 | а | Х6 | 11,18 | 11 | 0,236 | 2,596 |
| б | Х6 | 6,75 | 7 | 0,236 | 1,652 |

На основі порівняльного аналізу варіантів реалізації функцій за їх перевагами і недоліками і коефіцієнтів вагомості параметрів вибираємо наступні варіанти реалізації функцій:

1. F1б + F2а + F3а + F4а.
2. F1а + F2а + F3 + F4а.

Показник рівня якості k-того варіанта реалізації основних функцій виробу розраховується за формулою 6.19.

*Ктрк = Ктр(F1k) + Ктр(F2k) + ... + Ктр(Fzk)* (6.19)

де Ктр(F1k) – показник технічного рівня першої функцій k-того варіанту реалізації основних функцій виробу.

 = (1,270 +0,708)+(0,889+0,472)+( 0,546+0,127)+ 2,596 = 6,608.

 = (1,397+ 0,236)+(0,889+0,472)+( 0,546+0,127)+ 2,596 = 6,263.

Найкращим на етапі функціонального аналізу є варіант, якому відповідає найбільше значення коефіцієнта технічного рівня (формула 6.20).

(6.20)

де Kmpj – коефіцієнт технічного рівня j-того варіанта реалізації основних функцій виробу.

Як видно з розрахунків, кращим є перший варіант, у якого коефіцієнт технічного рівня має максимальне значення рівне 6,608.

Методика проведення вартісного аналізу. Аналіз функцій, які реалізуються програмним продуктом, повинен бути доповнений вартісним аналізом. Для цього розраховуються витрати, які необхідні для розробки ПП, тобто визначається функціонально-необхідна вартість виробу по всіх варіантах реалізації, які досліджуються. Якщо проводиться ФВА робіт, які мають науково-дослідний характер (наприклад, створення програмного продукту), то функціонально-необхідні витрати визначаються шляхом розрахунку кошторису витрат на проведення науково-дослідних робіт. Методика розрахунку кошторису витрат на НДДКР.

Далі наведено укрупнену методику визначення функціонально-необхідних витрат на розробку програмного продукту.

Функціонально необхідні витрати на створення ПП (Сф) визначаються за формулою 6.21.

*Сф = Сз + Свід+ См + Сн,* (6.21)

де Сз – оплата праці розробників, грн.;

Свід – відрахування на соціальні заходи (37,5% від фонду оплати праці), грн.;

См – вартість машинного часу, необхідного для розробки і налагодження ПП, грн.;

Сн – накладні витрати в розмірі 50-150% від витрат на оплату праці, грн.

Опис вхідних даних до формули 6.21:

* Сз – основна заробітна плата розробників, становить 451791,17 грн.;
* Свід – 37,5% від фонду оплати праці грн., становить 169421,69 грн.;
* См – вартість машинного часу становить 125,66 грн.;
* Сп – накладні витрати в розмірі 50-150% від витрат на оплату праці, грн., витрати за цією статтею не передбачені.

Розрахунок згідно формули 6.21:

= 451791,17 + 169421,69 + 125,66+ 0 = 621338.52 грн.

Вартісний аналіз варіантів реалізації функцій завершується визначенням коефіцієнта техніко-економічного рівня кожного варіанта (Kтерj), який розраховується за формулою 6.22.

, (6.22)

де Kтерj – коефіцієнт технічного рівня j-того варіанту;

Cфj – величина функціонально-необхідних витрат j-того варіанту, грн.

 = 9,535/8140,08= 1,2 • 10-3 ;

 = 9,027/8140,08= 1,1 • 10-3 ;

 = 8,827/8140,08= 1,0 • 10-4.

Найкращий варіант визначається за максимальним значенням коефіцієнта техніко-економічного рівня.

= 1,0 • 10-3 .

На заключному етапі проведення ФВА, якщо з’явиться можливість визначити базовий варіант, то бажано встановити область ефективного використання найбільш ефективного і базового варіантів. Для цього розраховують зведені витрати по базовому і новому варіантах (Зв) за формулами 6.23 і 6.24.

, (6.23)

, (6.24)

де ,  – річні зведені витрати відповідно по базовому і новому варіантах, грн.;

, – питомі поточні витрати на одиницю продукції відповідно по базовому і новому варіантах, грн.;

, – питомі капітальні витрати відповідно по базовому і новому варіантах, грн.;

 – витрати на розробку виробу, визначаються за кошторисом витрат, грн.;

 – нормативний коефіцієнт ефективності;

 – річні продуктивні можливості нового варіанту (наприклад, кількість розрахунків у рік певної задачі за допомогою ПП), шт.

Розрахунок за формулами 6.23 та 6.24:

 = (40 \* 3 \* 53,3 \* 1,3 \* 1,375 \* 1,67 + 40 \* 4,8) \* 1,2 = 23141,82 грн.;

= (12 \* 53,3 \* 1,3 \* 1,375 \* 1,67 + 12 \* 4,8) \* 1,2 = 2360,25 грн.;

= ;

= ;

= 632633,3 / 20 = 31631,67 грн.;

= 632633,3 / 20 = 31631,67 грн.;

 = 0,33;  = 0,33;

 = 632633,3 грн.;

 = 20 шт., = 20 шт.;

= (23141,82 + 0,33 \* 31631,67) \* 20 =   
= 671605,43 грн.;

= (2360,25 + 0,33 \* 31631,67) \* 20 + 0,33 \* 632633,3 = 464743,02 грн.

Основні техніко-економічні показники базового і нового варіантів зведені у таблиці 6.13.

Таблиця 6.13 – Основні техніко-економічні показники.

|  |  |  |  |
| --- | --- | --- | --- |
| Показник | Одиниці виміру | Варіанти | |
| Базовий | Новий |
| Питомі поточні витрати на одиницю продукції | Гривні | 23141,82 | 2360,25 |
| Питомі капітальні витрати | Гривні | 31631,67 | 31631,67 |
| Коефіцієнт ефективності | Безрозмірний | 0,33 | 0,33 |
| Річні продуктивні можливості | Штуки | 20 | 20 |
| Витрати на розробку виробу | Гривні | 632633,3 | 632633,3 |
| Річні зведені витрати | Гривні | 671605.43 | 464743.02 |

Область ефективного використання варіантів зображено на рисунку 6.3.



Рисунок 6.3 – Область ефективного використання

Отже з проведеного аналізу видно, що найкращим є варіант реалізації програмного продукту при завантаженні повного об’єму даних енциклопедії в окремому етапі (при цьому виконуючи паралельну обробку даних), вибірці множини зв’язаних з цільовою статей засобами сховища даних, а також пошук фактів і побудова зв’язків у режимі реального часу. Саме цей варіант і було реалізовано у даній роботі.

Як видно з рисунку 6.3 базовий варіант доцільно використовувати лише у випадках, коли задача використання виникає менше ніж 14 разів, інакше рекомендовано використовувати новий варіант програми.

# ВИСНОВКИ

У ході виконання даної роботи було розроблено високопродуктивну довідково-інформаційну систему на основі Wikipedia, що виконує побудову дидактичних зв’язків між статтями онлайн-енциклопедії на основі автоматичного аналізу їх тексту для полегшення процесу вивчення обраної користувачем тематики. Було виявлено та застосовано оптимальні для даної задачі засоби розробки, такі як мова програмування Elixir, що забезпечує високу швидкість виконання програмного коду і надає можливості паралельної обробки даних, та пошукова платформа Elasticsearch, яка використувується у якості локального сховища даних та забезпучає оптимальність процесів читання та запису данних. Запропоновано нову архітектуру системи, в якій процес завантаження даних с онлайн-енциклопедії відокремлюється в незалежний компонент. Оптимізовано параметри звернення до API енциклопедії для найбільшої швидкості процесу завантаження. Використано паралельний аналіз даних під час завантаження за допомогою можливостей паралелізму віртуальної машини Erlang. Дані, що завантажуються з енциклопедії, збережено у найбільш вигідній для побудови дидактичних зв’язків структурі. Розроблено метод пошуку множини зв’язаних з цільовою статей, завдяки якому ця задача виконується засобами сховища даних та максимально швидко. Використано паралельну обробку даних під час пошуку фактів та побудови дидактичних зв’язків. Виконано повну обробку англомовної Вікіпедії за прийнятний час. Забезпечено високу швидкість побудови дидактичних зв’язків за наявності потужного апаратного забезпечення.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Jimmy Wales on 60 Minutes. CBS News. [Електронний ресурс] —   
   Режим доступу: http://www.cbsnews.com/news/wikipedia-jimmy-wales-morley-safer-60-minutes.
2. Cohen, Noam (February 9, 2014). Wikipedia vs. the Small Screen. The New York Times. [Електронний ресурс] — Режим доступу: https://www.nytimes.com/2014 /02/10/technology/wikipedia-vs-the-small-screen.html.
3. Ковальов Е. А. Система аналізу онлайн-енциклопедії для генерації дидактичних рекомендацій. / Е. А. Ковальов, С. В. Титенко // Сучасні проблеми наукового забезпечення енергетики: Матеріали ХIV Міжнародної науково-практичної конференції аспірантів, магістрантів і студентів, присвяченої 85 річчю теплоенергетичного факультету, м. Київ, 18–21 квітня 2016 р. У 2 т. – К. : НТУУ «КПІ», 2016. – Т. 2. – С. 190.
4. Brusilovsky P., Yudelson M., Sosnovsky S. An adaptive E-learning service for accessing interactive examples. In: J. Nall and R. Robson (eds.) Proceedings of World Conference on E-Learning, E-Learn 2004, Washington, DC, USA, November 1-5, 2004. — AACE, 2004. — P. 2556-2561.
5. Adams, Douglas (2003). Geoffrey Perkins (ed.), additional Material by M. J. Simpson., ed. The Hitchhiker's Guide to the Galaxy: The Original Radio Scripts (25th Anniversary ed.). Pan Books. ISBN 0-330-41957-9..
6. Reagle, Joseph Michael (2010). «Interpedia». Good Faith Collaboration: The Culture of Wikipedia. Cambridge, Mass.: MIT Press. pp. 32-34. ISBN 978-0-262-01447-2.
7. Википедия — что это такое, ее история, wiki-движки, разметка и создание статей в народной энциклопеции Wikipedia. [Електронний ресурс] — Режим доступу: http://ktonanovenkogo.ru/seo/smo/vikipediya-narodnaya-enciklopediya-wiki-dvizhki-viki-razmetka-statej-wikipedia.html.
8. Интернет-энциклопедии. [Електронний ресурс] — Режим доступу: http://www.encyclopedia.ru/news/enc/detail/3917/.
9. История создания Википедии. [Електронний ресурс] — Режим доступу: http://www.seoded.ru/istoriya/internet-history/wikipedia.html.
10. Популярные wiki-проекты на русском языке. Часть 1. Русская Википедия. [Електронний ресурс] — Режим доступу: http://xbb.uz/web/Populjarnyje-wiki-projekty-na-russkom-jazyke-Chast-1-Russkaja-Vikipedija.
11. Шестаков В. К. Извлечение онтологии из Wiki-системы [Електронний ресурс] / В. К. Шестаков – Режим доступу: http://ntv.ifmo.ru/file/article/799.pdf.
12. Portland Pattern Repository [Електронний ресурс] — Режим доступу: http://c2.com/ppr.
13. Ward Cunningham. Correspondence on the Etymology of Wiki [Електронний ресурс] / Ward Cunningham. — 20 03. — Режим доступу: http://c2.com/doc/etymology.html.
14. Leuf B. The Wiki way: quick collaboration on the Web / B. Leuf, W. Cunningham. — Boston, MA: Addison-Wesley Longman Publishing Co., 2001.
15. Ahrens F. Death by Wikipedia: The Kenneth Lay Chronicles [Електронний ресурс] / Frank Ahrens // The Washington Post. — 2006. — Режим доступу: http://washingtonpost.com/wp-yn/content/article/2006/07/08/AR2006070800135.html.
16. Стеценко Г.В. Практичне використання вікі-енциклопедії в навчально-виховному процесі // Комп’ютер в школі та сім’ї, 2009 — № 5. — С.34-39.
17. Стеценко Г.В. Педагогічний потенціал вікі-енциклопедії та її використання в навчально-виховному процесі. // Наукові записки Тернопільського національного педагогічного університету імені Володимира Гнатюка. Серія: Педагогіка. — 2008. — № 7. — С. 53-57.
18. Бєлєнька Г. Використання wiki-технологій у сучасному дошкільному навчальному закладі [Електронний ресурс] / Г. Бєлєнька, Н. Кудикіна, Л. Дерев’янко — Режим доступу: http://www.psyh.kiev.ua/Бєлєнька\_Г.,\_Кудикіна\_  
    Н.,\_Дерев’янко\_Л.\_Використання\_wiki-технологій\_у\_сучасному\_дошкільному\_  
    навчальному\_закладі.
19. Elizabeth L. Collaborative exam creation [Електронний ресурс] / Lawley Elizabeth — Режим доступу: http://mamamusings.net/archives/2007/01/24/  
    collaborative\_exam\_creation.php.
20. Koren Y. Matrix Factorization Techniques for Recommender Systems / Y. Koren, R. Bell, C. Volinsky. // IEEE Computer Society. — 2009. — № 8. — P. 42–49.
21. Melville P. Content-Boosted Collaborative Filtering for Improved Recommendations [Електронний ресурс] / P. Melville, R. Mooney, R. Nagarajan — Режим доступу: http://www.aaai.org/Papers/AAAI/2002/AAAI02-029.pdf.
22. Hossain N. Why the Interest Graph Is a Marketer’s Best Friend [Електронний ресурс] / Nadim Hossain — Режим доступу: http://mashable.com/2012/06/19/interest-graph-marketer.
23. Левченко Я. А. Автоматизована побудова дидактичної онтології на основі Wikipedia / Я. А. Левченко, С. В. Титенко // Международная научная конференция имени Т.А. Таран “Интеллектуальный анализ информации” ИАИ-2015, Киев, 20–22 мая 2015 г.: cб. тр. — К.: Просвіта, 2015. — 131-137 с.
24. Dave Thomas. Programming Elixir. — Pragmatic Bookshelf, 2014. — 280 p. — ISBN 978-1-937785-58-1.
25. Chris McCord. Metaprogramming Elixir. — Pragmatic Bookshelf, 2015. — 120 p. — ISBN 978-1-68050-041-7.
26. Simon St. Laurent, J. David Eisenberg. Introducing Elixir. — O'Reilly Media, Inc., 2014. — 210 p. — ISBN 978-1-4493-6999-6.
27. Чезарини Ф., Томпсон С. Программирование в Erlang — М.: ДМК Пресс, 2012. — 488 с. — ISBN 978-5-94074-617-1.
28. Armstrong, Joe (January 2007). “A History of Erlang” in HOPL III. Proceedings of the Third ACM SIGPLAN Conference on History of Programming Languages: 6−1—6-26, San Diego, California: ACM. DOI:10.1145/1238844.1238850. 978-1-59593-766-7.
29. Bruce A. Tate. Seven Languages in Seven Weeks: A Pragmatic Guide to Learning Programming Languages. — Pragmatic Bookshelf, 2010. — ISBN 9781934356593.
30. Simon St. Laurent. Introducing Erlang. — O’Reilly Media, Inc., 2013. — 185 p. — ISBN 978-1-449-33176-4.
31. Elasticsearch: RESTful, Distributed Search & Analytics. [Електронний ресурс] — Режим доступу: <https://www.elastic.co/products/elasticsearch>.
32. DB-Engines Ranking — popularity ranking of search engines. [Електронний ресурс] — Режим доступу: <https://db-engines.com/en/ranking/search+engine>.
33. Getting Started. [Електронний ресурс] — Режим доступу: https://www.elastic.co/guide/en/elasticsearch/reference/current/getting-started.html.
34. Buchanan B. G. Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project / B. G. Buchanan, E. H. Shortliffe //. — MA: Addison-Wesley, 1984. — 769 p.
35. Титенко, С. В. Побудова дидактичної онтології на основі аналізу елементів понятійно-тезисної моделі / С. В. Титенко // Наукові вісті НТУУ “КПІ”. — 2010. — № 1(69). — С. 82-87.