НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

«На правах рукопису»
УДК _____

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____2022 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

зі спеціальності 123 «Комп'ютерна інженерія»

на тему: «Електронний словник підвищеної швидкодії на основі хеш-адресації без колізій»

Виконав (-ла):
студент (-ка) VI курсу, групи ІО-12мп
Гуменюк Інна Олександрівна _____

Науковий керівник:
доцент кафедри ОТ, к.т.н.,
Марковський Олександр Петрович _____

Консультант з нормоконтролю:
професор кафедри ОТ, д.т.н.,
Кулаков Юрій Олексійович _____

Рецензент:
професор кафедри ПЗКС, к.т.н.,
Дичка Іван Андрійович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2022 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Рівень вищої освіти – другий (магістерський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Гуменюк Інні Олександрівні

1. Тема дисертації «Електронний словник підвищеної швидкодії на основі хеш-адресації без колізій», науковий керівник дисертації Марковський Олександр Петрович, доцент каф. ОТ, к.т.н., затверджені наказом по університету від «09»листопада 2022 р. №5115-с

2. Термін подання студентом дисертації

3. Об'єкт дослідження: *процес пошуку в електронних словниках для систем комп'ютерного перекладу.*

4. Предмет дослідження: *методи організації швидкісного пошуку в електронних словниках на основі хеш-адресації.*

5. Перелік завдань, які потрібно розробити: *дослідити спосіб організації швидкодіючого електронного словника системи комп'ютерного перекладу.*

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: *рисунок прикладу здійснення 1-го етапу пошуку для запропоної організації електронного словника.*

7. Орієнтовний перелік публікацій:

Humeniuk I.O. Hash search organization in e-dictionaries using block ciphers/ I.O. Humeniuk. O.P. Markovskyi. O.M. Shevchenko // Information, Computing and Intelligent systems. —Kyiv. Ukraine. —2020. —C.34-42.

Humeniuk I.O. Method to improve the efficiency of electronic dictionaries with content search / I.O. Humeniuk. O.P. Markovskyi. O.M. Shevchenko // International Conference ICSFTI2019 Security, Fault Tolerance, Intelligence. —Kyiv. Ukraine. —2020. —C.152-159.

Гуменюк І.О. Організація пошуку в електронних словниках систем комп'ютерного перекладу / І.О. Гуменюк, О.П. Марковський, О.М. Шевченко // Альманах науки. —Kyiv. Ukraine. —2019. —С.40-43.

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	*проф., д.т.н., Кулаков Ю. О.*		

9. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	*Вивчення літератури*	*01.02.2022*	
2	*Складання і узгодження технічного завдання*	*01.04.2022*	
3	*Написання вступної частини та огляд рішень*	*22.05.2022*	
4	*Моделювання розробленого способу*	*20.09.2022*	
5	*Оформлення документації ДП*	*10.10.2022*	
6	*Попередній захист та проходження нормативного контролю*	*12.12.2022*	
7	*Подання ДП рецензенту*	*13.12.2022*	
8	*Захист*	*19.12.2022*	

Студент Інна ГУМЕНЮК

Науковий керівник Олександр МАРКОВСЬКИЙ

РЕФЕРАТ

на магістерську дисертацію

виконану на тему: Електронний словник підвищеної швидкодії на основі хеш-адресації без колізій

студентом: Гуменюк Інна Олександрівна

Робота складається із вступу та чотирьох розділів. Загальний обсяг роботи: 84 аркушів основного тексту, 1 ілюстрація, 10 таблиць. При підготовці використовувалася література з 61 джерела.

Актуальність. Сучасні системи комп'ютерного перекладу передбачають аналіз великої кількості даних задля отримання найбільш релевантного результату. Це в свою чергу ставить якісно нові вимоги до електронних словників що ними використовуються, а саме до їх швидкодії та об'єму контекстних даних що можуть ними зберігатись. Задача підвищення швидкості віднаходження релевантних результатів перекладу потребує значного прискорення доступу до потрібних контекстних даних в електронних словниках.

Мета і завдання дослідження. Метою магістерської роботи є прискорення пошуку за ключем в електронгому словнику за рахунок використання потенційно найбільш швидкодіючої технології пошуку —хеш-адресації без колізій.

Для досягнення мети дослідження поставлено і вирішено такі завдання:

- аналіз вимог до сучасних електронних словників, орієнтованих на використання в системах комп'ютерного перекладу;
- критичний огляд з позицій сучасних тенденцій структур та принципів побудови наявних електронних словників;
- визначення найбільш перспективних напрямків досліджень для підвищення швидкості пошуку в електронних словниках;

- теоретичне обґрунтування, розробка та дослідження організації швидкого контекстного пошуку в електронних словниках з використанням хеш-адресації без колізій;
- моделювання роботи запропонованого способу організації електронного словника та аналіз отриманих результатів **.**

Об'єкт дослідження – процес пошуку в електронних словниках для систем комп'ютерного перекладу.

Предмет дослідження – методи організації швидкісного пошуку в електронних словниках на основі хеш-адресації.

Методи досліджень. Для досягнення поставлених в магістерській роботі задач, використано методи теорії обчислень, теорії ймовірностей, статитичного аналізу, а також методи імітаційного моделювання.

Наукова новизна одержаних результатів роботи полягає у наступному:

- запропоновано спосіб організації пошуку в електронному словнику, який відрізняється використанням хеш-адресації без колізій для наперед заданого фіксованого набору слів;
- розроблено метод підбору хеш-перетворення, що не утворює колізій для заданого постійного масиву ключів, в основі якого покладено використання єдиного адресного простору для зберігання в словнику ключових слів та контекстних уточнень. В якості хеш-перетворення запропоновано використовувати стандартизований симетричний шифр, а ключ - як механізм налаштування хеш-перетворення;
- змодельовано запропонований спосіб з використанням підходів імітаційного моделювання для оптимізації параметрів та експериментальної оцінки ефективності.

Практична цінність отриманих результатів визначається тим, що застосування запропонованої організації електронних словників для комп'ютерного перекладу дозволяє значно прискорити процес пошуку і, тим самим, відкриває можливості аналізу більшої кількості варіантів перекладу, що в кінцевому підсумку забезпечує підвищення якості комп'ютерного перекладу.

Проведене дослідження дає можливість використання запропонованого способу організації електронного словника в системах комп’ютерного перекладу та обробки тексту.

Особистий внесок здобувача. Магістерське дослідження є самостійно виконаною роботою, в якій відображено особистий авторський підхід та особисто отримані теоретичні та прикладні результати, що відносяться до вирішення задачі прискорення електронних словників що можуть бути інтегравані в системи обробки тексту та системи комп’ютерного перекладу. Формулювання мети та завдань дослідження проводилось спільно з науковим керівником.

Апробація результатів дисертації. Основні результати магістерської дисертації доповідались та обговорювались на міжнародній науково-технічній конференції Intelligence: proceedings of the International Conference ICSFTI2020, Kyiv, Ukraine, May 21 2020. м. Київ.

Публікації. За темою дисертації опубліковано дві статті, одна з яких в фаховому виданні України. .

Ключові слова

Хеш-адресація, електронні словники, системи комп’ютерного перекладу, хеш-перетворення, симетричні шифри.

Пояснювальна записка

до магістерської дисертації

на тему: «Електронний словник підвищеної швидкодії на основі хеш-адресації без колізій»

Київ – 2022

ЗМІСТ

Стр.

Стр.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

МК	Мікроконтролер
AES	Симетричний алгоритм блочного шифрування (Advanced Encryption Standard)
DES	Симетричний алгоритм блочного шифрування (Data Encryption Standard)
MRD	Машиночитані словники (Machine-Readable Dictionaries)
ASCII	Таблиця кодування символів (American Standard Code for Information Interchange)
NMT	Нейроний машинний переклад (Neural Machine Translation)
МП	Машинний переклад
ПЗ	Програмне забезпечення
LMF	Lexical Markup Formal

CAT	Автоматизований переклад (Computer Assisted Translation)
TEI	Text Encoding Initiative

ВСТУП

За останні два десятиліття у світі відбувся процес перерозподілу домінуючих центрів виробництва та наукової діяльності. До останнього часу відбувалося інтенсивне перенесення матеріального виробництва в країни Далекого Сходу. Сьогодні ці країни є лідерами в багатьох галузях науки, особливо в сфері технологічних досліджень. Результатом таких процесів є об'єктивнна тенденція до нарощення обсягу науково-технічної інформації, якою обмінюються Схід і Захід. Однак у процесі збільшення потоку інформації однією з ключових перешкод є мовний бар’єр. Він зумовлений об’ктивно ю різницею мов що належать то різних мовних груп, зокрема як для східних та західних мов. Практика показала, що розв'язання проблем традиційним способом вивчення мови широкою групою фахівців галузі не дає бажаного ефекту. Все це підриває обмін інформацією між Сходом і рештою міжнародної спільноти.

З точки зору обміну науково-технічною інформацією, найбільш перспективним шляхом подолання мовних бар'єрів є використання більш досконалих та точних систем комп'ютерного інтелектуалізованого перекладу. Технологічні досягнення в розвитку комп'ютерних систем створюють важливі передумови для успішної реалізації цього підходу. Останні наукові розробки в галузі штучного інтелекту та машинного навчання відкривають можливість досягнення прийнятної семантичної валідності комп’ютерних перекладів для науково-технічних публікацій.

Варто зауважити що ці методі методи включають аналіз численних альтернативних варіантів і вимагають швидкого багаторазового доступу до електронних словників в межах пошуку найрелдевантніших опцій перекладу. Таким чином, використання сучасних і прогресивних

технологій комп’ютерного перекладу висуває на порядок вищі вимоги до швидкості та ефективності пошуку в електронних словниках. Задовільнення цих вимог потребує розробки нових методів контекстного пошуку, що враховують можливості наявних на сьогодні технологій. Тому, на сучасному етапі розвитку інформаційних технологій, наукова задача прискорення пошуку в електронних комп’ютерних словниках і комп’ютеризованих системах перекладу є актуальним і важливим.

РОЗДІЛ 1
ОГЛЯД ІСНУЮЧИХ ЕЛЕКТРОННИХ СЛОВНИКІВ ДЛЯ СИСТЕМ КОМП'ЮТЕРНОГО ПЕРЕКЛАДУ

Динамічний і невпинний розвиток комп'ютерних систем, їх швидка ітеграція в всі сфери життія та одночаний прогрес технологій штучного інтелекту і машинного навчання стали поштвхом до стрімкого розвитку високякісних систем комп'ютерного перекладу. Рівень отриманого перекладу за допомогою використання таких стситем мало чим поступається перекладу зробленого живими людьми — фахівцями-лінгвістами. Однак, з іншого боу всупереч наявності таких якісних передумов, прогрес сиситем високоякісного комп'ютерного перекладу потребує наявності електронних словників що забезпечували б достатню швидкодію. Вимоги які наразі висуваються до елктронних словників, зумовлені потребою подальшого розвитку систем комп'ютерного перекладую Це однозначно педалює розвиток електронних словниківі в якості найвищого пріорітету сиавиться швидкість пошуку в цих електронних словниках.

Враховуючи сьогоденний попит на підвищення якості систем комп'ютерного перекладу і їх повсякмістне використання, то задача організації електронних словників, що зможуть задовільняти потреби в швидкодії пошуку в мають високу практичну цінність. Значною мірою зібільшеня інтересу до цього питання викликане стрімкою глобалізацієї світу впродовж останніх десятиліть. Саме через це, попри наявність численної кількості електронних словників, орієнтованих для використання у складі

систем комп'ютерного перекладу, потреба їх вдосконалення лишається актуальною.

1.1 Аналіз сучасних вимог до контекстних електронних словників

Комп’ютерна лексикографія є відносно новою галузю, яка почала розвиватися в 1960-х роках. Цьому сприяло кілька факторів, зокрема серед них була потреба трансформувати словники у машиночитаний формат і вигляд, щоб з'явилась можливість створювати нові словники вже з використанням комп’ютерних технологій. Спочатку були створені електронні словники, які досконало повторювали свої паперові еквіваленти [1]. Іншими словами, йшов процес оцифрування наявний наробків лексикографії. Яскравим прикладом є Оксфордський словник англійської мови [2].

Проте час йшов. і незабаром почали з’являтися нові словники в електронному вигляді. На цьому етапі почалася розробка форматів для зберігання даних у словниках. що дозволяло більш ефективну обробку. Серед перших словників. випущених в електронному вигляді. можна виділити Collins Dictionary of English 1979 року [5]. Один із найпопулярніших словників сучасності. Оцифровування Longman Dictionary of Contemporary English [4] можна вважати початком комп’ютерної лексикографії.

Історично словники випускалися у двомовному форматі. Насамперед такі словники забезпечують збереження лексичних описів двох мов у формі, яка однозначно співставляє мовні структури однієї мови з іншою. Вихідні синтаксиси, які чітко відповідають призначенню, називаються лінгвістичними еквівалентами [6,7]. Тому електронні словники будуються за принципом зберігання даних лінгвістичної структури як пошукових ключів та їх мовних еквівалентів як пошукових даних [8]. Однак виявилося що цього обсягу інформації недостатньо для розробки якісного машинного перекладу. Словники, які побудовані за таким принципом, можуть використовуватися

лінгвістами, але не відповідають потребам машинного перекладу. Таким чином, електронні словники нового покоління мають дещо складнішу організацію [9].

Для словників нового покоління почали використовувати паралельні, тобто порівняльні, текстові корпуси. Паралельні корпуси означають, що приклади людського перекладу на іншу мову почали додаватися до мовних струкнтур що зберігаються словником [10]. Порівняння складаються з фрагментів її тексту двома мовами на певні теми. Головною перевагою використання паралельних і порівняльних корпусів є те що словники почали враховувати контекст. Це в свою чергу якісно сприяло підвищенню точності перекладу.

Обмежженння яке варто враховувати при використанні словників такого типу є те що побудова перекладного словника з використанням порівняльного корпусу має від початку передбачати обмеження в кількості паралельних корпусів, доступних для багатьох мовних пар. З іншого боку, це робить це завдання вдосконалення електронних словникі ще більш актуальним. Адже наразі підхід словників з паралельними корпусами вимагає більш складних методів для вилучення еквівалентів перекладу з таких випадків,де тотожний переклад виконатиє малореальним. В будь-якому разі всі наявні на сьогодні методі визначеннння паралельних корпусів використовують “попередній словник”, тобто словник, що складається з невеликого набору еквівалентних перекладів для заданої мовної пари. Переважно “попередні словники” зберігають інформацію дотичну до простих мовних конструкцій, на відміну від словників з паралельним корпусом. І длцього виду словників здійснється пошук «схожих» слів у вхідній мові та порівняльні переклади тексту.

Також пи обудові словників з паралельним корпусом часто використовують графові моделі [11]. Якщо вихідний словник невеликий. подібні дерева розбору використовуються для виразів різними мовами. Можливі різні підходи що залежать від ряда критерії, наприклад таких як використовуваної метрики подібності, побудови вектора контексту тощо [12]. Деякі автори пропонують використовувати непрямі зв'язки між словами для вдосконалення методів заснованих на порівнянні. Для споріднених мов пропонується метод, який використовує спеціальний початковий словник, що містить слова спільні для обох мов. Цей метод легко застосовується і даєконкурентні результати з мовами однієї мовної групи (слов'янської, германо-франкської, тощо). Згідно з цією ідеєю, спочатку вилучаються слова, що належать до однієї з тем із найбільш контактними та найавторитетнішими перекладами. Покращення словникового запасу також можна досягти шляхом додавання схожих слів і використання методів лексичного усунення неоднозначності. Однак створити такий словник для первинного аналізу проблематично [13].

Однак використання пералельних корпусів вносить багато технічних нюансів. Основним з них є розташування відповідностей у паралельному корпусі між двома різними мовними фрагментами [14,15]. Інформація зберігається в електронному словнику в базі даних або у форматі представлення на основі XML. спеціально розробленому для цієї мети. Наприклад, у рекомендаціях Text Encoding Initiative є окремий розділ про зберігання лексикографічних даних. Елемент введення може бути використаний для зберігання різноманітної інформації що міститься в словникових статтях, зберігання ключів пошуку у різноманітних словниках тощо [16].

В даний час прийнято виділяти наступні типи подання електронних словників:

1) Друкарське представлення - представлення словників в вигляді прийнятному для поліграфії, тото для їх розповсюдження в друкованому вигляді.

2) Представлення редагування - збереження чистого тексту без метаінформації.

3) Лексична репрезентація - вид представлення словників, який містить структуровану за категоріями інформацію зі словника. До категорій можутьналежати зокрема граматичні області, функціональні особливості тощо [16]. Записи в цьому представленні можуть містити спеціальні позначення, наприклад окремий розділ омографа. Кожен розділ містить форму слова, граматичне позначення, інтерпретацію і джерело. Детальним вивченням цього питання займалась міжнародна ініціативна група здослідження цифрових гуманітарних дисциплін TEI [17].

Інший варіант представлення LMF тобто Lexical Markup Framework використовує тільки останній тип (лексичне представлення), аргументуючи це тим, що цей формат призначений для зберігання різноманітних словників для систем обробки текстів [18]. Серед прикладів використання цього вузькоспеціалізованого виду представлення даних найбільш типовою є відома система CLARINE-Translate[19].

Наступним аспектом функціонування електронних словників є компонент що виконує пошук та редагуванні статей. Як згадувалося раніше. багато електронних словників пропонують веб-інтерфейс, що передбачає взаємодію користувача з базо даних, в форматі якої вреалізовано словник [20].

Такж широго розповсюдження набили лагіни та розширення до програм на комп’ютерах або мобільних пристроях які часто містять містять

підключення додаткових словників. Наприклад такі плагни дл ябрузерів як Grammarly, ReversoContext, Toucan [21].

Першочерговою метою цих плагінів є не завжди безпосередньо переклад з однієї мови на іншу. Вони також можуть мати навчальну мету як наприклад Toucan. Чи використовуючи технології подібні до тих що й в системах коп'ютерного інтелектуалізованого перекладу аналізувати заданий текст і пропонувати виправлення, як граматичного так і стилістичного характеру , як наприклад Grammarly. Другий вид виправлень потребує аналізу великої кількості контекстних дани для виявленнь стилістичних закономірностей вживання певних мовних структур, тобто пропонує швидкого багатократного доступу до доних в електронних словниках.

Наступною темою яку потрібно висвітлити для повноти огляду сучасного стану електронних словників та технологій орієнтованих на покращення кякості програм що оброблюють текст є системи фразових таблицяь, ще відомі під назвою з таблиць автокомпіляції. Вони маєть широке використання для перекладу зважених послідовностей слів і враховуються під час машинного перекладу текстів [22]. Суть їхньої роботи подлягає в тому що преклади витягуються з двомовного тексту на основі вирівнювання речень. Алгоритм вирівняювання, відомий в світі алгоитмів обробки тексту впід назвою модель IBM. Першочерговою метою цього алгоритму була побудова моделей, тобто схем, трансляції. Вони в свою чергу використовують алгоритмиоцінки ймовірності перекладу.

Станом на сьогодні значна частина цих алгоритмів веключає в себе застосувананя штучного ітелекту. Просто трапляються й такі що використовують статиистичні методи оцінки ймовірності перекладу. Для алгоритмів другого типу часто дозволяється ітераційна модель оцінювання доречності перекдаду на частині даних. Такі алгоритми зачасту є двоетаними

[23]. Ці моделі можуть бути застосовані до рівнів перекладу як на основі простих елементів таких як слова, так і з використанням складних чи сладених елементів, зокрема фраз. У цьому відношенні основні відмінності між словниками системи машинного перекладу та звичайними словниками полягає в тому що перші включають не лише фрази та рядки слів,. але також представлення слів і колокацій в доповненях до канонічної форми і вирішують проблему перекалду слів чи фраз які не мають прямих відповідників в іншій мові [25,26].

Починаючи від останнього десятиліття попереднього століття було проведено багато досліджень на тему автоматичного виділення еквівалентів перекладу з двомовних корпусів [27]. Отриману систему лексикографічної організації було названо MRD, тобто машинночитанними словниками. Однак спочатку машинночитанні словники застосувавуалися більшою мірою як додаткове джерело інформації для сеціалістів перекладачів, а не як самостійний інструмент що можуть бути інтегрований в систему комп’ютерного перекладу [28]. Основніа більшість машинночитаннх сдловників спершу містила тільки інформацію про можливі еквіваленти перекладу. Значно рідше зустрічались такі варіанти словників що містили також інформацію про частоту вживання тих слів чи частосту, яку можна було б використатити для вирахування ймовірності перекладу. І зховсім невеликимй об’єм тих словників містив власне маркування ‘моврності перекладів. Такж такі словники перебачали зберігання морфологічної інформації вхідних слів, колокацій і фраз [29].

Як було згадано раніше, існує ряд методі вилучення, в сенсі виділення, еквівалентів перекладу з текстів які розроблялись впродовж тривалого часу. Одним з найвизначніших досягнень осьтанніх десятиліть в сфері комп’ютеризованої лексикографії є різні методи фільтрації шумних

перекладів із фразових таблиць [30,31]. Тому головним завданням при створенні словників машинночитанних словників є включення допоміжної інформації, яку можна утримати шляхом аналізу тексту такими системами. Це дає мождливість якісного підвишщення користувацького досвіду словниками за рахунок навяної організації серій синонімічних рядів,Ю які в свою чергу індексовані і впорядковані за частото вживання.

На відміну від традиційних паперових словників, застосування яких має універсальний та багатоіункціональний характер, машиночитанні словники доволі часто орієнтовані на конкретний клас текстів. Тобто можлна вважати що велика кількість машиночитанних словників є укладена за тематичними ознаками. Більш детально це питання розглядається автоматизоваими системами обробки текстів. Основні правила конструювання машиночитанного словника засновано на логічному та інтуїтивному підході, тобто на тих самих методах на які опираються також автори традиційних словників [32]. Ці моделі можуть бути застосовані до всіх рівнів перекладу починаючи вд пошуку відповідників на етапі аналізу слів, кодного окремо, і закінчучи фразами, та еченнями.У цьому відношенні основні відмінності між словниками системи САТ, тобто словниками систм автоматизованого перекладу, в порівнянні зі звичайними словниками є кілька фраз і слів, як правило, стабільних[33].

Згідно з результатами проведених аналітичнихдосліджень основних характеристик поточного етапу розвитку технологій комп’ютерного перекладу було сформульовано висновки щодо основних тенденцій розвитку цього свпектру систем обробки інформації а також зведено перелік вимог до такого їх компонента як електронні словники. На підставі отриманих результатів цього етапу роботи було підсумовано найбільш впливові чинники подальшого розвитку систем перекладу,серез яких можна виділити

використання хмарних технологій, застосування технологій штучного інтелекту, а також нарощення якісних характеристик компютерних систем.

Детальний аналіз використання вишеописаних технологій в складі систем комп’ютерного перекладу засвідчив те що критичним моментом можливості подального поліпшення якості рперекладу є питання швидкісного пошуку в електронних словниках. Таким чином, було зроблено висновок, що пемспективи покращення систем перекладу передбачають в собі прискорення роботи електронних словників.

1.2. Огляд методів організації пошуку в електронних словниках

Відомим методом реалізації електронних словників, який гарантує якісні зв'язки з контекстами і, таким чином, якісний контекстний пошук є організація словників в основу яких взято графи. Слова зберігаються у вузлах графа, а контекстні зв’язки між словами визначаються ребрами. Таким чином. коли відбувається пошук за заданим словом, в результаті вдається отримати набір слів симантично чи синтаксичноо пов’язаних з переданим в якості аргументу шуканим словом. Для прикладу, якщо буде введенно в якості слова іменник, то серез результатів пошуку окрім найближчтго відповідника перекладу також буде група слі серез яких ймовірно будуть як прикметників, так і дієслів [36]. Саме ці слова що належать до відмінних частин мови від тієї до якої належить шукане слово можна використовувати в якості контексту.

Серед інших знаних варіантів організації електронних словників найбільшою популярністю користуються електронні словники на основі деревовидної структури. Ці словники реалізовані сотні раз, ітому легко знайти десятки різних варіантів модифікацій дерев пошуку що лежать в основі таких словників. Головною перевагою словників на основі деревовидних

структур є логіка представлення слів як змінних, що розгалужуються із загальної основи. Тобто як нескінченного набору символів і сімей із спільної основи, адже таке формулювання поняття широко вживане і є легко зрозумілим спеціалістам з галузей на межі технічних наук та філології. Іншою об'єктивною причиною такого підходу є високий рівень заповнення пам'яті що використовується і низький кофіцієнт використаннянадлишкових ресурсів пам'яті. Для реалізації таких електронних словників до сьогоднішнього часу було розроблено цілий ряд деревовидних структур що можуть відрізнятись різним методами зберігання даних і містити незначні модифікації .

Найбільш частим вимпадком електронних словників цього типу є електронний словник на основі бінарного дерева. Для цих словників передбачається така конструкція за якої вузли дерева гарантовано зберігають повний обсяг контекстної інформації відповідного ключа пошуку. Перевага такої організації словника зумовленазручністю реалізації в технічному контексті і наявністю великої кількості вже готових бібліотек та фреймфорків. Враховуючи те, що такі словники можуть використовуватися як частина сучасних комп'ютерних систем перекладу, їхнім найбільшим недоліком є відносно низька швидкість пошуку. Це важливо на даному етапі розвитку системи комп'ютерного перекладу.

Більшість електронних словників створених впродовж останніх років базуються на принципах роботи баз даних [34]. До цього часу було розроблено спеціальну модель для зберігання мовної інформації для цих електронних словників. Перевага такого методу, який застосовує принципи роботи з базами даних, значною мірою заключаєтсья в тому що це надає можливість використання вже готових і добре відтестованих рішень. Попри це цей тип словників має ключовим недоліком те що сильно обмежений в перспективах

підвищення швидколсті доступу до даних. Як наслідок це робить використання цих словників системами перекладу нераціональним.

Висновки до розділу 1

За результатами виконаних досліджень, які складають перший розділ магістерської дисертації можуть бути сформульовані наступні висновки:

1) Проведений аналіз особливостей сучасного етапу розвитку технологій комп’ютерного та комп’ютеризованого перекладу виявлено основні тенденції розвитку цього важливого классу систем обробки інформації показав, що найбільш важливими чинниками для їх швидкого вдосконалення є зростання характеристик компютерних систем, використання хмарних технологій, а також прогресивних технологій штучного інтелекту.

Детальний аналіз застосування вказаних трендових технологій в системах комп’ютерного перекладу показав, що найбільш вузьким місцем при їх застосуванні залишається проблема пошуку в електронних словниках. відповідно, можна зробити висновок, що подальше вдосконалення систем перекладу неможливе без кардинального прискорення роботи електронних словників.

2) Аналіз наявнив ресурсів для організації контекстних електронних словників показав, що загальна методологія побудови електронних словників на основі деревоподібної структури має переваги в контексті зручності маніпулювання мовними структурами. Перш за все, така конструкція полегшує розв'язання задачі пошуку в структурі якщо ключ був наданийз незначною помлкою. По-друге. цей підхід є зручним для сприйняття людьми, а саме фахівцми лінгвістами, адже маж чіткі паралелі між їхнім професійним баченням проблеми лексикографічної організації словників та їх технологічною реалізацією. Цей аргумент на користь словників на основі деревних структур ймовірно є ключовим фактором їх широкого розповсюдження і частого використання. Проте детальний аналіз багатьох

знаних підходів до організації електронних словників, в основу яких покладено деревоподібні структури, показав низку обмжень в їх ефективності. Станом на сьогодні ці обмеженя стають критичними, адже унеможливлюють пришвидшення пошуку в таких словниках і відповідно сповільнюють прогрес комп'ютерних систем контекстного перекладу.

3) Аналіз відомих організацій електронних словників що базуються на технології хеш-адресації свідчить про те що до недавнього часу їх продуктивність була сильно обмежена через виникнення колізій і потребу в надлишкових часових ресурсах на вирішенняцизх колізій. Це робить такі словникі не придатними до використання в сучасних комп'ютерних системах перекладу.

Здійснений аналітичний огляд існуючих способів швидкого пошуку слів в електронних словниках засвідчив, що вони не задовольняють сучасним вимогам в плані швидкодії, тобто їх використання не дозволяє аналізувати велику кількість варіантів контекстного перекладу, що є головним чинником якісного комп'ютерного перекладу.

4) Аналіз показав. що одним із найперспективніших шляхів прискорення роботи комп'ютерно-орієнтованих електронних словників є використання безколізійної хеш-адресації, яка теоретично гарантує високу швидкість пошуку.

Щоб зробити цей вид адресації практичним для словників. потрібно вирішити багато складних технічних проблем, більшість з яких пов'язана із технічною реалізацією вибору perfect хеш-перетворення для кількатисячного масиву ключів.

РОЗДІЛ 2

РОЗРОБКА МЕТОДУ ОРГАНІЗАЦІЇ ЕЛЕКТРОННОГО СЛОВНИКА НА ОСНОВІ ХЕШ-АДРЕСАЦІЇ БЕЗ КОЛІЗІЙ

Від початку двадцять першого століття в країнах Сходу відбувається значне пожвавлення наукової діяльності. Для міжнародної спільноти лишається гострим питання обміну науково-технічною інформацією, а мовний бар’єр є чи не основною перешкодою для збільшення потоків обміну інформації між Сходом і Заходом.

Через особливості східних мов доступні наразі методи машинного перекладу не в стані подолати цю проблему та прийнятному практичному рівні. Тому інснує гостра необхіднітсь розробити нові підходи та методи для вирішення цієї проблеми. Першим можливим кроком до розв'язання цієї задачі є об’єднання зусиль фахівців з інформатики та лінгвістики. Подальшим потенційним кроком є критичний аналіз елементів існуючих систем комп'ютерного перекладу. При цьому підвищення продуктивності будь-якого з доступних компонентів, безсумнівно, призведе до поліпшення загальної ситуації. Тому оприскорення пошуку в електронних словниках є актуальної задачею для сучасного етапу розвитку інформаційних технологій.

2.1. Організація доступу до контесних даних за ключами пошуку

Для досягнення поставлених цілей моєї магістерської роботи було проведено аналіз функціональних мовних технологічних моделей електронних словників. Інакше кажучи, словників які відповідають вимогам контекстного електронного словника, який може бути використаний в якості елементу

сучасних систем комп'ютерного перекладу. Для таких словників в якості аргументів з якими асоціюються контекстні дані можу вистепати слово. колокація, мовна конструкція або складний елемент — речення.

Згідно з функціональною мовною моделлю технічно орієнтованих електронних контекстних словників, інформацію, що в них записується, можна розділити на два типи. Першим типом інформація, яка зберігається в електронному контекстному словнику і включає до себе дані, які стосуються контексту та варіантів перекладу. Для простоти наділі буде позначатися терміном контекстні дані. Інший тип інформації це ключі з якими асоціються контекст який буде шукатись в словнику.

Однією з особливостей згаданої вище моделі контекстного електронного словника є те, що кількість даних, які є необхідними для контекстного перекладу значно перевищує кількість контекстних даних. Іншими словами, об’єм пам’яті необхідних для розміщенняслів-аргументів, що зберігаються в словнику, становить набагато менше, ніж контекстні дані.

Іншою особливістю яка визначає розвиток контекстних електронних словників є те, що ефективність електронних словників залежить від розв’язння компромісу між швидкістю пошуку контекстного словника та розміром таблиць пошуку, які зберігаються в ньому.

У наведеній вище моделі контекстного електронного словника разом із аргументом пошуку ключового слова, пов’язане контекстне слово аргументу ключового слова також надсилається до контекстних даних та наприклад речень, що є прикладами використання. У процесі пошуку інформації в контекстному електронному словнику пріоритет перекладу визначаються параметри на основі заданих слів-аргументів.

Загальновідомо. що ключові характеристики якості отриманого контекстного перекладу перекладу прийнято вважати релевантність і адекватністю Вони в свою чергу безпосередньо залежать від обсягу контекстної інформації. Кою оперує система п, тобто яку вона можу отримати з контекстного електронного словника.

Сучасні підходи до комп'ютерного перекладу базуються на ідеї аналізу великої кількості варіантів, тому ці технології вимагають швидкого збору даних для аналізу. Цей момент є критичним і його не можна відкидати на другий план у зв'язку з потребою у великій кількості альтернативних даних. Можна зробити висновок, що подальший розвиток інтелектуальних систем перекладу, також відомих під назвою NMT, напряму залежить від компромісу між швидкістю пошуку та обсягом даних в електронних словниках.

Оскільки якість перекладу залежить від обсягу контекстних даних, наступний висновок полягає в тому. що для суттєвого підвищення якості перекладу необхідно багатократно збільшити кількість контекстних даних. що відповідають одному клечеві пошуку. Підсумовуючи все це. можна коротко зробити висновок, що одна з головних особливостей розвитку комп'ютерної інтелектуальної системи контекстного перекладу — це кількакратне нарощення обсягу контекстної інформації, що зберігається в словниках.

У ході створення електронних словників, що можуть бути викоистані в якості елементів комп'ютерних систем перекладу, потрібно враховувати багаторівневу структуру комп'ютерних систем — технічну особливість організації пам'яті сучасних комп'ютерних систем. Причому кожен рівень має різні характеристики зумовлені певним підходом до роз'вязання компромісу між швидкістю доступу до даних на цьому рівні пам'єяті і обсягом цьог рівня пам'яті.

Відомо. що процес обміну даними між різними рівнями пам’яті займає у багато разів більше часу, ніж фактична обробка даних на вищих рівнях пам’яті. Для електронних словників це має таке значення, що швидкість безпосередньо залежить від кількості циклів обміну даних між різними рівнями пам’яті, що потрібно виконанити у процесі пошуку найкращого варіанту перекладу. Для отримання високої швидкості пошуку принципово необхідно вирішити проблему мінімізації циклів своппінгу даними. що виконуються в процесі семантичного пошуку відповідності.

Адже процесор працює з даними, що завантаження в пам'ять найвищого рівня - швидкодіючої кеш пам'ять невеликого обсягу. В той час великі обсяги даних зберігаються на низьких рівнях пам’яті, які працюють у сотні разів повільніше. І для обробки процесором будь-яких даних вони спершу мають бути транспортовані з пам'яті нижчих рівнів до швидкодіючої пам'яті вищих рівнів. Процес завантаження цих даних називається свопінгом.

2.2. Розробка механізму організації хеш-адресації

Виконаний в рамках роботи над магістерською дисертацією аналіз показав. що найшвидшим робочим методом для пошуку та доступу до даних відповідно до певного ключа пошуку є пошук на основі технології хешування. Техногія хешування. тобто хеш-адресації. полягає у збереженні даних у пам’яті за адресою, яка залежить від самих даних і в той же час гарантує. що час пошуку не залежить від розміру таблиці пошуку. Таким чином використанні технологій на основі хешування в організації електронних словників у системах комп’ютерного контекстного перекладу може вирішити проблему пошуку компромісу між розміром словників та швидкістю пошуку за конкретним ключовим словом пошуку.

До недавна широке застосування хеш-адресації як практичного технічного рішення для організації електронних словників у складі комп’ютерних систем перекладу було значною мірою обмежено властивими недоліками хеш-адресації. А саме можливістю того що хеш-перетворення створить однакову хеш-адресу для різних ключових слів цю ситуацію конфлікту адрес прийнято називати колізією.

На сьогоднісшній час наявний широкий ряд варіантів вирішення ситуацій колізії. Попри це значна частина з із них передбачає повторювані звернення до пам’яті за даними, що відповідають певному ключовому слову. Наприклад метод зондування є найвідомішим і широко використовуваним методом вирішення конфліктів. Технологія опитування реєструє слово, яке конфліктує з іншим словом у місці пам’яті яке є послідовно найближчим до попередньо збереженого слова.

Відомо, що ефективність застосування методів зондування та реальна ймовірність зіткнення безпосередньо залежить від ступеня використання пам’яті який можна охарактеризувати коефіцієнтом заповнення пам’яті β.

Використовуючи методи пробінгу, ще відомого як технологія зондування, в якості методу вирішення проблеми зіткнень, середня кількість ітерацій звернень до пам’яті для запису або вилучення визначається як:

$$\eta = \frac{1}{1-\beta} \tag{2.1}$$

Отже, можна зробити висновок, що ефективність хеш-пошуку днапряму залежить від використання пам’яті і чим більший рівень надлишковості тим швидшим є пошук.

У реальних системах із хеш-пошуками коефіцієнт навантаження пам’яті зазвичай вибирається близьким до 0.65, тобто 65% пам'яті заповнено корисними даними і 35% пам'яті не викроистовується.

Основним контраргументом для практичного застосування техлологй хешування для організації електронних словників є наявність колізій. Тому в роботі пропонується використовувати perfect хеш-адресацію. Тобто хеш-адресаці без колізій. В якості perfect хеш перетворення проноуються застосовувати стандартні блоки шифрування симетричних алгоритмів шифрування. таких як DES. AES і т. д. Таким чином пропонується вирішити проблему пошуку спеціального способу вибору хеш-перетворень.

Іншим властивим недоліком хеш-пошуків в сенсі його використання як технічного рішення для організації електронних словників невелика є те що зміна в ключовому слові пошуку повністю змінює хеш-адресу і вона вказує на зовсім інші області пам'яті. Тобто спільнокореневі слова чи близькі за написанням конструкції розкидуються по різних частинах словника.

Пропонується, щоб наведена вище синтаксично конвергентна функція обробки ключових слів була вирішена шляхом попереднього вибору елементів або основ і використання їх як ключових слів пошуку. Рекомендується зберігати варіації та модифікації цих ключових слів як контекстні дані. Дл цього можна застосувати комп'ютеризовані методи аналізу слів за їх морфологічними ознаками [50] з мето видяління виділення основ слів [51,61].

Під час первинного аналізу характеристик хеш-адресації було визначено намір використовувати дворівневий пошук для підвищення ефективності електронно-комп'ютерних словників у системах контекстного перекладу, де в якості ключового слова виступає корінь виділеного слова або його основа. В езультаті першого етапу пошуку має бути отримане адресне посилання на початок блоку контекстної інформації що відноситься до аргументу пошуку. Фактичний контекстний пошук варіантів перекладу виконується в межах другого рівня, наприклад з використанням технології лінійного пріоритетного пошуку. Деталі технічної реалізації пошуку другого

рівня лишаються на вибір конретної системи комп'ютерного перекладу і вони не впливають на організаціюпершоого етапу пошуку який розглянуто детально в подальших викладеннях цього розділу.

У цьому випадку в якості коду калібрування хеш-перетворення пропонується використовувати ключ симетричного шифроблоку. Це власне й втілює собою запропоновану ідею вирішення проблеми підбору perfect хеш-перетворення для заданого і незмінного масиву можливих аргументів пошуку.

Таким чином, запропонована в рамках магістерської роботи ідея допускає таку послідовність виконанння першого етапу пошуку релевантного перекдау.Необхідно передати ключове слово, за яким виконується пошук, в якості аргументу стандартного симетричного криптографічного алгоритму, а відповідний отриманий в результаті цього зашифрований текст або частину цього зашифрованого тексту, отримана в результаті хеш-перетворення. служить як хеш-адреса.

Серед відомих переваг використання блоку симетричного алгоритму шифрування як хеш-перетворення особливо важливим є те що він безпечує рівномірний розподіл хеш-адрес. Також варто відзначити високий потенціал використання стандартних ключів блоку шифрування, які використовуються як механізм налаштування хеш-перетворення. Ще одна важлива сильна сторона використання криптографічних блоків алгоритмів симетричного шифрування як хеш-перетворювачів полягає в тому що вони можуть досягти високої продуктивності схем, які реалізують стандартизовані криптографічні блоки на апаратному рівні.

Проте детальний аналіз запропонованої ідеї використання хеш-адресації без колізій як технічного рішення для організації електронних словників у системах комп’ютерного перекладу також виявляє деякі проблеми

в її реалізації. Ми знаємо, що пам'ять містить $2u$ комірок, де u - кількість бітів у хеш-адресі. У результаті коефіцієнт заповнення пам'яті β обчислюється як:

$$\beta = \upsilon/2^{u}. \quad (2.2)$$

Нехай $P(\beta, \upsilon)$ буде ймовірністю, що випадково вибране хеш-перетворення поміщає υ ключових слів у пам’ять із цим коефіцієнтом завантаження пам’яті β, не спричиняючи колізії. Наближеним формульним предствленям $P(\beta. \upsilon)$ є:

$$P(\beta, \upsilon) = e^{\frac{\upsilon \cdot \beta^2}{2}} \cdot e^{-\frac{3\beta}{3-\alpha}} \quad (2.3)$$

Це значення у свою чергу обернено пропорційне значенню N, де N – кількість вибірок для вибору досконалого хеш-перетворення, що залежить насамперед від значень коефіцієнтів β і υ.

$$N = 1 \Big/ P(\beta, \upsilon) = e^{\frac{\upsilon \cdot \beta^2}{2}} \cdot e^{--\frac{3\beta}{3-\alpha}} . \quad (2.4)$$

На практичному рівні основним обмеженням у процесі пошуку досконалого хеш-перетворення є час який потрібно витратити на віднаходження цього перетворення. Нехай він позначатиметься δ. Проаналізувавши процес вибору досконалого хеш-перетворення більш детально можна зробити висновок, що щоб знайти повне хеш-перетворення. хеш-перетворення для постійної таблиці υ-ключів необхідно обчисити хеш-адресу для кожного з υ ключів. У випадку колізії, тобто якщо хеш-адреса поточного ключового слова збігається з хеш-адресою одного з попередньо оброблених ключових слів, необхідно припинити подальшу роботу з поточним хеш-перетворенням. І процедура повторюється для наступного хеш-перетворення. Якщо всі υ ключів були оброблені без конфліктів, тобто

колізій, можна зробити висновок що досконале хеш-перетворення знайдено для даної сталої υ таблиці ключів.

Нехай символ μ позначає математичне сподівання кількості ключових переходів в одному хеш-перетворенні. Тоді час δ необхідний для віднаходження досконалого хеш-перетворення для постійного масиву υ ключів можна висловити формулою:

$$\delta = t_h \cdot \mu \cdot N = t_h \cdot \mu \cdot e^{\frac{\upsilon \cdot \beta^2}{2}} \cdot e^{-\frac{3\beta}{3-\alpha}}, \qquad (2.5)$$

де t_h - це час для обрахування однієї хеш-адреси за допомогою вказаного хеш-перетворення без колізій.

Математичне очікування μ кількості циклів обходу ключів під час перевірки кожного можливого хеш-перетворення без колізій обчислюється наступним чином:

$$\mu = \gamma_2 \cdot 2 + \sum_{l=3}^{\upsilon-1} \zeta_l \cdot \gamma_l \cdot l + \zeta_m \cdot \upsilon, \qquad (2.6)$$

де ζ_ι — це ймовірність того. що на ι-тому кроці перебору ключів буде виявлено колізію, γ_ι – це ймовірність того що колізій не буде виявлено до ι кроку пошуку за ключовим словом при $\iota \in \{2.3 \ldots . \upsilon\}$. Імовірність γι може бути обчислена як добуток ймовірностей того. що на жодному з попередніх кроків пошуку за ключовими словами не трпаилося колізій:

$$\gamma_l = \prod_{i=1}^{(l-2)\left(1-\frac{i}{2^n}\right)}. \qquad (2.7)$$

Значення ζι є відношенням кількості заповнених до ι-го кроку комірок хеш-пам’яті до кількості всіх комірокхеш-пам'яті 2^u.

$$\zeta_\iota = (\iota - 1)/2^u. \qquad (2.8)$$

У цьому випадку поєднуючи все вищесказане ми можемо більш точно виразити значення математичного сподівання μ циклу вибору ключа яке безпосередньо залежить від ймовірностей деталізованих вище. Формула для визначення матиматичного очікування μ кількості циклів обходу ключів набуває вигляду:

$$\mu = \gamma_2 \cdot 2 + \frac{\sum_{l=3}^{(\upsilon-1)(\iota-1)} \square}{2u} \cdot \prod_{i=1}^{(l-2)\left(1-\frac{i}{2^n}\right)} \square \cdot l + \zeta_\upsilon \cdot \upsilon \quad . \qquad (2.9)$$

У ході теоретичних розрахунків на основі наведених вище співвідношень ми знайшли можливі значення коефіцієнта заповнення пам'яті β. З цим значенням ми можемо знайти хеш-перетворення без колізій для заданої кількості υ ключів за обмеження часу δ. Результати виконаних розрахунків зведено до Таблиці 2.1.

Виконані розрахунки мають в своїй основі той факт, що вбудована реалізація алгоритму DES у криптографічний процесор Cortex-M4 використовується як хеш-конвертер. Згідно з цим документом Microcircuits [59]. час обробки для блоку даних. тобто. обчислення хеш-адреси для вибраного алгоритму становить 7.6 · 10 · e^-8 с. Значення часу δ для вибору хеш-перетворення без колізій наведені в таблиці 2.1. є оціночними. Цей час можна зменшити на коефіцієнт s, використовуючи хмарні технології для організації вибору на s кількості процесорів.

Наприклад розглянемо можливі значення коефіцієнта заповнення пам’яті β для обсягу словника в 70000 слів. Це значеня є наближеним до реального можливого значення ількості слів в словниках, які є частиною систем контекстного перекладу. Щоб знайти хеш-перетворення без колізій нам потрібно обмежити його до 500 годин. Для таких параметрів коефіцієнт

заповнення пам'яті β становить 0.02. це означає, що пам'ять має бути заповнена ключовими словами не більше ніж на 2% від загальної ємності.

Надано покроковий розрахунок вищеописаної процедури:

Таблиця 2.1

Залежність коефіцієнту β від *υ* та граничного часу *T* підбору хеш-перетворення без колізій.

υ	Значення коефіцієнта β при наведених значеннях δ(год)					
	δ = 10	δ = 50	δ = 100	δ = 500	δ = 1000	δ= 10000
10000	0.0595	0.0625	0.0641	0.0665	0.0675	0.0715
14000	0.0489	0.0505	0.0515	0.054	0.0545	0.0582
18000	0.0415	0.0435	0.0445	0.0465	0.0477	0.0523
24000	0.0379	0.0391	0.0395	0.0415	0.0426	0.0445
28000	0.0335	0.0355	0.0367	0.0375	0.0385	0.0405
32000	0.0319	0.0325	0.0335	0.035	0.0355	0.0375
36000	0.0298	0.0308	0.0317	0.0325	0.0338	0.0354
46000	0.0275	0.0293	0.0295	0.0305	0.0319	0.0332
52000	0.0266	0.0275	0.0284	0.029	0.0295	0.0311
56000	0.0245	0.0264	0.0265	0.0275	0.0282	0.0295
62000	0.0235	0.0251	0.0255	0.0271	0.0275	0.0285
66000	0.0225	0.0242	0.0245	0.0255	0.0261	0.0275

Продовження таблиці 2.1.

72000	0.0225	0.0233	0.0235	0.0245	0.0259	0.0265
76000	0.0219	0.0224	0.0225	0.0235	0.0243	0.0255
82000	0.0205	0.0215	0.0224	0.023	0.0235	0.0245
86000	0.0209	0.021	0.0215	0.022	0.0225	0.0242
92000	0.0195	0.0205	0.0205	0.0215	0.0224	0.0236
96000	0.0192	0.0195	0.0282	0.021	0.0214	0.0225
100000	0.0185	0.0189	0.0196	0.0208	0.0213	0.0221

Детальний аналіз результатів обчислень. наведених у таблиці 2.1 показує що значення коефіцієнта заповнення хеш-пам’яті β досить слабо залежить від часу пошуку межі δ. Зокрема для $\upsilon = 70000$ час пошуку збільшується на три порядки від 0.021 до 0.025. що становить лише 0.004.

Крім того. під час аналізу було виявлено. що значення β напряму залежить від кількості ключових слів υ. Коли значення υ збільшується в 10 разів. коефіцієнт заповнення пам'яті β зменшується в середньому в 5 разів. Це веде до заключення, що використання пам’яті є відносно неефективним при організації словників у реальному масштабі визначеному вимогами сучасних комп’ютерних систем контекстного перекладу.

Було запропоновано подолати цей недолік наступним чином. Пропонується щоб контекстна інформація фактично необхідна для перекладу. пов’язана з ключовими словами пошуку зберігалася безпосередньо в хеш-пам’яті. Це означає що велика кількість контекстної інформації може ефективно заповнювати прогалини між клітинками які фактично адресуються за допомогою повного хеш-перетворення ключового слова пошуку.

Ця ідея передбачає таку реалізацію при якій адреса отримана в результаті перетворення хешу ключового слова пошуку містить адресне посилання на початок контекстної інформації, що відповідає цьому ключу. Щоб реально реалізувати цю ідею нам потрібно заповнити хеш пам’яті в два послідовних кроки.

Першим кроком є виділення пам’яті адреса якої посилається на контекстну інформацію для відповідного ключового слова. Очікується, що другим кроком є заповнення контекстної інформації для розділів хеш-пам’яті в проміжках між основними посиланнями. У той же час.заповнення повинно дозволяти первинному посиланню та відповідній контекстній інформації завантажуватися в кеш без заміни, наскільки це можливо слово за словом.

Фактична реалізація запропонованої конструкції електронного словника полягає в тому що адресне посилання на початок контекстної адреси що відповідає даному ключовому слову зберігається за адресою, створеною в результаті виконання повного хеш-перетворення ключового слова. Зауважте, що як адресне посилання на контекстну інформацію, так і сама контекстна інформація зберігаються в хеш-пам’яті. Це означає, що запис у хеш-пам'ять має виконуватись у два етапи.

Спочатку пропонується вибрати і позначити область хеш-пам'яті де повинна бути розміщена адресна посилання на початок контекстної інформації. Для цього нам потрібно зробити хеш-перетворення пошукового терміну. Коли цей етап буду ітеративно виконана для всіх υ ключів, можна переходити до другого етапу.

Другий етап створення хеш-пам'яті має зберігати актуальну корисну контекстну інформацію, необхідну для перекладу. Для цього спочатку визначте розподілену організацію цих контекстних даних для υ ключів з

обсягами $v1$, $v2$, $....vυ$ з інтервалами $b0$, $b1$, $....bυ$ між необхідною зарезервованою пам'яттю, адреси посилання $l1$, $l2$, $....lυ$.

Якщо середній розмір контекстної інформації для кожного з υ ключових слів пошуку є значенням v, тоді середня ємність між адресними посиланнями. отриманими в результаті хешування ключових слів, дорівнює b.

Рішення проблеми організації розміщення контекстної інформації в хеш-пам'яті полягає у визначенні порядку збереження і завантаження контекстної інформації в швидкий кеш.

Зауважимо, що ефективність побудови словника з точки зору швидкості безпосередньо залежить від кількості ω блоків даних, які надсилаються в кеш протягом одного циклу обміну. Невеликі значення ω роблять цикл обміну набагато швидшим, але не всі контекстні дані завантажуються в одному циклі обміну. У цьому випадку результуюча витрата часу негативно впливає на ефективність запропонованої моделі словника. З іншого боку велике значення ω уповільнює процес обміну. Тому вибір обсягу ω блоків даних вимагає розумного компромісного рішення.

Більш детальний огляд процедури внесення даних до словника виглядає так: по-перше, пам'ятайте що словник спрощення має бути розділений на дві частини. У першій частині пропонується розмістити первинне посилання l та основну частину доступних текстових даних. Відтепер цю частину пам'яті називають хеш-пам'яттю. Друга частина пам’яті пропонується для розміщення решти контекстної інформації, яку не вдалося зберігати в частині хеш-пам'яті відповідно до запропонованого алгоритм. Ця друга область пам'яті надалі буде називатися пам'яттю переповнення. Зверніть увагу що частина хеш-пам'яті зберігає частину контекстної інформації яку можна прочитати за один цикл обміну.

Для реалізації описаної ідеї організації пам'яті було вирішено виділити чотири формати пам'яті для використання в словниках. Це формати для позначення пам’яті виділеної для зберігання словників і мають програмне забезпечення, а не вбудовані реалізації.

Перший формат пам’яті, є форматом вільної пам’яті та ідентифікується символом U1. який зберігається в першому байті комірки пам’яті прийнятого розміру. У цьому контексті комірка пам'яті являє собою найменшу незалежну від апаратної реалізації область пам'яті. яка обробляється запропонованими алгоритмами для введення та пошуку даних в електронних словниках. Це означає? що комірка пам’яті в цьому контексті не обов’язково дорівнює найменшій апаратно адресованій одиниці. що дорівнює 1 байту в сучасних комп’ютерних системах. Розмір комірки пам'яті. яку має на увазі залежить від її функції. Отже, перший формат пам’яті означає. що комірки цього формату містять маркерний символ U1 у першому байті а сам формат займає вільну пам’ять.

Було запропоновано використовувати другий формат пам'яті для зберігання дійсно корисної контекстної інформації. Для більш ефективного використання пам'яті комірки другої форми не позначаються символом і розпізнаються як такі. що містять виключно корисні дані починаючи з першого байта.

Рекомендується зберігати адресні посилання створені хешуванням ключових слів пошуку у клітинках третього формату. Комірку пам'яті третього формату можна розділити на три секції. Перший байт якого містить символ U2. В другій і третій частини пропонують зберігати адресні посилання на контекстну інформацію. Тобто другий розділ містить адресне посилання на початок контекстної інформації для конкретного ключового слова пошуку, а

третій розділ містить адресу останнього розташування в пам’яті відповідної контекстної інформації.

З метою збереження доповнення контекстної інформації які не могли зберігатися в області хеш-пам’яті було введено четвертий формат комірки пам’яті. В комірках пам'яті четвертого формату зберігається адресне посилання на продовження контекстної інформації певного ключового слова в зоні пам'яті переповнення.

Після визначення функціонального завдання кожного формату комірок пам'яті стає можливим визначити необхідний обсяг пам'яті для цих комірок. Розмір комірок пам'яті усіх форматів є однаковим. Аналіз наведених вище функцій доводить, що для більшості даних цей розмір визначається комірками другого формату. Обсяг необхідної пам'яті залежить від u-розрядного значення хеш-адреси. Через те що комірки 2-го формату повинні містити один макросимвол і два адресних посилання. Якщо для збереження адресного посилання потрібно u/8 байт, то мінімальний розмір комірки пам'яті становить 1+u/8*2 байти.

Для ззбереженя даних у словнику потрібно вибрати символи тегів, які використовуються для позначення різних форматів пам’яті. Тобто вони повинні відрізнятися від можливих корисних даних. Для цього підходить будь-який службовий символ ASCII.

У першій ітерації цієї магістерської роботи був запропонований наступний алгоритм запису інформації.

1. Символ тегу U1 вільної пам’яті записується в перший байт кожної комірки хеш-пам’яті.

2. Для кожного ключового слова розраховується хеш-адреса. Перший байт адресованої комірки пам'яті записується символ U2.

3. Початковий номер поточного слова j: $j = 1$. Реєстрація починається з пошуку контекстних даних слова адресованого найменшою хеш-адресою lj. Початок q переповнення пам'яті встановлюється відразу після області хеш-пам'яті. Тобто $q = 2u + 1$. де u - розрядність хеш-адреси.

4. Для ключового слова номер j починаючи з адреси lj - ω/2 шукається вільну ділянку хеш-пам'яті позначена символом U1.

5. Далі початкова позиція контекстної інформації для ключового слова номер j зберігається в клітинці l. а значення адреси l збільшується на 1: $l = l + 1$.

6. Якщо весь обсяг контекстних даних j-го слова знаходиться в хеш-просторі. то в третє поле комірки пам'яті за адресою lj необхідно ввести адресу l-1. Розміщення даних продовжується з кроку 11.

7. Якщо адреса l адресує комірку пам’яті третього формату $l \leq lj$, тоді адреса l має бути інкрементована $l = l + 1$. Продовжте процес розміщення даних з пункту 5.

8. Якщо припустити що комірка за адресою $l + 1$ містить тег U2 *і* $l > lj$. або $l = lj + \omega/2$, комірка l позначена символом тегу U3 як посилання на переповнення пам’яті. Там зберігається поточне значення q. Перейдіть до кроку 10.

Якщо комірка з $l = lj + \omega/2$ або l + 1 має тег U2 і $l > lj$. то перший байт комірки з l позначається символом тегу U3 як посилання на переповнення пам’яті. Поточне значення q зберігається як посилання на решту контекстної інформації. Процес запису продовжується з кроку 10.

9. Щоб продовжити запис перейдіть до кроку 5.

10. Залишки даних що не було розміщнгго раніше записуються в пам’ять переповнення починаючи з адреси q. Адреса останньої записаної комірки зберігається в комірці q.

11. Значення j інкрементується: $j = j + 1$. Якщо значення j <= υ, тоді знайдіть наступну комірку формату 2 посилання та поверніться до кроку 4.

12. Завершено запис даних для υ ключових слів.

В ході роботи над цим магістерським проектом., було вирішино трохи оптимізувати процедуру зберігання інформації слів у словнику. У фінальному варіанті процедура розміщення даних у словнику виглядає як:

1. Уся пам'ять позначається як вільна пам’ять першого формату.

2.Хеш-перетворення виконується для кожного ключового слова, а отримані комірки позначаються тегом головного посилання U2.

3. Кількість поточних слів j дорівнює 1: $j = 1$. Адреса q початку переповнення встановлюється на початок пам'яті переповнення $q = 2u + 1$.

4. Знайдіть вільну комірку хеш-пам'яті l з тегом U1 для j-го слова. починаючи з адреси lj - ω/2.

5. Запишіть службову інформацію в комірку l,таким чином вона трансформується в комірку пам’яті другого формату. Значення a інкрементується: $a = a + 1$.

6. Якщо вся контекстна інформація зберігається в хеш-пам’яті для j-го слова, тоді адреса l - 1 встановлюється в 3-е поле клітинки lj. Перехід до кроку 11.

7. Якщо за адресою l йде комірка основного посилання і в той же час $l \leq lj$. то адреса l оновлюється l = l + 1 і повертаємося до пункту 5. Тобто запис продовжується і закінчується відразу після комірки адресного посилання.

8. Якщо клітинка з $l = lj + ω/2$ або $l + 1$ містить тег U2 основного посилання та $l > lj$ одночасно перший байт клітинки l записується символом U3 що трансформує комірку до четверного формату. Перейдіть до кроку 10.

9. Перейдіть до пункту 5.

10. Решта контекстної інформації для ключового слова j записується послідовно для переповнення пам’яті за адресою *q*. Адреса останньої заповненої комірки пам'яті зберігається в третьому полі клітинки *lj*. а та сама адреса + 1 записується в q.

11. Збільшити *j* на 1: *j* = *j* + 1. Якщо *j* не перевищує υ. переходимо до пункту 4.

12. Контекстні дані ключових слів записано до пам'яті.

Таблиця 2 наочно показує поведінку та ефективність запропонованої процедури для впорядкування даних контекстних ключових слів на прикладі збереження десяти слів та їхніх контекстних даних у пам’яті в областях хеш-пам’яті та переповнення пам’яті. У ході прикладу розрахунків були використані наступні вхідні дані.

- ω дорівнює 120.
- Значення коефіцієнта β для заповнення хеш-пам'яті вихідними посиланнями становить 0.001.
- Ємність хеш-пам'яті 1000 комірок.
- Переповнення пам'яті починається з адреси клітинки 1000.

Тому для заданих параметрів 10 біт достатньо для адресації слова в запропонованого обсязі хеш-пам'яті. Це означає що використовуючи формулу (4). оптимальний розмір комірки пам’яті для вашого конкретного прикладу становить 4 байти.

Розглянемо розміщення одного зі слів докладніше. Візьмемо для прикладу третє слово хеш-адреса якого обчислюється рівна 256. Якщо ω дорівнює 120 то відповідно до запропонованої області зберігання для контекстної інформації цього слова 256 - 120/2 до 256 + 120/2 тобто від 196 до 316.

Таблиця 2.2.

Приклад розподілу хешів і переповнення пам’яті для 10 ключових слів і пов’язаної контекстної інформації

№	Результат хеш-функції для заданого ключогово слова. який використовується як адреса	Контекстні дані ключового слова				
		Розмір даних слова	Адреси в хеш-пам'яті		Адреси в пам'яті переповнення	
			Початок	Кінець	Початок	Кінець
1	103	252	42	163	1	132
2	200	13	187	199		
3	256	183	201	279	133	261
4	280	63	280	340	262	265
5	365	77	341	425	266	268
6	490	30	430	460		
7	560	21	500	521		
8	572	72	522	592		
9	700	90	640	730		
10	820	115	740	875		

Враховуючи що попереднє слово заяйняло простір до 199 комірки включно. а такж комірка за адресою 200 щайнята хеш адресою другого слова. Тому контекстні дані третєго слова починають розміщуватись з 201 комірки пам’яті. Запис до крайньої межі не ї можливим. а саме до 316 комірки. оскільки наступне слово має хеш адресу 280. Тому країньої межею запису даних третьго слова в хеш-пам’яті є комірка 279. Решта контекстних даних записується в пам'ять переповнення. починаючи з поточного значення q = 133 відповідно до пункту 10 запропонованої процедури.

Обчислені результати наведені в таблиці 2.2 і показують що для більшості ключових слів (навіть 60% ключових слів) відповідна контекстна інформація зберігається лише в хеш-пам’яті. Таким чином. обсяг контекстної інформації в пам'яті переповнення становить лише 26.8% від обсягу в хеш-пам'яті. Це приводить мене до висновку що пам’ять використовується дуже ефективно.

Що стосується пошуку, дотримуючись запропонованої концепції дворівневого пошуку, першим кроком є завантаження блоку пам’яті що містить контекстну інформацію для даного ключового слова у швидкий кеш. Упорядкування цих даних у монолітні блоки та отримання лише корисних даних. не перериваєтся службою комірки пам’яті. Це означає що всі фрагменти отриманої інформації про контекст ключового слова повинні бути об’єднані в один інтегрований блок.

На другому етапі шукається найбільш семантично відповідний варіант перекладу для певних важливих мовних конструкцій. Такий пошук перекладу базується на контекстній інформації завантаженій у швидкий кеш. Технічна реалізація цього процесу здійснюється за участю інтелектуальних прийомів структурної лінгвістики та лексико-семантичного комп’ютерного аналізу.

Розглянемо перший етап пошуку докладніше. Пропонується зробити це.дотримуючись запропонованої процедури для отримання контекстної інформації про задане ключове слово s. Ця процедура також включає завантаження цієї інформації в швидкий кеш вищого рівня. Тепер перейдіть до другого етапу компонування та пошуку. Це зроблено у такому порядку:

1. На основі заданого ключового слова пошуку s обчислюється його perfect хеш-адреса l: l = h(s).

2. Блок у клітинці ω завантажується з хешу в кеш. починаючи з адреси l - ω/2. Адреса першого байта. кешованого в блоці контекстної інформації. позначається ζ.

3. Зчитує початкову адресу контекстних даних для ключового слова s у хеш-пам’яті у змінну V. Це значення береться з комірки за адресою ζ + (ω/2)·(u/4+1).) + 1. Адреса Q початку контекстних даних у пам’яті обчислюється наступним чином: Q = ζ + (V - l + ω/2) · (u/4 + 1).

4. Кінцева адреса даних контексту ключового слова s у хеші зчитується в змінну R. Це значення є основною адресою посилання ζ + (ω/2)·(u/4 +1) у кеші.

5. Кінцева адреса v контекстних даних у кеші встановлена на нуль: W = 0+

6. Припускаючи. що R < l + ω/2. значення W задано як W = ζ + (R – l + ω/2) · (u/4+1).

7. Початковою адресою x у складеному блоці даних є ζ: x = ζ. а початковою адресою j у блоці даних контексту читання є Q: j = Q.

8. На цьому етапі дані зводяться в один блок. У кеші байт за адресою j передається байту за адресою x. і обидві адресиінкрементуються (x = x + 1. j = j + 1).

7. Якщо байт адресований j містить маркерний символ U2, що зустрічається в коміркх третього формату. Значення j збільшується на розмір клітинки в байтах. Тобто j = j + u/4 + 1.

8. Якщо j - 1 = W, перейдіть до кроку 11.

9. Якщо байт адресований j містить маркерний символ U3, що є типовим для комірки 4-го формату, решту даних контексту потрібно шукати в області переповнення. У цьому випадку адреса продовження контекстної інформації зберігається в змінній S і яка витягується за адресою j + 1. Перейдіть до кроку 12.

10. Перейдіть до пункту 8.

11. Контекстна інформація для всіх знайдених завантажених і впорядкованих ключових слів. Блок у кеші має початкову адресу ζ і кінцеву адресу x - 1. Ініціалізація другого кроку пошуку. Перейти до пункту 16.

12. Деяка контекстна інформація для ключового слова s знайдена. завантажена в кеш і відсортована. Блок у кеші має початкову адресу ζ і кінцеву адресу x - 1. Ініціалізація другого кроку пошуку.

13. Якщо на другому етапі пошуку знайдено відповідні та релевантні переклади. пошук можна вважати завершеним. Перейти до пункту 16.

14. Частина контекстної інформації ключового слова s знаходиться в пам’яті переповнення за адресою S. Кількість клітинок. зайнятих цією частиною інформації. дорівнює p і обчислюється таким чином: p = R - S. Переповнення пам'яті кешує блоки комірок починаючи з b.

15. Адреса першого байта кешованого блоку контекстної інформації позначається γ. Адреса W кінця кешованих контекстних даних обчислюється наступним чином: $W = \gamma + p \cdot (u/4 + 1)$, продовження другого етапу пошуку.

16. Пошук завершено.

Рис.2.1. схематично показує процедуру виконання першого кроку пошуку на прикладі слова s. контекстна інформація якого знаходиться не тільки в хеш-пам'яті. але також і в пам'яті переповнення.

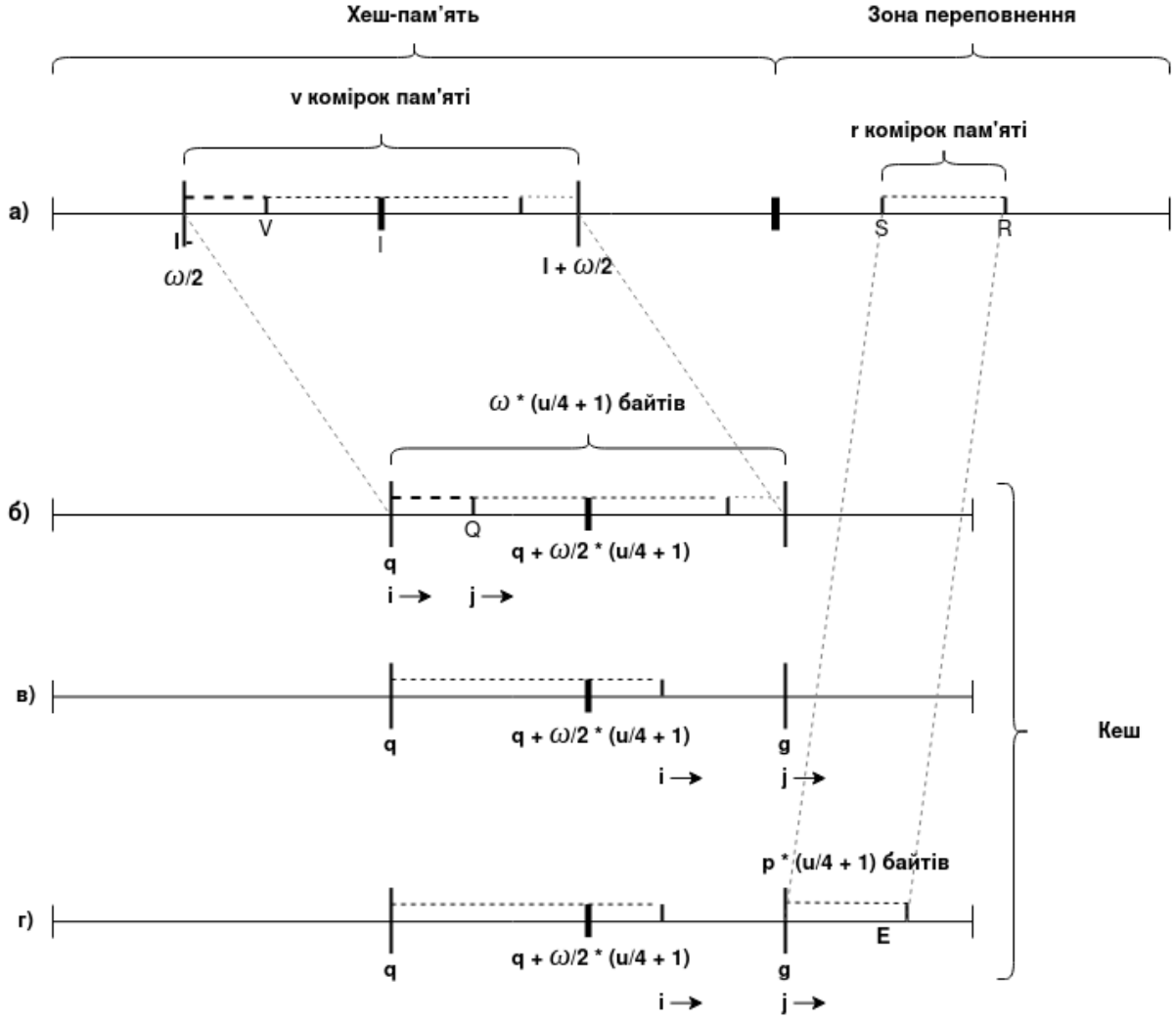

Рис.2.1. Приклад здійснення 1-го етапу пошуку для ключового слова *s*

У пункті *а* малюнку 2.1. показано розміщення контекстної інформації для слова s у пам’яті нижчого рівня або з точки зору цього дослідження у хеш-пам’яті та пам’яті переповнення. У той же час слово s адресується адресним посиланням l. Межі блоків пам'яті розташованих у хеш-пам'яті. визначаються розміром блоку комірки ω. Саме цей обсяг ω комірок

контекстної інформації може бути завантажений у кеш один раз у першому циклі обміну.

У пункті *б* малюнку 2.1. показано розміщення блоків даних завантажених із хешів у швидкий кеш блоків даних. Коли цей блок розміщується в монолітному макеті. значення ζ і Q зберігаються в змінних *x* і *j* відповідно.

У пункті *в* малюнку 2.1. показано вже скомпільовану контекстну інформацію. Він не містить службових даних і готовий для подальшого аналізу на другому етапі пошуку відповідних перекладів.

У пункті *г* малюнку 2.1. показано як блок пам’яті завантажується з кешу переповнення якщо не знайдено відповідного перекладу для даних завантажених із блоку, що зберігається в хеш-пам’яті. Цей блок містить лише корисну контекстну інформацію тому додаткова верстка не потрібна.

2.3. Оцінка ефективності

Першим кроком в оцінці ефективності конфігурації електронного словника для комп’ютерної системи перекладу є визначення критеріїв ефективності. Це дозволяє об'єктивно оцінювати і порівнювати з існуючими аналогами. Ці критерії визначаються сферою застосування та пріоритетами її розвитку на поточному етапі розробки.

Таким чином. встановлено, що ключовими критеріями оцінки ефективності електронних словників. призначених для застосування в системах комп’ютерного перекладу є::

- ефективність споживання пам'яті.
- швидкість пошуку за ключем.

Характерною особливістю сучасних комп'ютерних систем є багаторівнева організація пам'яті. Для таких систем час пошуку $\delta_в$ складається з двох компонентів: часу. витраченого на заміну поточної інформації в кеш-пам'яті і фактичного часу пошуку на контекст. Отже. час пошуку $\delta_в$ розраховується за формулою:

$$\delta_в = \eta \cdot t_\varphi + t_n, \tag{2.10}$$

де t_φ — час виконання одного циклу обміну. t_n — час (контекстного) пошуку, η — кількість циклів обміну.

Тоді час що витрачається t_φ для вздійснення одного циклу обміну даними між пам'яттю різних рівнів пропорційний об'єму ω буфера даних що власне завантажується:

$$t_\varphi = \omega/a, \tag{2.11}$$

де a — швидкість завантаження даних в кеш-пам'ять.

Як перше наближення середній час t_n можна вважати пропорційним середньому обсязі контекстної інформації θ:

$$t_n = \mathrm{r} \cdot \theta, \tag{2.12}$$

де r виражає собою коефіцієнт швидкості пошуку найбільш релевантного та правльного перекладу. А значння цього коефіцієнту визначається технологічним рішенням другого етапу пошуку, згідно з моделі запропонованого електронного словника.

Проведенний детальний огляд і розбір обчислювальних процесів котрі лягли в основу механізмів роботи сучасних електронних словників свідчить про те що час t_φ що затрачається на ззавантаження даних у кеш з пам'яті нижніх рівнів в рази перевищує час t що затрачається на власне другий еап пошкук. Адже пошук релевантного перекладу здійснюється відносно даних що заантаженя в пам'ять вищих рівнів. Роблячи висновок з цього середній час пошуку електронних словників можна оцінити за середньою кількістю циклів

обміну φ необхідних для доступу до контекстної інформації заданого шуканого слова.

Рівень ефективності споживання пам'яті можна оцінити як: нехай υ буде середнім обсягом контекстної інформації на ключове слово тоді загальний обсяг лінгвістичної інформації o що зберігається в словнику. дорівнює $o = r \cdot \upsilon$.

Окрім певного обсягу лінгвістичної інформації усі електронні словники містять певний обсяг службових даних які забезпечують доступ до певної контекстної інформації. Це означає що фобсяг пам’яті. який використовують фактичні словники завжди більший за обсяг виключно лінгвістичної інформації.

Показник надмірності використання пам’яті λ електронного словника можна визначити відношенням загального обсягу ресурсів пам’яті θ. призначених для його реалізації. до обсягу l контекстної інформації що зберігається в ньому:

$$\lambda = \theta / o = \theta /(\cdot \upsilon). \qquad (2.13)$$

Для дослідження показників ефективності запропонованої організації електронного словника шляхом експериметтальної перевіки було розроблено програмне забезпечення статистичного моделювання. Його використання допомогло встановити вплив параметрів організації даних у словнику на показники ефективності.

У рамках моделювання кількість слів у словнику υ було прийнято рівним 15000 а середній розмір контекстної інформації для всіх ключових слів становила 500 байт. Для віднаходження цих показників було виконано статистичне дослідження словників та описових комп’ютерних термінів [52][60].

Перший цикл досліджень запропонованої конфігурації електронних словників має на меті визначити вплив розміру буфера обміну ω на кількість циклів φ і (загальний) час обміну tφ. Актуальність проведення такого дослідження зумовлена комплексним характером впливу величини буфера обміну ω на час tφ його виконання. З одного боку збільшення значення ω збільшує ймовірність того що контекстна інформація ключового слова буде прочитана в повному обсізі за один цикл обміну, зменшуючи загальну кількість циклів обміну φ. З іншого боку, це також впливає на подовження тривалості самого циклу обміну.

Результати виконаного моделювання було зведено до таблиці 2.3. При обчисленях вважалосся значення коефіцієнта β заповнення адресного простору хеш-пам'яті рівним 0.001.

Таблиця 2.3

Залежність часу та кількості циклів обміну від буферу обміну

Довжина ω буферу обміну. байт	Ймовірність зберігання всого обсягу контекстної інформації в зоні хеш пам’ять	φ	Відносний час обміну
500	0.426	1.564	1
700	0.456	1.555	1.513
850	0.462	1.541	1.97’
1100	0.482	1.537	2.464
1400	0.493	1.509	2.891

Результати представлені в таблиці 2.3 показують відсутність чіткої залежності числа циклів обміну φ від об’єму буфера обміну ω. Це підводить до висновку про раціональність вибору такого значення ω яке дорівнює середньому обсягу контекстної інформації для ключового слова.

Ключовим показником за яким визначається ефективність запропонованої організації словника з використанням хеш-адресації без колізій. прийнято вважати коефіцієнт використання адресного простору хеш-пам'яті β. У контексті проектування програмного забезпечення. коефіцієнт β — це відношення υ клітинок. зайнятих посиланнями на базову адресу. до загальної кількості хеш-комірок.

Для експериментального дослідження впливу значення β на основні показники ефективності електронного словника проведено статистичне моделювання. результати наведено в табл. 4. контекстну інформацію. яку він містить. ψ – відношення ємності пам’яті переповнення до ємності хеш-пам’яті.

Таблиця 2.4

Залежність ефективності використання пам’яті та кількості циклів обміну від коефіцієнта адресного заповнення простору хеш-пам’яті

β	Ймовірність зберігання всієї контекстної інформації ключового слова в хеш-пам’яті	φ	ψ	α
0.001	0.436	1.100461	0.026349	0.099417
0.002	0.393	1.124957	0.108774	0.183199

Продовження таблиці 2.4

0.003	0.362	1.164662	0.071992	0.282691
0.004	0.332	1.188315	0.295162	0.321932
0.005	0.302	1.228969	0.138845	0.443438
0.006	0.254	1.257986	0.384826	44078132
0.007	0.221	1.276357	0.295726	0.549878
0.008	0.191	1.323034	0.75484	0.463059
0.009	0.167	1.331641	0.728821	0.526848
0.010	0.141	1.378572	0.738741	0.581087
0.011	0.12	1.392039	0.728089	0.643327
0.012	0.114	1.404917	0.714208	0.707058
0.013	0.103	1.500654	0.751686	0.749508
0.014	0.086	1.454478	0.738964	0.813068
0.015	0.057	1.492829	0.779215	0.851498

У запропонованій технології організації електронного словника кількість φ циклів підзавантажденя даних з пам’яті нижніх рівнів визначається значенням коефіцієнту β завантаження адресного простору хеш-пам'яті. Деталі на яких було висловлено це твердження зведено до Таблиці 2.4. Найбільш доцільно виконувати оцінку роботи такої організації електронного словника за допомогою порівняння з знаними існуючими розробками відповідно до виділених критеріїв оцінки.

Порівняльний аналіз продуктивності вимагає визначення кількості циклів обміну φ для яких доступна контекстна інформація.

При аналізі електронних словників інтелектуальних комп’ютерних систем перекладу котрі побудовані на деревовидних структурах кількість циклів обміну φ в межах яких яких здійснюється під-завантаження даних до пам’яті вищих рівнів сильно залежить від обраної деревовидної структури. Для прикладу для збалансованого бінарного дерева середня кількість пам’яті доступу задається формулою log2υ Для нестійких та інших варіантів побудови дерева це значення. швидше за все. буде вищим.

Під час пошуку контекстної інформації для певного слова в дереві кожен крок пошуку залежить від попереднього кроку що призводить до серії звернень до пам’яті. Також це вводить великі обмеженя в можливість оптимізації процесу пошуку.

Перший підхід, який пропонується розглянути, це організація електронних словників на основі деревовидних структур. Це метод організації із збереженням розподіленого еталонного дерева та контекстної інформації, де кожен вузол такого дерева містить підслово маршрутизації та адресні посилання на наступні вузли. Відсоткова частка службової інформації в таких словниках рідко перевищує 4% від обсягу всього словника. Решта пам'яті словника містить контекстні дані. Такі словники є дуже ефективними з точки зоку використання пам'яті адже не потребують її розрідження для конфігурації їх ефективності.

Іншим підходом який потребує бути розглянутим є ще одно організації електронних словників на основі дерева. Проте в цьому випадку зберерення контекстних даних відбувається оркемо від дерева посилань. Ефективність споживання пам'яті таких словників навіть вище за попередні. Індекс надлишковості пам'яті λ дорівнює відношенню + 6 до і становить близько

1,01 для еальниз контекстних словників. При цьому вони набагато більш швидкодіючі порівняно х попереднім варіантом, адже доступ до даних не потребує завантеження котекстної інфорації інших слів.

Наступною технологією яку доцільно розшлянути є технологія лектронних словників для якої хеш-пошук з колізіями, які розв’язуються за допомогою технології зондування. ДЛя таких словників перебачається окреме зберігання контекстних даних від службової хеш-таблиці з адресамми для навігації простогором власне контекстної інформації. Ця технологія передбачає такий порядок дій про пошуку. Щоб отримати доступ до контекстних даних слова, спочатку вичитується місце хеш-пам’яті, яке містить посилання на фактичний блок контекстних даних для заданого відповідного ключового слова. у наведеному прикладі. Розмір такої комірки 14 байт. Тому для $\beta = 0,65$ коефіцієнт резервування пам'яті λ виходить рівним 1,04.

Провіши детальний аналіх отриманих результатів, котрі зведено до таблиці 2.5, було чітко підтверджено що запропонована конфігурація електронного словника є конкурентноспроможною в порівнянні з аналогами та придатною до використання системами комп'ютернног перекладу.

Таблиця 2.5

Порівняння ефективності запропонованого методу організації електронних словників з існуючими

Оорганізація електронного словника	Критерії ефективності			
	Швидкість пошуку за клчем		Надлишковість споживання пам'яті	
	φ	φ/φ_0	λ	λ/λ_0

Продовження таблиці 2.5

Електронний словника на основі деревовидноїх структури з записом контекстної інформації в вузлах	14.29	9.121	1.01	0.58
Електронний словник на основі дерева з винесеним записом контекстної інформації від дслужбового дерева	5.3	3.47	1.04	0.59
Електронний словник на основі хеш-пошуку з колізіями та винесеним записом контекстної інформації від службової хеш-таблиці для збереженя адрес	2.1	1.39	1.04	0.561

Використання запропонованої моделі електронного словника, як частини комп’ютеризованої системи контекстного перекладу, дозволяє забезпечити найвищу швидкість пошуку в порявнянні з розглянутими аналогами. Такий ефект було отримано завдяки використанню технології хеш-пошуку, досі маловживаної тезхніки при розробці електронних словникаів систем комп’ютерного перекладу. Ціна за досягнуту продуктивність дещо більше надлишковість сподивання пам'яті. Проте такий

розподілиресурсоефективності цілком виправданий в сучасних умовах здешевлення апаратної пам'яті.

Висновки до розділу 2

У результаті проведених досліджень в рамках другого розділу магістерської роботи, метою яких була а розробка структури електронного словника з безконфліктною хеш-адресацією, а також математична модель і алгоритм роботи в системі перекладу цього словника. За результатими цього можна зробити такі висновки:

1) Для прискорення пошуку в електронних словниках пропонується використовувати безколізійну хеш-адресацію в розріджених адресних просторах. Щоб використання пам'яті була ефективним, пори її розріжденність пропонується використовувати хешпам'ять також і як простір для зберігання контекстних даних. А саме рекомендується заповнювати пробіли між хешами, контекстною інформацією ключових слів.

2) В результаті роботи над розробкою організацією електронних словників з використанням безколізійних хеш-адрес, було визначено чотири формати подання даних в такому словнику, які здатні забезпечити ефективну роботу електронного словника як частини системи комп'ютерного перекладу.

3) Розроблено порядок розмішення данних в хеш-пам'яті, які дозволяє оптимізувати доступ до них в процесівзчитування враховувючи ієрархічну будову пам'яті в сучасних комп'ютерних системах. Це в свою чергу допомагає значним чином зменшити час, необхідний для вичитування контекстної інформації з електронних словників.

4) Висунуто теоритичне обґрунтування вищеописаних ідей, а також розроблена математично модель елеектронного словника що базується на них. Теоретично доведено відсутність технічних обмежень для реалізації такого словника для ряду популярних сьогодні оптимізації обчислювальних платформ.

5) Взявши за основу розроблену математичну модель було здійснено оцінку ефективності запропонованої конструкці електронних словників. Було виявлено що за рахунок збільшення обсягу пам'яті, нвсього на на 20% можливо досягти удвічі більшої швидкості пошуку порівняно з знаними способами організації електронних словників.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО КОМПЛЕКСУ МОДЕЛЮВАННЯ РОБОТИ ЗАПРОПОНОВАНОГО ЕЛЕКТРОНОГО СЛОВНИКА

В рамках третього розділу магістерськоїх роботи пропрноується опис комплексу програмного забезпечення, що був розроблений для оцінки запропонованої конструкції електронного словника шляхом експерементального дослідження. ПЗ що є розультатом виконаної в рамках цьогорозділу робботи покликане вирішувати наступні задачі:

- Статистичне тестування ефективності розробленого електронного словника в обробці результатів моделювання моделювання доступу до словника.

- Експериментальна перевірка наявності залежностей між різними параметрами хеш-пам'яті. Віднахдження цих залежностей відіграє ключову роль в у процесі більшої оптимізації запропонованого рішення для організації електронних словників.

В якості мови програмування дбула обрано C++. Такий вибір обгрунтовуєтсь тим що ця мова дозволяє нв найбільш повному обсязі користуватися апартними можливостями обчислювальної платформи, але й при цьому пропонує зручні інструменти розробки програмного забезпечення в парадигмі об'єтно-орієнтованого програмування.

3.1 Налаштувння параметрів словника

Ключоваою характеристикою хеш-пам'яті в контексті електронних словників є кількісний обс збережених ключових слів. Розроблене прогармне забезпечення передбачає функцію зміни біжаною кількость слів в електронному словнику. Змінна в якій зберігається значеня кількості слів в словнику позначається keyWordsAmount.

Оскільки було прийнято рішеня реалізовувати механізми налаштування даного програмного комплексу згідно до потерну програмування декоратор, то доступ до цієї змінною слід здійснювати через метод setKeyWordsAmount(int).

Іншим основним параметром хеш-пам'яті є коефіцієнт завантаження. Він виражає співвідношення між кількістю інформаційних об’єктів, які вже записано до хеш-пам'яті до заданої верхньої межі кількості таких об'єктів в хеш-пам'яті.

Запропонована конфігурація хешу використовує два коефіцієнти навантаження: фізичний та інформаційний. Коефіцієнт завантаження хешу, який позначено змінною physLoad типу double позначає відношення обсягу корисної інформації, бу розміщена в зоні хеш пам'яті відносно повного об'єму хеш пам'яті. Коефіцієнт завантаження хешу, який зберігається в змінній infoLoad типу double, визначає відношення кількості слів, за якийми буде відбуватись пошук, до загальної кількості слів, які фізично можуть бути збережені в хеші. Це і є інформаційним коефіцієнтом завантаженя. Є очевидним те, що другий коефіцієнт значною більший від першого. Міра відношення першого коефіцієтту до другого напряму залежить від того наскільки обсяг контекстною інформації словника перевищує обсяз аргументів пошуку. Іншими словами, для розробленого програмного

комплексу обсяг хеш пам'яті, нехай буде позачено змінною hashMemSize типу double, може бути обчислено як hashMemSize = keyWordsAmount / infoLoad. Це підтверджує залежність обсягу хеш-пам’яті встановлену в хзоді виконання другого розділу цієї магістерської роботи.

Значення змінних infoLoad та physLoad обчислюються з використанням інших конфігурацій що може вносити користувач,. Ці дані від не налаштовує, але має доступ до них за геттерами getInfoLoad() та getPhysLoad() що повертаєть значення типу double.

Проте для зручності роботи користувач має вказати свої обмеження до hashMemSize, тобто встановаити максимальний допустимий розмір хеш-пам’ятв. Зазвичай це значення диктується об’єктивними технічними обмежеенями комп’юторнеї системи. Для цього користувач має викликати методі setMaxHashMemSize() передавши верхн можливу межу розміру хеш-пам’яті в якості параметру типу long в байтах.

Іншим значущим параметром, що має приналедність до параметрів налаштування конфігурації електрронного словника є математичне очікування довжини контекстної інформації пов'язаної зі словами. Тобто сережднь остсатист ичне значення розміру контекстної інформації клчових слів. Позначено цей параматр змінною contextAverageSize типу double. Задавати параметр потрібно за допомгою методу setContextAverageSize(double).

Іншим значущим параметром, що має приналедність до параметрів налаштування конфігурації електрронного словника є математичне очікування довжини контекстної інформації пов'язаної зі словами. Тобто сережднь остсатист ичне значення розміру контекстної інформації клчових слів. Позначено цей параматр змінною contextAverageSize типу double.

Задавати параметр потрібно за допомгою методу setContextAverageSize(double).

Для цілей статистичного моделювання програма створює характеристики розподілу для кожного слова, що зберігається в електронному словнику. Для цього використовується адресна таблиця addressTable для зберігання адрес заданої кількості keyWordsAmount в електронних словниках, які будуть використовуватися як ключі пошуку в розробленій програм. Кількість бітів коду адреси, що записана в таблиці addressTable, визначається виділеним об'ємом пам'яті і може бути обчислена як log2(hashMemSize).

3.2 Розробка програмних компонентів

Основний цикл статистичного моделювання задає хеш-адреси слів та обсяг їх контекстних даних, тобто ключові параметри даних що ззберігаються словником. рийнята модель передбачає, що хеші в адресному просторі розміщенні рівномірн. Для цього в якості хеш-перетворвача використовуєтсья спеціальний програмний модуль який здатний випадковим чином генерувати хеш-адреси. Тобто здійснювати хеш-перетворення використовуючи в якості аргументу ключвоі слова і резуолтатом якого ї рівномірно розташовані в адресному просторі.

На другому етапі програми здійснюється генерація великого обсягу контекстної інформації для кожного терміну пошуку. технічго процес генерації в математичному моделванні представляє собою почергову генерацію випадкових чисел. Для генерації цих контекстних даних використовується 14 вбудованих генераторів вибадкого бітового значення.Згідно з центральною граничною теоремою, результати такого генератоа відповідають нормальномцу розподілу. Для цього програмою використовується дві змінні.

Перша matExp містить значення математичного сподівання, а інша stdError відображає значення середнього квадратичного відхилення. Обидві змінні є приватними і не можуть бути отрмані ззовні.

Головна якісна перевага хеш-адресації без колізій полягає в тому, що час пошуку за заданим словом-аргументом визначається часом разового звернення до пам’яті. Таким чином досягається висока швидкість пошуку що не має залежності від розміру пошукової матриці [55].

Існує багато можливих рішень завдання отримання безконфліктного хеш-перетворення для заданого масиву термінів пошуку в електронних словниках. Проте має бути врахований факт того що обсяг використовуваних обчислювальних ресурсів обмежений. Загалом складність вищенаведеної проблема є експоненційною. Однак це вірно, лише якщо коефіцієнт заповнення хешу близький до 1. Якщо коефіцієнт завантаження невеликий цю властивість можна використовувати для легкого створення хеша без конфліктів, для прикладу використавши мультиплікативні перетворення на скінченних полях Галуа.

Таблиця metaTable зберігає діаграму розташування записів у хеш-пам'яті і вона заповнюється рекурсивно в процесі зберігання даних словника в пам'яті. Щоб уникнути зіткнень і шукати ключі якнайшвидше, процес зберігання даних у словнику починається зі збереження кожного шуканого слова в хеші з використанням хеш-адреси, наведеної в таблиці addressTable. Якщо таких колізій не відбувається, тобто якщо контекстна інформація вільно знаходиться в діапазоні адрес між сусідніми словами, запис створюється та зберігається в таблиці metaTable. Процедура здійснюється ітеративно для кожного з keyWordsAmount слів.

В разі якщо контекстні дані та пробіли не можна вставити між суміжними адресами слів-аргументів пошуку, виконується процедура

фрагментації запису контекстних даних. Для цього максимальний розмір фрагмента визначається кількістю слів, розміщених в одному стоп-сегменті. Це призначено для швидкого аналізу контекстної інформації, зменшуючи можливість заміни кількох сегментів у швидкому буфері.

Для переходу між фрагментами використовуються адресні посилання. Коли пам’ять заповнюється словами та контекстних даних. Також це відображає статистичні дані про середню довжину фрагмента і середню кількість фрагментів, які використовуються для зберігання контекстної інформації, дотичної з пошуковими термінами.

Розроблене прогрмане забезпечення враховує ієрархічну організацію пам’яті сучасних комп’ютерних систем та імітує розміщення та пошук слів з відповідною контекстною інформацією. Таким чином, модель, використана в цьому дослідженні, має два рівні ієрархії пам’яті. Першим з ни є швидкісні буфери, які можна умовно асоціювати з кеш-пам'яттю. Інший являє собою на кілька порядків гірші, але в рази більші, які можна співвіднести з основною пам'яттю в сучасних комп'ютерних системах. Метою цієї конфігурації є зменшення кількості циклів обміну між високорівневою та низькорівневою пам’яттю в процесі обробки контекстної інформації кожного слова.

Значну частину розробленого програмного комплексу становить статистичне моделювання випадкових потоків слів в електронних словниках. Розроблена програма пропонує два типи такого моделювання: пошукові слова визначаються за допомогою частотних таблиць слів у реальних вправах комп’ютерного перекладу або ж псевдовипадковим чином за допомогою вбудованого генератора. Перший підхід дозволяє більш реалістично оцінити ефективність використання хеш-адресації в електронних словниках для прискорення процесу пошуку. Мультиплікативні операції на полях Галуа використовується як хеш-перетворення без колізій для попередньо визначеної

таблиці ключів. Ці слова шукаються в електронних словниках, щоб забезпечити хороший розподіл статистичних даних і можливість гнучко налаштовувати їхні хеші.

Функція long multiMont (long x, ong y, long sigma, int n) виконує прискорене множення поля Галуа за допомогою скорочення Монтгомері. Ця функція має певні параметри, так як: sigma — числовий код, що відповідає поліному генератора поля Галуа, n-й порядок поліному P(x) генератора поля Галуа,a і y — коефіцієнти, . Ця функція повертає добуток у просторі поля Галуа.

Функція multiMont організовує цикл, який проходить по числах множника y, починаючи з найменшого. Отже, цикл виконується стільки разів, якого порядку є поліном. На кожному колі циклу проводиться перевірка значення поточного розряду множника i, якщо воно дорівнює 1 =, множникх додається до проміжного результату r. Після такого сумування, якщо молодша цифра проміжного результату дорівнює 1, до неї плюсується число, що відповідає поліному генератора заданого поля Галуа. Після підрахунку проміжних результатів робиться корекційна поправка. Обернене до цього числа отримано циклічним методом [53]. А отже, це означає, що зведення розгорнутої функції здійснюється безпосередньо під час множення, на відміну від традиційної схеми, де ппочергово спочатку виконується множення, а потім зведення. Такий інноваційний підхід дозволив уникнути компіляції продуктів з розрядністю більше 64 бітів. Таким чином, виходить що операція спрощення полінома полягає в підсумуванні числа, що відповідає твірному поліному, до поточного залишку. Ця операція передбачає знаходження позиції старшої цифри поточного залишку зсув коду полінома генератора та та його додавання до поточного залишку.

Тобто, у середньому для виконання декременту потрібні n-розрядні тестові операції, 2×n операцій зсуву (зсув поліноміального коду генератора та тестового коду, що містить 1 одиницю) і 0,5×n логічних операцій АБО має бути виконано. Це зменшує кількість логічних операцій для зведення поля Галуа у квадрат порівняно з множенням різних чисел. Середня кількість логічних доповнень, які можна усунути без погіршень ефективності та точності, за допомогою певних властивостей полінома в квадраті, становить 0,5n, з врахуванням припущення, що значення бітів у значущих бітах однаково можливі нулі або одиниці. Це також усуває операції розбору з кількома зсувами та найменшими бітами, що виконуються в кожному з n циклів.

Функція int ifOrtogonal(int *h, int n) використовується для визначення ортогональності квадратних масивів. Індекс h є початком координат, а кількість стовпців і рядків становить n. Шукаються всі можливі значення j від 1 до 2n-1. Двійкове представлення даного числа j представляє рядок елементарної матриці додавання. Це означає, що для кожного значення j деякі підмножини рядків матриці підсумовуються та порівнюються з нульовим рядком. Функція повертає значення null, якщо вказаний рядок знайдено. Функція повертає 1, якщо обчислено всі можливі суми рядків указаної матриці, і ці суми не дорівнюють нулю.

Функція long expInc (long x, ong e, long sigma) призначена для експоненціального зростання кінцевих полів Галуа. Параметри функції: x – число, яке потрібно звести до степеня, e – індикатор степеня, sigma – число, яке корелює з поліномом генератора поля Галуа. Масштабування поля Галуа використовує варіант класичного алгоритму з передачею двійкових чисел, починаючи з найдавнішого. Ця варіація спрощує процес експонування, оскільки цикли експонування передбачають множення на фіксовані числа. Для виконання цього множення використовується варіант, поєднаний із

скороченням Монтгомері. Це пояснюється тим, що коригування можна зробити лише один раз.

Функція unsigned long multFinite (long x, long y, long sigma) виконує множенняd в просторі заданого кінцевоно поля Галуа. В якості параметрів в цю функцію передаться Параметри цієї функції визначаються наступним чином: коефіцієнти та числовий код, корельований з поліномом, що генерує поле Галуа. Множення виконується за класичним алгоритмом з диференціюванням циклів множення поліномів і зведення з поліномами, що утворюють поле. Якісний виграш від використання цієї функції полягає в тому, що ця функція не вимагає виправлення додаткового множення на обернене множення 2c. А дже конверсія – це досить складне завдання, яке потребує значних часових ресурсів. При цьому програма використовує функції множення з модифікаціями. Якщо множення виконується одноразово, то зручніше використовувати розроблені функції multFinite. Коли операція множення використовується як ступінь степеня), більш вигідно використовувати опцію множення в комбінованому скороченні Монтгомері, оскільки робиться лише одне виправлення. Коли операція аналізу множника вимкнена, корекція Монтгомері може бути виконана негайно з урахуванням двозначних значень поточного коду залишку.

Висновки до розділу 3

Результатом проведених експериментальних досліджень є третя частина цієї магістерської роботи. Метою цього розділу було створення програмного забезпечення для виконання дослідження і оцінки ефективності запропонованої конструкції електронного словника. За підсумками виконання цієї частини роботи, можна зробити такі висновки:

1. Було розроблене ПЗ (програмне забезпечення), за допомогою якого стало можливим статистично моделювати роботу електронних словників які побудовані за запропонованим методом з використанням хеш-адресації без колізій. Це дозволило експериментально визначати зв'язки між заданими параметрами електронного словника. Результати цього дослідження та їхній подальший детальний аналіз допомогли оптимізувати використання запропонованої ідеї з врахуванням об'єктивних практичних обмежень ресурсів (обчислювальних потужностей, об'єму пам'яті).

2.Практичним шляхом було визначено, що хороший результат розподшлу хеш-адрес в пам’яті можна забезпечити використавши і якості хеш-перетворення поліном Галуа. Іншими словами мультиплікативні перетворення на скінчених полях Галуа. Такий вибір хеш-функції здатний гарантувати хороші статистичні дані розподілу, і окрм цього такі хеш-функції відразу мають механізм гнучкого налаштування.

3. За допомогою розробленого ПЗ було визначено зв'язки між кількістю слів в електронному словнику, коефіцієнтом завантаження хеш-пам'яті і часом виділеним на підбір досконалої хеш-функції. Результати отриманні на даному етапі експерементального дослідження підтвердали аналітичні розрахинки оцінки ефективності запропонованої колнструкції

електронного словника що було отримано в рамках другої частини цієї магістерської роботи, тобто на теоретичному етапі.

4. Експерементально підтверджено прискорення доступу до даних для запропонованої конструкції словника в порівнянні з нявнми аналогами.

5. За допомогою розробленого програмного забезпечення встановлено зв'язки між рівнями пам’яті, розміром блоку обміну, та рівнем фрагментації контекстної інформації в електронних словниках. Це дозволяє оптимізувати певні параметри словника в процесі йог розробки, щоб максимізувати швидкість пошуку найбільш відповідного перекладу.

РОЗДІЛ 4

РОЗРОБЛЕННЯ СТАРТАП–ПРОЕКТУ

4.1. Опис проблеми

Впродовж останніх двох десятиліть людство спостерігає за ти як відбуваються динамічні зміни у світі технологій. ВДедалі частіше Країни Сходу у численних галузях досліджень виходять на перший план. В контексті глобалізаціх це лише посилює інтерес до і так гострого питання питання інформаційного метаболізму з рештою світової спільноти.

На шляху до нарощення обсягів інформаційних потоків між країнами Заходу та Сходуо однією з клчовиз перепон є мовний бар'єр. Одним з перших можливих кроків на шляху до подолання цієї проблеми є об’єднання зусиль спеціалістів як з галузі лінгвістики., так й інформаційних технологійю Наступним послідовним кроком можна виділити проведення критичного аналізу компонентів необхідних для швидкого та точного перекладу. Виявивши найбільш вразливі станом на сьогодні компоненстти систем перекладу, в контексті їх швидкодії, за умови нарощення обсягу контекстних даних перекладу стане можливим їх подальший детальний розбір та оптимізація.

Проведений в рамках цієї магістерської роботи аналіз показав що одними з основних компонентів систем перекладу шо потребують вдосконалення та прискореннння є електронні словники. Це підводить до висновкку одніє з умов виникнення можливості збільшення інформаційних потоків між країнами Заходу і Сходу є створення нових, якісно більш гшвилдкодіючих електронних словників що можуть бути використані в складі систем ітнелекнуалізованого комп’ютерного перекладу.

Отже обрана наукова задача віднаходження методів прискорення пошуку в електронних словниках, які можна використати в якості складових систем комп'ютерного та комп'ютеризованого перекладу має практичний сеснс та є актуальною для сучасного етапу розвитку інформаційних технологій.

В рамках магістерської роботи була розроблена конструкція електронного словника на основі хеш-адресації без колізій. Такий словник передбачений для моливості легкої інтеграції в популярні типуи систем комп’терного перекладу. Основні напрямки використання запопонованої розробки та її переваги , які користувач отримає від її застосування на практиці було зведено до таблиці 4.1. Запропонована розробки має високу практичну цінність для широкого кола користувачів для яких важливим є досягнення якшавищої швидкодії при пошуку в електронних словниках., що можуть бути інтегровані як моддуль стстеми комп’ютерної оьробки тексту, інтелектуалізованого перекладу, тощо.

Було виконана аналіз технічних та економічних переваг запропонованої ідеї. А також виконанао критичне поівняння запропонованоїх можлі електронного словника з існуючими аналогами. В ході цього аналізу було здіснено наступне:

• визначено ряд техніко-економічни характеристик, обосливостей та властивостей запропоноваої ідеї;

• визначено наявних станом на сьогодні конкурентів ринку. Для цього було поаналізовано як вже добре відомі подукти, так і подібні діеї що перебувають в стані розробки чи висунення стартапу;

• було збрано, класифікоано та оброблено чисельну кількіть даних техніко-економічних показників проектів конкурентів. Також було проведено оцінку запропонованї ідеї за тими ж критеріями;

• виконано порівняльний критичний аналіз критеріїв та показників запропонованої ідеї зщо що мають а) гірші значення (W, слабкі); б) аналогічні (N, нейтральні) значення; в) кращі значення (S, сильні). Результати вищеописанного аналізу було зведено до таблиці 4.1.

Таблиця 4.1.

Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Організація електронного контекстного словника системи ітелектуалізованого комп’ютерного перекладу.	1. В системах котекстного комп’ютерного перекладу.	Підвищення якості та релеватності отримуваного комп’ютерного перекладу за рахунок використанні більш швидкодіючих електронних словників. І як наслідок, можливості перебору більшої кількості альтернативних варіантів при виборі найоптимальнішого з них.

Продовження таблиці 4.1.

	2. В системах аналізу тексту з метою виправлення стилістичних, граматичних, пунктуаційних та інших помилок. Інакше кажучи в системах редагування тесту	Покращення якості редагування тестів та підвищення адекватності виправень, що досягатиметься за рахунок процесування більшої кількості контекстних даних за один і той самий чаас з використанням запропонованої моделі швидкодіючого електронного словника в порівнянні з аналогами.
	3. В системах перевірки академічної доброчесності чи будь-яких системах аналізу тескту на оригінальність.	Підвищеня точності отримуваних результатів перевірки тексту на оригінасльність за рахунок можливості пошуку відповідників серед більшої кількості даних за аналогічний час що й при використання конкурентних розробок.

4.2 Аналіз ринкових можливостей запуску стартап-проекту

Основними характеристиками розробленого проекту організації електронного словника що може бути використаний в якості компоненту системи перекладу, комплексу редагування тексту чи системи перевірки тексту на плагіаь можна вваважати:

- Швидкість доступу до данних - виміри подані у відсотковому співвідношщенні даних конкурентів ринку до досягнутих показників для запропонованої розробки. Відповідно запропонована розробка оцінена в 100%.
- Ефективність використання пам'яті - дані наведено в відсотковому співвідношенні корисних контекстних даних словника до повного об'єму його пам'яті включаючи як пусті незаповнені облесті пам'яті так і службову інформацію.
- Ефективність роботи з великими об’ємів контекстних даних - оцінка здійснювалась в порявняні з запропонованою моделю словника, відповідно мій проект мав коефіцієнт 1.
- Ефективність роботи з великими пошуковими масивами ключових слів - ооцінка здійснвалась на основі співвідношення змін в швидкодії словника від збільшення кількості клчових слів що він зберігає. Для словників засновних на хеш-адресації при збільшенні слів об'єм пам'яті збільшувавсяттаким чином зщоб зберігалис рекомендовані інструкцією до їх моделей заповнення пам'яті.

Вирішення компромісу між швидкістю доступу до даних і об'ємом використання пам'ят в конукуруючих проектах реалізовано також на достатньо високому рівні, але вони здатні забезпечити на порядки меншу оптимзації пошуку конктекстних даних. А саме це є критичним на сьогоднішній момент в контексті розвитку сисстем комп'ютерного перекладу.

Таблиця 4.2.

Визначення сильних, слабких та нейтральних характеристик ідеї проекту.

№	Техніко-економічні характеристики ідеї	Мій проект	Проект конкурент GRAMR (на основі деревнох структури)	Проект конкурент TOUC (на основі хеш адресації з колізіями)	W	N	S
1	Швидкість доступу до даних	100%	35%	80%			+
2	Ефективність використанні пам’яті	65%	97%	80%	+		
3	Ефективність роботи з великими об’ємів контекстних даних	100%	-100%	100%			+
4	Ефективність роботи з великими пошуковими масивами ключових слів	60%	15%	30%			+

Також значна частина з конкуруючих розробок сильно втрачає в швидкодії при нарощеннні об'єму контекстних даних що відповідають ' слову пошуку. Це унеможливлює їх використання для систем редагування текстів і перевірки текстів на оригінальніть. Також це обмежує можливості систем кмп'ютерного перекладу до підбору найбільш релевантних опцій. Отож, попри те що запропонований в проекті метод потребує дещо більших ресурсів пам’яті, він забезпечує високу швидкодію іє конкурентноспроможним на ринку електронних словників.

Наступний етапом роботи над стартапом є аудит технології, з використанням якої можлива подальша реалізація проекту. Очікуваним отримати в розультаті проекту програмний комплекс. А детермінація його придатності до технічної реалізації переждбачає оцінку таких складових :

• яка технологія є найбільш дало для реалізації програмний продукту для заданої ідеї проекту ?

• чи є в наявності вже часткові готові рішення якими можна користуватись для реалізації проекту ?

• чи в авторів проекту є доступ та право користувтися тими технологіями?

В результаті проекту очікується отримати готове програмне забезпечення, яке електронний контекстний словник на основі хеш-адресації без колізій. В якості хеш-перетворення було обрано стандартизований шифроблок алгоритмів симетричного криптографічного шифрування, який є достатньо дослідженим і доступним в швидкодіючих апаратних реалізаціях, зокрема такі шифри як DES AES.

Для практичної реалізації запропонованої організації електронного словника ряд програмних засобів незалежних від типів обчислювальних платформ. Тобто, розробка такого програмного продукту передбачає

використання добре відомих інструментів та парадигмпрограмної інженерії. Більшість тезнологій небхідних для реалізації запропонованої ідеї є широковідомими та загальнодоступними.

Наступним кроком в розробці стартапу є оцінка ринкових можливостей програмного забезпечення що реалізує ідею проекту. Цей крпок виконується з метою оцінки можливосте які в подальшому можна використати під час ринкового просування проекту, та для оцінки факторів ризику на ринку, які можуть перешкодити успішному просуванню проекту. Виконання такої оцінки дає змогу ліпше спланувати керунки розвитку проекту і при цьому врахувати не лише сучасний урахуванням стану ринкового середовища, а й потенційні потреби клієнтів та можливі вектори розвитку проектів-конкурентів.

Дані таблиці 4.3 показують, що в загальному, за початковою оцінкою, ситуація є привабливою для виходу на ринок електронних швидкодіючих контекстних словників.

Таблиця 4.3.

Попередня характеристика потенційного ринку стартап-проекту

№	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	150
2	Загальний обсяг продаж, (для сектора)	65000
3	Динаміка ринку (якісна оцінка)	Зростає

Продовження таблиці 4.3.

4	Наявність обмежень для входу (вказати характер обмежень)	Велика кількість компанй звикла звертатись до рішень на основі деревних структур через їхню зрозумілість в контексті сприйняття таких словників людьми.
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі (або по ринку), %	27%

Особливої ринкової привабливості проекту додає наявний сьогодні розвиток систем корегувавння помилок тексту на основі аналізу великоїх кількості контекстних даних що містять приклади вживання шуканих мовних контрукцій.

Також великою популярність користуються системи інтелектуалізованого контекстного перекладу. Згідно з [39] створення таких комплексів для масового обміну науково-технічною інформацією країн мови яких належать до різних мовнизх груп (як наприклад між країнами Сходу та Заходу) лишатиметься пріоритетним напрямком найближчого десятиліття. Одне з ключових місць в таких ситемах посідають електронні контекстні словникі., а від їх швидкодії залежить адекватність перекладую

Наступним невід'ємним кроком у розробці стартапу є визначення потенційних клієнтів на ринку програмних продуктів.

Таблиця 4.4

Аналіз ринку збуту

№	Потреба що вормує ринок	Цільові сегменти ринку	Відмінності у поведінці різних цілоьових груп	Вимоги споживачів до товару
1	Потреба покращення релевантності контекстного перекладу	Системи комп1ютерног о переклдаду	Жорсткі часові обмежунення	Максимальна швидкість покуку за заданим ключем
2	Потреба підвищеня якості корекції помилок	Системи корегування помилок тексту	Необхідність аналізу значних об’ємів даних за умови значних часових обмежень	Можливість швидкого доступу о значної кількості контекстних даних
3	Пришвидшеня і покращення якості перевірки тесту на пагіат	Наукові оргацізації, університети, дослідницькі центри	Необхідність аналізу великого об’єму даних	Можливість пошуку в дуже великих об’ємах даних

Також великий ітерес до розробки проявляють університити, інститути, наукові оргацізації, дослідницькі центри, центри акредитації освіти через можливість даної розробки бути використаною в складі комп'ютерних систем перевірки на академічну доброчесність.

Крім цього існує певний спектр компаній що займаюсть створенням систем коригування граматичних, стилістичних, лексикографічних та інших помилок тексту. В цих системаз такоєж є гостра пореба пришвидшення доступу до даних контекстних соловників з метою поліпшеня кості сервісу що вони надають.

Таким чином, можна дійти висновку що запропонований проект задовольняє інтереси кількох великих груп учасників ринку і при цьому є спроможним до чесної ринквої конкуренції. У таблиці 4.5 подано підсумковий аналіз SWOT і вигляді матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities).

Таблиця 4.5

SWOT- аналіз стартап-проекту

Сильні сторони:	Слабкі сторони:
Запропонована організація електронного словника забезпечує вдвічі більшу швидкодію в порівнянні з відомими існуючими роззробками. Така швидкість пошуку досягнута завдяки втикористанню хеш-адресації в якості технічного рішення організації електрооного словника.	Слабкою стороною запропонованого електронного словника є потреба в надликовому викорстанні пам’яті задля забезпеченння ефективної роботи хеш-адресації без колізій. Також відповідальність за захищенність зберігання секретних ключів методу несе користувач.

Продовження таблиці 4.5

Можливості:	Загрози:
Розширення ринку збуту не лише в в системах комп’ютерного перекладу, а й в таких системах обробки тексту як сервіси з коригування правопису та системи перевірки текстів на оригінальність. Останні є дуже затребуваими в сьогоденних реаліях частих випадків плагіату та недоброщесного використання чужої інтелектуальної власності.	Використання переваг швидкодіючого словника потребує попереднього підборуу хеш-перетворення для заданого масиву слів. Тобто для укладання кожного окремо го словника ця процедура виконується незалежно і з початку. Це унеможливлює легко коригування масиву слів вже наявного словника.

Висновки до розділу 4

Проведення дослідження, спрямованого на визнання та оцінку можливості ринкової реалізації розробленого проекту, яким є магістерська робота, дозволяє зробити наступні висновки:

1. Розроблене програмне забезпечення та ідея на його основі мають високу конкурентоспроможність у зв’язку з тим, що вони здатні забезпечити на порядок вищу ефективність щодо прискорення пошуку в електронних словниках, що є основою для гейт-дослідження. Крім того, розроблений метод конкурує з існуючими проектами в області забезпечення безпеки оброблюваних даних. Крім того, розроблений проект електронного словника враховує специфіку ієрархічної структури пам’яті сучасних комп’ютерних систем і дозволяє пришвидшити доступ до контекстної інформації електронного словника за рахунок скорочення часу зберігання.

2. Широке коло потенційних клієнтів та покупців програмних продуктів для швидкого пошуку в електронних словниках. Основними учасниками ринку, зацікавленими в запропонованій розробці, є компанії, що займаються якісним інтелектуальним контекстним перекладом, системами пошуку граматичних, синтаксичних і стилістичних помилок тесту з можливістю їх виправлення. Крім того, ринок може бути розширений науковими установами, дослідницькими центрами, університетами та невеликими академіями наук, які зацікавлені у використанні швидкого співтекстуального пошуку як частини систем перевірки тексту на оригінальність.

3. Виконаний аналіз засвідчив досить великий інтерес до теми на ринку конкурентних розробок. Проте більшість з низх мають в основі технологію

пошуку в деревоподібних структурах, що значно програють в аспекті шидкодії в порівнянні з використанням хеш-адресації.

ВИСНОВКИ

В результаті виконання магістерської дисертації, метою якої є підвищення швидкодії операцій пошуку в електронник словниках, орієнтованих на застосування в системах комп'ютерного перекладу отримані певні результати, які можуть бути сформульовані наступним чином:

1. Проведений аналіз особливостей сучасного етапу розвитку технологій комп'ютерного та комп'ютеризованого перекладу виявлено основні тенденції розвитку цього важливого классу систем обробки інформації показав, що найбільш важливими чинниками для їх швидкого вдосконалення є зростання характеристик компютерних систем, використання хмарних технологій, а також прогресивних технологій штучного інтелекту. Детальний аналіз застосування вказаних трендових технологій в системах комп'ютерного перекладу показав, що найбільш вузьким місцем при їх застосуванні залишається проблема пошуку в електронних словниках. відповідно, можна зробити висновок, що подальше вдосконалення систем перекладу неможливе без кардинального прискорення роботи електронних словників.

2. Здійснений аналітичний огляд існуючих способів швидкого пошуку слів в електронних словниках засвідчив, що вони не задовольняють сучасним вимогам в плані швидкодії, тобто їх використання не дозволяє аналізувати велику кількість варіантів контекстного перекладу, що є головним чинником якісного комп'ютерного перекладу.

3. Теоретично обґрунтовано, розроблено та досліджено запропоновано спосіб організації пошуку в електронному словнику, який відрізняється використанням хеш-адресації без колізій для наперед заданого фіксованого

набору слів, що забезпечує прискорення пошуку слова за рахунок того, що пошук здійснюється за одне звернення до пам'яті.

4. Розроблено та реалізовано метод підбору хеш-перетворення, що не утворює колізій для заданого постійного масиву ключів, в основі якого покладено використання єдиного адресного простору для зберігання в словнику ключових слів та контекстних уточнень. В якості хеш-перетворення запропоновано використовувати стандартизований симетричний шифр, а ключ - як механізм налаштування хеш-перетворення.

5. Розроблено математичну модель запропонованої організації хеш-пошуку без колізій в електронних словниках, яка дозволяє оптимізувати вибір параметрів організації з урахуванням обмежень на об’єм пам'яті та час віднаходження хеш-переторення, що не породжує колізій.

6. Розроблено програмне забезпечення для моделювання запропонованої організації хеш-пошуку без колізій в електронних словниках, яке дозволяє здійснити експериментальне дослідження показників ефективності.

СПИСОК ЛІТЕРАТУРИ

1. Pastor V. Searching Techniques in Electronic Dictionaries: A Classification for Translators / V.Pastor. A. Ampago // International Journal of Lexicography.- 2010.- Vol.23.- № 23.- P.307-354.

2. Marchuk U.N. Computational linguistics. –AST. Vostok-Zapad. 2001.-165

3. Marchuk U.N. Computational linguistics. –AST. Vostok-Zapad. 2001.-165

4. Chen W. Guided Alignment Training for Topic-Aware Neural Machine Translation / W. Chen. E. Matusov. S. Khadivi. J.-T. Peter // Association for Machine Translation in the Americas (AMTA). — jul 2016. — C. 300–311.

5. Humeniuk I.O. Hash search organization in e-dictionaries using block ciphers"/ I.O. Humeniuk. O.P. Markovskyi. O.M. Shevchenko // Conference of Information. Computing and Intelligent systems. —Kyiv. Ukraine. —2020. —C.34-42.

6. Гуменюк І.О. Метод віддаленої середньоарифметичної фільтрації зображень / І.О. Гуменюк, О.Є. Слюсаренко // Альманах науки.- 2019.- № 11 (32).- С.40-43.

7. Markovskyi O.P. Interactive template method of computer translation of scientific and technical publications / O.P.Markovskyi. O.M.Mykolayivna. F.Chunlei // Visnik of NTUU "Igor Sykorsky KPI" Informatika. upravlinnia ta obchisliuvalna tekhnika. —№59. —2013 . —C.86-97.

8. Agapova N.A. On the principles of creating an electronic dictionary of the linguocultorological type: to the problem statement / N.A. Agapova . N.F.Kartofeleva // Vestnik Tomskogo gosudarstvenogo universiteta. № 382 -2014.- .6-10.

9. Bender M.A. On the optimal time/space tradeoff for hash tables / M.A. Bender. M. Farach-Colton. J. Kuszmaul. L. Mingmou // STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing. — 2022.

10. Марковський О.П. Організація хеш-пошуку в електронних словниках / О.П. Марковський. О.М. Шевченко. Т.А. Руденко // Вісник Національного технічного університету України «КПІ» Інформатика. управління та обчислювальна техніка. — 2017. — №65. — С. 43–48.

11. Zuo P. A Level Hashing: A High-performance and Flexible-resizing Persistent Hashing Index Structure / P. Zuo. Y. Hua. J. Wu // ACM Transactions on Storage 15(2):1-30. — 2019.

12. Бойко О.А. Обґрунтування архітектури функції хешування з використанням паралельних обчислень / А.О. Бойко. І.Д. Горбенко // Вісник Харківського національного університету. — 2010. — №890. — С. 29–36.

13. Wang J. DHash: Dynamic Hash Tables With Non-Blocking Regular Operations / J. Wang. D. Liu. X Fu // IEEE Transactions on Parallel and Distributed Systems 33(12):1-1 — 2022.

14. Kuszmaul W. A Hash Table Without Hash Functions. and How to Get the Most Out of Your Random Bits / W. Kuszmaul // arXiv:2209.06038 — 2022. — С. 1–22.

15. Jongejan B. Automatic training of lemmatization rules that handle morphological changes in pre-. in- and suffixes alike. / B. Jongejan B. and H.Dalianis // Proceeding of the ACL-2009. Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics of the Asian Federation of Natural Language Processing. Singapore. – 2009. - P. 145-153

16. Knott G.D. Hash table collision resolution with direct chaining / Knott G.D.. de la Torre P. // Journal of Algorithms. —1989. — №10. — С. 20–34.

17. Lin Y. Learning hash index based on a shallow autoencoder / C. Rodriguez. Y. Lin. Z. Huang. Y. Li // Applied Intelligence — 2022.

18. Daoud A. M. Efficient data structures for information retrieval systems / A. M. Daoud // Perfect Hashing with Lightning Fast Unsuccessful Search — 1990.

19. Kitamura M. Automatic extraction of word sequence correspondences in parallel corpora / M. Kitamura. Y. Matsumoto // Proceedings of the 4th Workshop on Very Large Corpora. — 1996. — С. 79–87.

20. Zhang X. An improved English to Chinese translation of technical text/ X.Zhang. S.Tao. Z.Gong. B.Wu. R.Wang. B.M.Wilamowski // IEEE 19th International Conference on Intelligent Engineering Systems (INES). — Bratislava. Slovakia. — №9 — 2015.

21. Daoud A. M. Minimal Perfect Hash Functions Review / A. M. Daoud // Perfect Hashing with Lightning Fast Unsuccessful Search — 2022.

22. Amble O. Ordered hash tables / O. Amble. D. E. Knuth // The Computer Journal. — 1994. — Vol. 17. № 2. — С. 135–142.

23. Wang M. A critical evaluation of bilingual Chinese/English dictionaries for elementary and intermediate Mandarin learners at Stellenbosch University / M.Wang. — University of Stellenbosch. — 2012. — 150 c.

24. Astrizi T. L. Redacting Blockchain Without Exposing Chameleon Hash Collisions / T. L. Astrizi. R. Custódio. C. Villar //Cryptology and Network Security — 2022.

25. Johnson R. Using finite state transducers for making efficient reading compression dictionary / R Johnson. L Antonsen. T Trosterud // Proceeding of 19-th Nordic Conference of Computational Linguistics- NODALIDA 2013.- Olso. Norway.- 2013.- P.75-87.

26. Amjad M. D. Minimal Perfect Hash Functions Review / M. D. Amjad // Perfect Hashing with Lightning Fast Unsuccessful Search — 2022. — С. 1–3.

27. Daoud A. M. Practical Perfect Hashing for very large Key-Value Databases / A. M. Daoud // CICEM 2012 — 2012.

28. Zuo P. A Write-friendly Hashing Scheme for Non-volatile Memory Systems / P. Zuo. Y. Hua // Conference: the 33rd International Conference on Massive Storage Systems and Technology (MSST). — 2017.

29. Виноградов Ю.М. Метод корекції подвійних помилок в каналах передачі цифрових даних зі спектральною модуляцією / Ю.М. Виноградов. Т.А. Руденко // Одинадцята міжнародна науково-технічна конференція "Проблеми телекомунікацій" і дев’ята міжнародна науково-технічна конференція студентів та аспірантів «Перспективи розвитку інформаційно-телекомунікаційних технологій та систем». — 2017. — С. 60–62.

30. Fuertes-Olivera P.A. E- Lexicography: the internet. digital initiatives and lexicography / P.A. Fuertes-Olivera. H. Bergenholtz.- London. Continuum International Published Group.- 2011 – 282 P.

31. Giron A. A. TLS 1.3 Handshake Analyzer / A. A. Giron. F. Schardong. R. Custódio // Conference: Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais — 2022.

32. Лужецький В.А. Узагальнений метод хешування байтової форми представлення інформації / В.А. Лужецький. Д.В. Кисюк // IV міжнародна науково-практична конференція «Інформаційні технології та комп'ютерна інженерія». — 2014. — С. 184–186.

33. Выдрин Д.В. Реализация электронного словаря с использованием n-грамм / Д.В. Выдрин. В.Н. Поляков // Науково-теоретичний журнал "Штучний інтелект". — 2002. — №.4. — С. 180–183.

34. Vidrin D.V.. Polyakov V.N. Implementation of an electronic dictionary using n grams/ D.V. Vidrin. V.N. Polyakov//Shtuchnyi intellect. № 4 - 2002. -.180-183.

35. Zuo P. A Write-Friendly and Cache-Optimized Hashing Scheme for Non-Volatile Memory System / P. Zuo. Y. Hua // IEEE Transactions on Parallel and Distributed Systems PP(99). — 2017.

36. Skala V. “A new approach to hash function construction for textual data: A comparison / V.Skala. R.Petruska // 4th World Congress on Information and Communication Technologies (WICT) .— Bandar Hilir. Malaysia. — December 2014.

37. Botelho F.C. Indexing Internal Memory with Minimal Perfect Hash Functions / F.C. Botelho. H.R. Langbehn. G. V. Menezes. N. Ziviani // The Brazilian Symposium on Databases. — 2008. — C. 60–75.

38. Fuertes-Olivera P.A. Wiktionary as a prototape of collective free multiple-language internet dictionary/ P.A. Fuertes-Olivera // The functional theory of lexicography and electronic dictionary.- 2009.- P.99-108.

39. Chandramouli B. Concurrent Key-Value Store with In-Place Updates / B. Chandramouli. G. Prasaad. D. Kossmann. M. Barnett // TConference: the 2018 International Conference. — 2018.

40. Tsunakawa T. Building a Bilingual Lexicon Using Phrase-based Statistical Machine Translation via a Pivot Language / T. Tsunakawa. N. Okazaki. T. Tsujii // COLING (Posters). —2008. — C. 127–130.

41. Ball L.H. Heuristic evaluation of e-dictionary / L.H. Ball. Bothma T.J.D. // Library Hi Tech. —2018. —Vol. 36. —№ 2. —P. 319–339.

42. Нурышов Т.А. Построение и исследование частотной хеш-функции для поиска в массивах со специальной структурой / Т.А. Нурышов // Технические и математические науки. Студенческий научный форум. Электронный сборник статей по материалам IX студенческой международной научно-практической конференции. — 2018. — №9(9). — С. 10–15.

43. Bender M.A. All-purpose hashing / M.A. Bender. A. Conway. M. Farach-Colton. W. Kuszmaul. G. Tagliavini // arXiv preprint arXiv. — 2021.

44. Liang R. Word Sense Disambiguation Based on Semantic Knowledge / R.Liang. C.Luo. C.Zhang. T.Lei. H.Wang. M.Li // IEEE 2nd International Conference on Electronic Information and Communication Technology (ICEICT) . —Harbin. China. —January 2019.

45. Prashant P. IcebergHT: High Performance PMEM Hash Tables Through Stability and Low Associativity / P. Prashant. Bender M.A.. Bender Michael. Johnson R. // 10.48550/arXiv.2210.04068 — 2022. — C. 1–15.

46. Ross K. A. Efficient hash probes on modern processors / K. A. Ross // IEEE 23rd International Conference on Data Engineering. — 2007. — C. 1297–1301.

47. Poirei M. Development of English to Manipuri electronic dictionary: a database approach / M. Poirei. H.D. Mamata // National Conference on Innovations in Science and Technology. — 2017.

48. Ravi S. Scalable Decipherment for Machine Translation via Hash Sampling/S. Ravi// Conference: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics. —vol. 1.1. —August 2013.

49. Johnson R. “Using finite state transducers for making efficient reading compression dictionary / R.Johnson. L.Antonsen. T.Trosterud // Proceeding of 19-th Nordic Conference of Computational Linguistics- NODALIDA. — Olso. Norway. —2013— C.75-87.

50. Steven A. Cache and Memory Hierarchy Design: A Performance Directed Approach / A.Steven. M.Kaufmann // 1st edition. Elsevie. —2014. — C.111-158.

51. Mingmou L. Succinct filters for sets of unknown sizes/ L. Mingmou. Y. Yitong. Y. Huacheng // International Colloquium on Automata. Languages. and Programming (ICALP). Schloss Dagstuhl-Leibniz-Zentrum fur Informatik. — 2020.

52. Rodriguez C. A Computer Implementation of a Spanish Synonym Dictionary / C. Rodriguez. L. De Sopena. C. Villar // Literary and Linguistic Computing — 1989.

53. Tapia-Fernández S. Key Concepts. Weakness and Benchmark on Hash Table Data Structures / S. Tapia-Fernández. D. García-García. P. García-Hernandez // Algorithms 15(3):100. — 2022.

54. Xu X. Dictionary Learning Based Hashing for Cross-Modal Retrieval / X. Xu // Conference: the 2016 ACM . — 2016.

55. Dong D. Multi-task learning for multiple language translation / D. Dong. H. Wu. W. He. D. Yu. H. Wang // ACL (1). — 2015. — C.1723–1732.

56. Wu Y. Dictionary Learning based Supervised Discrete Hashing for Cross-Media Retrieva / Y.Wu. X.Luo. X.Xu // Conference: the 2018 ACM. —2018.

57. Amjad M. D. Perfect Hash Functions for Large Dictionaries / M. D. Amjad // Perfect Hashing with Lightning Fast Unsuccessful Search — 2020. — C. 67–72.

58. Weaver S. Constructing Minimal Perfect Hash Functions Using SAT Technology / S. Weaver. M. Heule // Proceedings of the AAAI Conference on Artificial Intelligence. —vol. 34.2. —2020.

59. Колганов Д.С. Обзор аналитической. статистической и нейронной технологий машинного перевода / Д.С. Колганов. Е.А. Данилов // Международный студенческий научный вестник. — 2018. — №3. — С. 301-303.

60. Hyeontaek L. SILT: A memory-efficient. high-performance key-value store / L. Hyeontaek. F. Bin. D.G. Andersen. M. Kaminsky // Conference: Proceedings of the 23rd ACM Symposium on Operating Systems Principles. — 2011.

61. Baotong L. Dash: scalable hashing on persistent memory / L. Baotong. H. Xiangpeng. W Tianzheng. E. Lo // Proceedings of the VLDB Endowment. — 2020.

ДОДАТОК А

Електронний словник підвищеної швидкодії на основі хеш-адресації без колізій

Лістинг програми для експериментального дослідження запропонованого електронного словника з хеш-адресацією без колізій

Аркушів 8

Київ – 2022 р.

```
include<stdio.h>
#include<fstream.h>
#include<conio.h>
#include<iostream.h>
#include<dos.h>
#include<snap.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
unsigned long UG(unsigned long,unsigned long, unsigned long);
unsigned long EG(unsigned long,unsigned long, unsigned long);
unsigned long UMO(unsigned long,unsigned long, unsigned long, int n);
unsigned long UMG(unsigned long,unsigned long, unsigned long, int n);
int ORTA(int*, int);
int main()
{
long n,e,m,h,k,j,i,q,u,r,t,mod,ste,v,v_a,v_b;
double a,M,p,b,d,c,s,g,z,l_av,sigma,gamma,pp,dt;
cout<<"\n";
long A[1000];
long L[1000];
long B[1000];
long F[1000];
long B_D[1000];
long F_D[1000];
ofstream FA("INNA_51.TXT",ios::out);
m=1000; a=0.02; b=1.0;
```

```
M=m/a; d= 1.0/a;
v=1.2*l_av; v=100;
l_av=100; sigma = l_av/6;
a=0.001; gamma = 0.5;
v_a= gamma*v; v_b=v - v_a;
FA<<"\n Average lenght 500"<<" Alfa 0.001 Number of key 10000";
do
{
M=m/a;
pp=0.0;
for(j=0;j<m;j++) A[j]=rand()*M/RAND_MAX;
do
{
e=0;
for(j=0;j<m-1;j++)
if(A[j]>A[j+1]) { t=A[j]; A[j]=A[j+1]; A[j+1]=t; e=1; }
}
while(e);
A[m]=M;
for(j=0;j<m;j++) // генерація нормально розподілених довжин
{
c=0; for(i=0;i<12;i++) c+=(0.0+rand())/RAND_MAX;
c=(c-6)*sigma+l_av; if(c<=1) c=1;
L[j]=c;
}
// RECORDING
r=m;
```

```
if( (v_a-A[0]) > 0 ) B[0]=0; else B[0]=A[0]-v_a;
F[0]=B[0]+L[0];
B_D[0]=0; F_D[0]=0;
if(F[0]>=A[1])
{
B_D[0]=0;
F_D[0]=F[0]-A[1];
F[0]=A[1]-1;
r--;
}
else
if(F[0]>A[0]+v_b)
{
B_D[0]=0;
F_D[0]=F[0]-(A[0]+v_b);
F[0] = A[0]-v_b;
r--;
}
dt=F_D[0];
pp+=(L[0]-dt)/L[0] +2*dt/L[0];
for(j=1;j<m;j++)
{
h=A[j]-v_a;
if(h>F[j-1]) B[j]=h; else B[j]=F[j-1]+1;
F[j]=B[j]+L[j];
B_D[j]=F_D[j-1]; F_D[j]=F_D[j-1];
if(F[j]>=A[j+1])
```

```
{
B_D[j]++;
F_D[j]=B_D[j]+F[j]-A[j+1]+1;
F[j]=A[j+1]-1;
r--;
}
else
if(F[j]>A[j]+v_b)
{
B_D[j]=F_D[j-1]+1;
F_D[j]=B_D[j]+ F[j]-(A[j]+v_b);
F[j] = A[j]+v_b;
r--;
}
if(j<m-1)
{
dt=F_D[j]-B_D[j];
dt=(L[j]-dt)/L[j] +2*dt/L[j];
pp+=dt;
}
}
b=(r+0.0)/m;
FA<<"\n "<<a<<"
"<<b<<"
"<<pp/m<<"
"<<(F_D[m-
1]-1.0)/M;
```

```
FA<<"
"<<(m*(l_av+1))/(M+F_D[m-1]);
FA<<"\n a "<<a<<" % in hash-memory only "<<(r+0.0)/m;
FA<<" V_DOP = "<<(F_D[m-1]+0.0)/M;
for(j=0; j<m; j+=1) FA<<"\n "<<j<<" "<<A[j]<<" "<<L[j]<<"
beg="<<B[j]<<" fin="<<F[j]<<" db="<<B_D[j]<<" "<<F_D[j];
a+=0.001;
}
while(a<0.016);
FA.close();
return 0;
}
void LIN(int *L, int q) // Building of the linear function
{
int k,j,i,h;
for(j=0;j<32;j++)
{
k=0; h=1;
for( i=1;i<=5;i++)
{
if(((j&h)&(q&h))!=0) k^=1;
h<<=1;
}
*(L+j)=k;
};
a=0; h=0;
for(j=0;j<7; j++)
```

```
for(q=j+1;q<8; q++)
{
i=0; h++;
for(k=0;k<8;k++) if ((k!=j)&&(k!=q)) C[i++]=F[k];
f=0;
for(i1=0;i1<6;i1++)
{
i=0;
for(k=0;k<6;k++) if(k!=i1) B[i++]=C[k];
if(ORT(&B[0],5)) f=1;
}
if(f) a++;
else FA<<"\n
"<<hex<<F[j]<<" + "<<F[q];
}
FA<<" Probability correction "<<(a/h);
FA<<"\n\n Triple
";
a=0; h=0;
for(j=0;j<6; j++)
for(q=j+1;q<7; q++)
for(i2=q+1;i2<8; i2++)
{
i=0; h++;
for(k=0;k<8;k++) if ((k!=j)&&(k!=q)&&(k!=i2)) C[i++]=F[k];
if(ORT(&C[0],5)) a++;
else FA<<"\n
```

```
"<<hex<<F[j]<<" + "<<F[q]<<" +
"<<F[i2];
}
return;
}
unsigned long UMG(unsigned long a,unsigned long b, unsigned long m, int n)
{
int j;
unsigned long y=0;
for(j=0;j<n;j++)
{
if(b&1) y^=a;
if(y&1) y^=m;
y>>=1;
b>>=1;
} MASK[0]=10; // 100
MASK[4]=8; // 001
for(i=0;i<dn; i++ )
if(( MASK[0]&X_P[i]) != MASK[0] ) X_P[i]=2048;// invalid
else X_P[i]=i; // valid
for(j=1; j<M; j++) // The main cycle
{
for(i=0; i< dn; i++) // forming new R and X
{
X[i]=i;
if( (MASK[j]&X[i])!= MASK[j] ) X[i]=2048; // invalid
R[i]=0;
```

```
}
for(i=0; i<dn; i++ ) // current
if(X[i]<2048)
// select valid only
for(k=0; k<dn; k++ ) // previous
if(X_P[k]<2048) // select valid only
{
if(R[i]<R_P[k]) R[i]=R_P[k];
t=3; //two bit probe 011
for(q=1;q<n;q++)
{
if( ( (X[i]&t)|(X_P[k]&t) )==0) // square 2 x 2 can be place
{
h=X[i]|t;
if(R[h]<R_P[k]+1) R[h]=R_P[k]+1;
}
t<<=1;
}
}
for(i=0; i<dn; i++ )
{
cout<<"\n "<<R_P[i]<<" ";
if(X_P[i]<2048)
cout<<X_P[i]/8<<" "<<(X_P[i]/4)%2<<" "<<(X_P[i]/2)%2<<"
"<<X_P[i]%2<<" ";
else cout<<" *
";
```

```
cout<<R[i]<<" ";
if(X[i]<2048)
cout<<X[i]/8<<" "<<(X[i]/4)%2<<" "<<(X[i]/2)%2<<" "<<X[i]%2<<"
else cout<<" *
";
}
cin>>h;
for(i=0;i<dn;i++)
{
R_P[i]=R[i];
X_P[i]=X[i];
}
}
h=0;
for(i=0; i<dn; i++) // maximum
if( R[i]>h) h = R[i];
y+=h;
return y;
}
unsigned long UMO(unsigned long a, unsigned long b, unsigned long m, int n)
{
unsigned long r=0; int k=n;
while(n>0)
{
if(a&1) r+=b;
if(r&1) r+=m;
r>>=1;
```

```
a>>=1;
n--;
}
a=1;
for(b=1;b<=k;b++) a<<=1;
a=a%m;
r=(a*r)%m;
return r;
}
unsigned long UG(unsigned long a,unsigned long b, unsigned long m)
{
unsigned long i,j,r,k;
r=0;
while(b!=0)
{
if(b&1) r^=a;
a<<=1;
b>>=1;
}
if(m==0) return r;
i=0x80000000;
do
{
while((r&i)==0) i=(i>>1)&0x7FFFFFFF;
j=m; k=0;
if(j<i)
while((j&i)==0) { j<<=1; k++; }
```

```
if(k>0) r^=j;
}
while(k>0);
j=0x40000000;
while((m&j)==0) j>>=1;
if(r>=j) r^=m;
return r;
}
unsigned long EG(unsigned long a, unsigned long e, unsigned long m)
{
unsigned long i,j,k,r;
r=1;
for(j=1;j<=e;j++)
r=UG(a,r,m);
return r;
}
int i,j,k,f,z,cj,d;
int A[100]={0};
for(i=0;i<n;i++)
{
A[i]=0; k=1;
for(j=0;j<n;j++) { A[i]+=*(h+n*i +(n-1-j))*k; k*=2; }
}
for(i=0;i<n;i++) if(A[i]==0) return 0;
z=1;
for(j=0;j<n;j++)
{
```

```
d=0;
for(i=0;i<n;i++) d+= A[i]&z;
if(d==0) return 0;
z<<=1;
}
f=1;
for(j=1;j<k;j++)
{
z=0; cj=j;
for(i=0;i<n;i++)
{
if(cj&1) z^=A[i];
cj>>=1;
}
if(z==0) return 0;
}
return 1;
}l
```