УДК 004.934

МАРКОВСЬКИЙ О. П.,
ШЕВЧЕНКО О. М.,
РУДЕНКО Т. А.

ОРГАНІЗАЦІЯ ХЕШ-ПОШУКУ В ЕЛЕКТРОННИХ СЛОВНИКАХ

В статті запропоновано підхід до прискорення виконання пошуку в електронних словниках систем комп'ютерного перекладу. В основу запропонованої організації пошуку покладено використання найбільш швидкої технології пошуку – хеш-адресації. Виявлені характерні особливості хеш-пошуку в електронних словниках і на цій основі запропоновано нові технології розміщення даних та пошуку. Для оцінки ефективності проведене моделювання, результати якого показують, що час пошуку в порівнянні з відомими рішеннями скоротився в 2 - 2.5 рази.

The paper considers an approach to accelerate the process of word search in electronic dictionaries. The described word search method is based on the use of the fastest searching technique - hash addressing. Some specific features of the hash-based search in electronic dictionaries were described and some new approaches to data placement/allocation and retrieval were proposed. To assess the effectiveness of this approach a simulation method was used. The results showed that the search time in comparison with other methods has decreased by 2 - 2.5 times.

Ключові слова: електронні словники, комп'ютерний переклад, хеш-пошук в незмінних масивах ключів, хеш-адресація, кеш-пам'ять.

Вступ

Однією з характерних рис технічного прогресу на початку нового тисячоліття є динамічний розвиток країн Далекого Сходу, таких як Китай, Японія та Південна Корея. Окрім значних досягнень в економічній та технічній сферах, спостерігається стійка тенденція росту питомої ваги наукових публікацій, зроблених науковцями країн Сходу. Це загострює проблему ефективного обміну науковими досягненнями між країнами Заходу та Сходу в аспекті мовного інтерфейсу [1], який в сучасних умовах стає істотною завадою на шляху вільного обміну технічною інформацією.

Вказані чинники стимулюють зростання значення комп'ютерних засобів перекладу. І в європейських наукових програмах, і в аналогічних програмах країн Далекого Сходу, значні за обсягом ресурси виділяються для якісного вдосконалення засобів комп'ютерного перекладу. Фактично, на теперішній час чітко виділилося два напрямки: комп'ютерний переклад та комп'ютеризований переклад. Перший орієнтовано на виконання перекладу виключно комп'ютерними засобами, без участі людини. Другий передбачає активну взаємодію фахівця з певної предметної області та програмних засобів. що допомагають йому ефективно здійснювати переклад [2]. В обох напрямках комп'ютерних засобів перекладу одну з вузлових ролей відіграють електронні словники [3].

Сучасні технології комп'ютерного перекладу передбачають відпрацювання багатьох варіантів перекладу з урахуванням контексту. Це вимагає багатократного звернення до електронних словників. Тому, швидкодія електронних словників значною мірою визначає часові характеристики перекладу, а також кількість варіантів, що можуть бути розглянуті. Іншими словами, швидкодія електронних словників значною мірою впливає на якісні характеристики систем комп'ютерного чи комп'ютеризованого перекладу. Особливою гостро проблема швидкодії електронних словників стоїть для систем перекладу, що працюють в реальному часі.

Отже, особливості розвитку технологій комп'ютерного та комп'ютеризованого перекладу вимагають вдосконалення роботи словників у напрямку прискорення їх роботи.

Таким чином, проблема прискорення роботи комп'ютерних словників є важливою та актуальною на сучасному етапі розвитку інформаційних технологій.

Аналіз існуючих технологій пошуку в словниках

Пошук в електронних словниках відрізняється від звичайного наступними характерними особливостями:

1) пошуковий масив практично не змінюється;

2) наявність омонімів – тобто однакових слів з різним семантичним значенням;

3) наявність для кожного елементу пошукового масиву блоку інформації, яка дозволяє більш точно визначити варіант перекладу. Причому довжина цього блоку суттєво різниться для елементів пошукового масиву.

Іншими словами, інформація словника складається з пошукового масиву та супутніх даних. Пошук виконується в пошуковому масиві, причому результатом пошуку є адресне посилання на блок супутньої інформації, яка використовується безпосередньо для перекладу.

В сучасних умовах значного зростання об'єму словників внаслідок залучення для перекладу значних об’ємів текстової інформації, зростає значимість швидкості пошуку.

До теперішнього часу розроблено ряд підходів та способів організації електронних словників [1]. Значна частина сучасних електронних словників будуються за принципом бази даних. При цьому використовуються спеціальні моделі зберігання даних, що мають за основу різні види організації морфів та афіксів.

Найбільшого поширення в електронних словниках набули технології пошуку, що мають за основу використання деревовидних структур [2]. Базова ідея полягає в тому, що перехід по гілкам дерева організовується в залежності від поточної літери слова. Таким чином, цю технологію пошуку в певному плані можна розглядати як кінцевий автомат. Відповідно, в кожному з вузлів дерева зберігається d адресних посилань, де d – кількість літер. Кількість вузлів може бути приблизно оцінена як $n\cdot(1+1/q)$, де n – кількість слів в словнику. Кількість рівнів дерева може бути оцінена як середня кількість літер в слові. Об’єм пам’яті в оціночному плані складає $n\cdot(1+1/q)\cdot q\cdot\log_2 n \approx n\cdot q\cdot\log_2 n$.

Суттєвим недоліком використання деревовидних структур є низька швидкодія пошуку, зумовлена такими чинниками:

1) багаторівнева організація пошуку, при якій середня кількість звернень до пам’яті визначається довжиною слова;

2) в сучасних комп’ютерних системах пам'ять має багаторівневу організацію і складається з кеш-пам’яті та основної пам’яті. Загрузка кеш-пам’яті здійснюється порівняно великими блоками і потребує значних ресурсів часу. При пошуку в деревовидних структурах адресні посилання розкидані по всій пам’яті і, реально, при кожному переході по дереву потрібно виконувати підкачування даних в кеш-пам’ять із основної пам’яті, що потребує багато часу.

3) швидкість пошуку слів не залежить від частотності їх використання.

Найбільшу швидкість пошуку забезпечує хеш-адресація, тобто розміщення ключів за адресами, що функціонально залежать від самих ключів. До теперішнього часу її широке застосування стримувалося наступними чинниками:

1) хеш-адресація потребує певних обчислювальних ресурсів для формування хеш-адреси з коду слова;

2) хеш-адресація ефективна лише за умови неповного використання об’єму пам’яті, тобто вона потребує додаткового об’єму пам’яті;

3) наявність колізій, тобто ситуацій, при яких декілька різних слів мають однакову хеш-адресу.

В сучасних умовах ці недоліки стають менш значимими: швидкодія процесорів динамічно зростає; успіхи мікроелектроніки та інтегральної технології дозволили значно збільшити ємкість мікросхем пам’яті[4].

Це дозволило активно використовувати технології хеш-адресації для створення швидкісних словників. Електронні словники, що використовують технології хеш-пошуку, передбачають попередній граматичний аналіз слова задля виявлення його лексичної основи [5] шляхом видалення афіксів. Лише після цього виконується пошук з використанням в якості ключа лексичної основи. Урахування афіксів для адекватності перекладу виконується на рівні аналізу пов’язаних з ключем даних.

Серйозною перешкодою при створенні ефективних електронних словників на основі хеш-адресації є суттєво різний об’єм супутньої інформації для кожного з пошукових ключів. Для вирішення цієї проблеми пам'ять організовується у вигляді двох окремих банків: для зберігання ключів та для зберігання супутньої ключам інформації [5]. Для взаємодії між банками пам’яті використовуються спеціальні адресні посилання. Це дозволяє суттєво зменшити об’єм пошукового масиву і знизити кількість циклів свопінгу за рахунок того, що значна частина (або ж увесь) пошукового масиву вдається загрузити в кеш-пам'ять.

З іншого боку, таке рішення не дозволяє розв’язати дилему між ймовірністю колізій та ступенем надлишковості пам’яті. Для того, щоб зменшити кількість циклів свопінгу, для обробки колізій в таких системах найчастіше використовується лінійний пробінг.

Таким чином, існуючі механізми пошуку в електронних словниках, в тому числі побудовані на основі хеш-адресації, не забезпечують потрібної швидкості пошуку.

Мета досліджень полягає у вдосконаленні технології хеш-пошуку в електронних словниках, задля забезпечення суттєвого підвищення швидкодії пошуку в порівнянні з існуючими механізмами.

Метод прискореного хеш-пошуку в електронних словниках

Для прискорення хеш-пошуку в електронних словниках, з урахуванням того, що інформація в словниках змінюється рідко і її можна вважати статичною, можуть бути задіяні наступні можливості:

1) зменшення ступеню заповнення хеш-пам'яті ключів (розріження хеш-пам'яті) задля зменшення колізій;

2) розміщення інформації для пошуку в два етапи: спочатку ключів, а потім пов'язаних з ними даних;

3) урахування частотності використання слів при розміщенні їх в хеш-пам'яті;

Розміщення даних у пам'яті пропонується організувати таким чином, щоб ключі та відповідні їм блоки інформації знаходились у межах однієї або суміжних сторінок. Це дозволить зменшити кількість циклів пробінгу в процесі пошуку, а отже значно підвищити швидкість виконання процедури пошуку.

Основна ідея пропонованого методу полягає у тому, щоб записувати у пам'ять за хеш-адресами ключі, а відповідні їм блоки даних записувати якомога ближче до самих ключів, щоб зменшити час свопінгу.

Для вирішення проблеми можливого виникнення колізій ключі, які мають однакові хеш-адреси з уже розміщеними у пам'яті ключами, записуються за вільними адресами після цих записаних раніше ключів. Завдяки цьому, при використанні запропонованого методу навіть у випадку виникнення колізій гарантується знаходження правильного блоку даних.

На початку кожного запису пропонується виділяти один маркерний байт. Останній його біт пропонується використовувати для позначення типу запису. Для записів, що містять значення ключів, цей біт пропонується встановлювати в значення 0, а для записів, що містять блоки даних — в значення 1. Сьомий біт пропонується використовувати для позначення того, чи існує за наступною після даної адресою запис, що містить значення ключа, якщо даний запис є ключем. Якщо наступний запис містить ключ, значення даного поля пропонується встановити в 1, інакше — в 0. Шостий біт пропонується використовувати в якості тегу продовження блоку даних і встановлювати його в 0, якщо наступний запис цього блоку знаходиться за наступною адресою; і в 1, якщо кінець цього запису є адресним посиланням на початок продовження цього блоку. П'ятий біт пропонується використовувати для позначення кінця блоку даних і встановлювати його в значення 1, якщо цей запис є кінцевим для цього блока і в 0, якщо цей запис не є кінцевим.

Сьомий та восьмий біти запропоновано використовувати для правильної обробки випадку, коли виконується пошук слова, якого не існує у словнику. П'ятий та шостий біти доцільно використати для обробки випадків, коли блоки даних розбиваються на декілька частин для більш ефективного використання пам'яті.

Записи, які містять значення ключів, пропонується організувати у вигляді трьох полів. Перше є маркерним байтом. Друге поле пропонується використовувати для збереження значення ключа. У третьому полі пропонується зберігати адресне посилання на відповідний цьому ключеві блок.

Виходячи із запропонованого способу розміщення даних у пам'яті, пропонується наступна методика пошуку відповідного блоку даних за деяким заданим ключем K_i.

1. Обчислюється хеш-адреса $H(K_i)$.
2. Приймається початкове значення $j = 0$.
3. Прочитується запис, який знаходиться у пам'яті за адресою $H(K_i) + j$.
4. Перевірка маркерного байту запису. Якщо значення його восьмого біту дорівнює 1, припинити пошук.
5. Порівняння значення першого поля прочитаного запису зі значенням ключа K_i.
6. Якщо ці значення не співпадають, перевірити значення сьомого біта маркерного байта. Якщо воно дорівнює 0, то покласти $j = j + 1$ і повернутися на п. 3. Інакше, припинити пошук, оскільки шуканого слова у словнику не існує.
7. Якщо значення співпадають, то адресне посилання на початок шуканого блоку даних знаходиться у другому полі запису.
8. Читається блок даних за отриманою адресою:

8.1. Перевіряється маркерний байт. Якщо його шостий біт дорівнює 0, читаються дані з поточного запису.

8.2. Якщо п'ятий біт дорівнює 0, відбувається перехід до наступного запису і повернення на п. 8.1.

8.3. Якщо шостий біт запису дорівнює 1, відбувається перехід за адресним посиланням у кінці запису і повернення на п. 8.1.

Заповнення хеш-пам'яті пропонується виконувати у два етапи. На першому етапі виконується запис ключів у пам'ять. На другому етапі відбувається запис у пам'ять блоків даних, що відповідають цим ключам, і заповнення полів адресних посилань у записах з цими ключами.

Для заповнення пам'яті значеннями ключів пропонується наступна процедура. Вхідними даними є множина ключів $K = \{K_1, K_2, \ldots, K_n\}$.

1. Виконується очистка пам'яті.
2. Множина ключів K впорядковується за спаданням частотності їх використання.
3. Приймається початкове значення $i = 1$.
4. Обчислюється хеш-адреса $H(K_i)$.
5. Якщо за адресою $H(K_i)$ у пам'яті записів немає, у пам'ять за адресою $H(K_i)$ у восьмий біт маркерного байту записується значення 0, а у друге поле — ключ K_i .
6. Якщо в пам'яті за адресою $H(K_i)$ існує деякий запис, у четверте поле запису за цією адресою записується значення 1 і виконується пошук найближчої до $H(K_i)$ вільної адреси:

6.1. Приймається $j = H(K_i) + 1$.

6.2. Якщо за адресою j записів немає, за адресою j записується ключ K_i. Інакше, у сьомий біт маркерного байту встановлюється в 1, покладається $j = j + 1$ і відбувається повернення на п. 6.2.

7. Покладається $i = i + 1$, якщо $i <= n$, повернутися на п. 4.

Для заповнення пам'яті блоками даних, що відповідають ключам, пропонується наступна послідовність дій. Вхідними даними є множина ключів $K = \{K_1, K_2, \ldots, K_n\}$ та множина даних, які відповідають ключам з множини K, $D = \{D_1, D_2, \ldots, D_n\}$.

1. Приймається початкове значення $i = 1$.
2. Обчислюється хеш-адреса $H(K_i)$.
3. Приймається початкове значення $j = 0$.
4. Прочитується запис, який знаходиться у пам'яті за адресою $H(K_i) + j$.
5. Значення першого поля прочитаного запису порівнюється зі значенням ключа K_i.
6. Якщо значення не співпадають, покладається $j = j + 1$ і виконується повернення на п. 3.
7. Виконується пошук найближчого до адреси $H(K_i) + j$ вільного блоку пам'яті,.
8. Блок даних D_i записується у знайдений блок пам'яті, заповнюється маркерний байт, а початкову адресу блоку записується у друге поле запису ключа K_i.
9. Якщо знайдений блок не може вмістити блок даних D_i повністю, виконується пошук наступного найближчого вільного блоку, значення шостого біта маркерного байта встановлюється в 1, п'ятого в 0, а в кінець запису записується адреса знайденого блоку, і відбувається повернення на п. 8.
10. $i = i + 1$, якщо $i <= n$, виконується повернення на п. 2.

Запропоновані процедури забезпечують виконання базових операцій, пов’язаних з розміщенням та пошуком. Критична за часом процедура пошуку є доволі простою і її реалізаціє не потребує значних обпилювальних ресурсів.

Оптимізація структури та оцінка ефективності

Основними критеріями ефективності технічної реалізації електронних словників є:

1) час доступу до блоку даних, з яким пов’язане ключове слово пошуку. Вважаючи на багаторівневий характер організації сучасної пам’яті, час доступу визначається двома чинниками: розміром s сегмента пам’яті, яким проводиться обмін даними (свопінг) між різними рівнями пам’яті; кількістю η циклів свопінгу для реалізації віднаходження блоку даних, пов’язаного з ключовим словом пошуку.

2) Рівень надмірності використання об’єму пам’яті, який характеризується відповідним коефіцієнтом β, що визначається співвідношенням реального об’єму пам’яті до мінімального значення: сумарного об’єму ключів та відповідних їм блоків даних.

Для систем пошуку на основі хеш-адресації вказані критерії ефективності пов’язані складним чином. Зокрема, довжина блоків даних являє собою випадкову величину. Проведені дослідження реальних словників [3] показали, що довжина блоків даних являє собою випадкову величину, розподілену за нормальним законом з математичним очікуванням m та дисперсією, що, в першому наближенні, також дорівнює m. До того ж, при оцінці ефективності запропонованої організації хеш-пошуку, слід врахувати різну частотність використання різних слів. Все це практично виключає можливість побудови

аналітичної моделі, що встановлює математичну залежність між величинами *s*, η та β. Разом з тим, така залежність потрібна для оптимізації структури технічної реалізації словника виходячи з заданих обмежень.

Для отримання залежностей між параметрами технічної реалізації електронного словника з запропонованою організацією проведено статистичне моделювання за допомогою спеціально створеного програмного комплексу. Основні задачі статистичного моделювання полягають у виявленні взаємозалежностей основних характеристик *s*, η та β при їх значеннях близьких до реальних, виявлення впливу впорядкування ключів за частотою їх використання. Виходячи з цього, на першому етапі була досліджена модель без урахування частотності використання слів. Результати наведені в таблиці 1.

Таблиця 1. Середня кількість η циклів свопінгу для пошуку даних за ключем.

Розмір *s* сегменту обміну	**Коефіцієнт β надмірності використання об'єму пам'яті**				
	0.91	**0.87**	**0.83**	**0.8**	**0.77**
m	2.57	2.32	2.09	1.99	1.90
1.2·*m*	2.14	1.94	1.74	1.66	1.59
1.4·*m*	1.84	1.66	1.49	1.42	1.36
1.6·*m*	1.6	1.45	1.31	1.24	1.19
1.8·*m*	1.43	1.29	1.16	1.11	1.05
2.0·*m*	1.28	1.16	1.04	1	1
2.2·*m*	1.17	1.06	1	1	1
2.4·*m*	1.07	1	1	1	1
2.6·*m*	1	1	1	1	1

Проте реальний час виконання свопінгу визначається загальним обсягом *v* даних, що переміщуються між рівнями пам'яті, який, в свою чергу, являє собою добуток середньої кількості η циклів свопінгу на об'єм *s* сегменту, що переміщується при свопінгу: $v=s\cdot\eta$. Дані, що відображають залежність середнього об'єму v даних, що переміщуються при свопінгу від розміру сегменту *s* та коефіцієнту β надмірності використання пам'яті, наведені в таблиці 2. Наведені дані переконливо свідчать, що час виконання свопінгу практично не залежить від розміру сегмента даних, що переміщуються між рівнями пам'яті.

Таблиця 2. Об'єм даних, що передається при свопінгу для пошуку даних за ключем.

Розмір *s* сегменту обміну	**Коефіцієнт β надмірності використання об'єму пам'яті**				
	0.91	**0.87**	**0.83**	**0.8**	**0.77**
m	2.57·*m*	2.32·*m*	2.09·*m*	1.99·*m*	1.9·*m*
1.2·*m*	2.57·*m*	2.33·*m*	2.09·*m*	1.99·*m*	1.91·*m*
1.4·*m*	2.57·*m*	2.32·*m*	2.09·*m*	1.99·*m*	1.9·*m*
1.6·*m*	2.56·*m*	2.32·*m*	2.1·*m*	1.98·*m*	1.9·*m*
1.8·*m*	2.57·*m*	2.32·*m*	2.09·*m*	2.0·*m*	1.89·*m*
2.0·*m*	2.56·*m*	2.32·*m*	2.08·*m*	2.0·*m*	2.0·*m*
2.2·*m*	2.57·*m*	2.33·*m*	2.2·*m*	2.2·*m*	2.2·*m*
2.4·*m*	2.57·*m*	2.4·*m*	2.4·*m*	2.4·*m*	2.4·*m*
2.6·*m*	2.6·*m*	2.6·*m*	2.6·*m*	2.6·*m*	2.6·*m*

На другому етапі моделювання досліджувалась залежність часу пошуку від упорядкування ключів за частотністю їх використання. Результати моделювання представлено у таблиці 3. У другому її рядку наведено об'єм даних свопінгу при пошуку із урахуванням частотності. Для порівняння, у першому рядку наведено об'єм даних свопінгу при пошуку без урахування частотності. У третьому рядку таблиці 3 наведені коефіцієнти прискорення, яке забезпечується використанням сортування за частотністю. Таким чином, результати дослідження показують, що сортування ключів за частотністю використання дозволяє суттєвим чином пришвидшити пошук. Також з результатів, наведених у таблиці, можна зробити висновок, що коефіцієнт прискорення буде більшим при більших коефіцієнтах β надмірності використання пам'яті.

Таблиця 3. Об'єм даних, що передається при свопінгу для пошуку даних за ключем.

	Коефіцієнт β надмірності використання об'єму пам'яті				
	0.91	**0.87**	**0.83**	**0.8**	**0.77**
Без урахування частотності	2.57·*m*	2.32·*m*	2.09·*m*	1.99·*m*	1.9·*m*
З урахуванням частотності	1.49·*m*	1.41·*m*	1.32·*m*	1.28·*m*	1.25·*m*
Коефіцієнт прискорення пошуку	1.72	1.65	1.58	1.55	1.52

Висновки

В результаті проведених досліджень, направлених на прискорення функціонування електронних словників систем комп'ютерного перекладу, запропоновано, розроблено та досліджено спеціальну організацію хеш-пошуку.

Основна особливість запропонованого підходу полягає в організації хеш-пошуку з урахуванням особливостей даних, що зберігаються в електронних словниках. Це дозволило суттєво покращити характеристики пошукових засобів словників. Для оцінки ефективності використано статичне моделювання. Отримані результати дозволили сформулювати важливі для практики положення організації розміщення даних та характеристик роботи з пам'яттю. З іншого боку, отримані результати дозволили оцінити ефективність запропонованої організації пошуку в електронних словниках в аспекті прискорення їх функціонування. Доведено, що в порівнянні з найбільш поширеними на практиці технологіями пошуку на основі дерев запропонована організація дозволяє в 2-2.5 рази прискорити пошук. Це дозволяє прискорити роботи систем перекладу і підвищити якість перекладу в силу того, що відкриває можливості аналізу більшого числа варіантів перекладу.

Основною сферою практичного використання розробленої організації пошуку в електронних словниках є високопродуктивні системи комп'ютерного перекладу, в першу чергу ті, що працюються в реальному часу, а також системи комп'ютеризованого перекладу.

Перелік посилань

1. Марковский А.П. Интерактивно-шаблонный метод компьютерного перевода научно-технических публикаций / А.П. Марковский, О.Н. Шевченко, Фань Чуньлэй // Вісник Національного технічного університету України "КПІ" Інформатика, управління та обчислювальна техніка, – Київ: ВЕК+ – 2013. – № 59. - С. 86-97.
2. Марчук Ю.Н. Компьютерная лингвистика.- АСТ, Восток-Запад, 2007.-165 с.
3. Агапова Н.А. О принципах создания электронного словаря лингвокультороло гического типа: к постановке проблемы / Н.А.Агапова, Н.Ф.Картофелева // Вестник Томского государственного университета. № 382 -2014.- С.6-10.
4. Кашеварова И. С. Электронный словарь как новый этап в развитии лексикографии // Молодой ученый. – 2010. – №10. – С. 145-147.
5. Выдрин Д.В., Поляков В.Н. Реализация электронного словаря с использованием *n* грамм / Д.В. Выдрин, В.Н. Поляков // Штучний інтелект. № 4 – 2002. – С.180-183.