УДК 004

Гоцалюк О. М., Ярмолюк Р. С.

Хмельницький національний університет

БАГАТОПЛАТФОРМЕНА СИСТЕМА ПЕРЕВІРКИ ОФОРГАФІЇ НА ОСНОВІ СЛОВНИКОВОЇ ТЕХНОЛОГІЇ HUNSPELL

Основною метою цього дослідження є розробка нового методу підходу до проектування підсистеми перевірки орфографії на основі технології перевірки орфографії за словником Hunspell. Запропоновано та описано структурну і об'єктно-орієнтовну схему підсистеми перевірки орфографії. Проведено аналіз ефективності запропонованих підходів на основі експерименту.

The main purpose of this study is to develop a new method of approach to the design of a subsystem spell checking technology based spell checking dictionary for Hunspell. Proposed and described structural and object-oriented scheme subsystem checker. The analysis of the effectiveness of the proposed approach based on experiment.

Постановка проблеми. Якість інформаційно-пошукових та інтеграційних можливостей у великій мірі залежить від якості даних, що містяться в базі даних. Під час роботи з даними якого можуть виникати ситуації, що призводять до появи різного типу помилок.

Формування цілі статті. В сучасному світі розробники інформаційних систем значну увагу приділяють перевірці орфографії, але багато цих засобів є платними і не є багатоплатформеними.

Підсистема перевірки орфографії, що реалізована на технології Java для операційних систем Windows, MAC OS, Linux, схематично має структуру, представлена автором на рис. 1.

Експерименти з оцінювання швидкодії системи. На основі описаних підходів у статті Ярмолюка Р. С. [1] було розроблено програмне забезпечення, що реалізовує можливості пошуку орфографічних помилок у базах даних на технології Java.

Технологія Java була обрана через те, що вона дозволяє створювати незалежні від платформи програми шляхом компіляції в проміжне представлення, яке називається байт кодом [2].

Процес установки з'єднання з SQL-сервером може продовжуватися набагато більший час, ніж безпосередня обробка

запиту до бази після установки з'єднання. Слідуючи логіці, треба уникати зайвих з'єднань до бази, не від'єднуючись від неї там, де це можна зробити, якщо надалі планується продовжити роботу з SQL-сервером [4]. Зокрема технологія Java завдяки багатоплатформеності дозволяє будувати клієнт/сервер програми, запити яких будуть працювати на різних платформах із нормальною швидкодією.

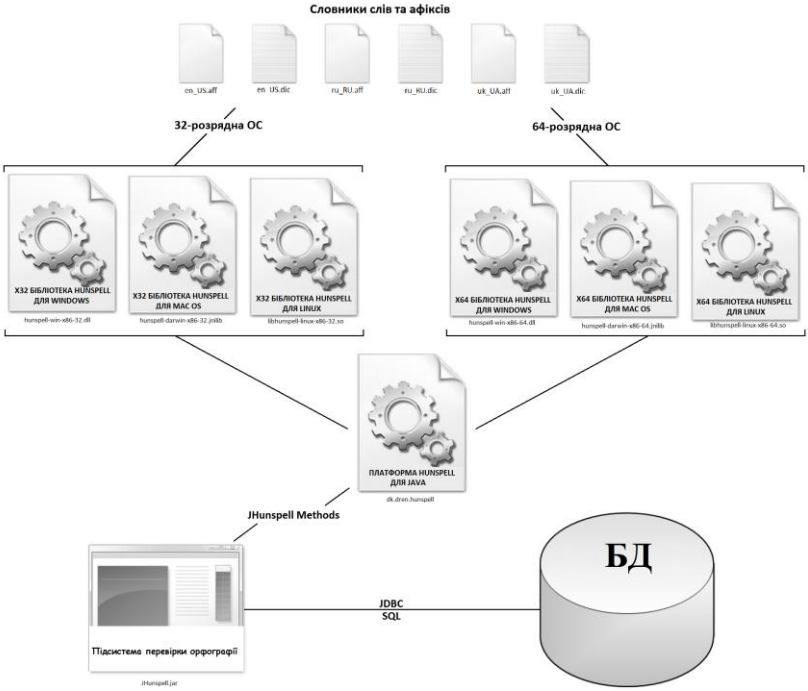


Рис. 1. **Схема структурної організації підсистеми перевірки орфографії бази даних**

Під час експлуатації системи виникла необхідність у досліджені швидкодії ядра системи Hunspell [3], а саме оцінку часу затраченого на перевірку великого масиву слів. Для оцінки верхньої часової межі для кожного слова повинен відбуватись певний перебір словника, тобто слово з великою ймовірністю повинно бути відсутнім у словнику.

Технологічні умови проведення експерименту. Для перевірки орфографії було обрано загальнолексичні словники

OpenOffice.org. Причина вибору обумовлюється їх безкоштовністю і постійним оновленням. Параметри кожного із словників наведені у табл. 1.

Тестування проводилось на двох комп'ютерних платформах на базі операційної системи Microsoft Windows. Критичні параметри кожної із систем наведені у табл. 2.

Таблиця 2

Параметри комп'ютерних платформ тестування

	Процесор	Оперативна пам'ять	Операційна система
Перша платформа	Intel(R) Pentium (R) CPU B970 2.30 GHz	4 ГБ	Microsoft Windows 7 Максимальная 64- розрядна
Друга платформа	AMD Athlon(tm) II X2 245 Processor 2.90 GHz	12 ГБ	Microsoft Windows XP SP3 64- розрядна

Дослідження швидкодії ядра системи Hunspell. На першому етапі під час експерименту для створення вхідних даних для перевірки було розроблено клас із методами генерації випадкових слів певної довжини. Метод randomWord.UA(int len, int nom), randomWord.RU(int len, int nom), randomWord.ENG(int len, int nom) (Рис. 2) використовуються для генерування nom кількості символьних послідовностей UNICODE (тестових слів довжини len) для українського, російського та англійського алфавіту відповідно. На другому етапі метод test.tspell (String[] s, int index) реалізує перевірку кожного тестового слова із масиву слів s за допомогою методу misspelled(s[i]) та повертає час (у мілісекундах), затрачений на перевірку nom кількості тестових слів довжини len за словником відповідної мови index.

Тестування проводилось для кожного масиву слів відповідної мови. Розмірність масиву складала 100 000 тестових слів відповідної довжини. Максимальна довжина слова 23 символів UNICODE. Оскільки тестові слова генерувались як довільна послідовність символів, то розглядається найгірший тестовий варіант: кожне із 100 000 слів з великою ймовірністю (близько 0,95) є неправильним. Результати оцінки верхньої часової межі швидкодії системи перевірки орфографії на основі технології Hunspell показані на рис. 3 та 4.

```
class randomWord {
    int a;
    String[] UA (int len,int nom) {
        Random generator = new Random();
        String[] ret=new String[nom];
        char[] word1 = new char[len];
        for (int i=0; i<nom;i++)
        {  for (int j = 0; j < len; j++)
            { word1[j] = (char)(generator.nextInt(32) + 1072);
                    switch (word1[j])
                    {   case 'ы': word1[j] = 'i'; break;
                        case 'ъ': word1[j] = 'i'; break;
                        case 'э': word1[j] = 'є'; break;  }   }
            ret[i]=new String(word1);   }
        return ret; }
String[] RU (int len,int nom) {
        Random generator = new Random();
        String[] ret=new String[nom];
        char[] word1 = new char[len];
        for (int i=0; i<nom;i++)
        { for (int j = 0; j < len; j++)
            {  word1[j] = (char)(generator.nextInt(32) + 1072); }
           ret[i]=new String(word1); }
        return ret; }
String[] ENG (int len,int nom) {
        Random generator = new Random();
        String[] ret=new String[nom];
        char[] word1 = new char[len];
        for (int i=0; i<nom;i++)
        {   for (int j = 0; j < len; j++)
            { word1[j] = (char)(generator.nextInt(26) + 97); }
            ret[i]=new String(word1);}
        return ret; }   }
```

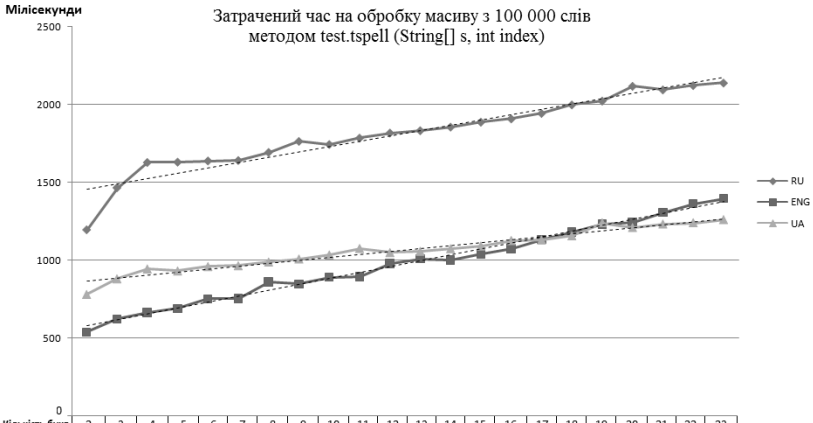
Рис. 2. **Код класу randomWord**

Аналіз результатів експерименту. Проаналізувавши отримані результати, можемо зробити такі висновки:

• Швидкість перевірки слова лінійно залежить від кількості символів, з яких складається слово, і ця тенденція зберігається незалежно від апаратної платформи або параметрів словників.

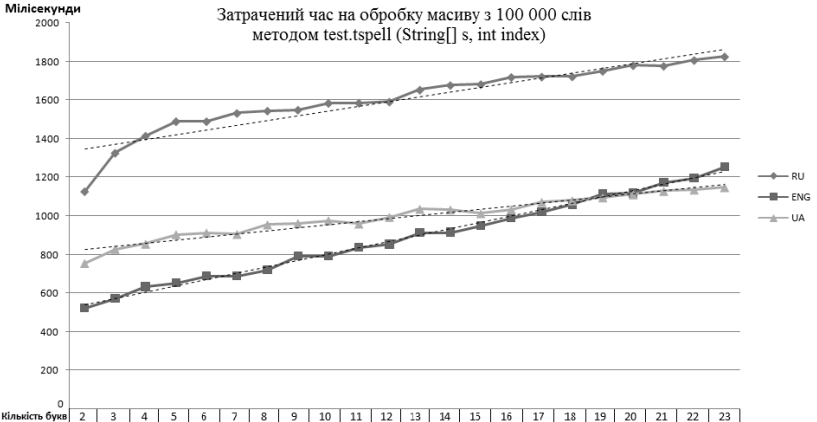
• Основним фактором, що впливає на швидкодію методу, є розмір словника основ слів.

• Збільшення затрачуваного часу для обробки великих слів на англійській мові може пояснюватись особливістю афіксної конструкції мови.



Кількість букв	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
RU	1194	1465	1628	1630	1638	1641	1693	1764	1744	1785	1818	1833	1856	1888	1909	1944	1999	2024	2119	2096	2124	2140
ENG	541	625	666	693	753	756	863	848	892	896	979	1009	1002	1041	1074	1134	1182	1231	1245	1306	1363	1396
UA	783	883	946	933	961	966	990	1009	1035	1076	1050	1058	1075	1092	1127	1131	1157	1240	1210	1232	1240	1260

Рис. 3. **Результати тестування на першій апаратній платформі**



Кількість букв	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
RU	1125	1325	1414	1489	1489	1532	1544	1549	1584	1585	1591	1653	1677	1682	1719	1721	1722	1751	1780	1777	1807	1826
ENG	523	572	633	654	688	690	721	791	792	838	853	912	915	949	951	1020	1059	1113	1118	1173	1197	1254
UA	753	827	854	903	912	904	954	961	975	958	993	1035	1032	1012	1033	1071	1079	1095	1111	1128	1135	1148

Рис. 4. **Результати тестування на другій апаратній платформі**

Висновки. У результаті проведеного дослідження було розроблено підхід для проектування підсистеми автоматичної корекції орфографічних помилок у записах бази даних на основі словникової технології. Було обрано відкриту технологію Hunspell для побудови на її основі даної підсистеми. Запропонована схема структурної

організації підсистеми на основі технології Java. Проведено експеримент із тестуванням швидкодії ядра системи.

Література

1. Ярмолюк Р.С. Підсистема перевірки орфографії електроного каталогу бібліотеки на основі технології Hunspell / Р.С. Ярмолюк //Вісник Національного Університету “Львівська Політехніка”, серія Інформаційні системи та мережі, - 2012. - № 743 – С.219-230.

2. Герберт Шилдт Java. Полное руководство. 7-е издание. / Г. Шилдт // Вильямс, 2007. –1034 с.

3. Surhone L.M. Hunspell / L.M. Surhone, M.T. Tennoe, S.F. Henssonow – Betascript Publishing, 2010 – 116 p.

4. Кузнецов М. С. Самоучитель MySQL 5 / М. С. Кузнецов // Петербург – 2007. – 560 с.