УДК 004.052.42

Р.С. Ярмолюк
Хмельницький національний університет

ПІДСИСТЕМА ПЕРЕВІРКИ ОРФОГРАФІЇ ЕЛЕКТРОННОГО КАТАЛОГУ БІБЛІОТЕКИ НА ОСНОВІ ТЕХНОЛОГІЇ HUNSPELL

Основною метою цього дослідження є розробка підходів до проектування підсистеми перевірки орфографії на основі технології перевірки орфографії за словником Hunspell. Запропоновано вимоги, що висуваються до підсистеми перевірки орфографії. Обґрунтовано вибір системи Hunspell. Розглянута структура словників слів та афіксів. Запропоновано та описано структурну та об'єктно-орієнтовну схему підсистеми перевірки орфографії. Проведено аналіз ефективності запропонованих підходів на основі експерименту.

Ключові слова: електронний каталог, система перевірки орфографії Hunspell, словник, афікс, АБІС, об'єктно-орієнтована модель, платформа .NET.

The main purpose of this research is to develop approaches to the design of subsystems spell checker Dictionary-based technologies Hunspell. A number of requirements which subsystem to spell checker. Substantiates the choice of Hunspell. The structure of dictionaries words and affixes. Suggested and described the structural and object-oriented scheme subsystems check spelling. The analysis of the effectiveness of the proposed approaches based on model tests.

Key words: OPAC, the system spelling checker Hunspell, dictionary, affixes, ILS, object-oriented model, platform .NET.

Вступ. Постановка проблеми

Система електронного каталогу має важливе значення у забезпеченні основних функцій сучасної бібліотеки. Від якості даних, що містяться в електронному каталозі, безпосередньо залежить якість інформаційно-пошукових та інтеграційних можливостей бібліотеки. Під час роботи з даними електронного каталогу бібліотеки можуть виникнути виняткові ситуації, що призводять до появи різного роду помилок. Основні атрибути запису електронного каталогу (автор, назва, видавництво, анотація джерела) мають текстовий тип. Відповідно, основними помилками в текстових атрибутах є символьні спотворення або орфографічні помилки, що зумовлені як об'єктивними, так і суб'єктивними чинниками. Від того, наскільки ефективно відбувається пошук та корекція орфографічних помилок в текстових атрибутах записів електронного каталогу, залежить якість вихідної інформації при пошукових запитах.

Аналіз останніх досліджень та публікацій

Значну увагу теоретичним і практичним основам проблеми пошуку та корекції орфограмфічних помилок в записах електронного каталогу у своїх працях приділяли М.И. Вершинин [1], А.С. Карауш [2], R. Nielsen [3], T. Ballard [4] та ін. У своїх працях та розробках дані автори приділяли увагу розробці теоретичних методів та засобів пошуку і корекції орфографічних помилок у бібліографічних записах. Але питання практичної реалізації на основі готових програмних рішень залишились поза увагою авторів.

У монографії М.И. Вершинин [1] приділяє значну увагу проблемам створення електронного каталогу різними технічними способами. У питанні пошуку орфографічних помилок запропоновано

лише загальні ідеї, що базуються на теорії нечітких множин, однак, напрямки практичної реалізації даних ідей запропоновані не були.

До основних алгоритмічних задач, що виникають під час розробки програмних засобів перевірки орфографії та аналізу текстових рядків, зараховують [5]:

- задача зіставлення текстових рядків;
- задача розрахунку відстані між текстовими рядками;
- задача нечіткого порівняння текстових рядків.

До основних алгоритмів зіставлення текстових рядків належать:

- алгоритм Кнута-Моріса-Прата[6];
- алгоритм Рабіна-Карпа[7];
- алгоритм Бояра-Мура [8] та його варіації.

До основних алгоритмів розрахунку відстані між текстовими рядками належать:

- алгоритм Вагнера-Фішера [9];
- алгоритм Хіршберга [10];
- алгоритм Ханта-Шиманського [11];
- алгоритм Уконена-Маєрса [12].

До основних алгоритмів нечіткого порівняння текстових рядків належать:

- алгоритм k – незбігів Ландау-Вишкин [13];
- алгоритм k – відмінностей Ландау-Вишкин [14].

Класичні підходи до аналізу текстових рядків висвітлені у роботах: Р.В. Хемминга, та Левенштейна В.И. Авторський аналіз вищезазначених алгоритмів зроблений у роботі [5].

Формулювання цілі статті

Аналіз можливостей сучасних автоматизованих бібліотечних інформаційних систем (АБІС) показав, що розробники даних систем недостатню увагу приділяють інструментам пошуку та корекції орфографічних помилок в записах баз даних електронного каталогу. Відсутність відповідного модуля для перевірки орфографії в таких поширених автоматизованих бібліотечних інформаційних систем (АБІС), як УФД/БІБЛІОТЕКА, МАРК-SQL, UNILIB, LIBER, ALEPH, Koha, ISIS, CDS Invenio, OpenBiblio, Evergreen, викликає багато труднощів для роботи редактора електронного каталогу. Редактор електронного каталогу – це співробітник бібліотеки, що відповідає за введення нових даних в бібліографічну базу даних електронного каталогу та їх перевірку. Зокрема, при роботі з електронним каталогом, виникає потреба ручної перевірки записів. Тому, розробка підходів щодо проектування підсистеми перевірки орфографії електронного каталогу є ***важливою та актуальною технічною проблемою*** при розробці програмних систем верифікації даних електронного каталогу бібліотеки (СВДЕК).

Для визначення ролі підсистеми СВДЕК пошуку орфографічних помилок у запропонованій авторській [13] структурно-функціональній схемі СВДЕК наведемо дану схему на рис. 1.

Підсистема пошуку орфографічних помилок є складовою частиною СВДЕК. СВДЕК – набір програмних засобів для пошуку, оцінки, виправлення та уточнення помилкових даних в електронних каталогах бібліотек. Відповідно до структурно-функціональної схеми СВДЕК підсистема пошуку орфографічних помилок є частиною блоку пошуку помилкових даних та модуля виправлення помилкових значень [13]. Функціонально підсистема реалізовує можливості, як пошуку помилкових даних, так і автоматичного виправлення помилок.

У авторській роботі [13] докладно описано функціональні можливості кожного з модулів СВДЕК (рис. 1), та обґрунтовано доцільність використання саме словникових методів при розробці підсистеми пошуку орфографічних помилок СВДЕК.

Основною метою роботи є представлення розроблених автором концептуальних підходів до вибору технології перевірки орфографії за словником та проектування підсистеми СВДЕК перевірки орфографії.

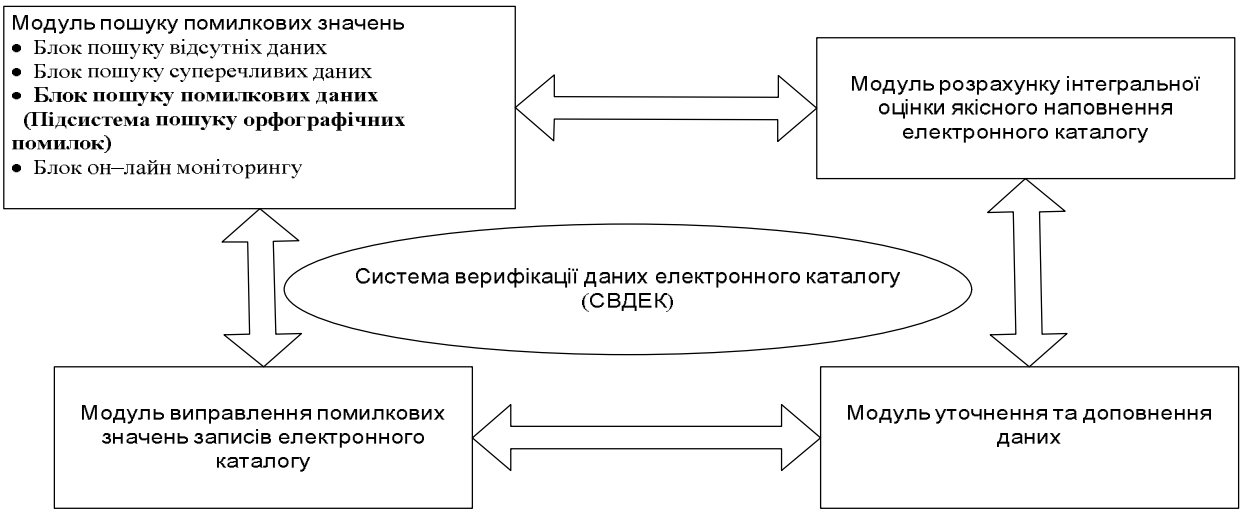


Рис. 1. Структурно-функціональна схема СВДЕК

Аналіз отриманих наукових результаів

Функціональні властивості підсистеми перевірки орфографії. Відповідно до структурних та логічних особливостей даних, що зберігаються в електронному каталозі бібліотеки, до підсистеми перевірки орфографії було висунуто авторські вимоги:

- можливість роботи з декількома різномовними словниками;
- можливість створення користувацьких словників, що відображають специфіку певної предметної області, або словників допустимих значень певних атрибутів;
- підтримка стандарту кодування символів UNICODE, зокрема, систем кодування UTF-8, UTF-16;
- доступність готових загальнолексичних словників для різних мов;
- наявність ефективного механізму морфологічного аналізу слова;
- наявність ефективного механізму пошуку помилкових слів на основі спів ставлення зі словником;
- наявність ефективного механізму пропозицій для помилкових слів (пошук слів у словнику, які найбільше підходять за певним критерієм для даного неправильного слова);
- реалізація багатопоточності при обробці даних;
- можливість інтеграції підсистеми, як в АБІС, так і в СВДЕК;
- кросплатформеність та незалежність від програмного забезпечення СКБД електронного каталогу та АБІС;
- безкоштовність.

Після проведеного аналізу даних вимог автором було відібрано дві безкоштовні системи для розробки на їх основі підсистеми перевірки орфографії у електронних каталогах бібліотек:

- Aspell (http://aspell.net/).
- Hunspell (http://hunspell.sourceforge.net/).

Однак, тільки у Hunspell є повна програмна підтримка UNICODE. Тому для розробки підсистеми перевірки орфографії було вибрано ядро системи Hunspell.

Hunspell – вільна система перевірки орфографії, що використовується у таких вільних програмних продуктах як: OpenOffice.org, LibreOffice, Thunderbird/Firefox, Opera, Google Chrome,

The Bat! та інших. Для даної системи реалізовано інтерфейси та програмні порти для платформ: Delphi, Java, Perl, .NET, Python, Ruby. Далі у дослідженні використовується безкоштовний набір бібліотек NHunspell для платформи .NET (http://nhunspell. sourceforge.net) та об'єктно-орієнтовна мова програмування C# [14].

Структура словників Hunspell. Для ефективного використання ядра системи Hunspell автором запропоновано аналіз структури словників технології Hunspell. Даний аналіз дозволив оцінити можливості даної системи для проектування на її основі підсистеми СВДЕК перевірки орфографії.

Для перевірки орфографії за допомогою системи Hunspell необхідно два файли. Перший файл – словник, який містить у собі основи слів (корені), другий – файл афіксів (під афіксом будемо розуміти частину слова, що вносить зміну у значення кореня). Мітки у системі Hunspell – це спеціально зарезервовані службові слова, що дозволяють задавати спеціальні правила перевірки за словником. За призначенням мітки(атрибути) умовно розділимо на групи:

- Загальні мітки (призначені для встановлення основних мовних налаштувань словника).
- Мітки для пропозицій (призначенні для реалізації механізму пропонування слів у випадку знайденої помилки).
- Мітки для створення складених слів (призначені для реалізації правил утворення складених слів, використовуються рідко).
- Мітки для створення афіксів (призначенні для визначення та розміщення афіксів).
- Інші специфічні мітки (використовуються рідко, в основному для мов угорської групи).

Повний перелік допустимих міток (атрибутів) наведено в табл. 1.

Таблиця 1

Мітки (атрибути) системи Hunspell

Загальні мітки.	Мітки для пропозицій.	Мітки для створення складених слів.	Мітки для створення афіксів.	Інші мітки.
SET FLAG COMPLEXPREFIXES LANG AF AM	TRY NOSUGGEST MAXNGRAMSUGS NOSPLITSUGS SUGSWITHDOTS REP MAP	BREAK COMPOUNDRULE WORDCHARS COMPOUNDFLAG COMPOUNDBEGIN COMPOUNDLAST COMPOUNDMIDDLE ONLYINCOMPOUND COMPOUNDPERMITFLAG COMPOUNDFORBIDFLAG COMPOUNDROOT COMPOUNDWORDMAX CHECKCOMPOUNDDUP CHECKCOMPOUNDREP CHECKCOMPOUNDCASE CHECKCOMPOUNDTRIPLE CHECKCOMPOUNDPATTERN COMPOUNDSYLLABLE SYLLABLENUM	PFX SFX	CIRCUMFIX FORBIDDENWORD KEEPCASE LEMMA_PRESENT NEEDAFFIX PSEUDOROOT WORDCHARS CHECKSHARPS

Файл словника, зазвичай, має розширення *.dic та містить список основ слів (коренів). У першому рядку словника вказується приблизна кількість слів в словнику. Це значення використовується для оптимального розподілу пам'яті. Після кожного слова може слідувати знак «/» і одна або більше міток, які відповідають афіксам і атрибутам. По замовчуванню мітка являє собою один (зазвичай алфавітний) символ. У файлі словника Hunspell також може існувати поле для морфологічного опису, що відділяється табуляцією. Формат морфологічного опису визначається користувачем [15].

Файл афіксів, зазвичай, має розширення *.aff та може містити у собі необов'язкові атрибути. Наприклад, SET для визначення типу кодування символів в файлах афіксів і словників. TRY визначає символи для замін. REP визначає таблицю замін для виправлення декількох символів. PFX і SFX визначають класи префіксів і суфіксів, які позначені мітками афіксів. Докладніше про структуру словника та значення кожної з міток можна прочитати у посібнику користувача на сайті проекту (http://hunspell.sourceforge.net/).

Окрім перевірки орфографії, система Hunspell має інструментарій для морфологічного аналізу слів, відокремлення основ, розміщення та аналізу переносів слів, пошуку синонімів, антонімів, омонімів, паронімів та побудови тезаурусів. Тому, у проект, окрім файлу словників та афіксів, також можуть входити файли hyph_*.dic (словник правил для перевірки та розміщення переносів), th_*.dat (словник – тезаурус синонімів), th_*.idx (індекс-файл синонімів). Також для Hunspell розроблено багато загальнолексичних словників, зокрема, найбільша колекція словників розміщена на сайті проекту OpenOffice.org (http://extensions.services.openoffice.org/en/dictionaries). Файли-словники для перевірки орфографії для OpenOffice мають розширення *.oxt і являють собою звичайні zip-архіви зі словниками Hunspell.

На основі представлених можливостей технології Hunspell та для вирішення поставлених у меті роботи проблем у наступній частині роботи вперше запропонована схема структурної організації підсистеми перевірки орфографії електронного каталогу бібліотеки.

Структурна схема підсистеми перевірки орфографії. Підсистема перевірки орфографії СВДЕК, що реалізована на платформі .NET для операційної системи Windows, схематично має структуру, представлену автором на рис. 2.

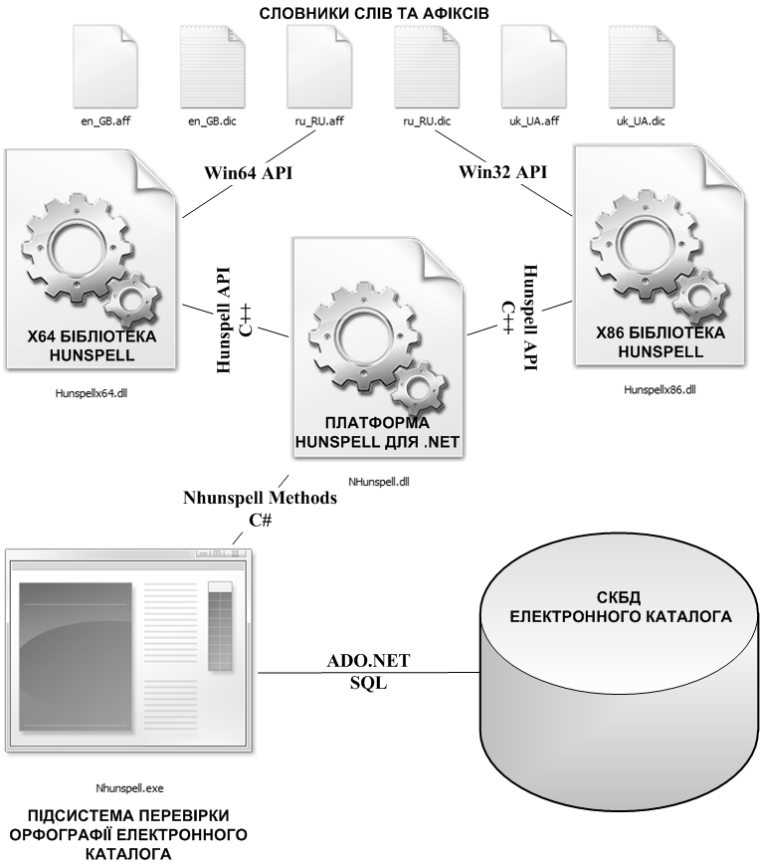


Рис. 2. Схема структурної організації підсистеми перевірки орфографії електронного каталогу бібліотеки

Запропонована авторська схема структурної організації підсистема перевірки орфографії СВДЕК складається з таких основних частин:

- Набір словників слів та афіксів – це файли з розширенням *.dic та *.aff відповідної структури Hunspell словників. Для проведення модельних тестів було обрано три словники відповідно української, російської та англійської мови. Для перевірки орфографії у полях бібліографічного запису, таких як «назва» або «анотація»(в яких містяться загальнолексичні слова), достатньо і загальних словників. Для перевірки орфографії інших полів бібліографічного запису (автор, видання, тощо) необхідно створювати спеціалізовані користувацькі словники. У даній системі не використовуються такі додаткові можливості системи Hunspell: морфологічний аналіз, розстановка переносів та побудова тезауруса. Але при потребі функціональність даної системи можливо розширити додаванням відповідних словників переносів та синонімів.
- Набір функцій та методів для роботи зі словниками Hunspell, що інкапсульовані відповідно до розрядності операційної системи у готові набори Hunspell API бібліотек (Hunspellx64.dll і Hunspellx86.dll). Дані бібліотеки написані на мові об'єктно-орієнтованого програмування C++ і доступні на сайті проекту (http://hunspell.sourceforge.net/).
- Набір методів для доступу до Hunspell API функцій, які інкапсульовані у класи відповідно до технології. У нашому випадку було обрано набір бібліотек для платформи .NET – Nhunspell (Nhunspell.dll). Далі у статті проводиться докладний аналіз об'єктно-орієнтованої моделі платформи NHunspell. Дана бібліотека написана на мові об'єктно-орієнтованого програмування C# і доступна на сайті проекту (http://nhunspell.sourceforge.net/).
- Підсистема перевірки орфографії і СКБД електронного каталогу бібліотеки – клієнт-серверний програмний комплекс, задача якого забезпечувати якість даних електронного каталогу. В залежності від платформи, на якій реалізована (у нашому випадку платформа C#.NET) клієнтська частина, тобто підсистема перевірки орфографії, та від типу СКБД (у нашому випадку Microsoft SQL Server), використовується відповідна технологія доступу до даних (у нашому випадку ADO.NET). Однак, запропонована схема структурної організації підсистеми перевірки орфографії є крос-платформною і не залежить від обраної СКБД або технології реалізації.

Для реалізації запропонованої авторської схеми структурної організації підсистеми СВДЕК перевірки орфографії на основні технології перевірки орфографії за словником Hunspell було проведено авторську модифікацію платформи Nhunspell. Суть модифікації полягала у розробці та представленні нового набору доступних класів та методів, які дають змогу перенести технологію Hunspell на платформу .NET.

Об'єктно-орієнтована модель модифікованої платформи NHunspell. Для повного розуміння запропонованих автором підходів далі у роботі зроблено короткий опис розроблених класів, методів та функцій. Дана інформація може бути корисною для розробників бібліотечного програмного забезпечення на платформі .NET та ілюструє один із можливих підходів до проектування підсистеми перевірки орфографії СВДЕК.

У просторі імен Nhunspell реалізовано 12 авторських класів для доступу до Hunspell API функцій ядра системи Hunspell. Перелік класів простору імен Nhunspell представлена на рис. 3.

Відповідно до схеми, класи згруповано за функціональним призначеннями. Зокрема публічний клас Hunspell реалізовує засоби перевірки орфографії, створення користувацьких словників слів та афіксів, морфологічний аналіз, механізм пропозицій для помилкових слів. Оскільки у даному класі реалізовані всі необхідні методи для забезпечення процесу перевірки орфографії, зупинимось на ньому докладніше.

Клас інкапсулює у собі 4 перезавантаження конструктора класу Hunspell(). Зазвичай використовують сигнатуру типу Hunspell(string affFile, string dictFile), де вхідні параметри affFile і dictFile є повним шляхом до словника афіксів і словника основ відповідно.

Метод Load(string affFile, string dictFile) має аналогічні перезавантаження, що й конструктор класу, і реалізовує можливість завантаження словників афіксів та основ у пам'ять.

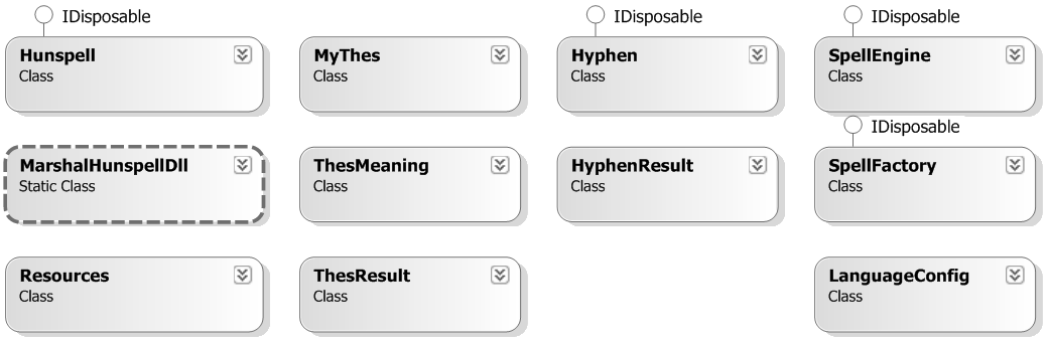


Рис. 3. Перелік класів Nhunspell

Методи Add(string word) і AddWithAffix(string word, string example) призначені для додавання слів-винятків (word) та афіксів слів-винятків за певним правилом (example), тобто слів, яких немає у основних словниках, але які вважаються правильними. Слід зауважити, що слова не заносяться до файлів основних словників, а лише зберігаються у оперативній пам’яті, тому для організації їх використання необхідно створювати окремі файли для зберігання даних слів і використовувати їх завантаження при кожному запуску системи. Метод повертає значення true, якщо додавання слова або афікса пройшло успішно, якщо ні – то false.

Для безпосередньої перевірки орфографії використовується метод Spell(string word), word – слово, що перевіряється. Метод повертає значення true, якщо дане слово є коректним (тобто наявне у словнику), інакше – false.

Методи Stem(string word), Analyze(string word) та Generate(string word, string sample) використовуються для морфологічного аналізу, зокрема, перші два – для отримання основи слова та основи слова за морфологією відповідно. Останній метод використовується для генерації всіх варіацій слова word, за шаблоном sample. Методи повертають значення типу список List<string>.

Реалізація механізму пропозицій для помилкових слів (помилковим вважається слово, що не пройшло перевірку за словником) здійснюється завдяки методу Suggest(string word), що повертає список (List<string>) можливих правильних слів для даного помилкового слова word.

На рис. 4 наведено перелік та сигнатуру усіх членів цього класу.

Класи Hyphen та HyphenResult призначені для перевірки та автоматичного розставлення переносів слів. Для використання можливостей даних класів необхідно додатково завантажувати словники з правилами розставлення переносів переносу hyph_*.dic.

Класи MyThes, ThesMeaning, ThesResult призначені для пошуку синонімів слів та створення тезаурусів. Для використання можливостей даних класів необхідно додатково завантажувати словники синонімів th_*.dat, th_*.idx (індекс-файл синонімів).

Структура даних класів наведена на рис. 5.

Класи SpellEngine, SpellFactory, LanguageConfig забезпечують можливість реалізації багато-поточності при обробці даних. Зокрема, забезпечують перевірку орфографії, розміщення переносів і роботу з тезаурусом у потокобезпечний спосіб. Структура даних класів представлена на рис. 6.

Розглянемо докладніше властивості класу LanguageConfig:

- HunspellAffFile – ім’я файла словника афіксів слів.
- HunspellDictFile – ім’я файла словника основ слів.
- HunspellKey – ключ файла словника основ.
- HuphenDictFile – ім’я файла словника правил переносу слів.
- LanguageCode – код мови.

- MyThesDatFile – ім’я файла тезауруса синонімів.
- MyThesIdxFile – ім’я індексованого файлу тезауруса синонімів.
- Processors – кількість процесорів, що може використовуватись під час роботи з певною мовою.

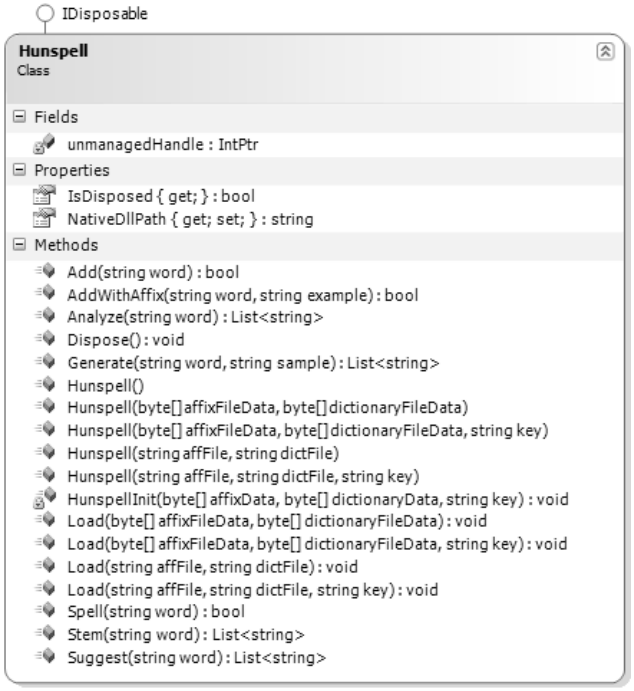


Рис. 4. Структура класу ***Hunspell***

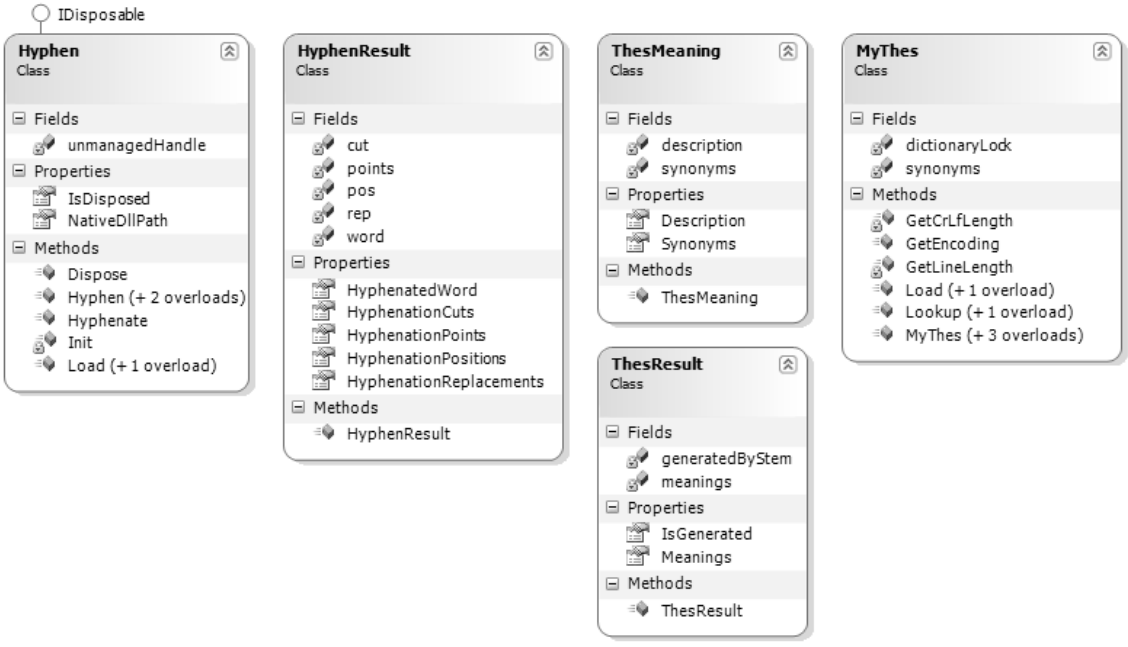


Рис. 5. Структура класів ***Hyphen, HyphenResult, ThesMeaning, ThesResult, MyThes***

За допомогою методу AddLanguage(LanguageConfig config) об’єкт класу LanguageConfig, що описує певну мову для перевірки, додається до списку об’єкта класу SpellEngine. Лише після цього, за допомогою індексатора, що реалізований в класі SpellEngine, об’єкти класу SpellFactory можуть застосовувати методи для перевірки орфографії та морфологічного аналізу паралельно за різними

словниками спосіб. Окрім описаних вище класів, у структурі платформи Nhunspell реалізовані ще два авторські класи – MarshalHunspellDll та Resources, що включають в себе делегати функцій та методів (під делегатами методу будемо розуміти структуру даних, що є вказівником на статистичні методи або методи екземпляра класу в .NET Framework) для доступу до Hunspell API. Методи та властивості класу MarshalHunspellDll призначені для виклику процедур і функцій основної бібліотеки Hunspell, що реалізовані на мові обєктно-орієнтованого програмування C++.

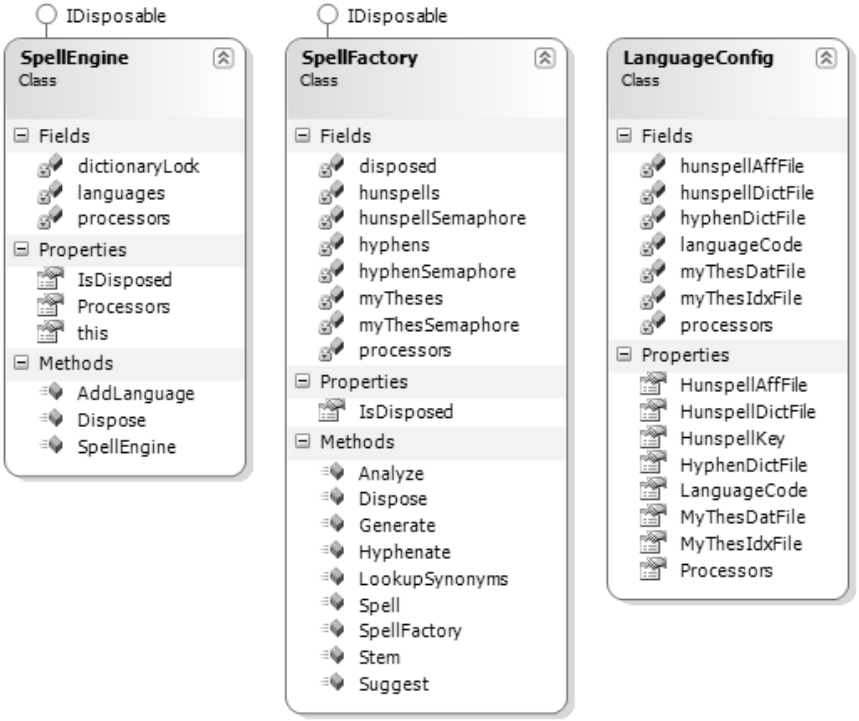


Рис. 6. Структура класів SpellEngine, SpellFactory, LanguageConfig

Експерименти з оцінювання швидкодії системи. На основні вищеописаних підходів та технологій було розроблено авторське програмне забезпечення, що реалізовує можливості пошуку орфографічних помилок у бібліографічних базах даних електронного каталогу бібліотеки.

Під час експлуатації запропонованої вище авторської системи виникла необхідність дослідження швидкодії ядра системи Hunspell, а саме, *оцінки часу затраченого на перевірку великого масиву слів*. Для оцінки *верхньої часової межі* для кожного слова повинен відбуватися повний перебір словника, тобто слово з великою ймовірністю повинно бути відсутнім у словнику.

На реальних бібліографічних базах даних електронного каталогу верхню межу по часу оцінити неможливо, адже вірогідність орфографічної помилки за статистичними дослідженнями не перевищує 0,1. Тому оцінка верхньої часової межі проводилась на даних, що генерувались генератором випадкових чисел. У такому випадку вірогідність відсутності у словнику генерованого слова прямує до одиниці. Для проведення експерименту було обрано словники з англійської, української та російської мови, оскільки вони є найбільш розповсюдженими у бібліографічних базах даних електронного каталогу.

Технічні умови проведення експерименту. Відповідно до поставлених цілей та передумов проведення експерименту, визначених у попередньому підрозділі роботи, для проведення тестування було обрано набори загальнолексичних словників OpenOffice.org. Причина вибору таких словників обумовлюється їх безкоштовністю, відкритістю та постійним оновленням. Параметри кожного із словників наведені у табл. 2.

Таблиця 2

Параметри словників для тестування

Мова	Назва файлу	Фізичний розмір файлу	Кодування	Кількість слів у словнику основ
Англійська	en_US.aff	64 145 байт	UTF-8	52890
	en_US.dic	641 025 байт	UTF-8	
Українська	uk_UA.aff	164 013 байт	UTF-8	115707
	uk_UA.dic	2 679 798 байт	UTF-8	
Російська	ru_RU.aff	394 839 байт	UTF-8	175302
	ru_RU.dic	5 141 236 байт	UTF-8	

Тестування проводилось на двох комп'ютерних платформах на базі операційної системи Microsoft Windows. Критичні параметри кожної із систем наведені у табл. 3.

Таблиця 3

Параметри комп'ютерних платформ тестування

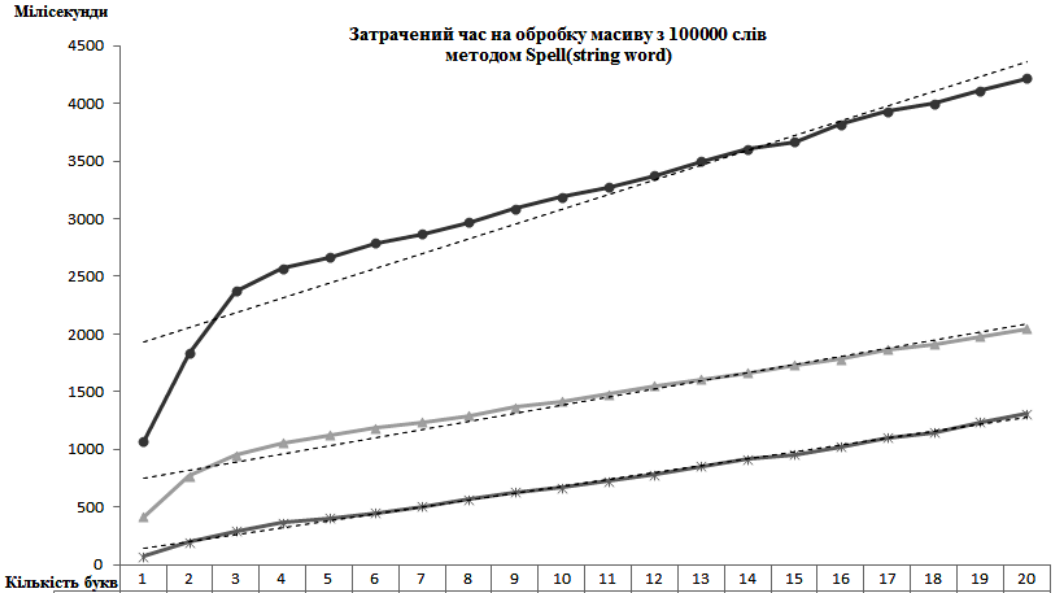
	Процесор	Оперативна пам'ять	Операційна система
Перша платформа	Intel Pentium Dual CPU E 2180 2,00 Ghz	2,00 ГБ	Microsoft Windows XP sp3 Professional 32 – розрядна
Друга платформа	AMD Turion X2 Dual Core Mobile RM-76 2,30Ghz	2,00 ГБ	Microsoft Windows 7 Professional 32 – розрядна

Дослідження швидкодії ядра системи Hunspell. На першому етапі під час експерименту для створення вхідних даних для перевірки було розроблено методи генерації випадкових слів певної довжини. Методи RandomWordRU(int n, int m), RandomWordUA(int n, int m) та RandomWordEN(int n, int m) використовуються для генерування n-ої кількості символьних послідовностей UNICODE (тестових слів довжини m) для російського, українського та англійського алфавіту відповідно. На другому етапі метод Test(string lang, int n, int m) реалізовує перевірку кожного тестового слова за допомогою методу Spell(string word) із класу Hunspell та повертає час (у мілісекундах), затрачений на перевірку n-ої кількості тестових слів довжини m за словником відповідної мови lang.

Тестування проводилось для кожного масиву слів відповідної мови. Розмірність масиву складала 100 000 тестових слів відповідної довжини. Максимальна довжина слова 20 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. 7 та 8.

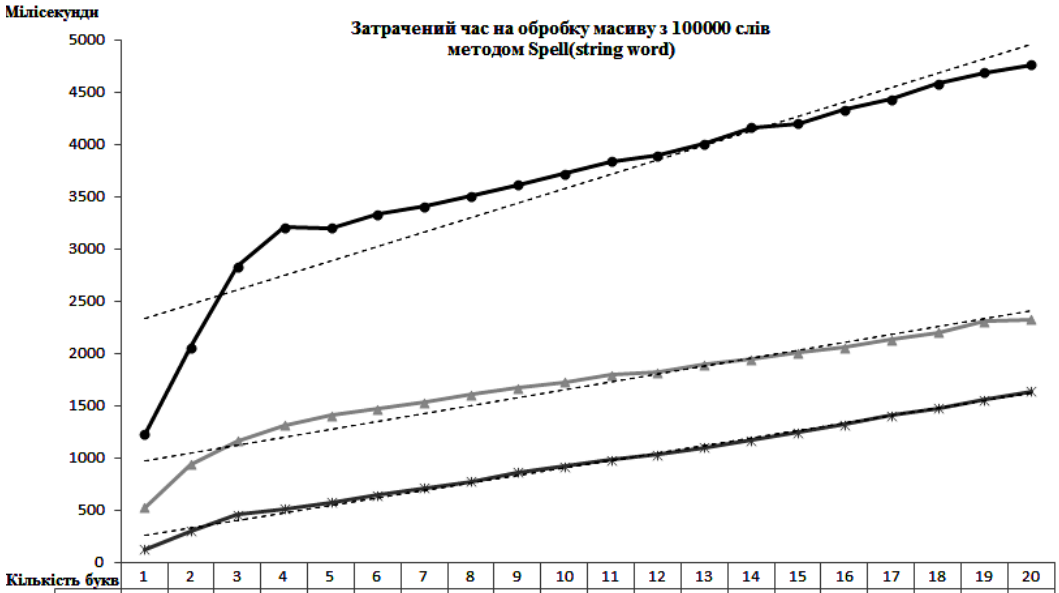
На рис. 7 та 8 наведено графіки залежності часу у мілісекундах (вісь ординат), витраченого на обробку масиву генерованих слів від довжини слова (вісь абсцис). Довжина слова змінюється з кроком 1. Максимальне значення довжини слова у експерименті бралось рівне 20.

На графіках (рис. 7 та 8) перша від осі абсцис лінія представляє залежність затраченого часу на перевірку масиву зі 100000 слів від довжини слова за словниками англійських слів, друга та третя відповідно за словниками українських та російських слів. Відповідно з графіків видно, що затрачений час прямо пропорційно зростає із обсягом словника.



Кількість букв	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Ru	1065	1830	2372	2570	2665	2789	2867	2969	3089	3190	3273	3376	3499	3606	3669	3820	3933	4001	4114	4217
Uk	415	772	947	1055	1123	1187	1235	1293	1365	1418	1478	1549	1606	1662	1729	1782	1864	1914	1977	2044
En	73	198	291	364	405	450	502	564	627	669	724	783	856	917	953	1021	1101	1146	1239	1311

Рис. 7. Результати тестування на першій апаратній платформі



Кількість букв	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Ru	1221	2053	2828	3207	3198	3332	3403	3504	3613	3721	3834	3894	4008	4161	4197	4334	4428	4582	4687	4762
Uk	523	936	1164	1310	1405	1470	1528	1606	1670	1724	1791	1817	1891	1943	2005	2057	2133	2198	2306	2324
En	124	296	457	514	573	643	708	776	861	915	981	1027	1101	1168	1242	1316	1408	1473	1556	1635

Рис. 8. Результати тестування на другій апаратній платформі

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.
- Для мов, що базуються на кирилиці (українська, російська), спостерігається відхилення від лінійної залежності для масиву генерованих слів довжиною приблизно 1-4 символи, що

зумовлене морфологічною специфікою мови. Також дане відхилення спричинене більш вірогідною генерацією коректного слова меншої довжини.

Висновки

У результаті проведеного дослідження було запропоновано авторські підходи для проектування підсистеми перевірки орфографії, що сформульовані у самій структурі статті та підрозділах її основної частини.

Проаналізувавши функціональні вимоги, було обрано відкриту технологію Hunspell для побудови на її основі підсистеми перевірки орфографії електронного каталогу бібліотеки. Запропонована схема структурної організації підсистеми на основі платформи .NET. Доведено на практиці, що на основі даної схеми можливо побудувати ефективну підсистему перевірки орфографії для електронного каталогу бібліотеки. Запропонована схема є новою і ілюструє технологічний процес перевірки орфографії на основні технології Hunspell.

Опис та аналіз об'єктно-орієнтованої моделі модифікованої платформи NHunspell показав, що запропонована модифікована платформа Nhunspell має у своєму складі необхідний інструментарій для перевірки орфографії, створення користувацьких словників основ слів та афіксів, морфологічного аналізу, механізму пропозицій для помилкових слів, роботи з тезаурусом.

Експерименти з оцінки верхньої часової межі ефективності ядра системи Hunspell показали лінійну залежність від характеристик комп'ютерної платформи і параметрів словників, що доводить можливість ефективно застосовувати дану технологію при розробці СВДЕК.

1. Вершинин М.И. Электронный каталог проблемы и решения / М. И. Вершинин. – СПб.: ПРОФЕССИЯ, 2007. – 233 с. 2. Карауш А.С. Утилиты для проверки и коррекции электронных каталогов / А. С. Карауш, Д.Ю. Копытков, А.С. Макаревич // Библиотечное дело. – 2005. – № 6 (30). – С. 21–24. 3. Nielsen R. Lost articles: Filing problems with initial articles in data bases / R. Nielsen, J. M. Pyle // Libr. Resources a. techn. Services. – 1995. – Vol. 39, №3. – P. 291– 293. 4. Ballard T. pelling and typographical errors in library databases: one libr. System for noting out spelling error / T. Ballard // Computer in libr. – 1992 – Vol. 12, №6. – P. 14–19. 5. Ярмолюк Р.С. Задача аналізу текстових атрибутів в електронному каталозі / Р.С. Ярмолюк // Вісник Хмельницьк. нац. ун-ту, серія Технічні науки. – 2011. – № 3. – С. 225–228. 6. Knuth D.E Fast pattern matching in string / D.E. Knuth, J.H. Morris, V.R. Pratt // SIAM j. on computing. – 1977 – Vol. 6, № 2. – P. 323–350. 7. Karp R.M. Efficient randomized pattern-matching algorithms / R.M. Karp, M.O Rabin // IBM Journal of Research and Development – 1987 – Vol. 3, №2. – P. 249–260. 8. Boyer R.S. A fast string searching algorithm / R.S. Boyer, J.S. Moore // Communications of the ACM – 1977. – Vol. 20, № 10. – P. 762–772. 9. Wagner R.A. The string-to-string correction problem / R.A. Wagner, M.J. Fischer // Journal of the ACM – 1974 – Vol. 21, № 1. – P. 168–173. 10. Hirschberg D.S. A linear space algorithm for computing maximal common subsequences / D.S. Hirschberg // Communications of the ACM. – 1975. – Vol. 18, №6, – P. 341–343. 11. Hunt J.W. A fast algorithm for computing longest common subsequences / J.W. Hunt, T.G. Szymanski // Communications of the ACM – 1977 – Vol. 20, №5, – P. 350–353. 12. Myers E.W. An O (ND) difference algorithm and its variations / E.W. Myers // Algorithmica. – 1986. – Vol. 1. – P. 251–266. 13. Ярмолюк Р.С. Структурно-функціональна модель системи верифікації даних електронного каталогу / Р.С. Ярмолюк // Сучасні проблеми діяльності бібліотеки в умовах інформаційного суспільства: матеріали третьої науково-практичної конференції. – Львів: Вид-во Нац. ун-ту "Львівська політехніка", 2011. – С. 217–224. 14. Schildt H. C# 4.0 The Complete Reference / H. Schildt –McGraw-Hill Osborne Media, 2010. – 976p. 15. Surhone L.M. Hunspell / L.M. Surhone, M.T. Tennoe, S.F. Henssonow – Betascript Publishing, 2010. – 116 p.