Хмельницький національний університет
Факультет програмування та комп'ютерних і телекомунікаційних систем
Кафедра комп'ютерних наук та інформаційних технологій

ДИПЛОМНА РОБОТА
МАГІСТРА

на тему *Інформаційна технологія автоматизованого формування семантичної структури інформаційного навчального матеріалу*

Галузь знань *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Виконав: *студент 2 курсу, група КНм-19-1* Підпис *В.О. Слободзян* Ініціали, прізвище

Керівник: *ст. викладач кафедри КНІТ* Підпис *О.В. Мазурець* Ініціали, прізвище

Нормоконтроль: *к.т.н., доцент кафедри КНІТ* Підпис *Р.О. Багрій* Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КНІТ, д.т.н., професор Підпис *О. В. Бармак* Ініціали, прізвище
7 12 2020 р.

Хмельницький 2020

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет *програмування та комп'ютерних і телекомунікаційних систем*

Кафедра *комп'ютерних наук та інформаційних технологій*

Освітній ступінь *магістр*

Галузь знань *12 – Інформаційні технології*

Спеціальність *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ

Завідувач кафедри *комп'ютерних наук та інформаційних технологій*

(підпис)

д.т.н., професор О.В. Бармак

«7» 09 *2020* року

ЗАВДАННЯ

НА ДИПЛОМНУ РОБОТУ МАГІСТРА

1. Тема дипломної роботи магістра: *«Інформаційна технологія автоматизованого формування семантичної структури інформаційного навчального матеріалу»*

2. Завдання видано студенту *Слободзяну Віталію Олександровичу*
(прізвище, ім'я, по батькові)

3. Керівник роботи *старший викладач Мазурець Олександр Вікторович*
(прізвище, ім'я, по батькові)

4. Затверджені наказом університету від «9» 09 2020 р. № 22

5. Зміст пояснювальної записки (перелік задач) та вихідні дані:

Мета роботи – розробка інформаційної технології автоматизованого формування семантичної структури інформаційного навчального матеріалу та дослідження практичної ефективності розробленої інформаційної технології шляхом створення відповідної експериментальної інформаційної системи та її експериментального тестування. Розроблена інформаційна технологія має дозволяти за вхідними даними в вигляді цифрового файлу електронного документу одержувати семантичну структуру інформаційного навчального матеріалу.

Реферат

Дипломна робота магістра присвячена розробці інформаційної технології автоматизованого формування семантичної структури інформаційного навчального матеріалу та відповідної інформаційної системи для дослідження практичної ефективності розробленої технології.

Актуальність теми. Освіта є одною з головних речей, що потребує людина у своєму житті. Чим вищий рівень освіти має людина, чим більше вона знає і краще вміє вчитися, тим швидше вона здатна засвоїти додаткові професійні навички, оволодіти новою професією, зорієнтуватися в зміні ситуації, прийняти правильне рішення. Не маючи належного рівня освіти, людина може взагалі не знайти підходящої роботи, не реалізуватися в професійному житті. Маючи вищий рівень освіти людина має менше конкурентів на ринку праці і тим ширшим є у неї вибір. Право здобути освіту має кожен. Якщо раніше на шляху до здобуття освіти могли стати обмежені можливості людини, нестача часу, чи відсутність навчальних закладів у віддалених регіонах, то на сучасному етапі розвитку освіти це вирішується з допомогою різних форм навчання, таких як заочна чи дистанційна, що дозволяють здобувати освіту будь кому, навіть з найвіддаленіших куточків планети. Все, що для цього потрібно – доступ до мережі Інтернет та базові вміння користування комп'ютером. Сучасні можливості ІТ роблять дистанційне навчання чи не найзручнішою формою здобування освіти, створюючи багато переваг перед очним навчанням.

Під дистанційним навчанням розуміють такий тип навчання де на різних етапах навчання взаємодія студента та викладача забезпечується в інтерактивному режимі за допомогою інформаційно-комунікаційних технологій (ІКТ). Невід'ємним елементом дистанційного навчання є дистанційний курс (ДК). ДК розробляються тьюторами (викладачі ДК) до початку навчання та наповнюють його необхідними навчальними матеріалами та засобами контролю знань (такими як самостійні роботи, контрольні роботи чи тестові завдання). В процесі навчання ДК може доповнюватися і змінюватися. Тьютор може сам

вирішувати, як буде виглядати ДК і які навчальні матеріали в ньому будуть використовуватися.

До навчальних матеріалів відносяться презентації, відео-презентації, практикуми, підручники, посібники, різного виду допоміжні ресурси такі як збірники документів і матеріалів, довідники, словники чи енциклопедії. Хоч різні мультимедійні навчальні матеріали вважаються найбільш ефективними та саме текстові матеріали є досі найпоширенішими.

Сучасний світ розвивається надзвичайно стрімко, постійно з’являються нові технології та нові теми для вивчення, через це будь які навчальні матеріали швидко стають не актуальними і потребують оновлення. Перевидання паперових видань може займати багато часу та потребувати значних фінансових витрат. Тому доцільно використовувати електронні навчальні матеріали (ЕНМ). Їх значно легше оновлювати та поширювати, крім того не потрібно витрачатись на передрукування та пересилання поштою студентам дистанційних форм навчання. Все що потрібно це оновити електронний документ та відправити його електронною поштою. Саме тому ЕНМ є найпоширенішою та чи не єдиною поширеною формою навчальних матеріалів для дистанційних форм навчання. На сьогодні в інтернеті знаходиться незліченна кількість електронних матеріалів. Лише в онлайн-енциклопедії «Вікіпедія» знаходиться приблизно 49.9 мільйонів статей, з яких більше 900 тисяч українською мовою.

Навчальний матеріал являє собою складну систему, яка має свою структуру зі специфічними елементами й зв'язками між ними. Будучи своєрідною "сировиною" процесу навчання, навчальний матеріал включає всю інформацію, яка подана для засвоєння й сприяє засвоєнню. Тобто навчальний матеріал ми розглядаємо як сукупність двох інформацій: основної та допоміжної. Кінцева мета подання основної інформації є перетворення її в знання або вміння (знання - це сприйнята, зрозуміла, усвідомлена й включена в систему наявних знань інформація).

ІНМ являється широким поняттям, що охоплює різні види інформаційних матеріалів, які використовуються в освіті. Це сукупність всієї інформації, яка

одержується і накопичується в процесі розвитку науки, культури, освіти, практичної діяльності людей і функціонування спеціальних пристроїв, що використовуються в суспільному виробництві та управлінні. До інформаційних ресурсів ДН належать інформаційно-навчальні матеріали: лекції, словники, посилання на літературні джерела, посилання на віддалені мережеві ресурси (бази даних, WWW-сервери, програмне забезпечення тощо). Ці інформаційні ресурси є основною складовою ДК.

Враховуючи великі обсяги електронних документів, єдиним шляхом до роботи з ними є автоматизація визначення їх семантичного вмісту. Відповідно до проведеного аналізу предметної області, поданням семантичного вмісту електронного слабо структурованого документу є ієрархічна система рубрикації та переліки ключових термінів рубрик. У системі освіти, для ІНМ таким поданням може бути структура заголовків документу та відповідні їх множини ключових термінів.

Мета і задачі роботи. Мета роботи полягає у розробці інформаційної технології автоматизованого формування семантичної структури інформаційного навчального матеріалу. Для досягнення поставленої мети визначенні наступні задачі дослідження:

1. Провести аналіз сучасних рішень з автоматизованого моделювання семантичної структури інформаційного навчального матеріалу з метою визначення нерозв’язаних задач з побудови семантичної структури інформаційних навчальних матеріалів.

2. Удосконалити інформаційну модель семантичної структури інформаційного навчального матеріалу, яка дозволяє зберігати всі елементи та атрибути інформаційного навчального матеріалу, необхідні для побудови його семантичної структури.

3. Удосконалити метод формування рубрикації інформаційного навчального матеріалу з метою одержання ієрархічної структури цифрового документу інформаційного навчального матеріалу та визначення відповідних елементам структури фрагментів контенту.

4. Удосконалити метод пошуку ключових термінів у рубриках інформаційного навчального матеріалу, що дозволяє одержувати сортовані за рівнем семантичної значущості множини ключових слів та словосполучень.

5. Розробити інформаційну технологію автоматизованого формування семантичної структури інформаційного навчального матеріалу, яка дозволяє за вхідними даними в вигляді цифрового файлу електронного документу одержувати семантичну структуру інформаційного навчального матеріалу.

6. Дослідити практичну ефективності розробленої інформаційної технології шляхом створення відповідної експериментальної інформаційної системи та її тестування.

Об'єкт дослідження – семантична структура інформаційного навчального матеріалу.

Предмет дослідження – моделі, методи та алгоритми автоматизації побудови семантичної структури інформаційного навчального матеріалу.

Методи дослідження, застосовані для вирішення поставлених завдань: для визначення ключових слів – метод дисперсійної оцінки; для роботи з корпусом слів української мови – методи автоматизованої обробки природної мови; для реалізації інформаційної технології – методології проєктування інформаційних систем та об'єктно-орієнтований підхід.

Наукова новизна одержаних результатів. В результаті проведеної роботи були отримані такі результати:

– Набула подальшого розвитку інформаційну модель семантичної структури інформаційного навчального матеріалу, яка відрізняється тим, що містить елементи та атрибути семантичної структури інформаційного навчального матеріалу, необхідні для роботи інформаційної технології.

– Набув подальшого розвитку метод формування рубрикації інформаційного навчального матеріалу, який відрізняється тим, що формує ієрархічну структуру рубрикації інформаційного навчального матеріалу та обмежує відповідні рубрикам фрагменти контенту.

– Набув подальшого розвитку метод пошуку ключових термінів у рубриках інформаційного навчального матеріалу, який відрізняється тим, що використовує дисперсійне оцінювання слів і словосполучень та визначає оцінку їх семантичної важливості в рубриці.

– Вперше розроблено інформаційну технологію автоматизованого формування семантичної структури інформаційного навчального матеріалу, яка забезпечує автоматизоване одержування семантичної структури інформаційного навчального матеріалу за цифровим файлом електронного документу.

Практичне значення одержаних результатів. На основі розробленої інформаційної технології було розроблено експериментальну інформаційну систему автоматизованого формування семантичної структури інформаційного навчального матеріалу, і було проведено її тестування й формування висновків щодо перспектив використання розробленої інформаційної технології.

Розроблена для дослідження практичної ефективності інформаційної технології експериментальна інформаційна система виконує наступні основні функції: зчитування та парсинг електронного ІНМ у форматі .docx; формування ієрархічної структури рубрикації інформаційного навчального матеріалу та обмеження відповідних рубрикам фрагментів контенту; пошук ключових термінів у рубриках інформаційного навчального матеріалу з використанням дисперсійного оцінювання слів і словосполучень та визначення оцінки їх семантичної важливості в рубриках; побудова семантичної структури інформаційного навчального матеріалу. При розробці автоматизованої інформаційної системи автоматизованого формування семантичної структури інформаційного навчального матеріалу для нормалізації слів вхідного тексту використано існуючу базу даних корпусу слів української мови.

За результатами дослідження, проведеного з використанням розробленої автоматизованої системи автоматизованого формування семантичної структури інформаційного навчального матеріалу, було доведено спроможність інформаційної технології розв’язувати поставлені задачі.

Результати експериментальної перевірки інформаційної технології автоматизованого формування семантичної структури інформаційного навчального матеріалу були визначені шляхом розробки й використання відповідної інформаційної системи. Результати експериментального тестування запропонованої інформаційної технології довели її спроможність розв'язувати поставлені задачі. При цьому, середня точність пошуку ключових термінів складає 73,2%, середня повнота пошуку – 69,7%. Знайдені ключові терміни містяться в середньому в 81,2% абзаців, або в 87,3% речень.

Апробація результатів дипломної роботи магістра та публікації. Основні наукові та практичні результати *доповідалися* на конференціях:

– доповідь на тему «Аналіз сучасних спеціалізованих програмних розширень для автоматизації роботи з цифровими документами навчальних матеріалів» на Міжнародній конференції молодих вчених «Сучасні технології в механіці» (Хмельницький, 18-21 квітня 2018 року, Хмельницький національний університет);

– доповідь на тему «Метод формального опису елементів моделей автоматизованого формування тестових завдань» на Всеукраїнській науково-практичній конференції «Інтелектуальний потенціал – 2018» (Хмельницький, 14-16 листопада 2018 року, Університет економіки і підприємництва);

– доповідь на тему «Використання програмного розширення spire.doc для автоматизації роботи з цифровими документами» на XI Всеукраїнській науково-практичній конференції «Актуальні проблеми комп'ютерних наук АПКН-2019» (Хмельницький, 14-15 листопада 2019 року, Хмельницький національний університет);

– доповідь на тему «Аналіз результатів автоматизованого пошуку ключових термінів у навчальних матеріалах» на XII Всеукраїнській науково-практичній конференції «Актуальні проблеми комп'ютерних наук АПКН-2020» (Хмельницький, 9-10 листопада 2020 року, Хмельницький національний університет);

– За темою дипломної роботи магістра автором виконано п’ять *наукових публікацій*, одна з яких у фаховому виданні, яке включено у перелік МОН України.

Структура та обсяг роботи. Дипломна робота магістра складається з завдання, реферату, змісту, переліку скорочень, вступу, 4 розділів, висновків, переліку посилань із 39 найменувань та 8 додатків. Загальний обсяг дипломної роботи магістра становить 193 сторінок, з них 95 сторінок основного тексту та 98 сторінок додатків. У роботі наведено 38 рисунків та 10 таблиць.

Ключові слова: ключові слова, ключові терміни, інформаційна система, інформаційна технологія, текстовий контент, дисперсійна оцінка, семантична структура, навчальний матеріал.

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
ІНМ	Інформаційний навчальний матеріал
НМ	Навчальний матеріал
БД	База даних
ІТ	Інформаційна технологія
ДС	Дистанційний курс
ІКТ	Інформаційно-комунікаційна технологія
КН	Комп’ютерні науки
MS	Microsoft
ІС	Інформаційна система
ПП	Програмний продукт
СКБД	Система керування базами даних
PIA	Primary Interop Assembly
TF-IDF	Term Frequency-Inverse Document Frequency

Вступ

Дипломна робота магістра присвячена розробці інформаційної технології автоматизованого формування семантичної структури інформаційного навчального матеріалу та відповідної інформаційної системи для дослідження практичної ефективності розробленої технології.

Актуальність теми. Освіта – одна з головних речей, що потребує людина у своєму житті. Чим вищий рівень освіти має людина, чим більше вона знає і краще вміє вчитися, тим швидше вона здатна засвоїти додаткові професійні навички, оволодіти новою професією, зорієнтуватися в зміні ситуації, прийняти правильне рішення. Не маючи належного рівня освіти, людина може взагалі не знайти підходящої роботи, не реалізуватися в професійному житті. Маючи вищий рівень освіти людина має менше конкурентів на ринку праці і тим ширшим є у неї вибір [1]. Право здобути освіту має кожен, держава забезпечує доступність і безоплатність професійно-технічної та вищої освіти в державних і комунальних навчальних закладах [2]. Якщо раніше на шляху до здобуття освіти могли стати обмежені можливості людини, нестача часу, чи відсутність навчальних закладів у віддалених регіонах, то на сучасному етапі розвитку освіти це вирішується з допомогою різних форм навчання, таких як заочна чи дистанційна, що дозволяють здобувати освіту будь кому, навіть з найвіддаленіших куточків планети. Все, що для цього потрібно – доступ до мережі Інтернет та базові вміння користування комп’ютером. Сучасні можливості ІТ роблять дистанційне навчання чи не найзручнішою формою здобування освіти, створюючи багато переваг перед очним навчанням.

Дистанційне навчання – це форма навчання де на різних етапах навчання взаємодія студента та викладача забезпечується в інтерактивному режимі за допомогою інформаційно-комунікаційних технологій (ІКТ). Невід’ємним елементом дистанційного навчання є дистанційний курс (ДК) [3]. ДК розробляються тьюторами (викладачі ДК) до початку навчання та наповнюють

його необхідними навчальними матеріалами та засобами контролю знань (такими як самостійні роботи, контрольні роботи чи тестові завдання). В процесі навчання ДК може доповнюватися і змінюватися. Тьютор може сам вирішувати, як буде виглядати ДК і які навчальні матеріали в ньому будуть використовуватися.

До навчальних матеріалів відносяться презентації, відео-презентації, практикуми, підручники, посібники, різного виду допоміжні ресурси такі як збірники документів і матеріалів, довідники, словники чи енциклопедії. Хоч різні мультимедійні навчальні матеріали вважаються найбільш ефективними та саме текстові матеріали є досі найпоширенішими.

Сучасний світ розвивається надзвичайно стрімко, постійно з’являються нові технології та нові теми для вивчення, через це будь які навчальні матеріали швидко стають не актуальними і потребують оновлення. Перевидання паперових видань може займати багато часу та потребувати значних фінансових витрат. Тому доцільно використовувати електронні навчальні матеріали (ЕНМ). Їх значно легше оновлювати та поширювати, крім того не потрібно витрачатись на передрукування та пересилання поштою студентам дистанційних форм навчання. Все що потрібно це оновити електронний документ та відправити його електронною поштою. Саме тому ЕНМ є найпоширенішою та чи не єдиною поширеною формою навчальних матеріалів для дистанційних форм навчання. На сьогодні в інтернеті знаходиться незліченна кількість електронних матеріалів. Лише в онлайн-енциклопедії «Вікіпедія» знаходиться приблизно 49.9 мільйонів статей, з яких більше 900 тисяч українською мовою [4].

Навчальний матеріал являє собою складну систему, яка має свою структуру зі специфічними елементами й зв'язками між ними. Будучи своєрідною "сировиною" процесу навчання, навчальний матеріал включає всю інформацію, яка подана для засвоєння й сприяє засвоєнню. Тобто навчальний матеріал ми розглядаємо як сукупність двох інформацій: основної та допоміжної. Кінцева мета подання основної інформації є перетворення її в

знання або вміння (знання - це сприйнята, зрозуміла, усвідомлена й включена в систему наявних знань інформація).

ІНМ – це широке поняття, що охоплює різні види інформаційних матеріалів, які використовуються в освіті. Це сукупність всієї інформації, яка одержується і накопичується в процесі розвитку науки, культури, освіти, практичної діяльності людей і функціонування спеціальних пристроїв, що використовуються в суспільному виробництві та управлінні. До інформаційних ресурсів ДН належать інформаційно-навчальні матеріали: лекції, словники, посилання на літературні джерела, посилання на віддалені мережеві ресурси (бази даних, WWW-сервери, програмне забезпечення тощо). Ці інформаційні ресурси є основною складовою ДК [5].

Враховуючи великі обсяги електронних документів, єдиним шляхом до роботи з ними є автоматизація визначення їх семантичного вмісту. Відповідно до проведеного аналізу предметної області, поданням семантичного вмісту електронного слабо структурованого документу є ієрархічна система рубрикації та переліки ключових термінів рубрик. У системі освіти, для ІНМ таким поданням може бути структура заголовків документу та відповідні їх множини ключових термінів.

Мета і задачі роботи. Мета роботи полягає у розробці інформаційної технології автоматизованого формування семантичної структури інформаційного навчального матеріалу. Для досягнення поставленої мети визначенні наступні задачі дослідження:

1. Провести аналіз сучасних рішень з автоматизованого моделювання семантичної структури інформаційного навчального матеріалу з метою визначення нерозв’язаних задач з побудови семантичної структури інформаційних навчальних матеріалів.

2. Удосконалити інформаційну модель семантичної структури інформаційного навчального матеріалу, яка дозволяє зберігати всі елементи та

атрибути інформаційного навчального матеріалу, необхідні для побудови його семантичної структури.

3. Удосконалити метод формування рубрикації інформаційного навчального матеріалу з метою одержання ієрархічної структури цифрового документу інформаційного навчального матеріалу та визначення відповідних елементам структури фрагментів контенту.

4. Удосконалити метод пошуку ключових термінів у рубриках інформаційного навчального матеріалу, що дозволяє одержувати сортовані за рівнем семантичної значущості множини ключових слів та словосполучень.

5. Розробити інформаційну технологію автоматизованого формування семантичної структури інформаційного навчального матеріалу, яка дозволяє за вхідними даними в вигляді цифрового файлу електронного документу одержувати семантичну структуру інформаційного навчального матеріалу.

6. Дослідити практичну ефективності розробленої інформаційної технології шляхом створення відповідної експериментальної інформаційної системи та її тестування.

Об’єкт дослідження – семантична структура інформаційного навчального матеріалу.

Предмет дослідження – моделі, методи та алгоритми автоматизації побудови семантичної структури інформаційного навчального матеріалу.

Методи дослідження, застосовані для вирішення поставлених завдань: для визначення ключових слів – метод дисперсійної оцінки; для роботи з корпусом слів української мови – методи автоматизованої обробки природної мови; для реалізації інформаційної технології – методології проєктування інформаційних систем та об’єктно-орієнтований підхід.

Наукова новизна одержаних результатів. В результаті проведеної роботи були отримані такі результати:

– Набула подальшого розвитку інформаційна модель семантичної структури інформаційного навчального матеріалу, яка відрізняється тим, що

містить елементи та атрибути семантичної структури інформаційного навчального матеріалу, необхідні для роботи інформаційної технології.

– Набув подальшого розвитку метод формування рубрикації інформаційного навчального матеріалу, який відрізняється тим, що формує ієрархічну структуру рубрикації інформаційного навчального матеріалу та обмежує відповідні рубрикам фрагменти контенту.

– Набув подальшого розвитку метод пошуку ключових термінів у рубриках інформаційного навчального матеріалу, який відрізняється тим, що використовує дисперсійне оцінювання слів і словосполучень та визначає оцінку їх семантичної важливості в рубриці.

– Вперше розроблено інформаційну технологію автоматизованого формування семантичної структури інформаційного навчального матеріалу, яка забезпечує автоматизоване одержування семантичної структури інформаційного навчального матеріалу за цифровим файлом електронного документу.

Практичне значення одержаних результатів. На основі розробленої інформаційної технології було розроблено експериментальну інформаційну систему автоматизованого формування семантичної структури інформаційного навчального матеріалу, і було проведено її тестування й формування висновків щодо перспектив використання розробленої інформаційної технології.

Розроблена для дослідження практичної ефективності інформаційної технології експериментальна інформаційна система виконує наступні основні функції: зчитування та парсинг електронного ІНМ у форматі .docx; формування ієрархічної структури рубрикації інформаційного навчального матеріалу та обмеження відповідних рубрикам фрагментів контенту; пошук ключових термінів у рубриках інформаційного навчального матеріалу з використанням дисперсійного оцінювання слів і словосполучень та визначення оцінки їх семантичної важливості в рубриках; побудова семантичної структури інформаційного навчального матеріалу. При розробці автоматизованої інформаційної системи автоматизованого формування семантичної структури

інформаційного навчального матеріалу для нормалізації слів вхідного тексту використано існуючу базу даних корпусу слів української мови.

За результатами дослідження, проведеного з використанням розробленої автоматизованої системи автоматизованого формування семантичної структури інформаційного навчального матеріалу, було доведено спроможність інформаційної технології розв'язувати поставлені задачі.

Результати експериментальної перевірки ІТ автоматизованого формування семантичної структури інформаційного навчального матеріалу були визначені шляхом розробки й використання відповідної ІС. Результати експериментального тестування запропонованої ІТ довели її спроможність розв'язувати поставлені задачі. При цьому, середня точність пошуку ключових термінів складає 73,2%, середня повнота пошуку – 69,7%. Знайдені ключові терміни містяться в середньому в 81,2% абзаців, або в 87,3% речень.

Апробація результатів дипломної роботи магістра та публікації. Основні наукові та практичні результати *доповідалися* на конференціях:

– доповідь на тему «Аналіз сучасних спеціалізованих програмних розширень для автоматизації роботи з цифровими документами навчальних матеріалів» на Міжнародній конференції молодих вчених «Сучасні технології в механіці» (Хмельницький, 18-21 квітня 2018 року, Хмельницький національний університет);

– доповідь на тему «Метод формального опису елементів моделей автоматизованого формування тестових завдань» на Всеукраїнській науково-практичній конференції «Інтелектуальний потенціал – 2018» (Хмельницький, 14-16 листопада 2018 року, Університет економіки і підприємництва);

– доповідь на тему «Використання програмного розширення spire.doc для автоматизації роботи з цифровими документами» на XI Всеукраїнській науково-практичній конференції «Актуальні проблеми комп'ютерних наук АПКН-2019» (Хмельницький, 14-15 листопада 2019 року, Хмельницький національний університет);

– доповідь на тему «Аналіз результатів автоматизованого пошуку ключових термінів у навчальних матеріалах на XII Всеукраїнській науково-практичній конференції «Актуальні проблеми комп'ютерних наук АПКН-2020» (Хмельницький, 9-10 листопада 2020 року, Хмельницький національний університет);

– За темою дипломної роботи магістра автором виконано п'ять *наукових публікацій* [35, 36, 37, 38, 39], одна з яких у фаховому виданні, яке включено у перелік МОН України [39].

Структура та обсяг роботи. Дипломна робота магістра складається з завдання, реферату, змісту, переліку скорочень, вступу, 4 розділів, висновків, переліку посилань із 39 найменувань та 8 додатків. Загальний обсяг дипломної роботи магістра становить 193 сторінок, з них 95 сторінок основного тексту та 98 сторінок додатків. У роботі наведено 38 рисунків та 10 таблиць.

Розділ 1

Аналіз сучасного стану проблеми моделювання семантичної структури інформаційного навчального матеріалу

1.1 Особливості використання інформаційних навчальних матеріалів

Навчальний матеріал представляє систему, що має структуру зі характерними елементами та зв'язками між ними [6]. Навчальний матеріал є невід'ємною частиною навчального процесу і містить всю необхідну для засвоєння інформацію, або для сприяння її засвоєнню. Отож необхідно розглядати навчальний матеріалі як об'єднання двох видів інформації: допоміжної та основної. Головною цілю основної інформації є трансформація її у вміння (готовність людини успішно виконувати певну діяльність, яка ґрунтується на знаннях і навичках) та знання (теоретично узагальнений суспільно-історичний досвід, результат оволодіння людиною дійсності, її пізнання). Водночас знання розглядають як зрозумілу, сприйняту, усвідомлену й включену в систему наявних знань інформацію [6]. Цілю допоміжної інформація є гарантування надійності, твердої впевненості у засвоєнні основної інформації.

Навчальний матеріал, зосереджений у навчальних посібниках, збірниках задач і вправ, підручниках, інструкціях, дидактичних матеріалах, довідниках, засобах наочності тощо, навчальний матеріал залежно від виконуваних функцій може бути розділений за далі переліченими групами: програмний, інформаційний, операційний, контролюючий, актуалізуючий, стимулюючий та діагностуючий (рисунок 1.1) [7]. Наведені види навчального матеріалу використовуються і комбінуються при формування навчальних курсів дисциплін.

Лише інформаційний, операційний і частково контролюючий види навчального матеріалу являються носіями основної інформації. Інші види виконують допоміжну роль, сприяючи більш міцному й більш оперативному

засвоєнню основної інформації, отож більш якісному. Допоміжний навчальний матеріал дозволяє привести зміст навчального предмету у відповідність до психологічних особливостей процесу пізнання.

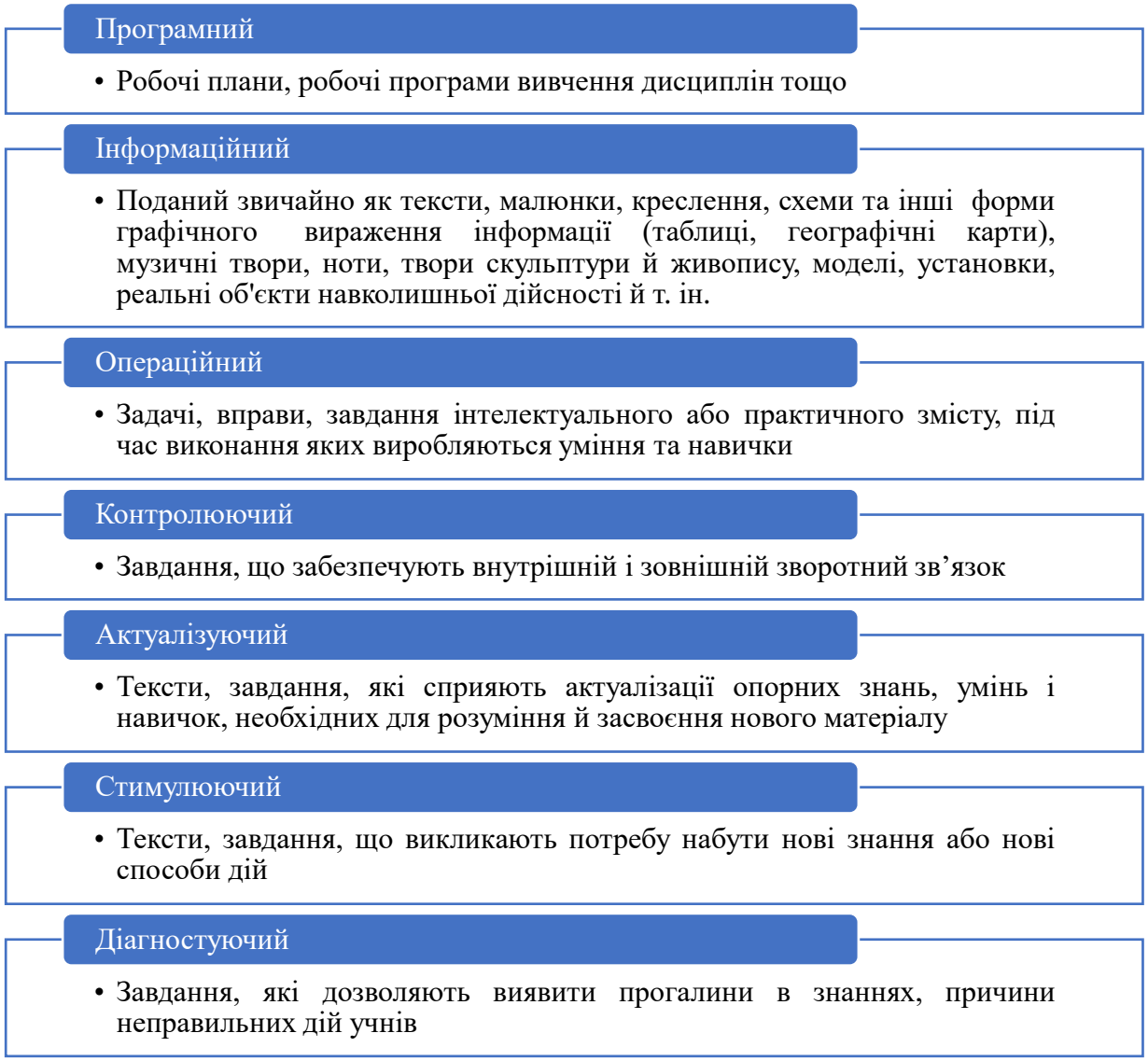


Рисунок 1.1 – Види навчальних матеріалів залежно від виконуваних функцій [7]

Ознакою структурної складності навчального матеріалу є середня кількість структурних елементів та зв’язків з одним елементом матеріалу. Складність розуміння навчальних матеріалів залежить від кількості структурних елементів та широти зв’язків: чим більше структурних елементів міститься у навчальних матеріалах, тим ширші їх зв’язки та вища складність [8].

Кожен структурний елемент включається до структури ІНМ за допомогою певних зв'язків, і різноманітність цих зв'язків також дуже велика. Наприклад, залежно від кількості ІНМ, зв'язки між структурними елементами можуть мати міжпредметний, міжцикловий і внутрішньопредметний характер. Між циклами предметів формуються міжциклові зв'язки, в цьому випадку вони відіграють роль структурних елементів. Внутрішньопредметні зв’язки особливо широкі; це можуть бути знакоутворюючі зв’язки, словоутворюючі, змістоутворюючі, функціонально-конструкційні, понятійні тощо [8].

Таким чином, ІНМ для включення у навчальний курс вимагає наявності визначеної семантичної структури та рубрикації, наявність яких покращує сприйняття матеріалу та дозволяє ефективно використовувати його складові у навчанні.

1.2 Дослідження структури інформаційного навчального матеріалу

Одним з атрибутів тексту є структура, виходячи з цього, ІНМ повинна відображати структуру тексту. Елементом структури ІНМ є заголовок – певна частина тексту виділена автором задля структуризації контенту ІНМ. Наявність такої структуризації сприяє кращому засвоєнню матеріалу та пришвидшує навігацію по ньому. На кількість заголовків та глибини їх структури впливає загальна кількість та складність тексту.

Кількість рівнів вкладеності елементів структури ІНМ може бути різною, нерівномірною в межах одного ІНМ, назви рівнів різних ІНМ можуть відрізнятись (наприклад, Дисципліна / Модуль / Розділ / Тема / Лекція / Параграф; Дисципліна / Розділ / Тема; Дисципліна / Розділ / Підрозділ / Параграф тощо). На рисунку 1.2 подано приклад, у якому назвами рівнів елементів структури ІНМ є «Дисципліна», «Модуль», «Розділ» та «Підрозділ» [9].

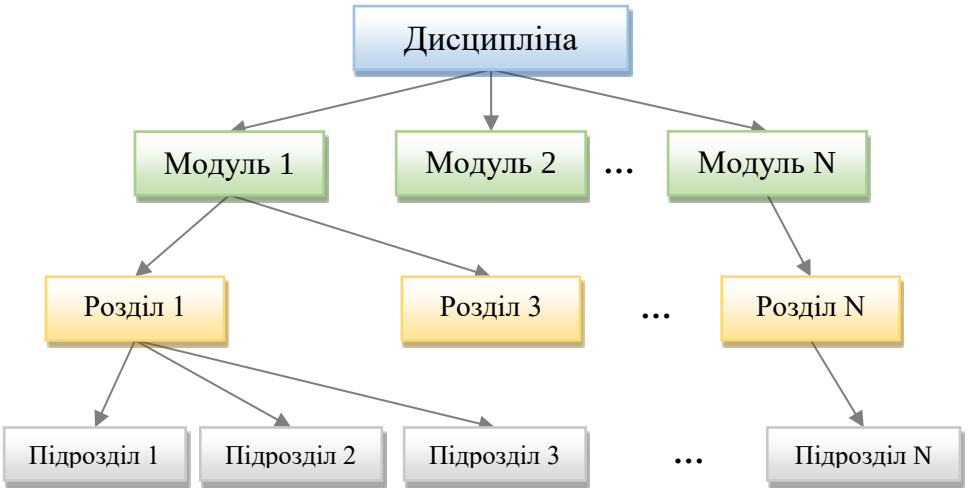


Рисунок 1.2 – Приклад формального подання структури ІНМ [9]

Тому окрім назви (семантичного ідентифікатора) кожного елементу структури ІНМ є важливим номер рівня цього елементу в ієрархічній структурі ІНМ, в той час як назва рівню не є семантично значущою. ІНМ мають структуру, потрібну для первинної структуризації всього контенту ІНМ, причому характерними властивостями такого підходу є наступне:

– елементи структури ІНМ розбивають контент ІНМ на окремі фрагменти;

– кожен елемент структури ІНМ, і відповідно фрагмент контенту, має назву – семантичний ідентифікатор (заголовок);

– кожен елемент структури ІНМ, і відповідно фрагмент контенту, може містити аналогічним чином підпорядковані елементи структури, що розділяють ці фрагменти на менші фрагменти з привласненням їм власних назв.

Отже, при формуванні семантичної структури ІНМ обов’язково слід враховувати наявну структуру інформаційного навчального матеріалу, подану в вигляді системи рубрикації. При цьому така структура має бути доповнена елементами, які подають семантичний вміст рубрик [9].

1.3 Аналіз сучасних підходів до автоматизованого формування семантичної структури інформаційного навчального матеріалу

Можна виділити три основні групи методів автоматичної побудови семантичної структури в залежності від області запозичення основного підходу: методи, які використовують лінгвістичні підходи, статистичні методи, методи, засновані на підходах з області штучного інтелекту [10].

Одним з методів автоматичної побудови онтологій на основі лінгвістичних засобів є підхід, який використовує лексико-синтаксичні шаблони, запропонований в [11]. Автор вважає, що для отримання онтології необхідно використовувати різні рівні дослідження природної мови: синтаксис, морфологію та семантику. Для автоматичної побудови онтології автор використовує метод лексико-синтаксичних шаблонів (один з рівнів семантичного аналізу текстів природною мовою). Як метод семантичного аналізу лексико-синтаксичних шаблонів давно використовуються в комп'ютерній лінгвістиці і являють собою характерні висловлювання й конструкції певних елементів мови. Дана методика семантичного аналізу не є спеціалізованою на певну предметну область. На основі лексико-синтаксичних шаблонів виділяються онтологічні конструкції. Зважаючи на складність завдання, оцінка результатів застосування підходу проводиться авторами опосередковано через аналіз результатів його використання в різних додатках Semantic Web. В цілому відзначається, що лексико-синтаксичні шаблони як метод семантичного аналізу текстів природною мовою – в разі великого обсягу колекції шаблонів – є ефективним засобом для автоматичної побудови онтологій [11].

Підхід на основі системи продукцій був запропонований в [12] і належить до групи методів автоматичної побудови онтологій, в основі яких лежать підходи з області штучного інтелекту. Автор стверджує, що ефективне автоматичне побудова онтологій може бути засноване на здатності методів штучного інтелекту отримання від тексту елементів знань і їх нетривіальною

переробці. Аналіз галузі природничо-мовної обробки тексту показує переважання використання різних правил при вирішенні завдань у розглянутій предметній області. Даний факт, а також декларативний характер уявлення методів автоматичної побудови онтологій, обґрунтовує застосування системи продукцій в якості моделі представлення знань про метод. Для створення методів автоматичної побудови онтологій автор розробляє модель генерації системи продукцій (на основі застосування генетичного програмування), модель генерації перетворювачів (на основі генетичного і автоматного програмування), модель генерації систем логічного висновку (також на основі генетичного і автоматного програмування) і модель апарату активації продукцій (на основі застосування автоматного програмування). Таким чином, автором методу пропонується модель автоматичної побудови онтологій у вигляді системи продукцій і застосуванні генетичного і автоматного програмування для створення необхідних моделей.

В [13] пропонується підхід на основі статистичних методів, де на першому етапі побудови онтології потрібно виділити входять до її складу класи. Раніше було відзначено, що поняття лінгвістичної онтології строго пов'язані з термінами. Таким чином, дана задача зводиться до визначення термінів розглянутої предметної області. Алгоритми вилучення термінів з текстів природною мовою можна розділити на дві групи: статистичні та лінгвістичні. Однак перші володіють певною перевагою, оскільки їх використання не залежить від лінгвістичних особливостей конкретної мови. Підхід до вилучення термінів у даній статті є переважно статистичним.

Таким чином, множини ключових термінів можуть бути використанні для доповнення системи рубрикації ІНМ. При цьому окремі множини ключових термінів є розширенням елементів структури ІНМ для розгорнутого подання семантичної структури інформаційного навчального матеріалу.

1.4 Методи пошуку ключових термінів у цифрових текстах

Терміном називається слово або словосполучення, що виражає чітко окреслене поняття певної галузі науки, мистецтва, техніки, культури, суспільно-політичного життя. Термін повинен мати конкретний зміст і межі поняття, логічне визначення, яке визначає суттєві ознаки значення поняття або предмета. Крім того термін повинен належати до певної термінологічної системи та бути однозначним та не мати синонімів та омонімів в її межах [14].

Ключовий термін – це термін, який використовуються для вираження змісту документу, тексту або його фрагменту. Ключовий термін має суттєве смислове навантаження та може служити ключем для пошуку інформації [15]. Ключовими термінами можуть виступати як слова, так і словосполучення й абревіатури (таблиця 1.1). Якщо абревіатура є сукупністю літер і може розглядатися як слово, то словосполучення є впорядкованою сукупністю слів.

Таблиця 1.1. Підходи до пошуку ключових термінів різних типів [9]

Тип терміну	**Підхід до формалізації**
Слово	Окремий елемент тексту, що характеризується неперервною (без прогалин/пробілів) сукупністю символів у тексті. До цього типу належать також скорочення, слова з апострофами, двокореневі слова та складені слова тощо. Одному слову відповідає один елемент множини унікальних слів документу. Приклади слів: «з'єднання», «авіарейс», «генерал-майор».
Словосполучення	Словосполучення є стійкими сукупностями важливих слів, що згруповані у визначеній послідовності та у такій комбінації неодноразово присутні в текстовому контенті. Синтакстично словосполучення як елемент тексту є сталою впорядкованою сукупністю (колокацією) слів. Одному словосполученню відповідає кілька елементів множини унікальних слів документу. Приклади словосполучень: «берег річки», «інформаційні технології».
Абревіатури та скорочення	Є стійкою зв'язною сукупністю символів, тому може визначатись та оброблятись як слово. Одній абревіатурі чи скороченню відповідає один елемент множини унікальних слів документу. Приклади абревіатур та скорочень: «ІНМ», «СКБД», «шт.».

Множина ключових термінів тексту є найбільш семантично стиснутим результатом семантичного аналізу тексту Для автоматизації пошуку ключових слів використовуються різноманітні методи аналізу текстів, серед яких широке розповсюдження одержали метод частотної оцінки TF, метод частотної оцінки та оберненої частоти документу TFIDF та метод дисперсійного оцінювання DE. Ці методи дозволяють поставлені у відповідність окремим словам або словосполученням тексту деякі певним чином визначені числові вагові значення, що вказують на міру їх важливості в досліджуваному тексті.

Частотна оцінка *TF* (term frequency) обчислює частоту появ певного слова *i* у тексті, що розглядається, й обчислюється наступним чином [16]:

$$TF_i = \frac{n(i)}{N}, \tag{1.1}$$

де *n(i)* – кількість появ слова *i* у тексті, *N* – загальна кількість слів у тексті.

Оцінка *TFIDF* є добутком частоти появ слова у тексті *TF* та зворотної документарної частоти слова *IDF* (inverse document frequency) [17]:

$$TFIDF = TF * IDF, \;\; IDF_i = \log \frac{D}{d_i}, \tag{1.2}$$

де *D* – кількість альтернативних документів для порівняння або фрагментів, на які розбивається текст при аналізі; d_i – кількість документів або фрагментів, у яких дане слово присутнє.

Дисперсійна оцінка *DE* (disperse evaluation) за змістом близька до оцінки TFIDF, та є оцінкою дискримінантної сили слів. Дисперсійний аналіз є статистичним методом оцінки зв’язку між факторними й результативними ознаками в різних групах, відібраний випадковим чином, заснований на визначенні розходжень (розкиду) значень ознак. В основі дисперсійного аналізу лежить аналіз відхилень всіх одиниць досліджуваної сукупності від середнього арифметичного. Як міра відхилень береться дисперсія – середній квадрат відхилень. Відхилення, викликані впливом факторної ознаки (фактора) порівнюються з величиною відхилень, викликаних випадковими обставинами.

Якщо відхилення, викликані факторною ознакою, більш істотні, ніж випадкові відхилення, то вважається, що фактор впливає на результуючу ознаку. Аналіз значень дисперсійної оцінки дозволяє відділити із загального переліку широковживаних у тексті слів слова, що розташовані рівномірно. Якщо деяке слово A в тексті, що складається з N слів, позначене як A_k^n, де індекс k – номер появи даного слова в тесті, а n – позиція даного слова в тексті, то інтервалом між послідовними появами слова при таких позначеннях буде величина [18]:

$$\Delta A_k^m = A_{k+1}^m - A_k^n = m - n\ , \tag{1.3}$$

де на m-й і n-й позиціях у тесті знаходиться слово A, яке зустрілось k+1-ий і k-й рази. Тоді дисперсійна оцінка розраховується наступним чином [18]:

$$DE = \frac{\sqrt{(\Delta A^2)-(\Delta A)^2}}{(\Delta A)} \tag{1.4}$$

де (ΔA) – середнє значення послідовності $\Delta A_1, \Delta A_2, \Delta A_k$; (ΔA^2) – послідовності A_1^2, A_2^2, A_k^2; K – кількість появи слова A в тексті.

Таким чином, задача пошуку ключових термінів у ІНМ може бути розв’язана з використанням одного з розглянутих методів пошуку ключових слів.

1.5 Засоби подання структури цифрового інформаційного навчального матеріалу

Структура ІНМ як електронних документів регламентується мовами розмітки документів (наприклад, WordprocessingML для XML чи Open Document Format) й реалізується через систему заголовків. По відношенню до онтології ІНМ навчальної дисципліни, заголовки співставленні її відповідним рівням, як це показано у таблиці 1.2.

Існує багато текстових процесорів які дозволяють будувати подібну структуру цифрового документу, як безкоштовних так і платних. Найпопулярніші з яких Microsoft Word, LibreOffice Writer, OpenOffice Writer та WPS Office Writer (рисунок 1.3).

Таблиця 1.2 Відповідність елементів структури цифрових документів рівням структури ІНМ

Порядок в ієрархії	**Рівень онтології ІНМ**	**Елемент структури цифрового документу**
1	Навчальна дисциплін	Заголовок 1
2	Модуль	Заголовок 2
3	Тема	Заголовок 3
4	Підтема	Заголовок 4

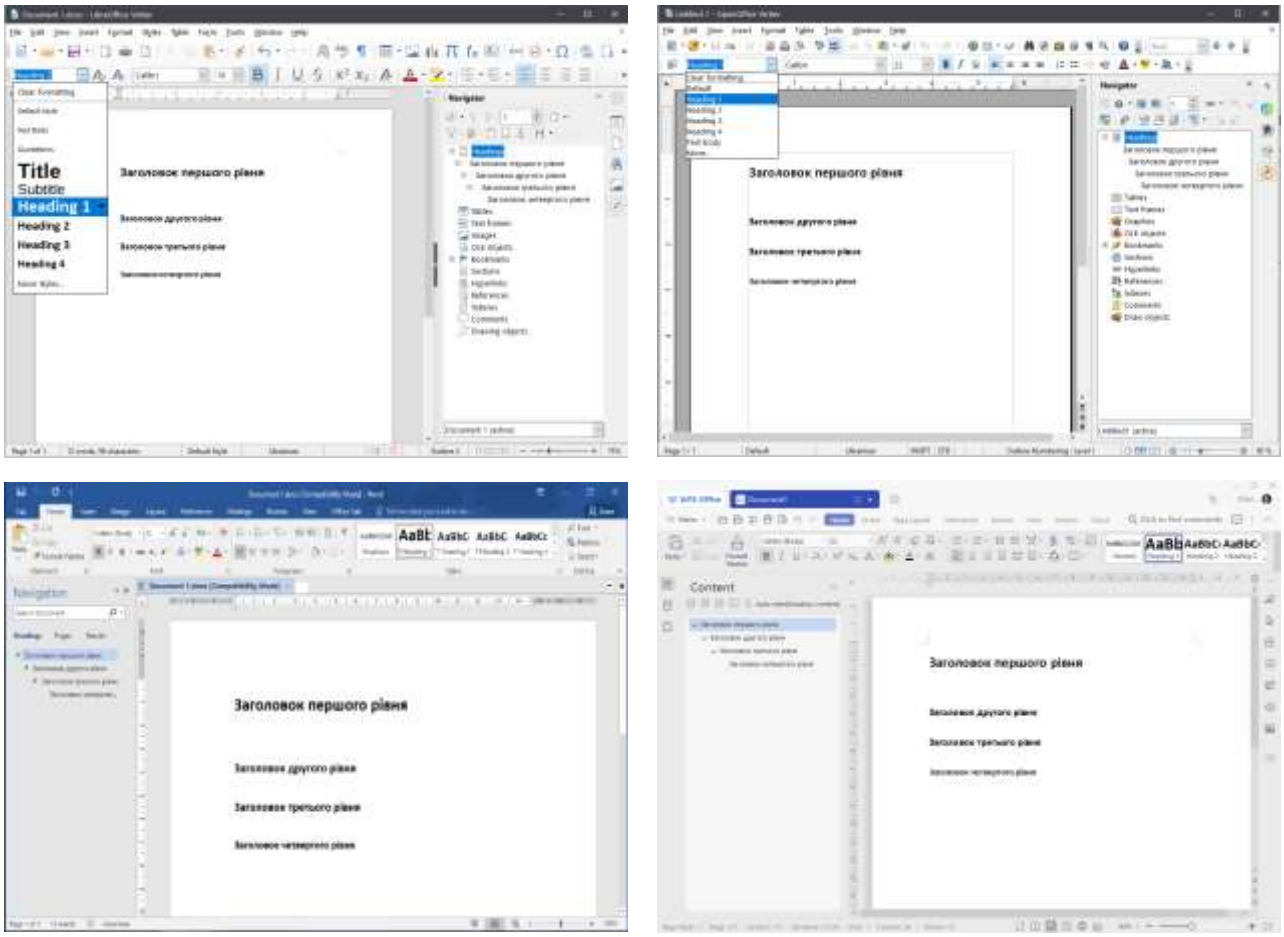


Рисунок 1.3 – Вигляд елементів структури документу в популярних текстових процесорах

Кожен з них має свої переваги та недоліки але їх використання дозволяє надати документу професійного вигляду. Звичайні текстові файли таких форматів як txt чи rtf не дають можливості будувати структуру документу. Проте

файли формату rtf дозволяють в певній мірі імітувати стилі заголовків з допомогою додання окремих характеристик форматування, таких як розмір кегля, тип шрифту тощо [19] (рисунок 1.4).

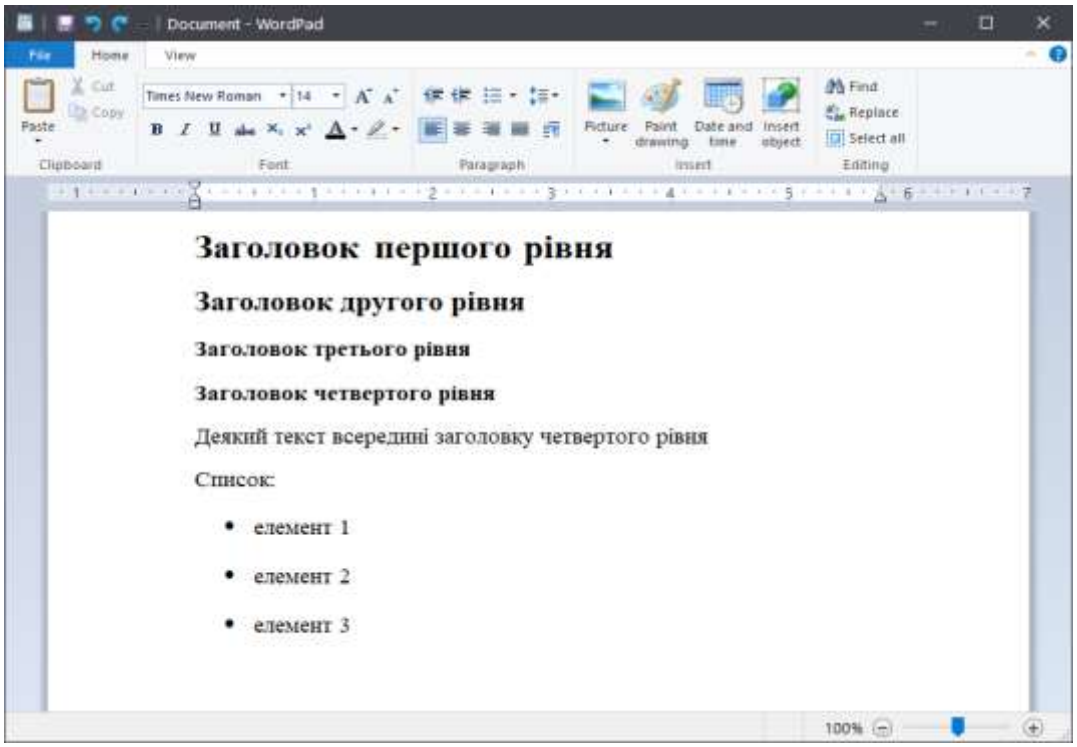


Рисунок 1.4 – Форматування тесту в rtf документі

Проблема такого форматування полягає в тому, що потрібно пам’ятати весь набір характеристик який застосовувався для конкретного рівню заголовку, абзацу чи таблиці для підтримки єдиного стилю форматування для всього документу. Натомість вбудовані стилі текстових процесорів так як містять відразу весь набір необхідних стилів.

Вбудовані стилі – це поєднання характеристик форматування, які можна застосувати до тексту, щоб швидко змінити його. Наприклад, застосування стилю «Заголовок 1» може зробити текст жирним шрифтом, Arial і 16 кеглем, а застосування стилю «Заголовок 2» робить текст жирним, похилим, Arial і 14 кеглем [20]. Таким чином для того щоб відформатувати всі заголовки другого рівня в документі все, що потрібно зробити це обрати відповідний вбудований стиль на панелі стилів, або створити власний. Якщо вимоги до форматування

змінилися, буде достатньо відредагувати необхідний стиль форматування і всі відповідні елементи автоматично зміняться відповідно до нових вимог (рисунок 1.5). Натомість якщо не використовуючи вбудовані стилі для таких змін потрібно буде замінити всі заголовки, таблиці чи абзаци по одному у всьому документі вручну, що, очевидно, займе значно більше часу та зусиль.

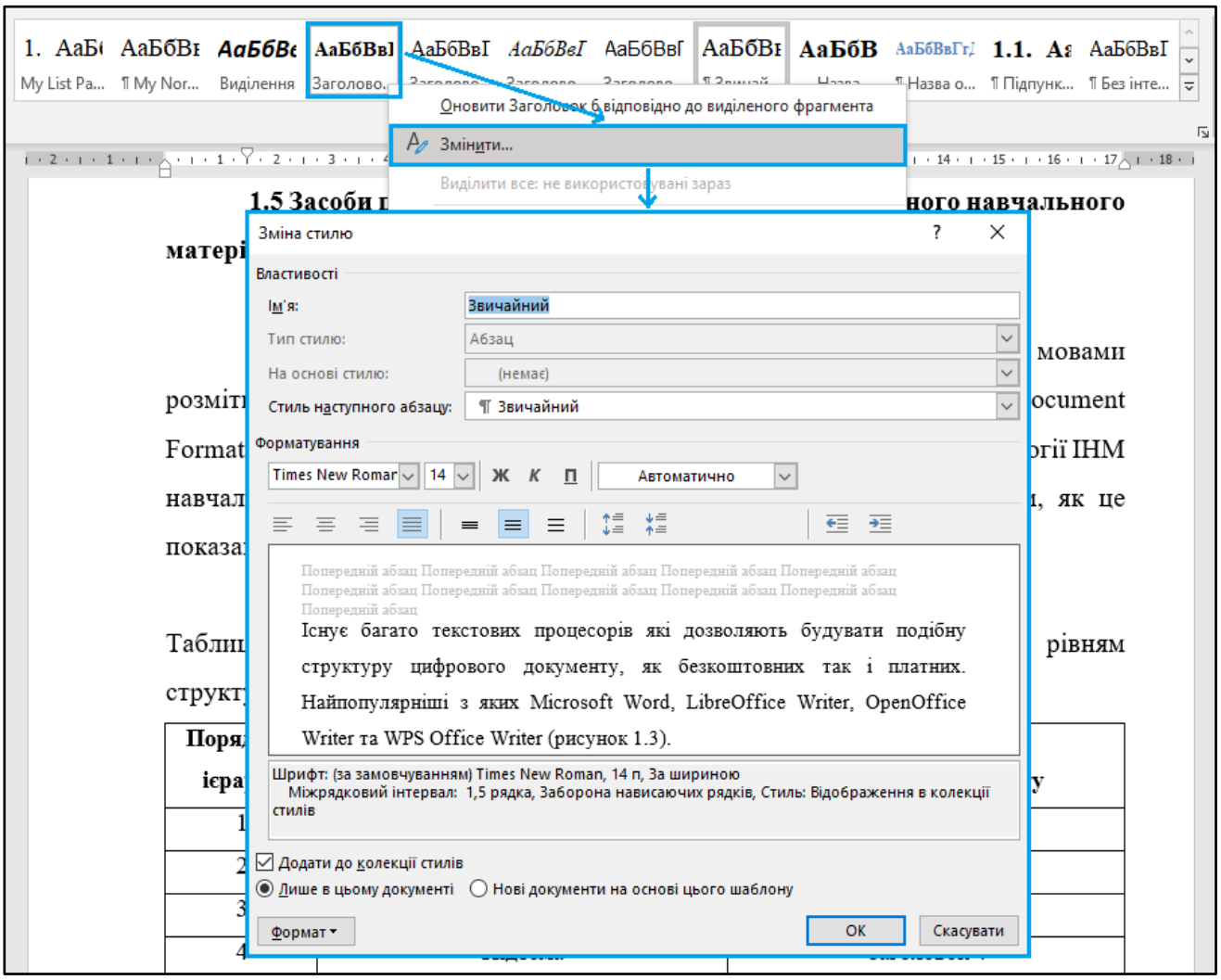


Рисунок 1.5 – Форматування наявних стилів форматування

Крім цього використання стилів форматування має ще ряд переваг, серед яких автоматична генерація змісту [21] та навігація по документу [22] (рисунок 1.6).

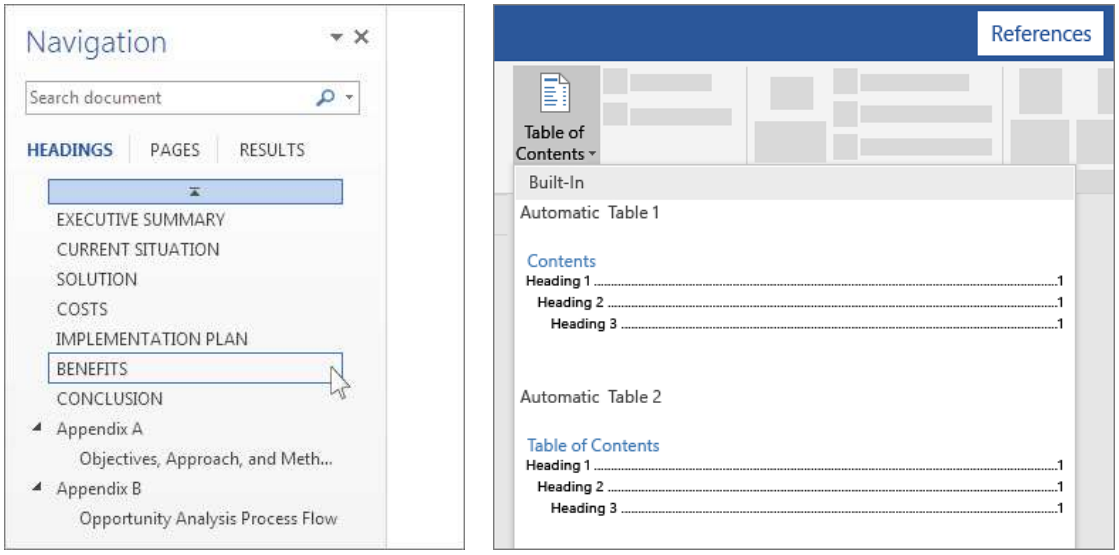


Рисунок 1.6 – Навігація по документу та формування змісту [21, 22]

Для програмного зчитування електронних документів існують спеціальні програмні бібліотеки, а саме: Microsoft.Office.Interap.Word [23], DocumentFormat.OpenXml [24], Spire.Doc [25], DocX [35]. DocX та Spire.Doc є комерційними продуктами, проте мають безкоштовний функціонал. DocX в більшій мірі розрахована на створення документів. Проведені дослідження [29] виявили переваги застосування бібліотеки Spire.Doc.dll для зчитування електронних документів.

Бібліотека Spire.Doc.dll надає можливість зчитування параметрів форматування текстових фрагментів [27], з допомогою чого можна реалізувати зчитування рубрикацій електронного документу.

Відповідно до об’єктної моделі документу [28], формат .docx структура документу складається з об’єктів документу, секцій, абзаців та текстових фрагментів (Text Range). Фрагменти Text Range є найменшими елементами структури що міститься в межах одного абзацу та використовує єдину множину параметрів форматування. Якщо одне слово розділене на декілька частин з допомогою параметрів форматування, воно буде знаходитися у різних фрагментах Text Range. Незважаючи на це, воно залишається одним словом, тому для пошуку ключових слів та термінів необхідно об’єднати всі фрагменти, що відносяться до одного слова.

Отож використання текстових процесорів для форматування та структурування електронних документів значно полегшує роботу та дозволяє дотримуватися єдиного стилю в всіх документах та уникнути стильових суперечностей. Одержання доступу до вмісту і форматуванню цифрових документів з ІНМ за допомогою спеціалізованих програмних бібліотек дозволяє автоматизовано визначати структуру ІНМ та відкриває можливості для пошуку ключових слів та термінів у ІНМ.

1.6 Постановка задачі

Метою дипломної роботи магістра є розробка інформаційної технології автоматизованого формування семантичної структури інформаційного навчального матеріалу.

Для досягнення мети необхідно розв’язати наступні *задачі*:

1. Удосконалити інформаційну модель семантичної структури інформаційного навчального матеріалу, яка дозволяє зберігати всі елементи та атрибути інформаційного навчального матеріалу, необхідні для побудови його семантичної структури.
2. Удосконалити метод формування рубрикації інформаційного навчального матеріалу з метою одержання ієрархічної структури цифрового документу інформаційного навчального матеріалу та визначення відповідних елементам структури фрагментів контенту.
3. Удосконалити метод пошуку ключових термінів у рубриках інформаційного навчального матеріалу, що дозволяє одержувати сортовані за рівнем семантичної значущості множини ключових слів та словосполучень.
4. Розробити інформаційну технологію автоматизованого формування семантичної структури інформаційного навчального матеріалу, яка дозволяє за вхідними даними в вигляді цифрового файлу електронного документу одержувати семантичну структуру інформаційного навчального матеріалу.

5. Дослідити практичну ефективності розробленої інформаційної технології шляхом створення відповідної експериментальної інформаційної системи та її тестування.

Розроблена для дослідження практичної ефективності створеної інформаційної технології експериментальна *інформаційна система* повинна виконувати наступні функції:

– зчитування та парсинг електронного ІНМ у форматі .docx;

– формування ієрархічної структури рубрикації інформаційного навчального матеріалу та обмеження відповідних рубрикам фрагментів контенту;

– пошук ключових термінів у рубриках інформаційного навчального матеріалу з використанням дисперсійного оцінювання слів і словосполучень та визначення оцінки їх семантичної важливості в рубриках;

– побудова семантичної структури інформаційного навчального матеріалу;

– відображення отриманих результатів користувачеві.

Висновки до розділу

В розділі було розглянуто особливості використання ІНМ. Було вияснено, що для включення у навчальний курс ІНМ вимагає наявності визначеної структури та рубрикації, наявність яких покращує сприйняття матеріалу та дозволяє ефективно використовувати його складові у навчанні. Крім того дослідження структури ІНМ встановлено, що при формуванні семантичної структури обов’язково слід враховувати наявну структуру інформаційного навчального матеріалу, подану в вигляді системи рубрикації. При цьому така структура має бути доповнена елементами, які подають семантичний вміст рубрик.

Такими елементами можуть виступати множини ключових термінів. При цьому окремі множини ключових термінів є розширенням елементів структури ІНМ для розгорнутого подання семантичної структури ІНМ. Задача пошуку ключових термінів у ІНМ може бути розв’язана з використанням одного з розглянутих методів пошуку ключових слів.

Дослідження засобів подання структури ІНМ показало ефективність використання текстових процесорів для форматування та структурування електронних документів. Спеціалізовані програмні комплекси дозволяють автоматизовано визначати структуру ІНМ, розроблених з допомогою текстових процесорів та відкриває можливості для пошуку ключових слів у них.

За результатом проведеного аналізу визначено доцільною розробку нової інформаційної технології, яка із використанням методу дисперсійної оцінки дозволить ефективно й автоматизовано формувати семантичну структуру інформаційного навчального матеріалу. Тому метою дипломної роботи магістра є розробка інформаційної технології автоматизованого формування семантичної структури інформаційного навчального матеріалу. Відповідно до поставленої мети, було визначено задачі дослідження, та сформовані функції експериментальної інформаційної системи для дослідження практичної ефективності створеної інформаційної технології.

Розділ 2

Інформаційна технологія автоматизованого формування семантичної структури інформаційного навчального матеріалу та її компоненти

2.1 Інформаційна модель семантичної структури інформаційного навчального матеріалу

Інформаційна модель семантичної структури ІНМ є цілісним поданням семантичної структури інформаційного навчального матеріалу, що може бути застосоване для реалізації інформаційної технології автоматизованого формування семантичної структури інформаційного навчального матеріалу і вирішенні ряду інших задач.

2.1.1 Формальне подання семантичної структури інформаційного навчального матеріалу

Загальноприйнятим є підхід до застосування ІНМ у вигляді слабо структурованих цифрових документів (наприклад, форматів .doc, .html, .pdf, .docx тощо) як інструменту навчання. В роботі пропонується формальне подання семантичної структури таких ІНМ (рисунок 2.1), до складу яких входить ієрархічна структура заголовків документу (рубрикація, наприклад: Дисципліна / Розділ / Тема) та співставленні заголовкам множини ключових термінів. Ключові терміни в таких множинах ранжовані за рівнем семантичної важливості в межах відповідних заголовкам фрагментів контенту документа.

Текстовий контент відповідного заголовку фрагменту ІНМ може бути поданий впорядкованою множиною слів. Якщо на її основі сформувати індексовану невпорядковану множину унікальних слів, то подання текстового контенту як впорядкованої множини слів можна замінити його поданням як впорядкованої множини індексів унікальних слів.

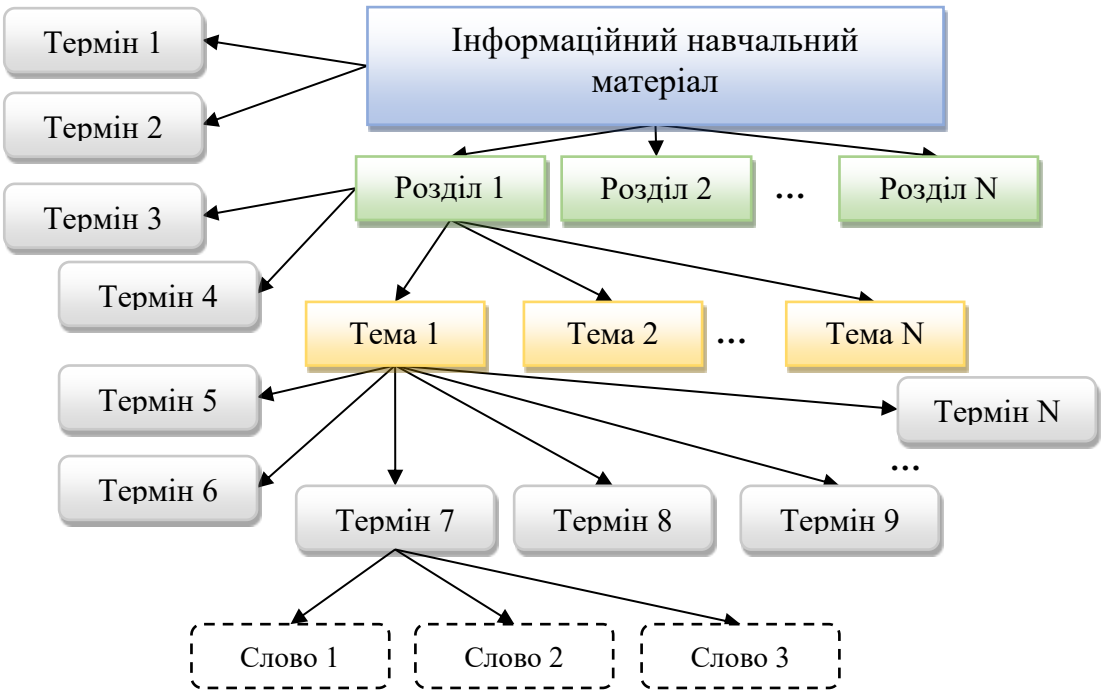


Рисунок 2.1 – Приклад формального подання семантичної структури ІНМ

Таким чином, при поданні семантичної структури ІНМ у вигляді мережі, можна визначити наступну множину сутностей ІНМ, що є сукупностями відповідних атрибутів (рисунок 2.2):

– підмножина заголовків (атрибути: унікальний ідентифікатор елементу, назва заголовку, рівень заголовку в ієрархічній структурі);

– підмножина слів (атрибути: символьна назва слова, символьна назва слова в нормалізованому вигляді, частина мови, показник лематизації);

– підмножина термінів (атрибути: символьна назва терміну, кількість слів у терміні, показник лематизації);

– підмножина зв’язків між сутностями (атрибути: тип зв’язку, перша сутність співвідношення, друга сутність співвідношення, характеристика зв’язку).

До підмножини зв’язків, які відображають факт і атрибути взаємозв’язку між сутностями, належать наступні різновиди:

– зв’язки між заголовками та заголовками;

– зв’язки між заголовками та ключовими термінами;

– зв’язки між ключовими термінами та словами.

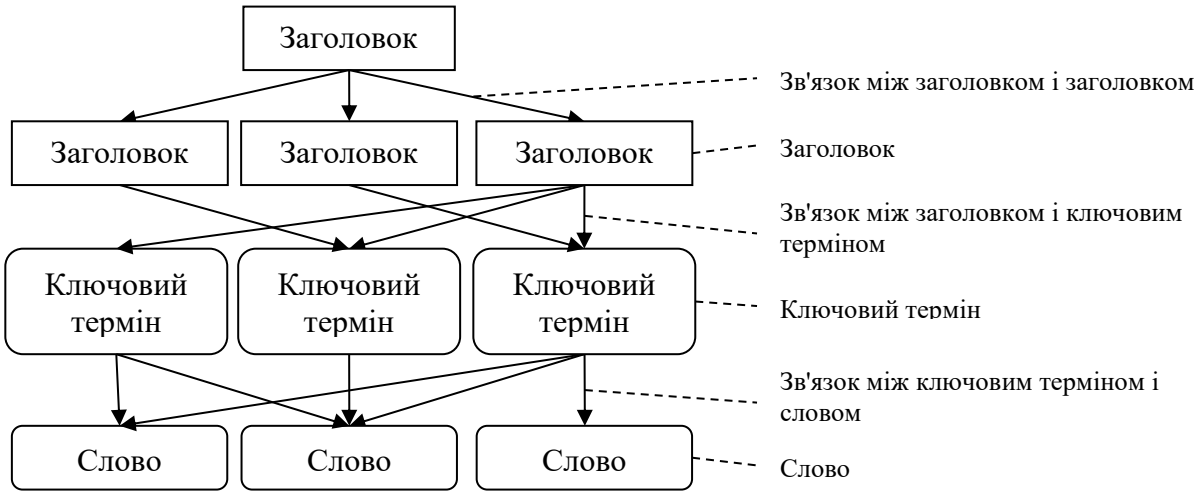


Рисунок 2.2 – Приклад використання елементів множини сутностей ІНМ для подання семантичної структури НМ

Формальне подання семантичної структури ІНМ у наведеному вигляді дозволяє використовувати її як модель для відображення семантичної структури ІНМ. Таке відображення відкриває шлях до вирішення ряду прикладних задач.

2.1.2 Модель семантичної структури інформаційного навчального матеріалу

Контент ІНМ зазвичай включає тексти, малюнки, таблиці та інші форми графічного вираження інформації. За характером інформації ІНМ подається у вигляді підручника, навчального посібника, конспекту лекцій тощо. Контент ІНМ містить слабо структуровану символьну (переважно текстову) інформацію, засвоєння якої забезпечує набуття відповідних знань та вмінь суб’єктом, що вивчає навчальний курс.

Семантична структура інформаційного навчального матеріалу (*ІЕМ)* має включати наступні множини:

– $M_{Heading}$ – множина заголовків;

– M_{Par} – множина абзаців;
– M_{Sen} – множина речень;
– M_{Term} – множина термінів;
– M_{Word} – множина слів;
– M_{Rel} – множина зв’язків.

Відповідно, кожен елемент множини елементів *IEM* є елементом підмножини $M_{Heading}$, M_{Par}, M_{Sen}, M_{Term}, M_{Word} або M_{Rel}:

$$IEM=\{x|x \in M_{Heading} \vee x \in M_{Par} \vee x \in M_{Sen} \vee x \in M_{Term} \vee x \in M_{Word} \vee x \in M_{Rel}\} \quad (2.1)$$

Наведена сукупність множин є достатньою для формування семантичної структури ІНМ.

Елементи множини слів M_{Word} виступають атомарними складовими наведеної структури. За допомогою її елементів визначаються елементи множини ключових термінів M_{Term}. Потреба у включенні множини слів M_{Word} до моделі семантичної структури ІНМ обумовлена тим, що одне слово може входити до складу кількох ключових термінів, тобто одному елементу множини слів M_{Word} може відповідати кілька елементів множини ключових термінів M_{Term}. Водночас у випадках, коли ключовим терміном є словосполучення, одному елементу множини ключових термінів M_{Term} може відповідати кілька елементів множини слів M_{Word}.

Множину зв’язків M_{Rel} можна розділити на підмножини, відповідно до множин, елементи яких зв’язують між собою її елементи:

– множина зв’язків між заголовками та заголовками $M_{H\text{-}H}$;
– множина зв’язків між заголовками і абзацами $M_{H\text{-}P}$;
– множина зв’язків між абзацами і реченнями $M_{P\text{-}S}$;
– множина зв’язків між реченнями і термінами $M_{S\text{-}T}$;
– множина зв’язків між термінами та словами $M_{T\text{-}W}$.

Таким чином, формально ІНМ представляється у вигляді деревовидної структури, в якій всі елементи являються семантичними вузлами, де ключові слова і терміни є найнижчим рівнем структури, а всі інші рівні це заголовки

рубрикацій. Визначення зв’язків між вузлами забезпечує формальне вираження взаємозв’язку між ними.

Відповідно до (2.1), модель семантичної структури навчального курсу навчальної дисципліни подається у наступному вигляді:

$$\{M_{Heading} \cup M_{Par} \cup M_{Sen} \cup M_{Term} \cup M_{Word} \cup M_{Rel}\} \subset IEM \subset EC, \quad (2.2)$$

де EC – курс навчальної дисципліни, IEM – інформаційний навчальний матеріал, $M_{Heading}$ – множина заголовків, M_{Par} – множина абзаців , M_{Sen} – множина речень, M_{Term} – множина термінів, M_{Word} – множина слів, M_{Rel} – множина зв’язків.

Далі розглянуто окремі складові розглянутих множин: множину заголовків $M_{Heading}$, множину абзаців M_{Par}, множину речень M_{Sen}, множину термінів M_{Term}, множину слів M_{Word} та множину зв’язків M_{Rel}.

2.1.3 Множина заголовків

В якості заголовку в роботі розглядається назва семантично цілого фрагменту контенту ІНМ (назва розділу, теми, параграфу тощо) визначеного автором. Кожен заголовок має власний рівень вкладеності.

Всі елементи множини заголовків $M_{Heading}$ можна подати наступним чином:

$$M_{Heading} = (HID, HName, Level, PHID), \quad (2.3)$$

де HID – ідентифікатор, $HName$ – назва заголовку, $Level$ – рівень заголовку в ієрархічній структурі, $PHID$ – ідентифікатор батьківського заголовку.

Для кожної з цих властивостей є відповідні множини:

– M_{HID} – множина унікальних цілих чисел, ідентифікаторів;
– M_{HName} – назви заголовків;
– M_{Level} – рівень вкладеності.

Таким чином з допомогою цих властивостей можна формувати множини $M_{Heading}$ наступного вигляду:

$$M_{Heading} = \{(HID, HName, Level, PHID) \mid$$
$$(HID, PHID) \in M_{HID} \in Z, HName \in M_{HName}, Level \in M_{Level} \}. \quad (2.4)$$

Множина M_{HName} включає всі назви заголовків.

Множина M_{Level} включає цілі числа, що означають рівень вкладеності заголовків. Таким чином M_{Level} лежить в межах від 0 до n, де 0 це корінний елемент (що відповідає всьому документу) та n дорівнює кількості рівнів вкладеності.

Множина M_{HID} включає унікальні цілі числа, що використовуються для однозначної ідентифікації елементів $M_{Heading}$.

2.1.4 Множина абзаців

Множина абзаців M_{Par} включає фрагменти контенту ІНМ розділені символом нового рядка. Символ нового рядка – спеціальний символ або послідовність символів, що позначають кінець рядка тексту. За стандартом таким символом може бути один із далі перелічених символів (або їх комбінація) [29]:

– LF (U+000A): англ. line feed – зміна рядка;

– CR (U+000D): англ. carriage return – повернення каретки;

– NEL (U+0085): англ. next line – перехід на наступний рядок;

– LS (U+2028): англ. line separator – роздільник рядків;

– PS (U+2029): англ. paragraph separator – роздільник абзаців.

Всі елементи множини абзаців M_{Par} можна подати наступним чином:

$$M_{Par} = (PID, Number, PHID, PType), \quad (2.5)$$

де *PID* – ідентифікатор, *Number* – номер по порядку в межах батьківського заголовку, *PHID* – ідентифікатор батьківського заголовку, *PType* – тип абзацу.

Для кожної з цих властивостей є відповідні множини:

– M_{PID} – множина унікальних цілих чисел, ідентифікаторів;

– $M_{PContent}$ – назви заголовків;

– M_{PHID} – множина унікальних цілих чисел, ідентифікаторів батьківських заголовків;

– M_{PType} – множина типів абзаців.

Таким чином з допомогою цих властивостей можна формувати множини M_{Par} наступного вигляду:

$$M_{Par} = \{(PID, Number, PHID, PType) \mid$$
$$PID \in M_{PID}, Number \in M_{Number}, PHID \in M_{PHID}, PType \in M_{PType}\} \quad (2.6)$$

Множини M_{PID} та M_{PHID} включають унікальні цілі числа, що використовуються для однозначної ідентифікації елементів M_{Par} та $M_{Heading}$ відповідно.

Множина M_{Number} включає цілі числа, що означають номер по порядку в межах батьківського заголовку.

Множина M_{PType} включає типи абзаців: заголовок, елемент списку, підпис до візуального елементу (рисунку, таблиці, тощо).

2.1.5 Множина речень

Множина речень M_{Sen} включає граматично та інтонаційно цілі фрагменти контенту.

Всі елементи множини речень M_{Sen} можна подати наступним чином:

$$M_{Sen} = (SID, SText, Number, PPID, SType), \quad (2.7)$$

де *SID* – ідентифікатор, *Text* – текст речення, *SType* – тип речення, *PPID* – ідентифікатор батьківського абзацу, *Number* – номер по порядку в межах батьківського заголовку в ієрархічній структурі.

Для кожної з цих властивостей є відповідні множини:

– M_{SID} – множина унікальних цілих чисел, ідентифікаторів;

– M_{SText} – множина текстів речень;

– M_{PSID} – множина унікальних цілих чисел, ідентифікаторів батьківських заголовків;

– M_{SType} – множина типів речень.

Таким чином з допомогою цих властивостей можна формувати множини M_{Sen} наступного вигляду:

$$M_{Sen} = \{(SID, SText, Number, PPID, SType) \mid$$

$$SID \in M_{SID}, SText \in M_{SText}, Number \in M_{Number}, PPID \in M_{PPID}, SType \in M_{SType}\}. \quad (2.8)$$

Множина M_{SText} включає тестові фрагменти відповідного речення.

Множини M_{PID} та M_{PPID} включають унікальні цілі числа, що використовуються для однозначної ідентифікації елементів M_{Sen} та M_{Par}.

Множина M_{Number} включає цілі числа, що означають номер по порядку в межах батьківського заголовку.

Множина M_{SType} включає типи речень які класифікуються за метою висловлювання: розповідний, питальний та спонукальний типи відповідно граматичних правил (в даній роботі української мови) [30].

2.1.6 Множина термінів

Множини термінів M_{Term} включає терміни, які розкриваються в рамках визначених структурою документу фрагментів контенту ІНМ.

Всі елементи множини термінів M_{Term} можна подати наступним чином:

$$M_{Term} = (TID, TText), \quad (2.9)$$

де *TID* ідентифікатор, *TText* – текст терміну.

Для кожної з цих властивостей є відповідні множини:

– M_{TID} – множина унікальних цілих чисел, ідентифікаторів;
– M_{TText} – множина текстів термінів.

Таким чином з допомогою цих властивостей можна формувати M_{Term} наступного вигляду:

$$M_{Term} = \{(TID, TText) \mid TID \in M_{TID}, Text \in M_{TText}\} \quad (2.10)$$

Множина M_{TText} включає в себе тестові назви певного терміну в ІНМ.

Множина M_{TID} включають унікальні цілі числа, що використовуються для однозначної ідентифікації елементів M_{Term}.

2.1.7 Множина слів

Множина слів M_{Word} включає слова, що присутні в рамках визначених структурою документу фрагментів контенту ІНМ хоча б один раз. Кожне слово не обов'язково є терміном або входить до складу одного з термінів ІНМ.

Властивості кожного елемента в множині є незалежними від конкретного тексту, в той час як множина M_{Word} формується шляхом включення до неї всіх елементів, що відповідають присутнім в тексті оригінальним (посимвольно унікальним) словам. Всі елементи множини слів M_{Word} можна подати наступним чином:

$$M_{Word} = (WID, WText, WTextInf, WType), \quad (2.11)$$

де *WID* – ідентифікатор, *WText* –назва слова, *WTextInf* –назва слова в нормалізованому вигляді; *WType* – частина мови, до якої відноситься слово.

Для кожної з цих властивостей є відповідні множини

- M_{WText} – множина назв термінів;
- $M_{WTextInf}$ – множина інфінітивних назв термінів;
- M_{WType} – множина частин мови.

Таким чином з допомогою цих властивостей можна формувати множини M_{Word} наступного вигляду:

$$M_{Word} = \{ (WText, WTextInf, WType) \mid$$
$$WText \in M_{WText}, WTextInf \in M_{WTextInf}, WType \in M_{WType}\} \quad (2.12)$$

Множина назв слів M_{WText} включає символьні (текстові) назви слів, що використовуються в інформаційному навчальному матеріалі. Множині належать всі оригінальні словоформи, чи леми, які присутні в навчальному матеріалі.

Множини інфінітивних назв слів $M_{WTextInf}$ включає відповідні символьні (текстові) назви слів у нормальній (інфінітивній) формі, що використовуються в інформаційному навчальному матеріалі. Множина містить лише по одному оригінальному елементу для кількох можливих словоформ.

Шляхом посимвольного порівняння атрибутів *WText* та *WTextInf* елементу множини M_{Word}, можна визначити чи перебуває він у нормальній формі.

Множину частин мови M_{WType} складають назви самостійних та службових частин мови, до яких належать іменник, прикметник, дієслово, займенник, прислівник, числівник, сполучник, прийменник або частка [31]. Тип слова встановлюється відповідно до граматичних правил (в даній роботі – української мови), які дозволяють визначати, до якої частини мови належить кожен елемент множини слів M_{Word}.

Як і в терміна, нормальна (інфінітивна) форма слова є його початковою формою, наприклад, для іменника це називний відмінок однини, для дієслова це теперішній час доконаного виду в однині тощо. Тому кожній лемі обов'язково відповідає певна визначена інфінітивна форма.

2.1.8 Множина зв'язків між елементами семантичної структури

Множина зв'язків M_{Rel} включає елементи семантичної структури ІНМ, що визначають наявність і характер зв'язку між іншими елементами семантичної структури, зокрема елементами множин $M_{Heading}$, M_{Par}, M_{Sen} та M_{Term}.

Елементи множини M_{Rel} визначаються як бінарні та однонаправлені. Бінарність визначає об'єднання зв'язком тільки двох елементів семантичної структури ІНМ. Однонаправленість вказує, що зв'язок спрямований від першого атрибути зв'язку до другого, але не навпаки. Окрім визначення зв'язку між двома атрибутами та його направленості, елементи множини M_{Rel} можуть містити додаткові атрибути, що визначають характер цього зв'язку.

Множини зв'язків M_{Rel} включає елементи, які визначають бінарні зв'язки між двома елементами з множин $M_{Heading}$, M_{Par}, M_{Sen} та M_{Term}. Відповідно, кожен елемент множини можна подати наступним чином:

$$M_{Rel} = (RelationTypeID, FirstEntity, SecondEntity, Property), \quad (2.13)$$

де *RelationTypeID* – множина унікальних цілих чисел, ідентифікаторів типу зв'язку; *FirstEntity, SecondEntity* – перша та друга сутності зв'язку; *Property* – властивість зв'язку.

У залежності від приналежності атрибутів *FirstEntity* та *SecondEntity* окремим множинам з переліку $M_{Heading}$, M_{Par}, M_{Sen} та M_{Term}, атрибут *RelationTypeID* приймає значення, вказані у таблиці 2.1.

Таблиця 2.1 – Визначення властивостей *RelationTypeID*

Назва	***RelationTypeID***	**FirstEntity**	**SecondEntity**	**Property**
M_{IEM-H}	1	M_{IEM}	$M_{Heading}$	-
M_{H-H}	2	$M_{Heading}$	$M_{Heading}$	-
M_{H-P}	3	$M_{Heading}$	M_{Par}	-
M_{P-S}	4	M_{Par}	M_{Sen}	-
M_{S-T}	5	M_{Sen}	M_{Term}	Назва терміну у тій формі, у якій він зустрівся у реченні
M_{T-W}	6	M_{Term}	M_{Word}	Порядковий номер слова у терміні
M_{H-T}	7	$M_{Heading}$	M_{Term}	Важливості ключового терміну

Відповідно до типів елементів, які сполучаються за допомогою елементів множини M_{Rel}, її структура може бути подана у вигляді:

$$M_{Rel} = M_{IEM-H} \cup M_{H-H} \cup M_{H-P} \cup M_{P-S} \cup M_{S-T} \cup M_{T-W} \cup M_{H-T}, \quad (2.14)$$

де M_{IEM-H} – множина зв’язків між ІНМ та заголовками, M_{H-H} – множина зв’язків між заголовками й заголовками, M_{H-P} – множина зв’язків між заголовками та абзацами, M_{P-S} – множина зв’язків між абзацами та реченнями, M_{S-T} – множина зв’язків між реченнями та термінами, M_{T-W} – множина зв’язків між ключовими термінами та словами, M_{H-T} – множина зв’язків між заголовками й ключовими термінами.

Відповідно до (2.14), кожен елемент множини M_{Rel} є елементом підмножини M_{IEM-H}, M_{H-H}, M_{H-P}, M_{P-S}, M_{S-T}, M_{T-W} або M_{H-T}:

$$M_{Rel} = \{x | x \in M_{IEM-H} \vee x \in M_{H-H} \vee x \in M_{H-P} \vee x \in M_{P-S} \vee x \in M_{S-T} \vee x \in M_{T-W} \vee x \in M_{H-T}\} \quad (2.15)$$

Таким чином, модель семантичної структури навчального курсу навчальної дисципліни містить наступні складові: $M_{Heading}$ – множина заголовків, M_{Par} – множина абзаців , M_{Sen} – множина речень, M_{Term} – множина термінів, M_{Word}

– множина слів, M_{Rel} – множина зв’язків. При цьому, визначено наступні підмножини множини зв’язків M_{Rel}: $M_{IEM\text{-}H}$ – множина зв’язків між ІНМ та заголовками, $M_{H\text{-}H}$ – множина зв’язків між заголовками й заголовками, $M_{H\text{-}P}$ – множина зв’язків між заголовками та абзацами, $M_{P\text{-}S}$ – множина зв’язків між абзацами та реченнями, $M_{S\text{-}T}$ – множина зв’язків між реченнями та термінами, $M_{T\text{-}W}$ – множина зв’язків між ключовими термінами та словами, $M_{H\text{-}T}$ – множина зв’язків між заголовками й ключовими термінами.

Для наповнення елементів моделі, необхідно застосувати:

– метод формування рубрикації інформаційного навчального матеріалу для визначення елементів моделі: множину заголовків, множину абзаців, множину речень, множину зв’язків між заголовками, множину зв’язків між заголовками та абзацами, множину зв’язків між абзацами та реченнями, множину зв’язків між реченнями та термінами, множину зв’язків між термінами та словами, множину зв’язків між ключовими термінами та заголовками;

– метод пошуку ключових термінів у рубриках інформаційного навчального матеріалу для визначення елементів моделі: множини зв’язків між заголовками і ключовими термінами.

Використання наведених двох методів дозволяє здійснювати повне визначення елементів моделі, що відкриває можливості для її практичного застосування.

2.2 Метод формування рубрикації інформаційного навчального матеріалу

2.2.1 Схема методу

Метод формування рубрикації документу ІНМ зіставляє всі елементи рубрик з відповідним текстовим контентом ІНМ таким чином формуючи ієрархічну структуру документу. Загальну схему методу наведено на рисунку 2.3.

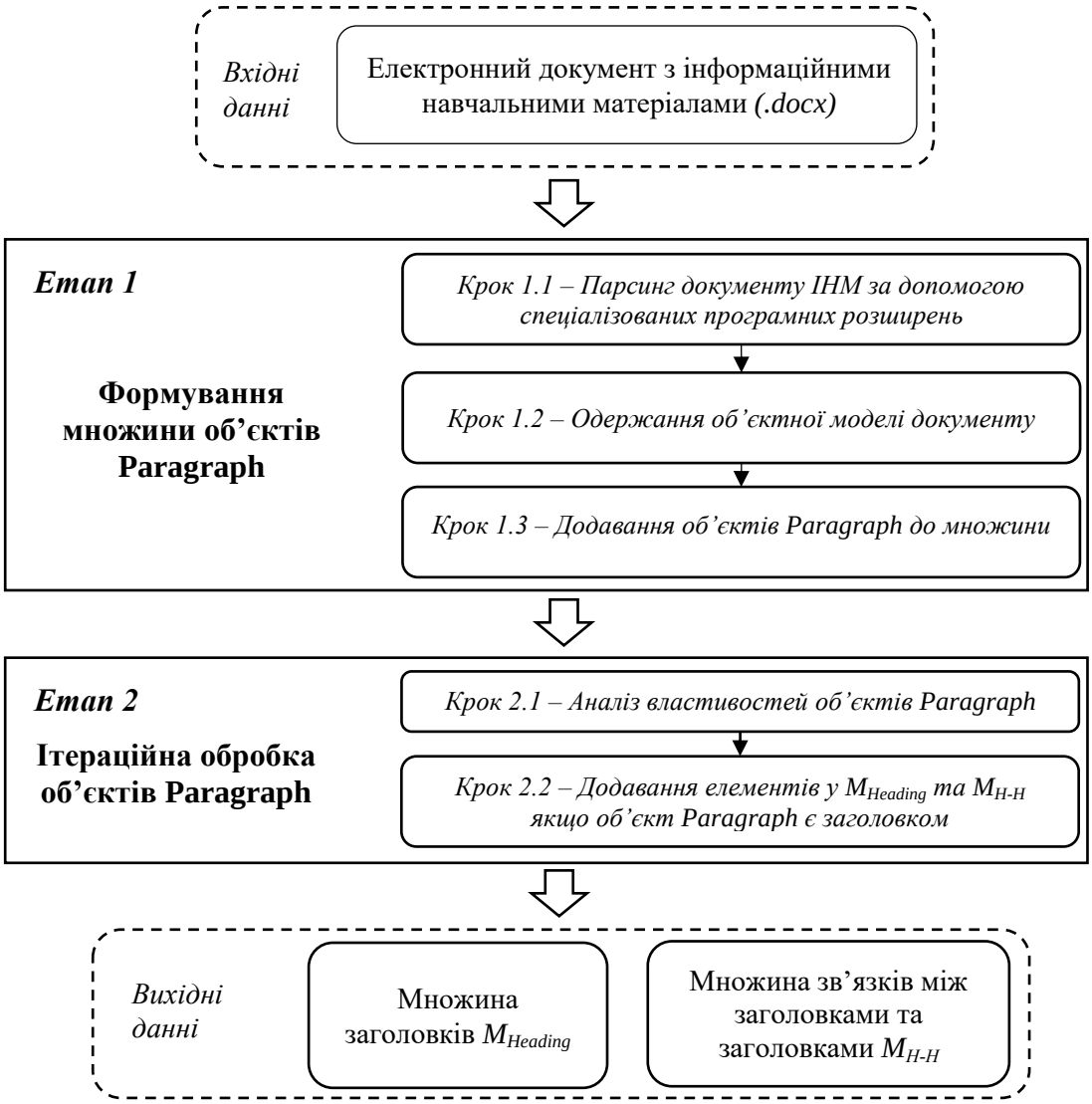


Рисунок 2.3 – Схема методу формування рубрикації ІНМ

Вхідними даними методу є електронний документ ІНМ з розширенням .docx, та параметри налаштування пошуку, а саме максимальна кількість слів з яких може складатися термін та коефіцієнти коригування оцінки за стилістичними ознаками.

У *Етапі 1* визначається структура документу, та формується зв'язок з відповідним текстовим контентом ІНМ. Для зчитування та парсингу електронного документу використовуються програмні засоби, описані в 1.5. Результатом першого етапу є об'єктна модель документу та множина об'єктів Paragraph.

У *Emani 2* об’єкти Paragraph ітераційно опрацьовуються для одержання ієрархічної структури рубрик документу. Для цього виконуються наступні дії.

Для всіх абзаців виконується аналіз стилів та визначається наявність стилів заголовку. Якщо в абзацу є такий стиль, весь текст абзацу вважається назвою заголовку *HName*. Порівнюючи попередній та поточний рівні заголовків *Level* формуються елементи множин зв’язків $M_{H\text{-}H}$. Для яких *FirstEntity* та *SecondEntity* отримують значення батьківського та дочірнього заголовків. Алгоритмічна схема обробки об’єктів Paragraph зображена на рисунку 2.4.

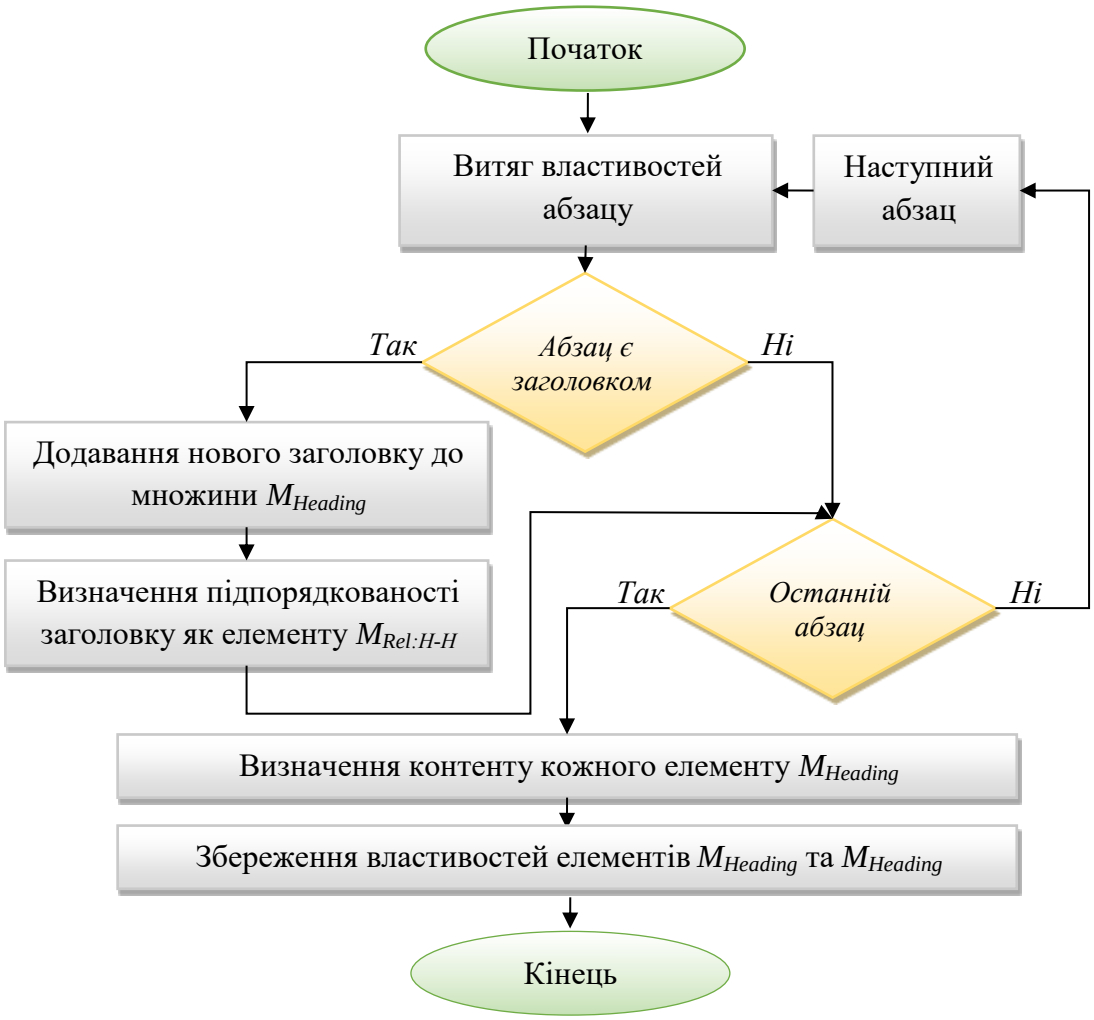


Рисунок 2.4 – Алгоритмічна схема обробки об’єктів Paragraph

Контентом заголовку вважається текст від абзацу поточного заголовку до наступного абзацу заголовку. Хоча контент дочірніх заголовків являється

одночасно і контентом батьківського заголовку проте не зберігається в батьківському заголовку для уникнення дублювання даних.

Вихідними даними методу є множина зв'язків між заголовками та заголовками $M_{H\text{-}H}$ та множина заголовків $M_{Heading}$.

2.2.2 Підхід до визначення елементів множини заголовків

Всі елементи множини заголовків $M_{Heading}$ подаються наступним чином $M_{Heading} = (HID, HName, Level, PHID)$, де властивостями є ідентифікатор *HID*, назва заголовку *Name* та рівень заголовку *Level* в ієрархічній структурі (див. 2.1.3). Визначення заголовків базується на зчитуванні параметрів форматування електронних документів визначених автором документу.

Під час форматування документу ІНМ автор визначає рівні заголовків з допомогою спеціальних стилів форматування, таким чином формуючи ієрархічну структуру документу. В даній роботі заголовком вважається текстовий фрагмент документу в межах ІНМ який знаходиться під абзацом заголовку та на одному рівні з ним до наступного абзацу заголовку. Наступний заголовок та його контент вважається дочірнім якщо його рівень нижчий рівня поточного. Тому ідентифікатор поточного заголовок визначається як батьківський і встановлюється у властивість *PHID* для такого заголовку.

Оскільки є ймовірність, що може існувати декілька заголовків з однаковою назвою та рівнем вкладеності в ієрархічній структурі є необхідність використовувати унікальний ідентифікатор для кожного заголовку.

Назвою заголовку вважається повний текст абзацу відповідного стилю.

Таким чином визначаються всі властивості елементів заголовків *HID*, *HName*, *Level*, *PHID*, що описують всі елементи множини $M_{Heading}$.

Визначені властивості заголовків, а саме *Level*, *PHID* дозволяють формувати елементи множини зв'язків. Для заголовків елементи цієї множини подаються у вигляді $M_{H\text{-}H} = (2, H_1, H_2, 0)$, де властивості H_1 та H_2 приймають

значення ідентифікаторів батьківського та дочірнього заголовків. Який з заголовків відіграє роль батьківського, а який дочірнього визначається з допомогою властивості заголовку *Level*, за принципом заголовок з меншим рівнем вкладеності – батьківський.

Таким чином одержані властивості елементів множини заголовків дозволяють визначити елементи множини зв'язків між заголовками, що в свою чергу дозволяє сформувати ієрархічну структуру заголовків та відкриває можливості до її заповнення іншими елементами інформаційної моделі семантичної структури ІНМ.

2.2.3 Визначення елементів множин інформаційної моделі семантичної структури

Всі елементи множини абзаців M_{Par} подаються наступним чином M_{Par} = *(PID, Number, PHID, PType)*, де властивостями є ідентифікатор *PID*, номер по порядку *Number*, ідентифікатор батьківського заголовку *PHID* та тип абзацу (див. 2.1.4).

Для несуперечливого доступу до абзаців встановлюється унікальний ідентифікатор *PID*. Визначення абзаців проводиться в контенті певного заголовку тому значення властивості *PHID* відповідає ідентифікатору заголовку *HID*. З допомогою визначених ідентифікатора формується зв'язок між абзацом та заголовком $M_{H\text{-}P}$ = (*3*, *HID*, *PID*, *0*).

Номер по порядку визначається в контексті певного заголовку, тобто абзац, що являється текстом заголовку має номер по прядку *Number = 0*, а всі інші абзаци в заголовку мають *Number* послідовно збільшене на одиницю та останній абзац має значення *Number*, що відповідає різниці кількості абзаців у заголовку та одиниці.

Типи абзацу *PType* визначаються відповідно до визначених автором параметрів форматування. Якщо для типів «Заголовок» та «Елемент списку»

зазвичай вказують відповідні стилі, то для зображень, таблиць чи інших візуальних елементів часто ігноруються можливість вказування спеціальних стилів для підпису, тому тип «Підпис до візуального елементу» не завжди можна визначити.

Всі елементи множини речень M_{Sen} подаються наступним чином M_{Sen} = *(SID, SText, Number, PPID, SType)*, де властивостями є ідентифікатор *SID*, текст речення *Text*, тип речення *SType*, ідентифікатор батьківського абзацу *PPID* та номер по порядку в межах батьківського заголовку *Number* (див. 2.1.5).

Для несуперечливого доступу до речень встановлюється унікальний ідентифікатор *SID*. Визначення речень проводиться в контенті певного абзацу тому значення властивості *PPID* відповідає ідентифікатору абзацу *PID*. З допомогою визначених ідентифікатора формується зв'язок між реченням та абзацом $M_{P\text{-}S}$ = (*4, PID, SID, 0*).

Текст речення *SText* визначається шляхом розбиття тексту який належить до певного абзацу по відповідним розділовим знакам. Оскільки крапка може використовуватися не тільки як показник завершення речення, а і як показник скорочення слова чи використовуватися як роздільник числах, додатково проводиться аналіз контенту на визначення таких виключень.

Номер по порядку визначається в контексті певного заголовку, тобто перше речення в заголовку має номер по прядку *Number = 0*, а всі інші речення в заголовку мають *Number* послідовно збільшене на одиницю та останнє речення має значення *Number*, що відповідає різниці кількості речень у заголовку та одиниці.

Під типами речень *SType* мається на увазі класифікація речень за метою висловлювання (розповідні, питальні й спонукальні). Типи речень встановлюються відповідно до символу закінчення речення: знак оклику – спонукальний тип, крапка – розповідний тип, знак запитання – питальний тип.

Всі елементи множини слів M_{Word} податься наступним чином M_{Word} = (*WID, WText, WTextInf, WType*), де властивостями є ідентифікатор *WID*, текст

слова *WText*, текст слова в нормальній формі *WTextInf* та тип слова *WType* (див. п. 2.1.7).

Для несуперечливого доступу до слів встановлюється унікальний ідентифікатор *WID*. З допомогою визначеного ідентифікатора та ідентифікатора терміну де зустрілося слово формується зв’язок між словом та терміном $M_{T\text{-}W}$ = (*6*, *TID*, *WID*, *0*).

Текст слова *WText* визначається шляхом розбиття тексту який належить до певного речення по пробілам та розділовим знакам враховуючи можливість використання символу дефісу та апострофа.

Під типами слів *WType* мається на увазі класифікація слів за частинами мови. Типи слів визначаються з допомогою бази даних корпусу українських слів.

Нормальна форма слова *WTextInf* визначається з допомогою бази даних корпусу українських слів.

Оскільки наявність відповідної бази даних не вимагається властивості *WType* та *WTextInf* можуть бути не заповненими.

Всі елементи множини термінів M_{Term} подаються наступним чином M_{Term} = *(TID, TText)*, де властивостями є ідентифікатор *TID* та текст терміну *TText* (див. п. 2.1.6).

Для несуперечливого доступу до термінів встановлюється унікальний ідентифікатор *TID*. З допомогою визначеного ідентифікатора та речення де зустрівся термін формується зв’язок між терміном та реченням $M_{S\text{-}T}$ = (*5*, *SID*, *TID*, *0*).

Текст терміну *SText* визначається шляхом співставлення сусідніх слів. Оскільки термін може складатися як з одного слова так і декількох формується множина всіх можливих комбінацій слів, які могли б виступати в ролі термінами.

Таким чином одержані властивості елементів множин дозволяють визначити всі елементи необхідних множин та зв’язки між ними, що в свою чергу дозволяє повністю наповнити інформаційну моделі семантичної структури

ІНМ. Сформована множина термінів є формальною та містить всі можливі комбінації слів які можуть формувати ключові терміни. В подальшому ця множина фільтрується та компактифікуються відповідно до методу пошуку ключових.

2.3 Метод пошуку ключових термінів у рубриках інформаційного навчального матеріалу

Метод пошуку ключових термінів у рубриках інформаційного навчального матеріалу проводиться в три етапи кожен з яких, в свою чергу, складається з трьох кроків.

Перший етап відповідає за пошук та оцінку термінів в елементах структури ІНМ. Вхідними даними для цього етапу є елетронний документ.

Першим кроком першого етапу є визначення термінів кандидатів. На цьому кроці створюється множина всіх можливих комбінацій найменших семантично цілісних вузлів які відеокремлені розділовими знаками чи стилями та не перевищують максимальну визначену кількість слів у ключовому терміні.

Другий крок першого етапу – визначення унікальних термінів та кількість їх входжень в рубрику ІНМ. На цьому кроці опрацьовується попередньо визначена множина термінів кандидатів. Після знаходження всіх дублікатів термінів визначаєтсья їхня кількість та позиція в рубриці – найважливіші параметри які необхідні для подальшої оцінки термінів.

Останнім, *третім кроком* першого етапу є визначення дисперсійної оцінки для терміну в контексті певної рубрики.

Вихідними даними цього кроку, і всього першого етапу, є множина термінів та їх дисперсійна оцінка.

Другий етап відповідає за компактифікацію термінів за граматичними та синтаксичними властивостями. Вхідними даними для цього етапу є множина термінів та їх дисперсійна оцінка.

На *першому кроці* другого етапу визначаються граматичні властивості слів. На цьому етапі з допомогою бази даних корпусу слів української мови для кожного із слів в термінах встановлюються наступні властивості слова: інфінітив, рід, відмінок, частина мови.

Другий крок другого етапу – фільтр термінів за синтаксичними та граматичними ознаками. Наприклад якщо термін є одним словом то згідно проведених досліджень він може бути тільки іменником (див. п. 4.1.2).

Останнім, *третім кроком* другого етапу є лематизація та компактифікація термінів. На цьому етапі проводиться лематизація кожного слова у кожній фразі. Після чого отримана множина термінів обробляється таким чином щоб уникнути повторень термінів. Терміни, що повторюються видаляються і залишається лише один термін з найвищою оцінкою.

Вихідними даними другого етапу є компактифікована за граматичними та синтаксичними властивостями множина термінів та їх дисперсійна оцінка. Для роботи цього етапу використовується база корпусу українських слів. Другий етап не являється обов'язковим.

Третій етап відповідає за формування множини ключових термінів. Вхідними даними для цього етапу є множина термінів та їх оцінка.

На *першому кроці* третього етапу виконується поглинання термінів. Оскільки на етапі формування множини термінів додавались усі можливі варіанти термінів в межах фраз без поглинання більшими словосполученнями менших, в даному блоці проводиться аналіз необхідності такого поглинання.

Другий крок третього етапу це – корекція оцінки терміну відносно стилів. Для корекції остаточної оцінки терміну поточна оцінка терміну множиться на зданий користувачем коефіцієнт відповідно до певного типу стилю тексту (bold, italic, underline).

Крок 3.3 відповідає за обмеження кількості термінів та формування множити ключових термінів. Відбувається сортування термінів за їхньою оцінкою після чого формується множина термінів з найбільшими значеннями

оцінки важливості. Обмеження термінів відбувається за певним відсотком вказаним в якості параметра користувачем ІС.

Результатом цього кроку є остаточний результат методу – множина ключових термінів у рубриках інформаційного навчального матеріалу. Наведені етапи схематично зображені на рисунку 2.5.

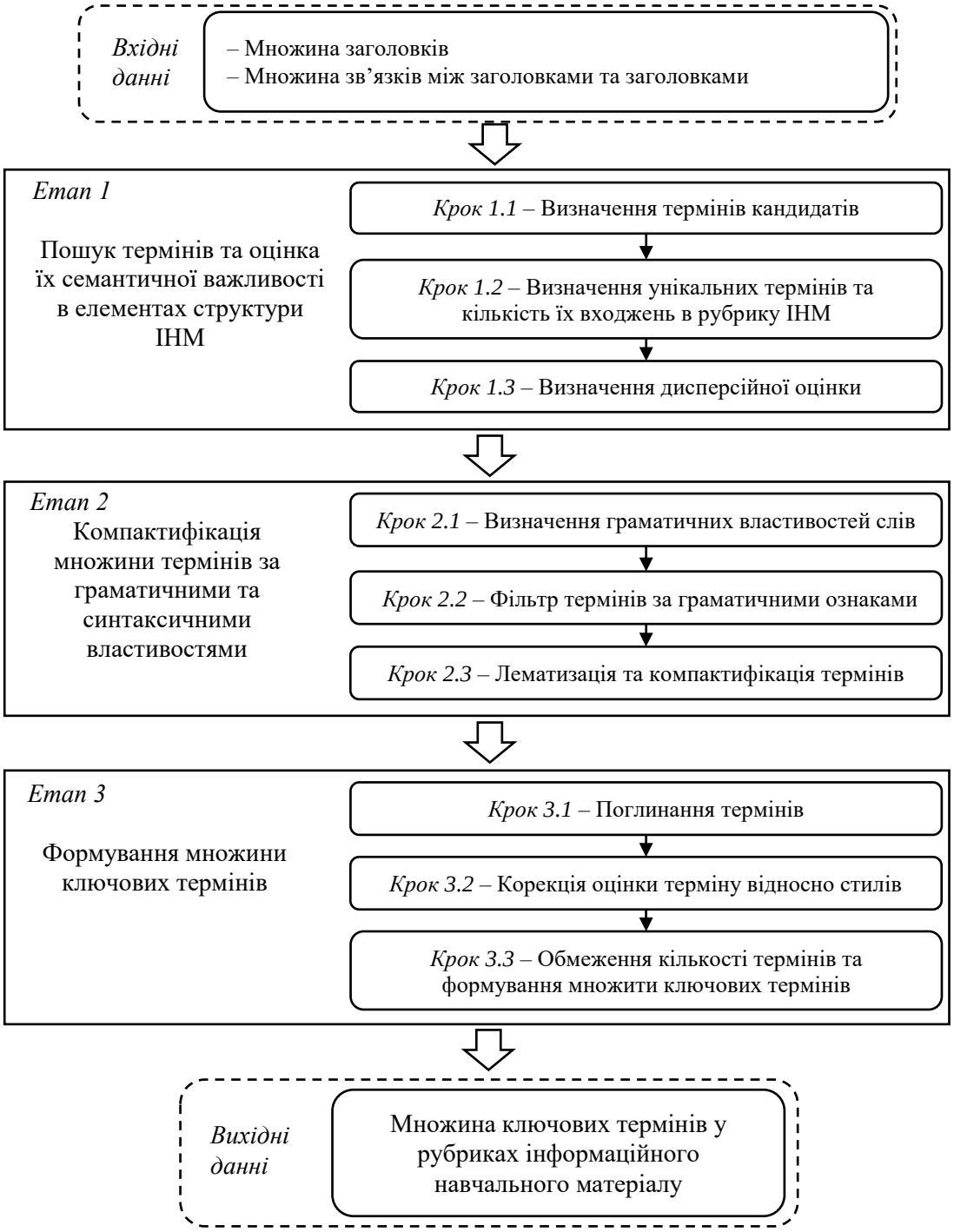


Рисунок 2.5 – Схема етапів методу пошуку ключових термінів у рубриках ІНМ

Таким чином, після пошуку термінів та оцінки їх семантичної важливості в елементах структури ІНМ, а також компактифікації множини термінів за граматичними та синтаксичними властивостями, формується множина ключових термінів та їх дисперсної оцінки в контексті певної рубрики ІНМ.

2.4 Схема інформаційної технології автоматизованого формування семантичної структури інформаційного навчального матеріалу

ІТ автоматизованого формування семантичної структури інформаційного навчального матеріалу використовує запропоновану в п. 2.1 інформаційну модель семантичної структури інформаційного навчального матеріалу та методи:

– метод формування структури (див. п. 2.2);

– метод пошуку ключових слів (див. п. 2.3).

Схематично застосування означених методів у ІТ подано на рисунку 2.6.

Рисунок 2.6 – Схема застосування методів ІТ

Як видно з наведеної схеми, досягнення мети ІТ відбувається шляхом послідовного застосування розглянутих методів (рисунок 2.7).

Вхідними даними ІТ є файл електронного документу з контентом ІНМ, наприклад, формату .docx. Файл має містити слабо структурований текстовий контент, для рубрикації бажане використання заголовків, які визначають структуру текстового контенту. Ніякої спеціальної попередньої обробки вхідних

даних не вимагається, втім підвищення рівня виконання вимог роботи з електронними документами (використання стилів) та вимог до роботи з навчальними матеріалами може суттєво підвищити якість вихідних даних ІТ.

Рисунок 2.7 – Послідовність кроків ІТ

Метод формування рубрикації інформаційного навчального матеріалу є першим етапом роботи ІТ. Результатом роботи методу є формування частини моделі семантичної структури ІНМ, а саме верхнього рівня структури ІНМ у вигляді рубрикації (структури заголовків).

При аналізі вмісту документу обробляється контент документу для визначення стилів його складових. Відповідно до них, формується структура заголовків, одним з атрибутів кожного з яких є приналежний йому обсяг контенту документу. В подальшому обробка контенту кожного з одержаних елементів відбувається окремо і незалежно. При цьому контент підлеглого елементу одночасно входить до складу контенту елемента-батька в ієрархічній структурі заголовків.

Метод пошуку ключових термінів у рубриках інформаційного навчального матеріалу є другим етапом роботи ІТ. Для кожного елемента в структурі ІНМ проводиться аналіз контенту для пошуку ключових термінів із використанням методу дисперсійного оцінювання. При цьому терміни-кандидати обираються в межах фраз, обмежених єдиним стилістичним оформленням та знаками пунктуації. Після визначення дисперсійної оцінки термінів, яка розглядається як показник їх важливості, відбувається поглинання термінів, що є частинами інших термінів з більшою кількістю слів. Одержана множина термінів сортується за значенням важливості терміну й обмежується за встановленим значенням показника щільності ключових термінів у відповідному контенті.

Результат роботи методу пошуку ключових термінів у рубриках інформаційного навчального матеріалу не прив’язаний до певної мови й на оцінку важливості терміну не впливають стильові ознаки в контенті. Втім, передбачена необов’язкова можливість коригування отриманих множин ключових термінів, що може підвищити якість вихідних даних:

– шляхом визначення нового показника важливості терміну за стилями документу, наприклад при виділенні в контенті терміну жирним, курсивом чи в складі заголовку;

– шляхом фільтрування за синтаксичними шаблонами, наприклад якщо термін є одним словом то згідно проведених досліджень він може бути тільки іменником;

– видаленням деяких термінів вручну у випадках, коли попри їх важливість контенті користувач не вважає їх вивчення метою даного фрагменту ІНМ.

Вихідними даними ІТ є семантична структура ІНМ, подана у вигляді ряду множин, яка може бути використана для автоматизованого вирішення ряду задач – визначення відповідності ІНМ вимогам, допомога при розробці ІНМ, оцінка відповідності наборів тестових завдань до ІНМ, допомога при створенні тестів й інших.

Висновки до розділу

У розділі було визначено інформаційну модель та два методи, які необхідно застосувати для наповнення елементів моделі: метод формування рубрикації інформаційного навчального матеріалу для визначення елементів моделі: множини заголовків, множини зв’язків між заголовками; метод пошуку ключових термінів у рубриках інформаційного навчального матеріалу для визначення елементів моделі: множини ключових термінів і множини слів, множини зв’язків між заголовками і термінами, множини зв’язків між термінами і словами.

Також була розроблена схема інформаційної технології та описано її послідовні кроки. Схема інформаційної технології описує процес повного визначення елементів моделі, що відкриває можливості для її практичного застосування. Інформаційна технологія автоматизованого формування семантичної структури інформаційного навчального матеріалу забезпечує автоматизоване одержування семантичної структури інформаційного навчального матеріалу за цифровим файлом електронного документу.

Розділ 3

Інформаційна система для автоматизованого формування семантичної структури інформаційного навчального матеріалу

3.1 Функціональне подання інформаційної технології як інформаційної системи

Функціональне подання запропонованих у ІТ методів перетворення інформації для автоматизованого формування семантичної структури зображено на рисунку 3.1 у вигляді IDEF0-діаграми.

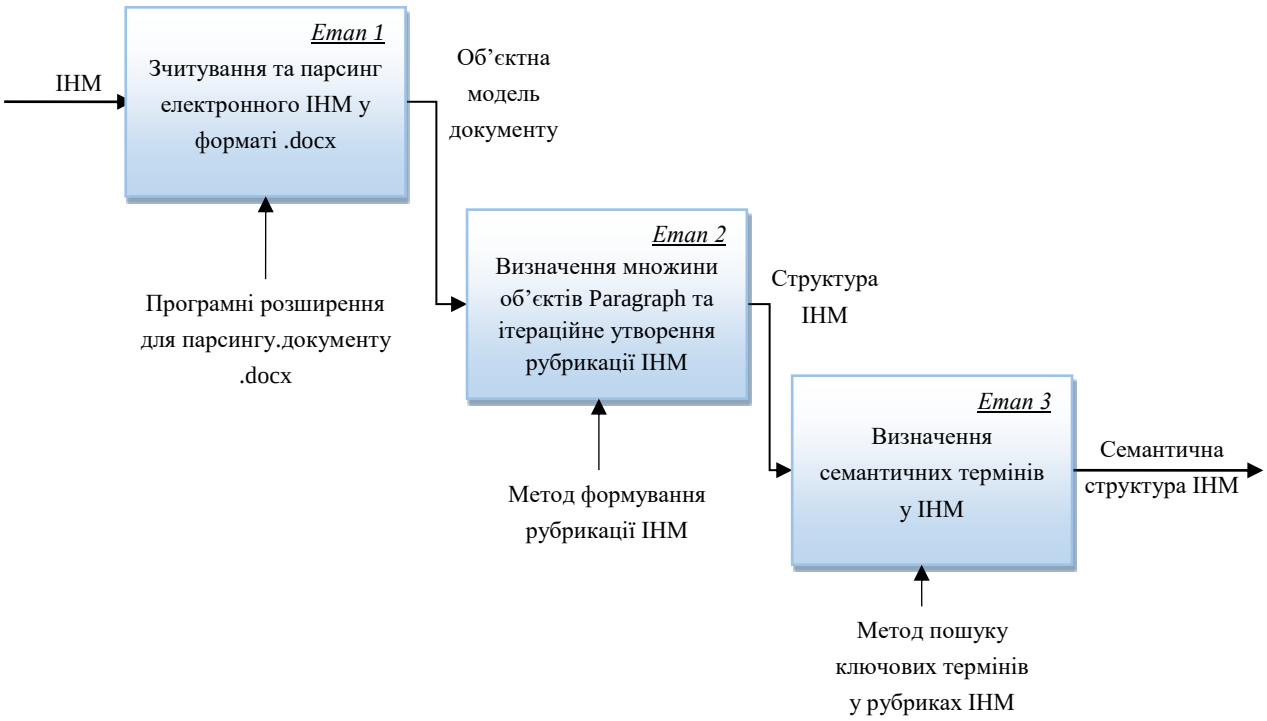


Рисунок 3.1 – Діаграма етапів вирішення задачі автоматизації створення семантичної структури ІНМ

Кожному з наведених етапів відповідає певна впорядкованість перетворення даних і компоненти системи, які прийматимуть у цьому участь. На рисунку 3.2 зображено відповідну схему розподілу функцій за компонентами системи.

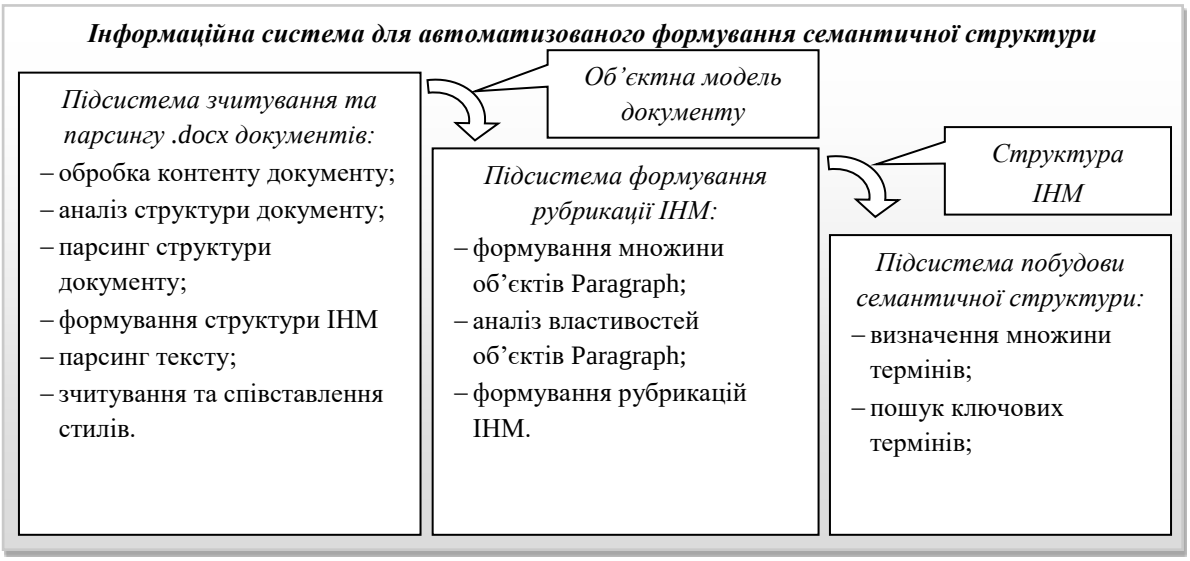


Рисунок 3.2 – Схема розподілу функцій за компонентами системи

Підсистема зчитування та парсингу .docx документів виконує функції обробки контенту вхідного документу, визначає всі необхідні властивості документу та його компонентів. Підсистема формування рубрикації ІНМ опрацьовує модель документу, для визначення рубрикацій ІНМ. Підсистема побудови семантичної структури відповідає за пошук термінів у рубрикація ІНМ та визначення множин ключових термінів.

Цей розподіл функцій за підсистемами визначає обсяг можливих дій користувача, які наведено на рисунку 3.3.

Функції користувача можна розділити на дві категорії:

1. Обов'язкові активні функції користувача, без яких робота системи є неможливою, до яких належить вибір файлу документу для обробки.

2. Необов'язкові активні функції користувача, які впливають на кінцевий результат роботи системи, до яких належать:

– встановлення етапу пошуку ключових слів;

– встановлення коефіцієнтів впливу стильових ознак;

– встановлення параметрів пошуку.

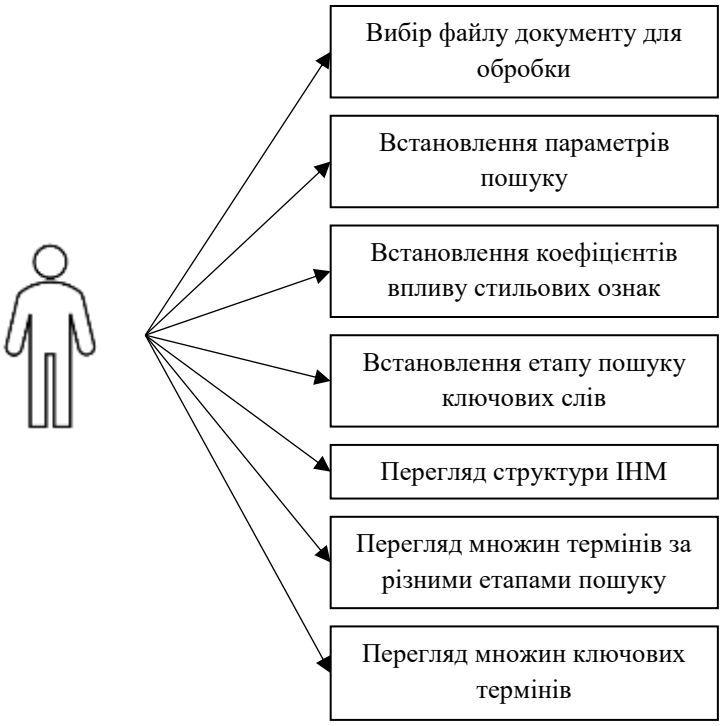


Рисунок 3.3 – Функції користувача інформаційної системи

Таким чином, обов’язковою функцією користувача системи є вибір файлу документу ІНМ для обробки. Подальші кроки, що приводять до формування семантичної структури, система здатна виконувати самостійно. Решта функцій користувача є необов’язковими (для покращення результату роботи системи) або спостережними (для розуміння процесу одержання проміжних і вихідних даних).

3.2 Комбінація засобів розробки інформаційної системи

Відповідно до поставленої задачі, ІС повинна зчитувати та парсити електронні ІНМ у форматі .docx, обробляти отримані дані, та відображати отримані результати користувачеві. Для розробки ІС необхідно обрати платформу, мову програмування та тип додатку.

В якості типу додатку було обрано веб-додаток, так як цей тип має низку переваг перед іншими. Невідмінно від настільних та мобільних додатків, веб-додатки не потрібно встановлювати, вони можуть бути використані з будь якого

пристрою де є доступ до інтернету та веб-браузер, тому для користувача такі додатки мають значно менші системні вимоги.

Платформа та мова програмування мають бути здатні виконувати поставленні задачі. Оскільки ІС, в більшій мірі, має працювати з текстом важливим критерієм для вибору платформи та мови програмування є наявність та функціональність інструментів для роботи з текстом. Одним із найважливіших таких інструментів є формальна мова пошуку і маніпуляцій з підрядками в тексті – регулярні вирази (Regular Expression або RegEx). Різні платформи та мови програмування реалізовують підтримку регулярних виразів з допомогою власних механізмів, які можу підтримувати різні функції. Згідно дослідження [32] було проведене порівняння підтримки функцій формальної мови пошуку різними платформами та мовами програмування (рисунок 3.4).

Підтримка фукцій регулярних виразів платформами та мовами програмування

Ім'я	"+" quantifier	Negated character classes	Non-greedy quantifiers	Shy groups	Recursion	Look-ahead	Look-behind	Backreferences	>9 indexable captures	Directives	Conditionals	Atomic groups	Named capture	Comments	Embedded code	Unicode property support	Balancing groups	Variable-length look-behinds	Кількість підтримуваних функцій
Perl	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	-	-	16
Python	+	+	+	+	+	+	+	+	+	+	+	+	+	+	-	+	-	+	16
.NET	+	+	+	+	-	+	+	+	+	+	+	+	+	+	-	±	+	+	15
Ruby / Onigmo	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	±	-	-	15
PHP	+	+	+	+	+	+	+	+	+	+	+	+	+	+	-	-	-	-	14
Tcl	+	+	+	+	-	+	+	+	+	+	-	+	-	+	-	+	-	-	12
Java	+	+	+	+	-	+	+	+	+	+	-	+	+	+	-	±	-	-	12
Haskell	+	+	+	+	-	+	+	+	+	?	?	?	?	?	-	-	-	-	8
JavaScript	+	+	+	+	-	+	+	+	+	-	-	-	-	-	-	±	-	-	8
Lua	+	+	±	-	-	-	-	+	-	-	-	-	-	-	-	-	-	-	3
OCaml	+	+	-	-	-	-	-	+	-	-	-	-	-	-	-	-	-	-	3

+	Функція підтримується	±	Функція підтримується частково
-	Функція не підтримується	?	Функція не документована

Рисунок 3.4 – Порівняння підтримки функцій регулярних виразів різними платформами та мовами програмування

Згідно проведеного порівняння можна виділити п’ять платформ та мов програмування які підтримують найбільшу кількість функцій, а саме: мови

програмування Perl, Python, Ruby, PHP та платформу .NET. Решта платформ та мов програмування не підтримують велику кількість функцій, в тому числі використання логічних умов (Conditions), що є великим недоліком.

Згідно з дослідженням рейтингу мов програмування [33] C# (платформа .NET) є найпопулярнішою мовою з вище виділених. C# дозволяє розробляти різні типи додатків: веб додатки, мобільні додатки, настільні додатки, мікро сервіси тощо [34].

Таким чином мова програмування C# задовольняє поставлені вимоги і була обрана для розробки ІС.

3.3 Аналіз програмних розширень для роботи з електронними документами навчальних матеріалів

Файл з розширенням .docx – це файл для зберігання електронних документів. На відміну від файлів .doc, які зберігають дані документа в одному бінарному файлі, файли .docx створюються за допомогою відкритого формату XML, в якому зберігаються документи, як збори окремих файлів і папок в стислому пакеті. .docx-файли містять XML-файли і три папки, docProps, Word, і _rels (рисунок 3.5), які містять властивості документа, зміст і відносини між файлами .docx файли розроблені, щоб зробити вміст документів доступним. Наприклад, текстовий документ зберігається за допомогою простих текстових файлів і зображень документів, зберігаються у вигляді окремих файлів зображення для одного файлу .docx.

Оскільки файл стилів та текст знаходяться в різних XML-файлах, потрібно зчитувати відразу декілька документів та на їх базі формувати об’єктну модель документа, що є не найкращим рішенням, оскільки XML-файли надто громіздкі внаслідок чого обробляти документи таким чином буде не раціонально.

Автоматизація – це процес, який дозволяє програмам, написаним мовами, такими як Visual C# .NET, програмно керувати іншими програмами.

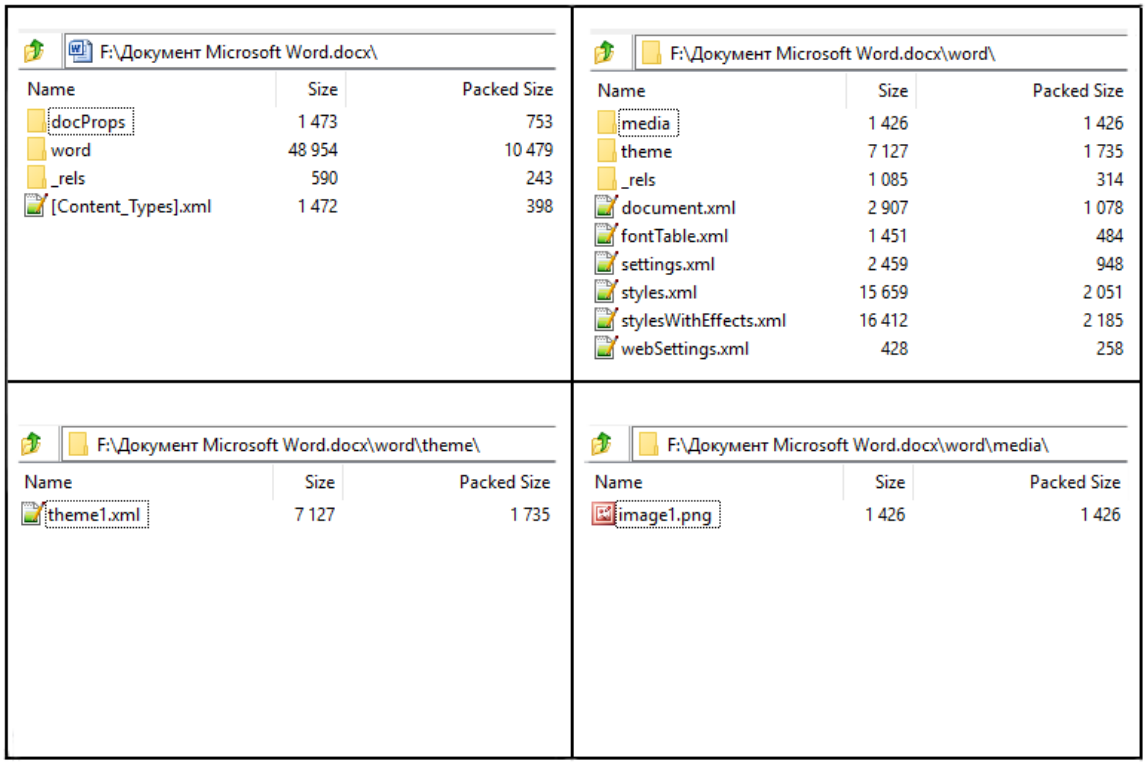


Рисунок 3.5 – Структура docx файлу

Автоматизація Microsoft Word (далі Word) дозволяє вам виконувати дії, такі як створення нового документу, додавання даних у документ або створення таблиць. З програмами Word та іншими програмами Microsoft Office (далі MS Office) практично всі дії, які ви можете виконувати вручну за допомогою користувацького інтерфейсу, також можуть бути виконані програмним шляхом за допомогою автоматизації.

Word виставляє цю програмну функціональність через об'єктну модель (рисунок 3.6). Об'єктна модель являє собою набір класів і методів, які служать аналогами логічних компонентів Word. Наприклад, існує об'єкт Application, об'єкт Document та об'єкт Paragraph, кожен з яких містить функціональність цих частин Word.

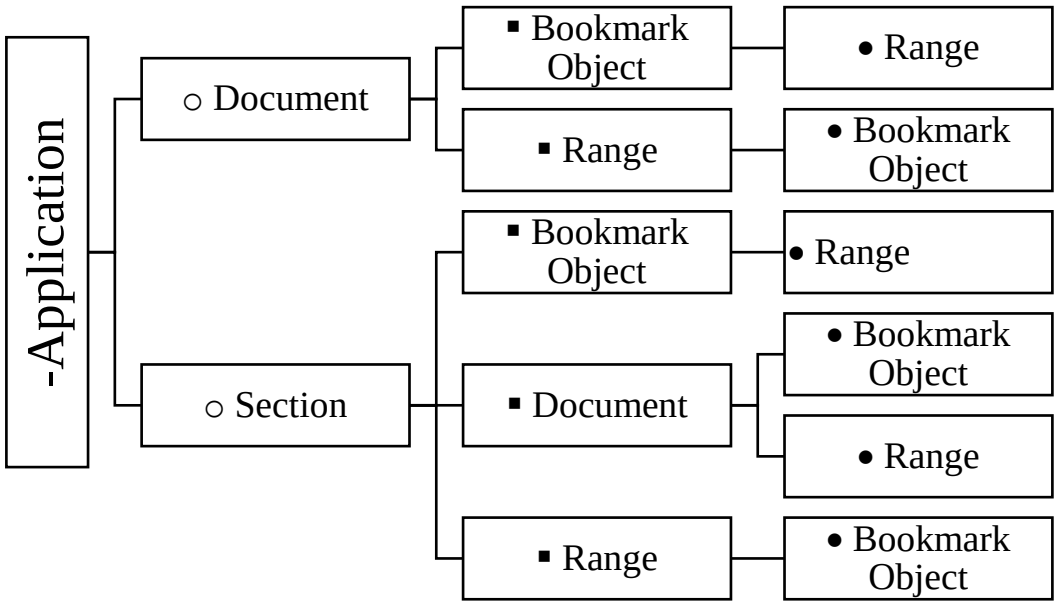


Рисунок 3.6 – Об'єктна модель Word

Щоб використовувати функції програми MS Office в своєму проекті, можна використовувати первинний взаємозв'язок (англ. Primary Interop Assembly далі PIA). PIA дозволяє керованому коду взаємодіяти MS Office та об'єктною моделлю на базі COM. Коли ви створюєте новий проект Office, Visual Studio додає посилання на PIA, необхідний для побудови проекту. У деяких сценаріях, можливо, доведеться додати посилання на додаткові PIA (наприклад, якщо ви хочете використовувати функцію MS Office Word в проекті для MS Office Excel).

Для роботи з Microsoft.Office.Interop.Word інтерфейсом необхідно встановити відповідну збірку, для цього потрібно зайти в меню Tools, обрати пункт NuGet Package Manager та запустити Package Manager Console та ввести наступну команду:

```
PM> Install-Package Microsoft.Office.Interop.Word
```

Після того як менеджер пакетів повідомить про успішне завершення встановлення можна починати працювати з необхідними інтерфейсами.

Хоч ця бібліотека надає доступ до всіх функцій MS Office, так як ми працюємо безпосередньо з PIA, для її роботи потрібна ліцензія для MS Office на

кожному клієнтському комп'ютері. Крім того при використанні Automation, MS Office завантажується у фоновому режимі, в наслідок чого займає певну кількість оперативної пам’яті та завантажує велику кількість файлів і DLL. Додатки MS Office були розроблені як додатки для користувацького інтерфейсу і через це Microsoft.Office.Interop.Word працює дуже повільно. Microsoft не рекомендує використовувати Office Automation (або будь-який Office Interop) на сервері.

Spire.Doc для .NET – це повністю незалежна бібліотека класів .NET Word, спеціально створена для розробників, яка дозволяє швидко генерувати, відкривати, писати, редагувати та зберігати документи Word з Word Version 97-2003 до 2013 року. Конвертація функціональних можливостей дозволяє розробникам легко здійснювати перетворення між Word в інші популярні формати документів, такі як PDF, EPub, HTML, RTF, Image, XML і т. д.

Для роботи з бібліотекою Spire.Doc необхідно встановити відповідну збірку, для цього потрібно запустити Package Manager Console та виконати наступну команду:

```
PM> Install-Package Spire.Doc
```

Після завершення встановлення можна починати працювати з бібліотекою.

Spire.Doc не вимагає встановлення в систему кожного користувача MS Office, тобто є можливість повністю незалежної від нього роботи; об’ємна документація з прикладами та поясненнями. Проте повна версія Spire.Doc не є безкоштовною, а безкоштовна версія, FreeSpire.Doc, має певні обмеження (наприклад обробка не більше 500 абзаців і 25 таблиць).

Обов’язковість наявності MS Office на машині користувача та його запуск в фоновому режимі є серйозними недоліками Microsoft Interop. Оскільки Spire.Doc немає цих недоліків та має значно більшу швидкодію для вирішення

поставленої проблеми було обрано цю бібліотеку. програмних засобів буде використано для розробки системи.

Отже, було обрано комбінацію платформи .NET та мови програмування C# для розробки основної частини ІС. Для керування базами даних було обрано СКБД MS SQL Server. Для роботи з електронними документами формату .docx бібліотеку Office Open XML (OOXML).

3.4 Архітектура інформаційної системи

3.4.1 Структура модулів інформаційної системи

ІС містить три логічні модулі (рисунок 3.7) один з яких модуль користувацького інтерфейсу (рисунок 3.8).

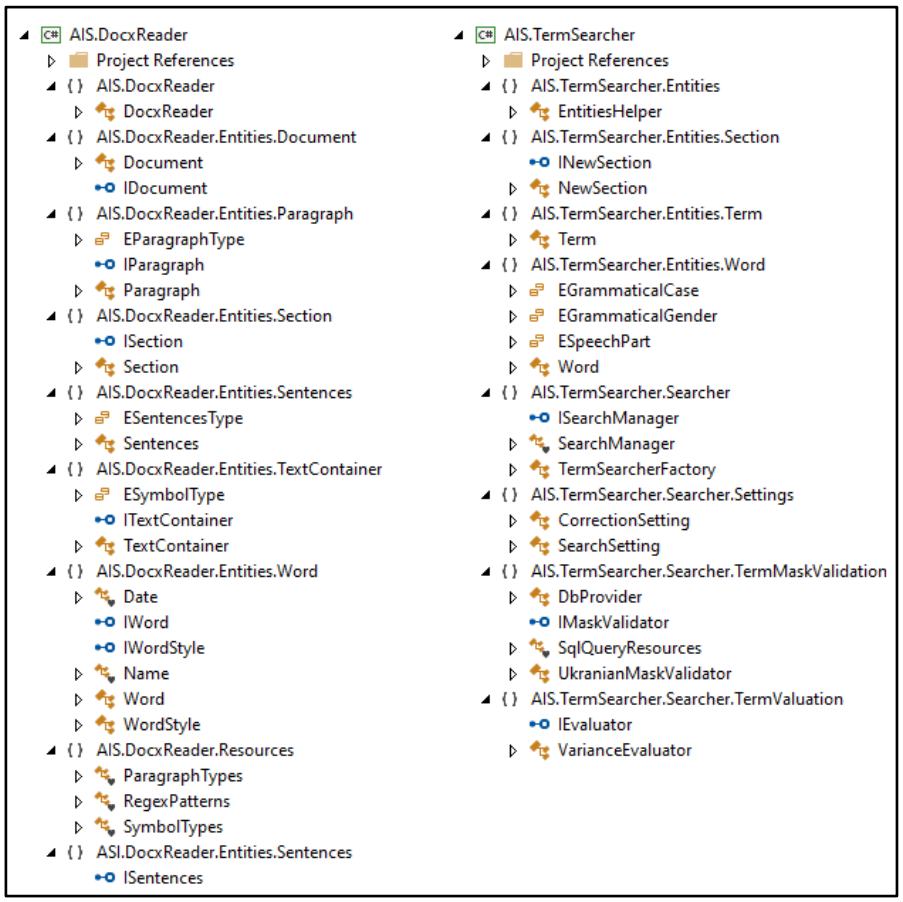


Рисунок 3.7 – Логічні модулі

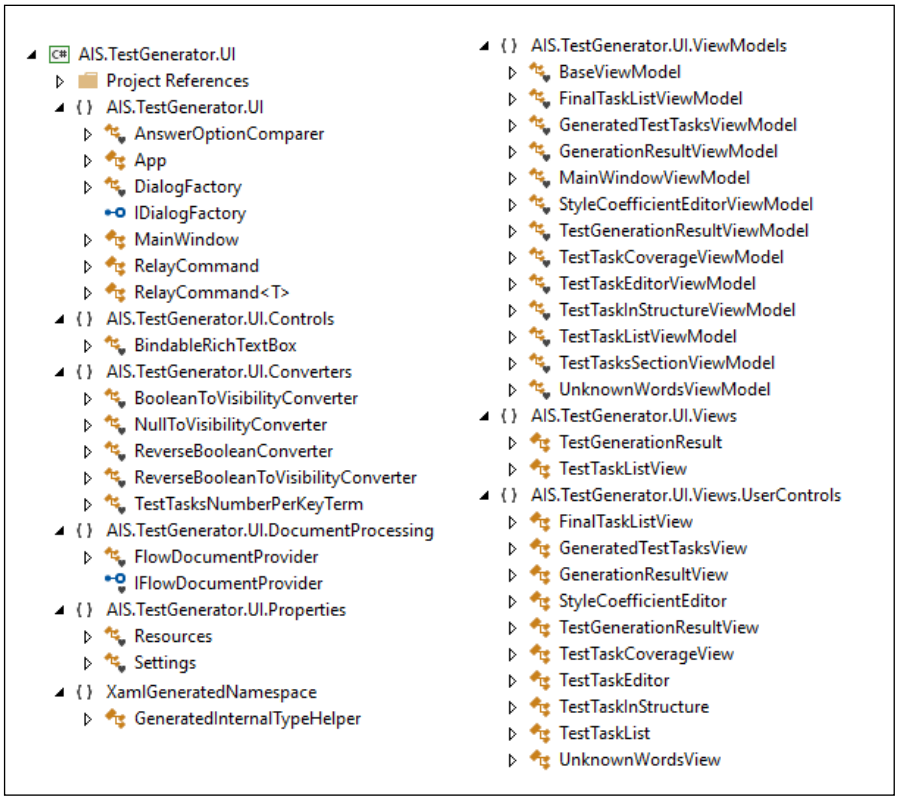


Рисунок 3.8 – Модуль користувацького інтерфейсу

AIS.DocxReader – модуль для роботи з електронним документом типу docx. Головною задачею цього модулю є зчитування та парсинг. Також модуль відповідає за побудову рубрик документу.

AIS.TermSearcher – модуль пошуку ключових термінів. Відповідає за формування множин термінів у межах специфічних рубрик їх компактифікацію, оцінку та фіксацію результатів.

AIS.TermSearcher.UI – модуль користувацького інтерфейсу.

Всі вище перераховані модулі послідовно взаємодіють між собою. Детально робота кожного модуля описана в розділах (3.3.2, 3.3.3 та 3.3.4).

3.4.2 Модуль «Робота з електронним документом»

Відповідно до описаної об’єктної моделі документу було розроблено структуру класів на основі інтерфейсів зображену на рисунку 3.9.

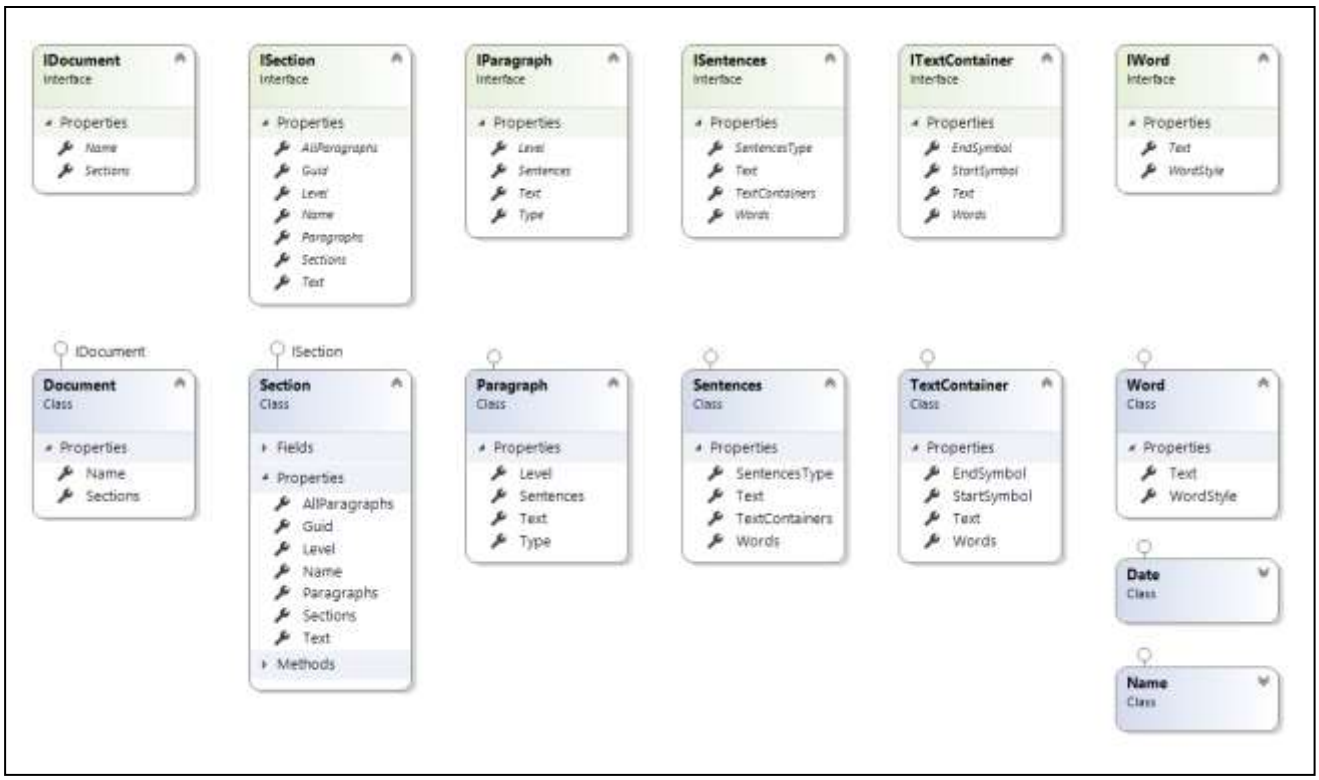


Рисунок 3.9 – Діаграма класів, що описують модель документу

Підхід проектування структури на основі інтерфейсів дозволяє послабити залежність від реалізації, що в результаті забезпечує гнучку розширюваність та полегшує процес автоматичного тестування. Нижче представлений детальний опис всіх класів.

Логіку зчитування документу описує клас DocxReader (рисунок 3.10).

Зчитування документу розпочинається викликом методу Read класу DocxReader з формування колекції абзаців які описані в класі Paragraph. Кожному абзацу встановлюється рівень який відповідає поточному заголовку та один із визначених в перелічені EParagraphType типів. Властивості класу Paragraph описані в таблиці 3.1.

Відповідно до тексту в абзаці методом GetSentences формується колекція речень. Модель речення описує клас Sentences.

Відокремлення окремих речень з тексту абзацу виконується за розділовими знакам: крапка, знак оклику, знак питання, знак три крапки. В результаті речення позначається одним із визначених в перелічені

ESentencesType типів (питальне, окличне, стверджувальне). Символ крапки також може бути використаний для позначення скорочень слів. Оскільки ці символи можуть бути сприйняті програмою як розділові знаки, для покращення точності відокремлення речень, часті скорочення (ін., тис., див., мал., Рисунок, т. д., т. п.) були додані до списку виключень.

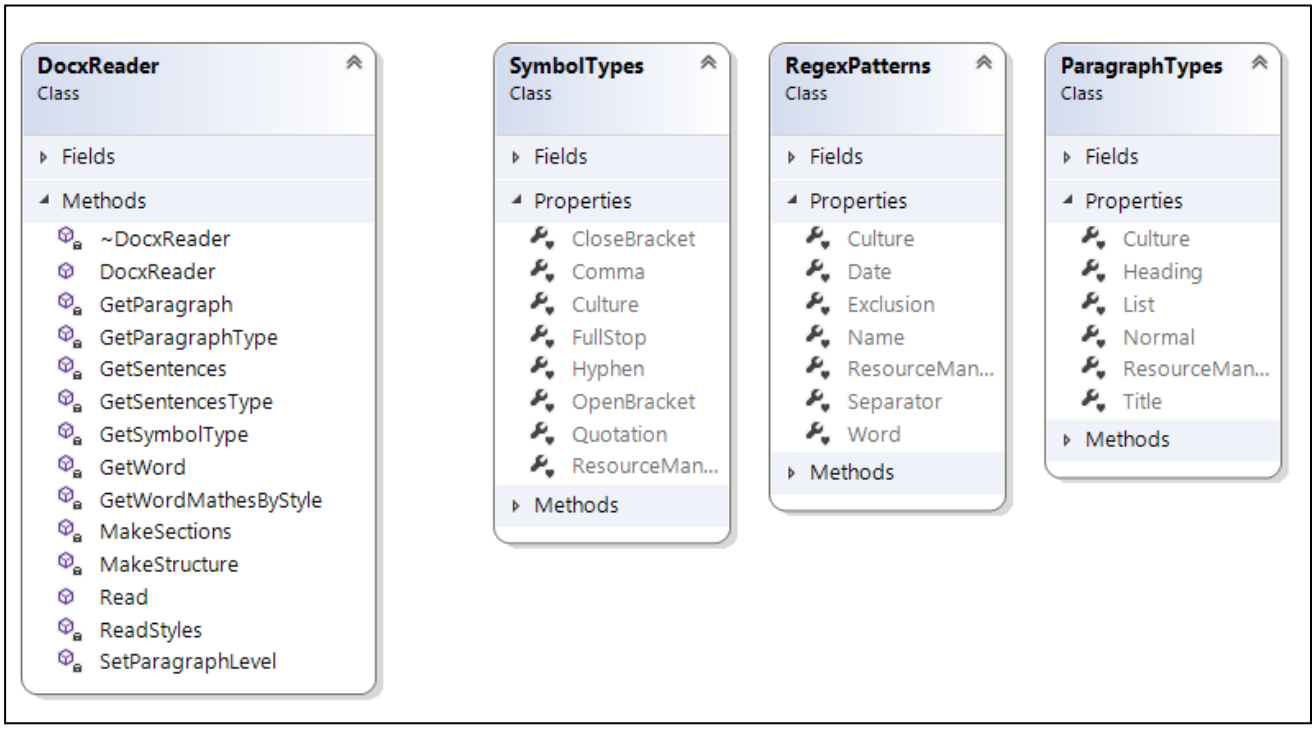


Рисунок 3.10 – Діаграма класу DocxReader та допоміжних класів

Таблиця 0.1 Властивості класу Paragraph

Назва	Опис
Type	Тип параграфу
Level	Рівень параграфу. Відповідає рівню батьківської секції
Sentences	Представляє рекурсивну колекцію речень
Text	Текст абзацу

Модель речення описується класом Sentences. Властивості класу описані в таблиці 3.2.

Таблиця 0.2 Властивості класу Sentence

Назва	**Опис**
ESentencesType	Тип речення
TextContainers	Представляє колекцію текстових контейнерів
Words	Представляє колекцію слів
Text	Текст речення

Відразу ж під час того як формується колекція речень, в результаті синтаксичного аналізу формуються колекції слів та текстових контейнерів в межах цього речення. Детально синтаксичний аналіз буде описаний нижче, після опису формування документу.

Моделі слів та текстових контейнерів описані в класах Word (таблиця 3.3) та TextContainer (таблиця 3.4) відповідно.

Таблиця 0.3 Властивості класу TextContainer

Назва	**Опис**
StartSymbol	Представляє символ початку контейнера
EndSymbol	Представляє символ закінчення контейнера
Words	Представляє колекцію слів
Text	Текст контейнера

Таблиця 0.4 Властивості класу Word

Назва	**Опис**
Text	Представляє текст слова
WordStyle	Представляє стиль слова

Після того як всі параграфи та їхній вміст сформовано відповідно до їх рівнів в методі MakeStructure, рекурсивними викликами MakeSections формуються секції та структура документу.

Модель секцій описується класом Section. Властивості класу описані в таблиці 3.5.

Таблиця 0.5 Властивості класу Section

Назва	**Опис**
Name	Назва секції. Відповідає заголовку
Level	Рівень секції. Відповідає рівню заголовку
Guid	Унікальний глобальний ідентифікатор секції
Text	Текст секції
Paragraphs	Представляє колекцію абзаців поточної секції
AllParagraphs	Представляє колекцію абзаців поточної секції та всіх підсекції
Sections	Представляє рекурсивну колекцію підсекцій

Клас Document описує модель електронного документу. Екземпляр цього класу є результатом роботи зчитування документу. Властивості класу описані в таблиці 3.6.

Таблиця 0.6 Властивості класу Document

Назва	**Опис**
Name	Назва документу. Відповідає назві фізичного docx файлу
Sections	Представляє колекцію секцій

Весь вищеописаний процес зчитування документу зображений на мапі коду, на рисунку 3.11.

Для синтаксичного аналізу речення використовується формальна мова пошуку – регулярні вирази. З допомогою визначеного патерну (шаблону) формується множина окремих слів та розділових знаків. Далі створений патерн розглядається поетапно.

Шаблон для відокремлення окремого слова має наступний вигляд:

```
(?<word>(?>\w+\S)*\w+|\w+)
```

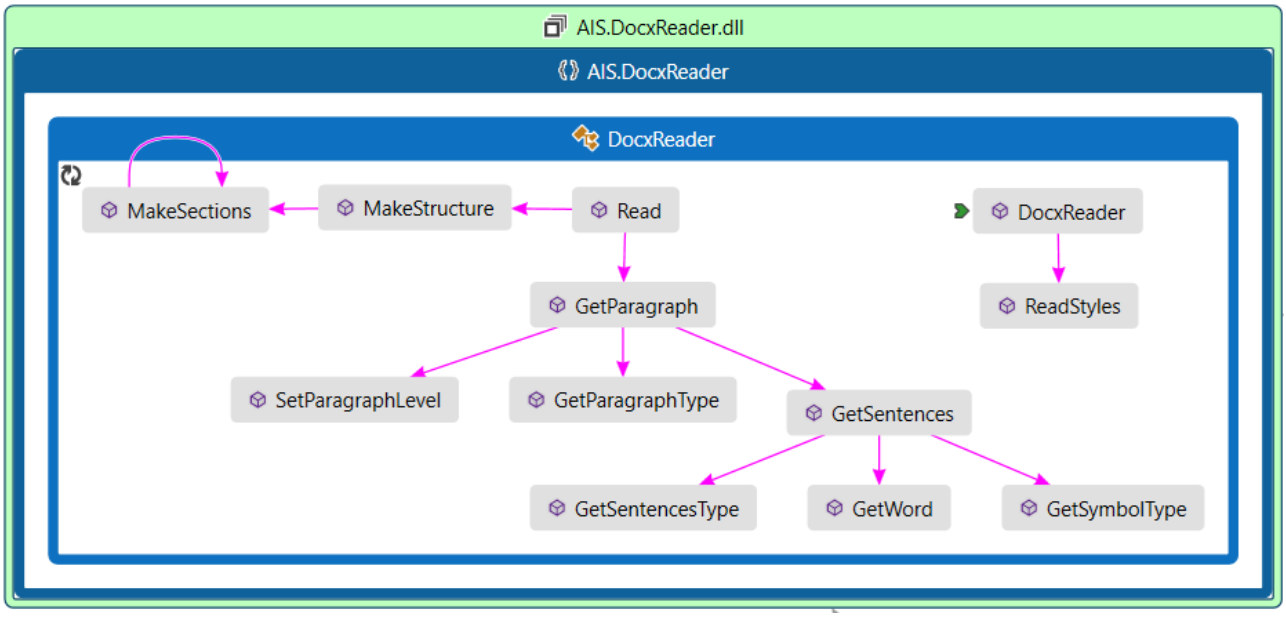


Рисунок 3.11 – Мапа коду – читання документу

Результат роботи патерну можна побачити в безкоштовному онлайн тестері регулярних виразів regexstorm.net (рисунок 3.12).

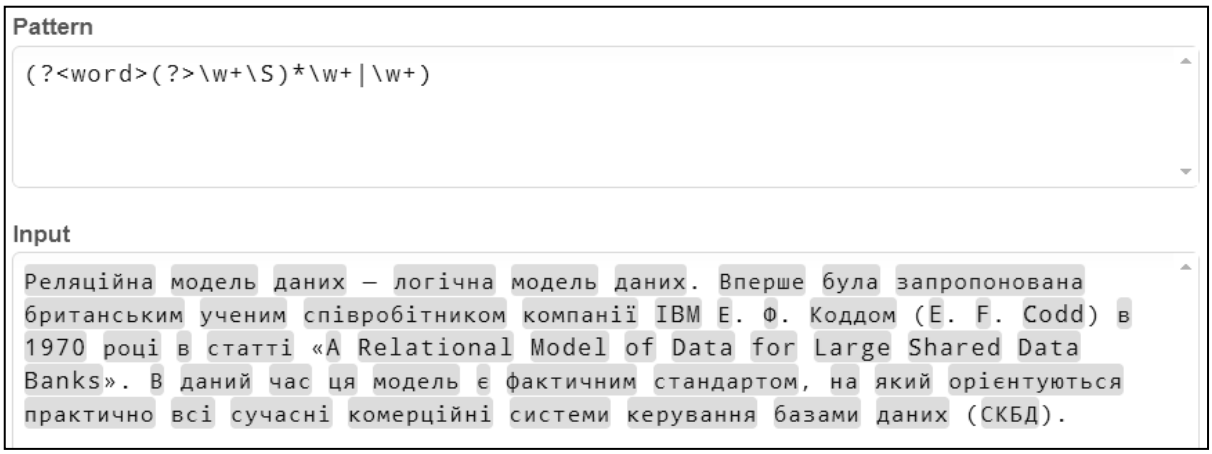


Рисунок 3.12 – Патерн для відокремлення слова

Для відокремлення символів використовується наступний патерн:

```
(?<separator>\S)
```

Як видно з рисунку 3.5 в наслідок об’єднання двох описаних патернів операцією “або” отримано непоганий результат, такий патерн дозволить

виділяти окремі слова та текстові контейнери. Але через скорочення імені (Е. Ф. Коддом) буде важко коректно сформувати речення оскільки символ крапки для скорочення імені може бути сплутаний з символом крапки для закінчення речення (рисунок 3.13).

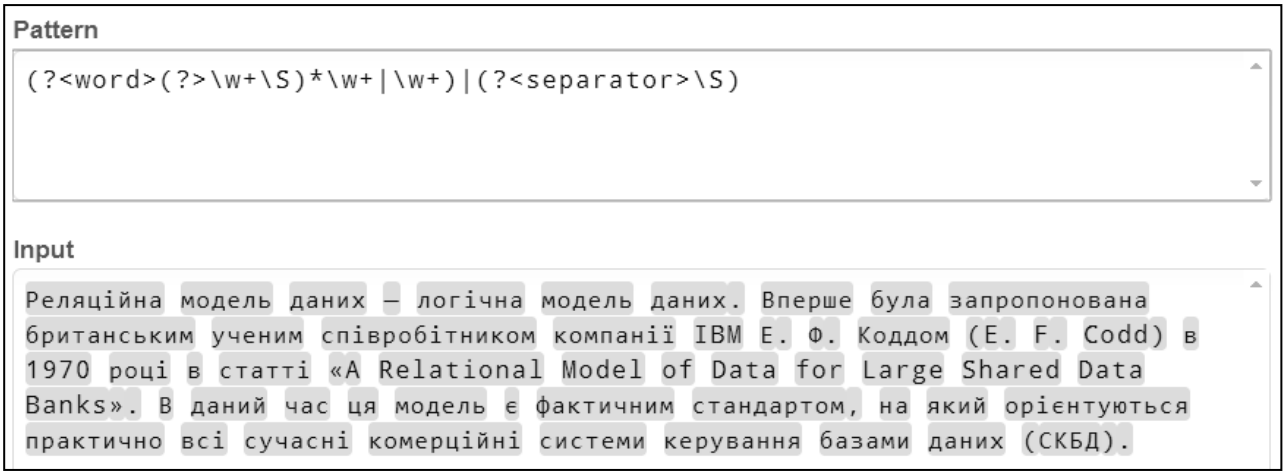


Рисунок 3.13 – Патерн для відокремлення слів та символів

Для вирішення цієї проблеми було сформовано окремий патерн для визначення імен в наступному форматі:

{Скорочення імені}. {Скорочення по батькові}. {Прізвище}

Де скорочення по батькові опціональне. Будуть опрацьовані лише скорочення які сформовані граматично правильно та відповідають описаному формату. Сформований патерн має наступний вигляд:

```
(?<=\s|\W|^)
(?<name>
  (?<initials>(?>(?>[А-ЯІЇЄ]|[A-Z])\.\s?(?>|$)){1,2})
  (?<lastname>(?>[А-ЯІЇЄ]|[A-Z])(?>(?>\w+\S)*\w+|\w+)(?>-(?>[А-ЯІЇЄ]|[A-Z])(?>\w+\S)*\w+|\w+)?)
)
```

Переноси рядків та пробіли були додані для покращення сприйняття, а перед використанням їх потрібно видалити з патерну.

Результат об’єднання зображено на рисунку 3.14.

Також додатково були розроблені патерни для та популярних виключень та шаблонів дат.

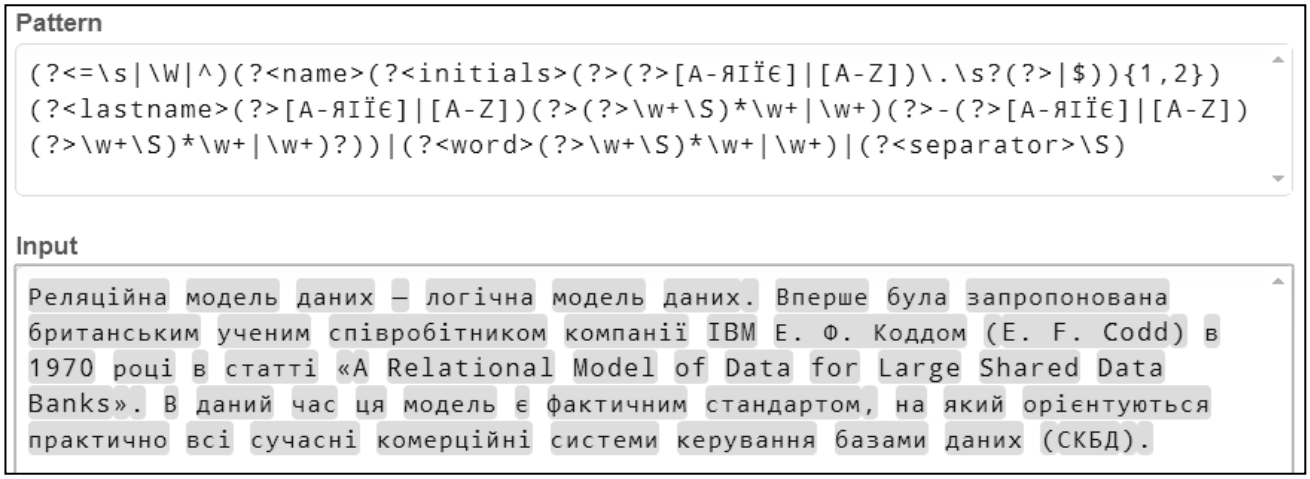


Рисунок 3.14 – Визначення імен

Патерн для виключень:

```
(?<exclusion>ін\.|тис\.|див\.|мал\.|рис\.|т\.д\.|т\.п\.)
```

Патерн для дат:

```
(?<day>\d{1,2}
  (?>\s+|-\d{1,2}\s+)
  (?>січня|лютого|березня|квітня|травня|червня|липня|серпня|вересня|жовтня|листопада|грудня)
\s+)
|
(?<year>\d{4}(?>\s*(?>р\.|рік|року|році))|\d{4}-\d{4}\s*(?>рр\.|років|роках))
```

Послідовність визначених підпатернів має велике значення оскільки, наприклад, крапка в ініціалах може бути сприйнята за розділовий знак до того як опрацьовується патерном для імен. Таким чином загальний патер має наступний вигляд:

```
(?<exclusion>ін\.|тис\.|див\.|мал\.|рис\.|т\.д\.|т\.п\.)|
(?<day>\d{1,2}
```

```
  (?>\s+|-\d{1,2}\s+)
  (?>січня|лютого|березня|квітня|травня|червня|липня|серпня|вересня|жовтня|листопада|грудня)
\s+)
|
(?<year>\d{4}(?>\s*(?>p\.|рік|року|році))|\d{4}-\d{4}\s*(?>pp\.|років|роках))
(?<=\s|\W|^)
(?<name>
  (?<initials>(?>(?>[A-ЯІЇЄ]|[A-Z])\.\s?(?>|$)){1,2})
  (?<lastname>(?>[A-ЯІЇЄ]|[A-Z])(?>(?>\w+\S)*\w+|\w+)(?>-(?>[A-ЯІЇЄ]|[A-Z])(?>\w+\S)*\w+|\w+)?)
)|
(?<word>(?>\w+\S)*\w+|\w+)|
(?<separator>\S)
```

Результат роботи загального патерну зображено на рисунку 3.15.

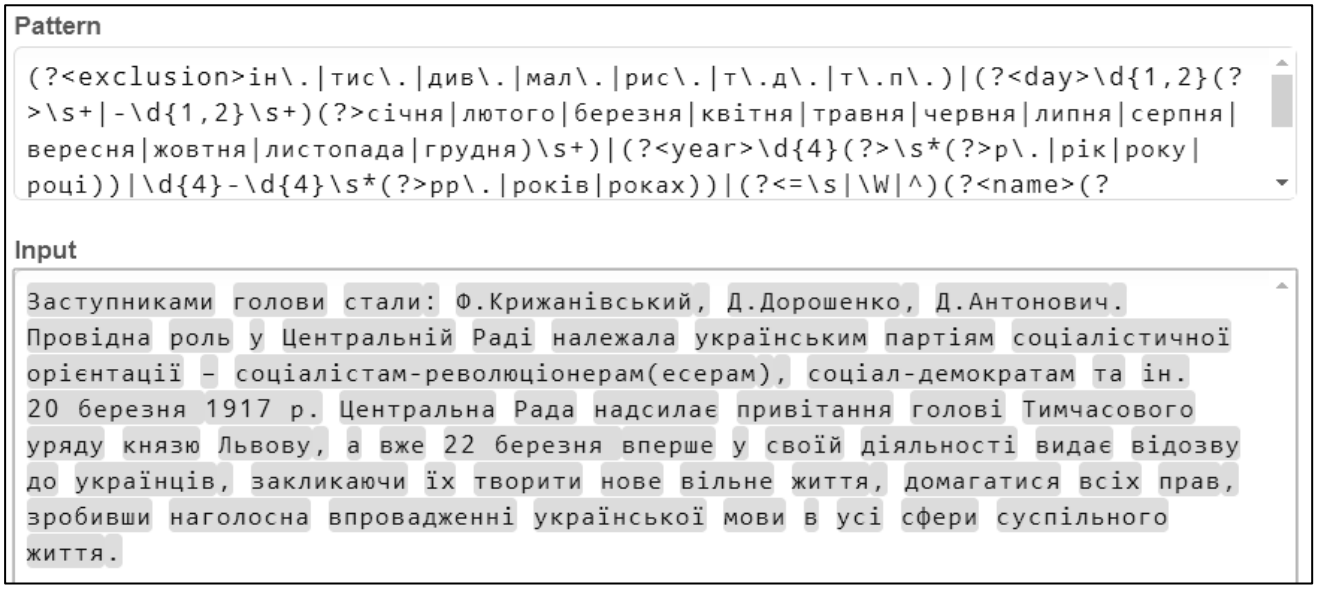


Рисунок 3.15 – Загальний патерн

Таким чином розроблений регулярний вираз формальної мови пошуку для синтаксичного аналізу речення дозволяє визначати слова, розділові знаки, популярні шаблони дат, імен з ініціалами та дозволяє враховувати визначенні виключення.

3.4.3 Модуль «Пошук ключових термінів»

Отриманий об’єкт зчитаного цифрового документу в попередньому модулі передається на обробку в модуль «Генерація ключових термінів» (рисунок 3.16).

Головним класом в модулі пошуку ключових термінів є SearchManager його екземпляр створюється через фабрику TermSearcherFactory.

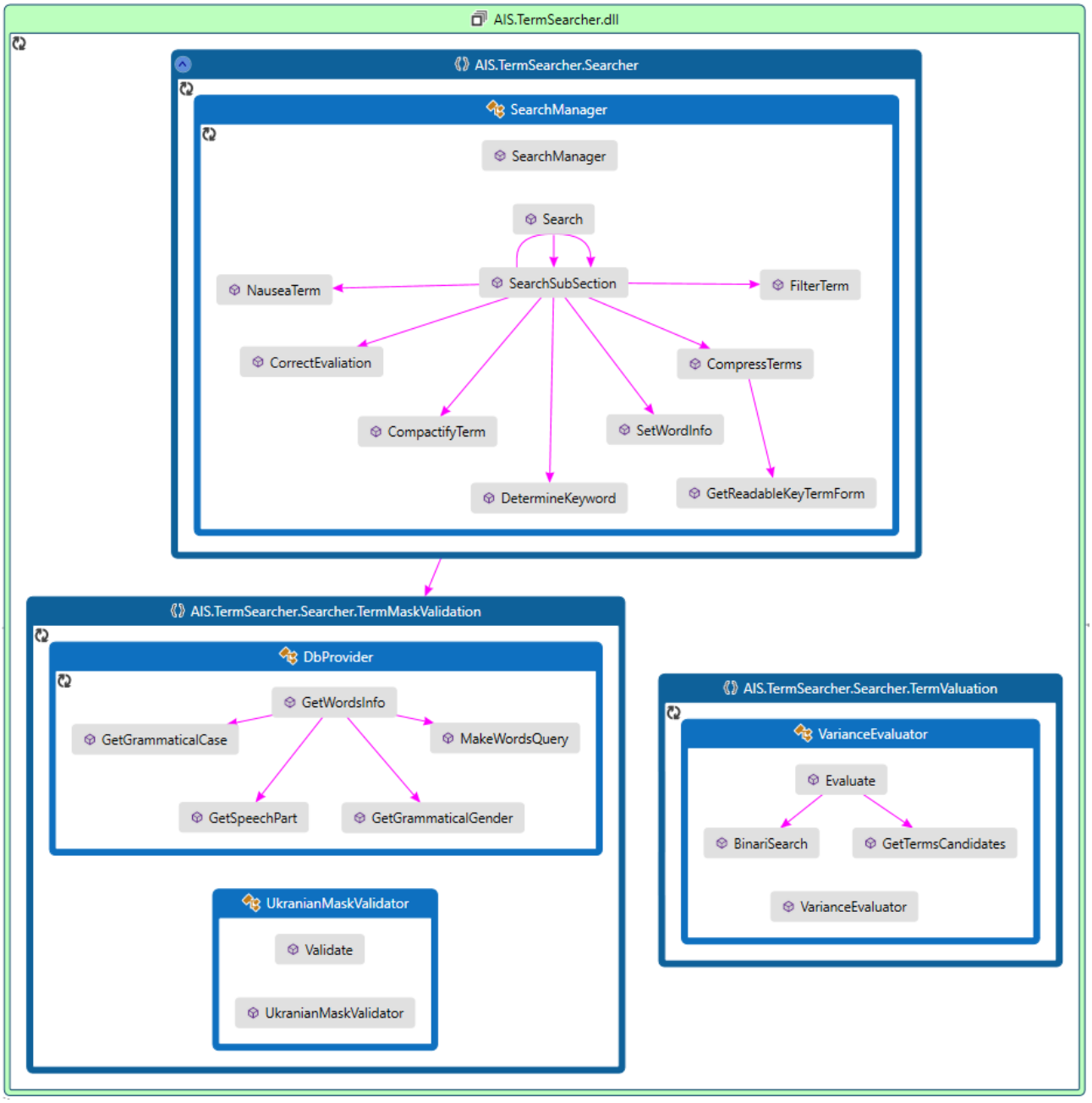


Рисунок 3.16 – Карта коду «Генераці ключових термінів»

В реалізованій фабриці створюється менеджер пошуку ключових слів з допомогою дисперсійної оцінки та з масками української мови. Підхід створення менеджера через фабрику дозволяє зручно розширювати проект, додавати нові фабрики та створювати менеджери різних типів не прикладаючи значних зусиль на оновлення структури проекту. Наприклад можна створити менеджера пошуку ключових слів з масками для англійської мови та використовувати інший алгоритм пошуку (оцінювач). Діаграма класів зображена на рисунку 3.17.

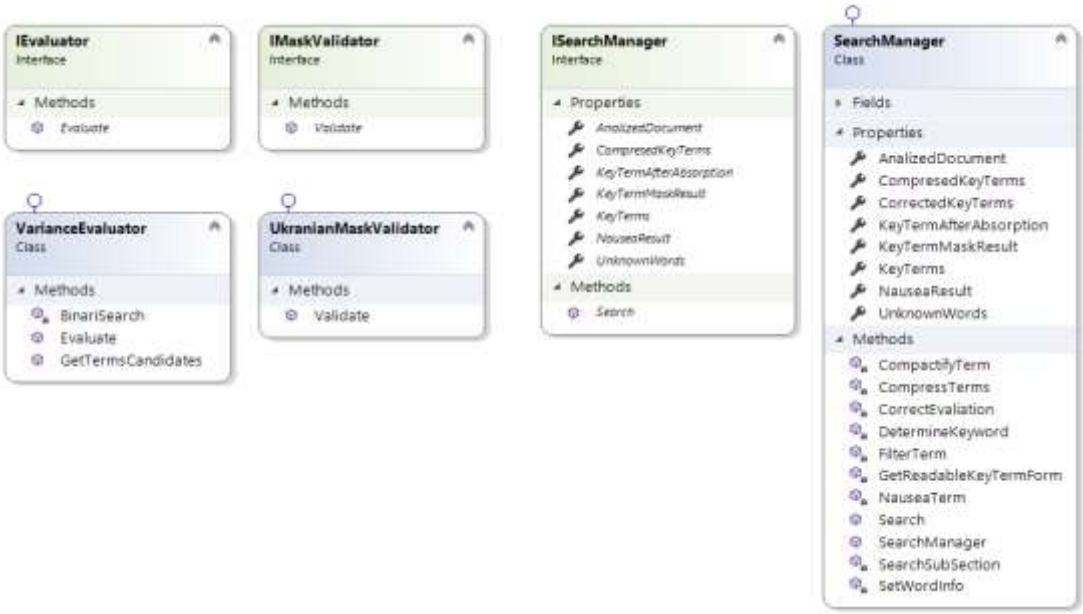


Рисунок 3.17 – Діаграма класів

Нараз реалізований валідатор для шаблонів українських термінів та алгоритм оцінки термінів на базі алгоритму для вираховування дисперсійної оцінки термінів.

Пошук термінів починається з виклику DetermineKeyword де з допомогою заданого оцінювача визначаються всім можливі кандидати на ключові терміни, а потім проводиться оцінка кожного з них.

Далі для сформованих термінів з бази даних встановлюються семантичні властивості (рід, відмінок, частина мови, лема). Для отримання необхідних даних із бази виконується наступний SQL запит.

Далі терміни фільтруються за вказаним валідатором відповідно до семантичних даних. В наступних етапах терміни поглинаються та компактифікуються. Проводиться корекція оцінки термінів відповідно до стилів форматування. Наступний етап обмеження результуючої множини термінів. Після цього етап пошуку ключових термінів завершено.

3.4.4 Модуль «Веб інтерфейс для відображення ключових термінів»

Модуль був розроблений з використанням ASP.NET MVC, JavaScript відповідає за візуалізацію результатів. За допомогою власної бібліотеки myTree.js відображається ієрархія рубрик документу та з використанням бібліотеки Tabulator.js відображається перелік ключових термінів. Схема класів зображена на рисунку 3.18.

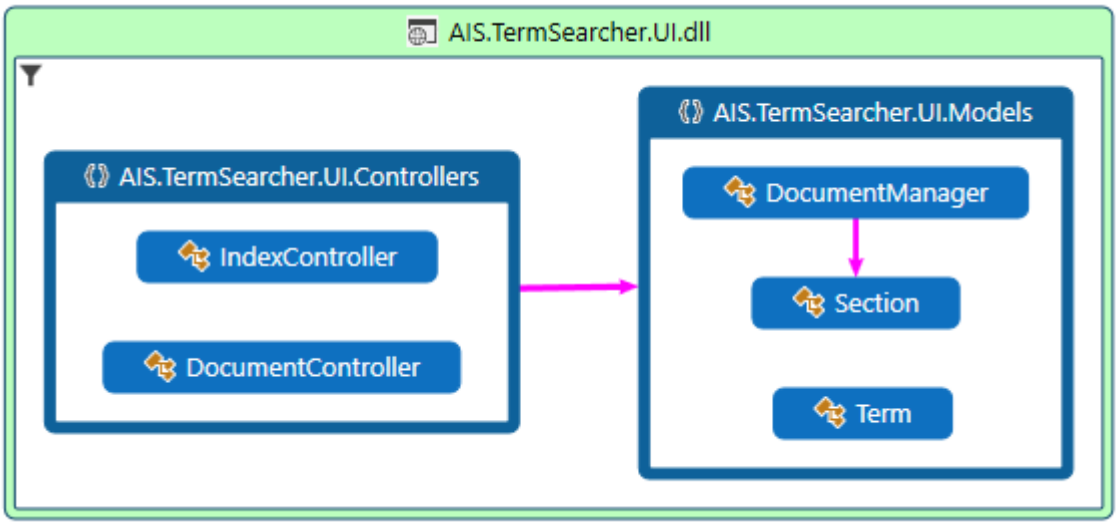


Рисунок 3.18 – Карта коду «Веб інтерфейс для відображення ключових термінів»

Проект побудований на класичній MVC архітектурі. Розроблено дві нових моделі які відповідають за опис даних для візуалізації секцій та термінів.

За їх взаємодію з інтерфейсом відповідає контролер DocumentController.

Таким чином, було розроблено ІС яка містить три логічні модулі: модуль для роботи з електронним документом типу docx, модуль пошуку ключових термінів та модуль користувацького інтерфейсу.

Висновки до розділу

В розділі було визначено обов’язкові користувача без яких система не здатна виконувати подальші кроки самостійно та не обов’язкові функції які виконуються для покращення результату роботи системи або спостереження за проміжними і вихідними даними.

Для розробки основної частини ІС було обрано комбінацію технологій з допомогою яких було розроблено ІС яка містить три логічні модулі: модуль для роботи з електронним документом типу .docx, модуль пошуку ключових термінів та модуль користувацького інтерфейсу.

Для дослідження функціональності ІС було проведено декілька видів тестування, а саме: навантажувальне тестування, тестування стабільності та автоматизоване тестування інтерфейсу користувача. Результати тестування показали, що ІС здатна працювати достатньо швидко з документами різного об’єму, працювати на протязі довгого часу з визначеним очікуваним навантаження та що інтерфейс користувача стабільний та виконує всі необхідні функції.

Розділ 4

Експериментальне тестування інформаційної технології автоматизованого формування семантичної структури інформаційного навчального матеріалу

4.1 Умови експериментального тестування інформаційної технології

Валідність запропонованої інформаційної технології автоматизованого формування семантичної структури інформаційного навчального матеріалу полягає в можливо підтвердити проведенням наступних досліджень для тестової вибірки документів з ІНМ:

1) дослідження функціональності та тестування інформаційної системи – для підтвердження відповідності розробленої системи до запропонованої інформаційної технології автоматизованого формування семантичної структури інформаційного навчального матеріалу й її придатності для проведення подальших досліджень;

2) дослідження синтаксичних параметрів ключових термінів – для визначення частки термінів, що можуть бути відкинуті з множини ключових за аналізом синтаксичних параметрів ключових термінів;

3) дослідження з вибору алгоритму пошуку ключових термінів – для обгрунтування доцільність вибору методу пошуку ключових термінів у навчальних матеріалах;

4) оцінка точності та повноти визначення множин ключових термінів для контенту рубрик ІНМ – шляхом порівняння одержаних автоматично множин із множинами ключових термінів, сформованих авторами ІНМ;

5) оцінка повноти покриття контенту ключовими термінами з створеної множини – шляхом визначення відношення кількості окремих семантичних фрагментів тексту із знайденими ключовими термінами до загальної кількості таких фрагментів тексту.

Для проведення сформованих досліджень було сформовано вибірку з ІНМ з відкритого доступу. Сформована вибірка містить 483 ІНМ, кожен з яких містить визначену автором множину ключових слів. Вибірка використовується для досліджень ефективності пошуку ключових термінів описаним методом. Далі наведено хід та результати досліджень.

Було проведено дослідження функціональності ІС з допомогою декількох видів тестування, а саме: навантажувальне тестування, тестування стабільності та автоматизоване тестування інтерфейсу користувача. Результати навантажувального тестування показали, що ІС здатна працювати достатньо швидко з документами різного об’єму. Результати тестування стабільності показали, що ІС здатна працювати на протязі довгого часу з визначеним очікуваним навантаження. Результати автоматизованого тестування з допомогою інструменту автоматизації Selenium показали, що інтерфейс користувача стабільний та виконує всі необхідні функції. Результати дослідження функціональності ІС наведено у Додатку Г та Додатку Д.

4.2 Дослідження синтаксичних параметрів ключових термінів

Одним із недоліків статистичних методів пошуку ключових термінів є висока оцінка слів та словосполучень які не можуть являтися термінами за синтаксичними ознаками (рисунок 4.1). Яскравим прикладом таких слів можуть бути сполучники такі як «в», «та», «і», «але», «а» тощо. Така проблема яскраво виражена в методі пошуку ключових термінів TF (розглянуто в п. 1.4): сполучники зустрічаються в тексті досить часто тому будуть оцінено високо, проте ключовими термінами являтися вони не можуть. Ця проблема частково вирішується модифікацією методу з використанням оберненої частоти документу – TF-IDF (розглянуто в п. 1.4), проте жодний з статистичних методів не вирішує всі синтаксичні недоліки.

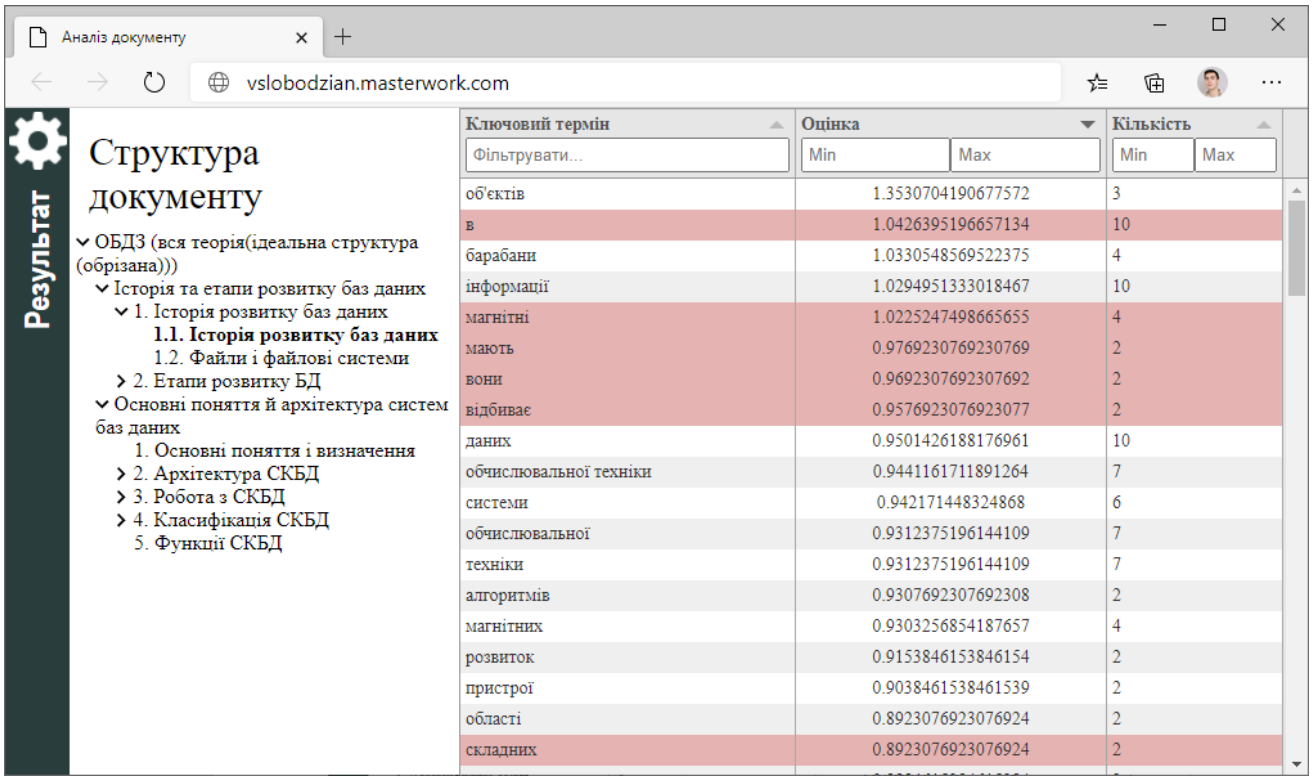


Рисунок 4.1 – Синтаксично некоректні терміни визначені методом дисперсійної оцінки

Тому передбачається, що розглянуті статистичні методи можуть бути вдосконаленні з допомогою доповнення їх певними синтаксичними евристиками.

Проаналізувавши визначені авторами множити ключових термінів в текстових матеріалах тестової вибірки було виявлено наступні закономірності:

– термін, що складається з одного слова може бути лише іменником та не може належати до будь якої іншої частини мови;

– термін, що складається з двох або більше слів має містити хоча б один іменник, та може містити прикметники та частки;

– існують складені терміни, що складаються з інших термінів з’єднаних сполучником.

Опираючись на виявленні закономірності було розроблено метод фільтрації термінів за синтаксичними евристиками.

Дослідження ефективності розробленого методу було проведено на множині ключових термінів знайдених методом дисперсійної оцінки для дисципліни «Організація баз даних та знань». Дослідження показало, що фільтрація множини ключових термінів за синтаксичними параметрами ефективно вилучає некоректні терміни з множини ключових термінів (рисунок 4.2).

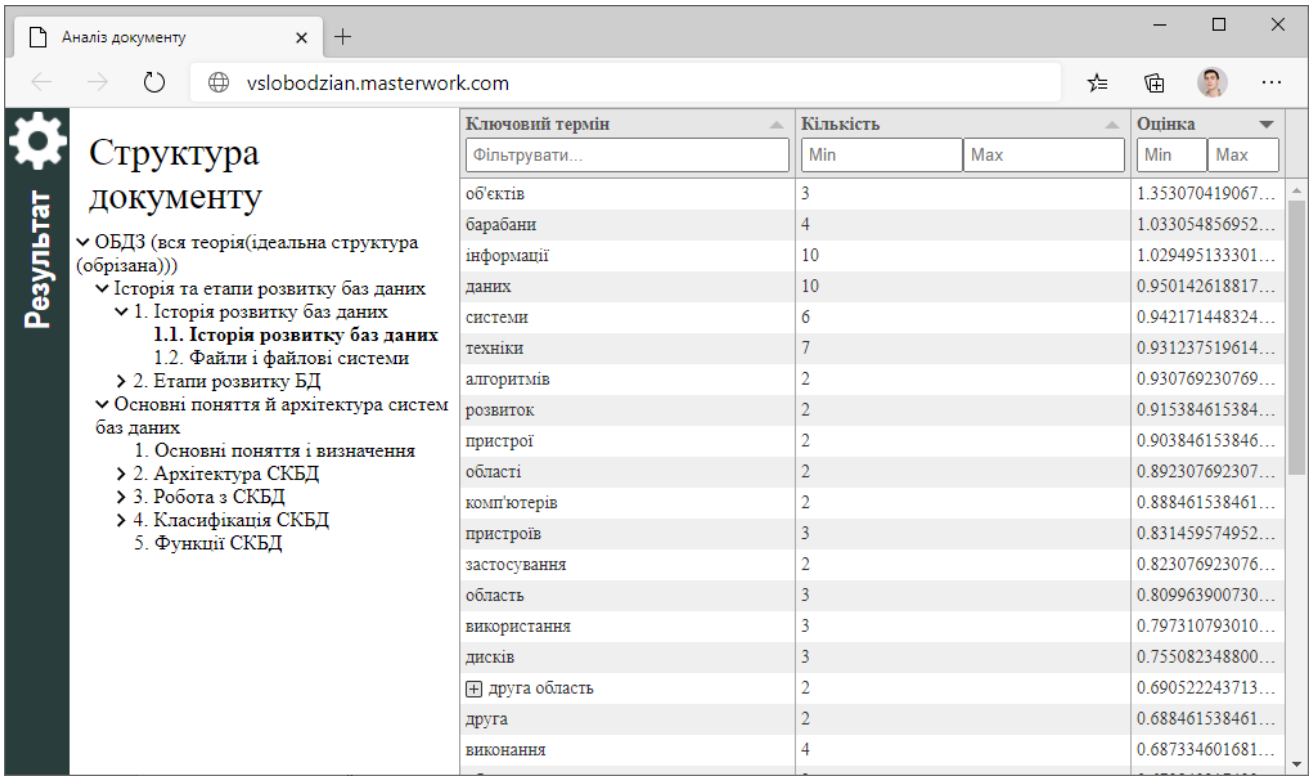


Рисунок 4.2 – Терміни визначені методом дисперсійної оцінки фільтровані за синтаксичними параметрами

Аналіз показав, що в середньому по дисципліні 65.45% оцінених термінів не можуть бути термінами за синтаксичними параметрами. На рисунку 4.3 зображено співвідношення некоректних термінів до коректних в рубрикаціях курсу дисципліни «Організація баз даних і знань».

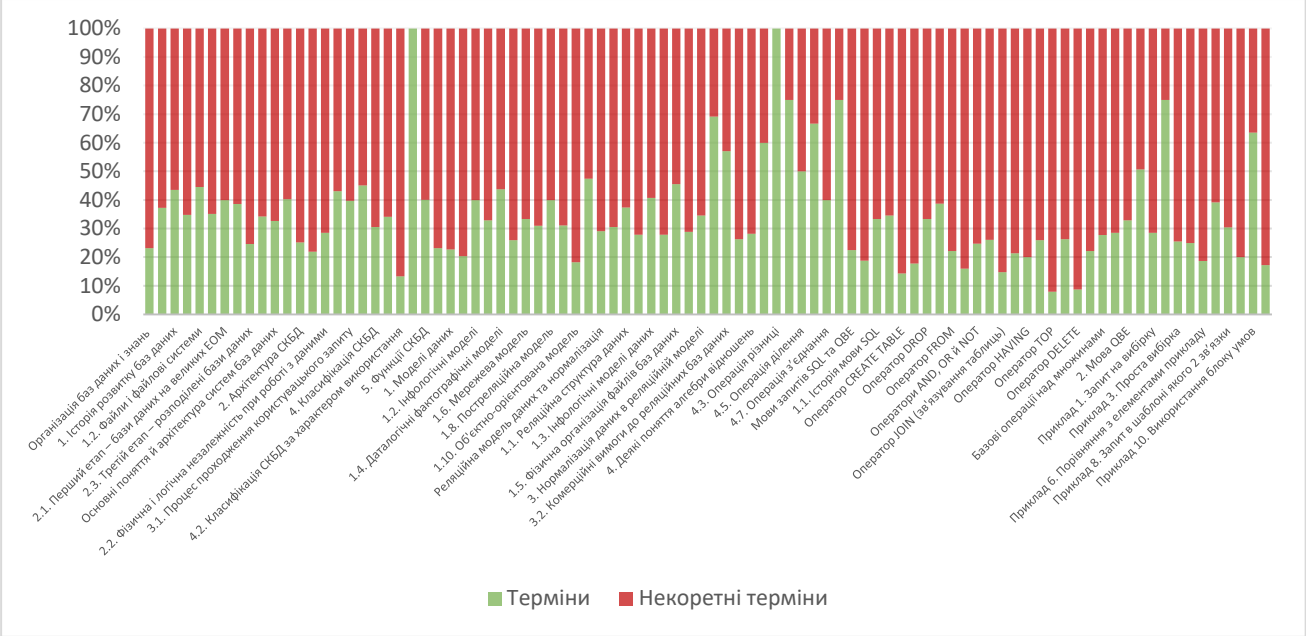


Рисунок 4.3 – Співвідношення некоректних термінів до коректних в рубрикаціях курсу дисципліни «Організація баз даних і знань».

Отож, за результатами проведеного дослідження можна зробити висновок, що використання синтаксичних параметрів для фільтрації множини ключових термінів покращує результуючу множину для розглянутих статистичних методів.

4.3 Дослідження з вибору алгоритму пошуку ключових термінів

Було проведено аналіз визначених в п. 1.4 відомих методів пошуку ключових слів, а саме: методу частотного аналізу TF, методу аналізу TFIDF та методу дисперсійного оцінювання. Під час дослідження порівнбвалися множини кнючових термінів вичначених автоматично та визначених експертами (авторами ІНМ). Для проведення експермент було розроблено експерментальне програмне забезпечення, що реалізовує три розглянуті методи пошуку ключових слів.

Кожна з множин ключових термінів отриматих з допомогою відповідного методу була обмежена до сталого розміру та включена у відповідні множини T_{TF},

T_{TFIDF}, T_{DE}, та порівнювалися з множиною ключових термінів визначених експертом T_{Expert}. Відповідно відсоток термінів, що співпали у досліджуваній множині та множині експертів вважати оцінкою ефективності: чим більший відсоток співпадіння тим ефективнійший метод пошуку ключових термінів у контексті ІНМ.

Таким чином ефективність досліджуваних методів визначається за наступною формулою:

$$E_i = \frac{N_{Ai}}{N_A} \cdot 100\%, \quad (4.1)$$

де N_{Ai} – кількість термінів у експертній множині термінів та сформованій автоматично за i-им методом пошуку ключових слів, що співпали, N_A – кількість термінів у множині термінів визначених експертом.

Досліджено було проведено на 23 ІНМ і для кожного з низ було обчисленно середню ефективність методів пошуку ключовиз термінів. В результаті дослідження було виявленно, що середня ефективність методу частотного аналізу TF склала 25,4%, оцінювання методу аналізу TFIDF – 41,9% та методу дисперсійного оцінювання – 89,2% (рисунок 4.4).

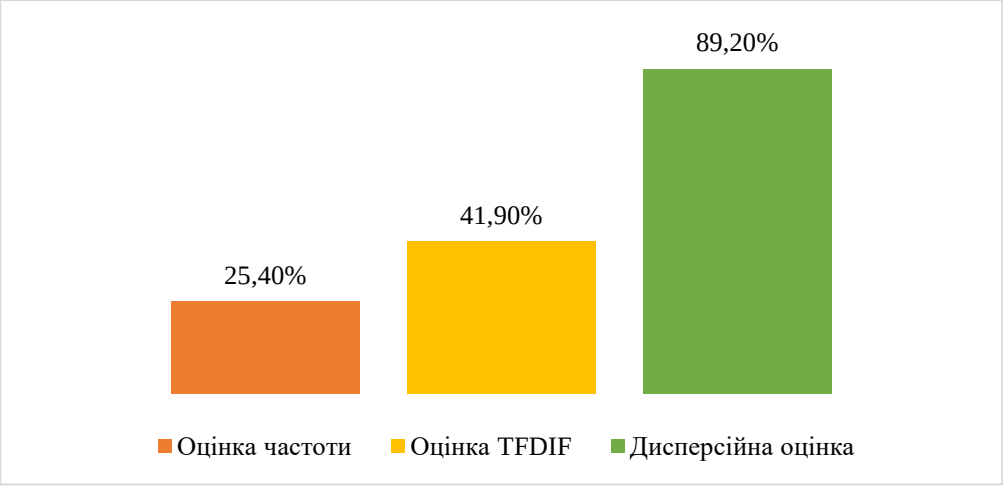


Рисунок 4.4 – Діаграма ефективності методів пошуку ключових термінів

Таким чином, метод частотного аналізу TF показав найнижчу ефективність порівнянно з іншими досліджуваними методами. Водночас метод

дисперсійного оцінювання показав найвищі результати, що довдить доцільність його використання для пошуку ключових слів у ІНМ.

4.4 Дослідження повноти і точності пошуку ключових термінів

Для перевірки здатності визначення множини ключових термінів у рубриках ІНМ з допомогою розробленої ІТ, здійснене порівняльне дослідження експертної множини ключових термінів (визначеною автором ІНМ) із знайденою множиною термінів.

Результати дослідження оцінювалися за показниками повноти та точності. Точність пошуку A (Accuracy) відношення кількості коректних ключових термінів, знайдених автоматично до всієї множини термінів в досліджуваному ІНМ.

Повнота пошуку C (Completeness) – відношення кількості коректних ключових термінів, знайдених автоматично до множини коректних ключових термінів в досліджуваному ІНМ.

Повнота пошуку та точність пошуку обчислюються за наступними формулами:

$$C = \frac{|M_E \cap M_A|}{|M_E|}, A = \frac{|M_E \cap M_A|}{|M_A|}, \quad (4.1)$$

де M_E – множина експертних ключових термінів; M_A – множина автоматично знайдених ключових термінів.

Отже, середня повнота пошуку $\overline{C}$ та середня точність пошуку $\overline{A}$ визначаються за наступними формулами:

$$\overline{C} = \frac{\sum_{i=1}^{k} C_k}{k}, \overline{A} = \frac{\sum_{i=1}^{k} A_k}{k}, \quad (4.2)$$

де k – кількість ІНМ у експерминтальній вибірці.

Під час дослідження приймаються до уваги підмножини M_E та M_A: множина спільних у M_A та M_E ключових термінів ($M_A \cap M_E$), множина тільки автоматично знайдених ключових термінів ($M_A \setminus M_E$) та множина тільки

експертних ключових термінів ($M_E \setminus M_A$. Для того щоб визначити співвідношення між множинами визначено показники D_A, D_S, D_E,

$$D_A = \frac{|M_A \setminus M_E|}{|M_E \cup M_A|}, D_S = \frac{|M_E \cap M_A|}{|M_E \cup M_A|}, D_E = \frac{|M_E \setminus M_A|}{|M_E \cup M_A|}, \quad (4.3)$$

де D_A – відношення кількості термінів присутніх лише у автоматично визначеній множині термінів до кількості унікальних термінів в об'єднаній множині; D_S – відношення кількості термінів які спільні для обох множин до кількості унікальних термінів в об'єднаній множині термінів; D_E – відношення кількості термінів присутніх лише у експертній множині термінів до кількості унікальних термінів в об'єднаній множині.

Результати проведеного дослідження представленні в таблиці 4.1 й зображено на рисунку 4.5.

Таблиця 4.1 – Результати дослідження повноти і точності пошуку ключових термінів

Показник	**Значення**
Середня точність пошуку $\overline{A}$	0,732
Середня повнота пошуку $\overline{C}$	0,697
Мінімальна точність пошуку A_{min}	0,512
Мінімальна повнота пошуку C_{min}	0,581
Максимальна точність пошуку A_{max}	0,929
Максимальна повнота пошуку дорівнює C_{max}	1,000

Рисунок 4.5 – Результати обрахунку точності та повноти пошуку ключових термінів

Середні показники для D_A, D_S та D_E обраховані за (4.3) та узагальнено аналогічно до (4.2) відповідають: $\overline{D_S}$= 60,10%, $\overline{D_A}$= 13,79%, $\overline{D_E}$= 26,11%, що відображено на рисунку 4.6.

Рисунок 4.6 – Результати аналізу входжень ключових термінів до результуючих множин

Аналіз результатів показав, що не завжди віднустнівть термінів визначених експертом може бути розіцінено як недолік досліджуваної ІТ. Адже автор може оцінювати ІНМ суб’єктивно і може виділити термін, який насправді не являється ключовим. Водочас не ключовий термін може бути розкритий надмірно і таким чином потрапити у множину автоматично визначених термінів, не зважаючи на свою другорядність. Тобто відмінність між множинами експертних термінів та автоматично визначених термінів здебільшого має суб’єктивний характер.

Отож, результати дослідження демонструють, що досліджувана ІТ може бути використана для одержання множин ключових термінів, релевантних до множин ключових термінів визначених авторами ІНМ. Виявленні неточності

пошуку обґрунтовуються людським фактором, а неточності повноти пошуку можуть бути виправлені шляхом коригування множини термінів передбаченими засобами ІТ.

4.5 Дослідження повноти покриття контенту ключовими термінами

Для того щоб визначити відсоток контенту ІНМ, що містить терміни включені результуючої множини ключових термінів було проведено дослідження повноти покриття контенту ІНМ ключовими термінами.

Для позначення повноти покриття ІНМ ключовими термінами узято відношення K_H, K_P і K_S кількості текстових блоків (відповідно для дочірніх рубрик N_H, абзаців N_P і речень N_S), які містять результуючі ключові терміни, до загальної кількості цих текстових блоків (відповідно N_{HH}, N_{PP} і N_{SS}) у межах кожного ІНМ:

$$K_H = \frac{N_H}{N_{HH}}, \quad K_P = \frac{N_P}{N_{PP}}, \quad K_S = \frac{N_S}{N_{SS}}, \tag{4.4}$$

$$\overline{K_H} = \frac{\sum_{i=1}^{h} K_{H,i}}{h}, \quad \overline{K_P} = \frac{\sum_{i=1}^{h} K_{P,i}}{h}, \quad \overline{K_S} = \frac{\sum_{i=1}^{h} K_{S,i}}{h}, \tag{4.5}$$

де $\overline{K_H}$, $\overline{K_P}$ та $\overline{K_S}$ – середні значення для відповідно K_H, K_P та K_S для експериментальної вибірки; h – кількість опрацьованих документів ІНМ у вибірці.

Для дослідження тестових блоків було використано додаткові показники: L_H, L_P та L_S для відношення кількості слів у текстових блоках (відповідно дочірніх рубрик, абзаців і речень), що містять результуючі ключові терміни, до загальної кількості слів у ІНМ F_W. Значення додаткових показників L_H, L_P та L_S та їхні середні значення $\overline{L_H}$, $\overline{L_P}$ та $\overline{L_S}$за експериментальною вибіркою обчислюються наступним чином:

$$L_H = \frac{F_H}{F_W}, \quad L_P = \frac{F_P}{F_W}, \quad L_S = \frac{F_S}{F_W}, \tag{4.6}$$

$$\overline{L_H} = \frac{\sum_{i=1}^{h} L_{H,i}}{h}, \quad \overline{L_P} = \frac{\sum_{i=1}^{h} L_{P,i}}{h}, \quad \overline{L_S} = \frac{\sum_{i=1}^{h} L_{S,i}}{h}, \tag{4.7}$$

де F_H, F_P та F_S – кількість слів відповідно у дочірніх рубриках, абзацах та реченнях, для яких знайдені ключові терміни.

Усього по експериментальній вибірці з h = 83 документів ІНМ за (4.5) одержано середні значення показників покриття текстового контенту $\overline{K_H}$ = 100%, $\overline{K_P}$ = 81,2% та $\overline{K_S}$ = 87,3%. Граничні значення цих показників відповідно склали: $K_{H,min}$ = 100%, $K_{H,max}$ = 100%, $K_{P,min}$ = 60,1%, $K_{P,max}$ = 100%, $K_{S,min}$ = 66,5%, $K_{S,max}$ = 100%. Відповідні середні показники покриття контенту за кількістю слів відповідно до (4.7) отримано $\overline{L_H}$ = 100%, $\overline{L_P}$ = 88,1% та $\overline{L_S}$ = 90,1%, при граничних значеннях: $L_{H,min}$ = 100%, $L_{H,max}$ = 100%, $L_{P,min}$ = 69,7%, $L_{P,max}$ = 100%, $L_{S,min}$ = 69,7%, $L_{S,max}$ = 100%. Результати аналізу покриття текстового контенту ключовими термінами візуально зображено на рисунку 4.7.

Рисунок 4.7 – Середні значення покриття текстового контенту ключовими термінами в рубриках, абзацах та реченнях

Таким чином, терміни з множини автоматично визначених ключових термінів містяться у всіх дочірніх рубриках ІНМ, в середньому в 81,2% абзаців, або в 87,3% речень. Підвищення значення граничної щільності ключових термінів Q мможе збільшити отриманні значення, однак може призвести до появи малозначущих термінів в результуючій множині, які не будуть мати ключового значення.

Отож за одержаними результатами можна зробити висновок, що ключові терміни здебільшого відсутні в абзацах, що складаються з малої кількості слів. Відповідний висновок можна зробити стосовно речень. Такі результати є природніми, оскільки не всі речення чи абзаци мають семантичний зміс, а несуть лише зв'язувальний характер.

4.6 Вимоги до вхідних даних інформаційної технології

Основною вимогою до ІНМ, що є вхідними даними для ІТ, являється подання тестового матеріалу у вигляді ASCII символів. ІТ не передбачає роботу з текстом представленим на рисунках, тощо. Це єдина обов'язкова вимога до формування ІНМ, дотримання якої впливає на коректність роботи ІТ.

Наступні додаткові вимоги до формування ІНМ не є обов'язковими, проте впливають на результат роботи ІТ.

Використання різних назв термінів може привести до некоректної оцінки важливості терміну. Термін може буде представлений в різному вигляді в різних фрагментах тексту, наприклад термін «База даних» може бути представлений абревіатурою «БД» або «DB». Таке представлення є допустимим, ІТ буде розцінювати такі терміни як різні, тому оцінка важливості буде вираховуватися для кожного з термінів окремо. Використання єдиної назви терміну в ІНМ дозволить покращити його оцінку.

Використання стилів форматування тексту може бути використано для покращення оцінки термінів, які були виділені автором ІНМ. Таким чином некоректне використання стилів може призвести до небажаного збільшення чи зменшення оцінки важливості певного терміну і з іншого боку при коректному використанні стилів оцінка може бути скорегована для покращення результатів.

Формування рубрик ІНМ дозволить формувати множини ключових термінів для конкретних розділів, що дозволяє показати більш повну семантичну структуру. Проте у випадку відсутності рубрикації буде автоматично

сформовано кореневу рубрику і в контексті цієї рубрики буде формуватися множина ключових слів.

Не правильне використання розділових знаків та регістру літер може призвести до некоректного формування множин абзаців, речень та слів, що також може призвести до некоректної оцінки важливості термінів. Для прикладу некоректне використання крапки для скорочення терміну може бути розцінене як закінчення речення, що призведе до зміни множини термінів кандидатів і відповідно призведе до змін результуючої множини термінів.

Описані додаткові вимоги до ІНМ, рекомендується використовувати не тільки задля покращення результатів роботи розробленої ІТ, а й в цілому для покращення сприйняття ІНМ. Так як некоректне форматування може погіршити сприйняття ІНМ, а використанні різних назв термінів може погіршити його запам’ятовування та призвести до не однозначного сприйняття матеріалу.

Таким чином для роботи ІТ є необхідним подання тестового матеріалу у вигляді ASCII символів, а для покращення ефективності ІТ може бути виконана попередня обробка тексту, що наближає ІНМ до описаних вище рекомендацій і здебільшого полягає у виправленні стилістичних помилок, та помилок форматування, уникненні неоднозначних назв термінів, некоректного використання розділових знаків та регістру літер.

Висновки до розділу

В розділі було проведено дослідження функціональності ІС, яке показало, що досліджувана ІС здатна працювати коректно та стабільний в умовах очікуваного навантаження при цьому виконуючи всі необхідні функції для забезпечення повноцінної роботи ІТ.

Результат дослідження використання синтаксичних параметрів для фільтрації множини ключових термінів показав ефективність для розглянутих статистичних методів.

Дослідження алгоритмів пошуку ключових термінів у ІНМ показав, що метод дисперсійного оцінювання демонструє найвищі результати, таким чином доводячи доцільність його використання для пошуку ключових слів у ІНМ.

Дослідження повноти і точності пошуку ключових термінів демонструють, що досліджувана ІТ може бути використана для одержання релевантних множин ключових термінів визначених авторами ІНМ. При цьому було обґрунтовано неточності пошуку ключових термінів. Дослідження повноти показало, що терміни здебільшого відсутні в абзацах та словах, що складаються з малої кількості слів оскільки деякі речення чи абзаци не є симантично важливими.

Було визначено вимоги до вхідних даних ІТ. Єдиною обов’язковою вимогою є подання тестового матеріалу у вигляді ASCII символів. А не обов’язковими вимогами являються попередня обробка тексту, що здебільшого полягає у виправленні стилістичних помилок, та помилок форматування, уникненні неоднозначних назв термінів та некоректного використання розділових знаків та регістру літер.

Загальні висновки

Дипломна робота магістра розв’язує науково-технічну задачу створення ІТ автоматизованого формування семантичної структури інформаційного навчального матеріалу.

У рамках роботи поставлені та вирішені такі завдання:

1. Проведено аналіз сучасних рішень з автоматизованого моделювання семантичної структури інформаційного навчального матеріалу з метою визначення нерозв’язаних задач з побудови семантичної структури інформаційних навчальних матеріалів.

2. Удосконалено інформаційну модель семантичної структури інформаційного навчального матеріалу, яка дозволяє зберігати всі елементи та атрибути інформаційного навчального матеріалу, необхідні для побудови його семантичної структури.

3. Удосконалено метод формування рубрикації інформаційного навчального матеріалу з метою одержання ієрархічної структури цифрового документу інформаційного навчального матеріалу та визначення відповідних елементам структури фрагментів контенту.

4. Удосконалено метод пошуку ключових термінів у рубриках інформаційного навчального матеріалу, що дозволяє одержувати сортовані за рівнем семантичної значущості множини ключових слів та словосполучень.

5. Розроблено інформаційну технологію автоматизованого формування семантичної структури інформаційного навчального матеріалу, яка дозволяє за вхідними даними в вигляді цифрового файлу електронного документу одержувати семантичну структуру інформаційного навчального матеріалу.

6. Досліджено практичну ефективність розробленої інформаційної технології шляхом створення відповідної експериментальної інформаційної системи та її тестування.

При створенні інформаційної технології автоматизованого формування семантичної структури інформаційного навчального матеріалу необхідно використати метод дисперсійного оцінювання слів тексту, який дозволяє одержати релевантну множину ключових слів документу.

В результаті проведеної роботи були отримані такі результати:

– Набула подальшого розвитку інформаційна модель семантичної структури інформаційного навчального матеріалу, яка відрізняється тим, що містить елементи та атрибути семантичної структури інформаційного навчального матеріалу, необхідні для роботи інформаційної технології

– Набув подальшого розвитку метод формування рубрикації інформаційного навчального матеріалу, який відрізняється тим, що формує ієрархічну структуру рубрикації інформаційного навчального матеріалу та обмежує відповідні рубрикам фрагменти контенту.

– Набув подальшого розвитку метод пошуку ключових термінів у рубриках інформаційного навчального матеріалу, який відрізняється тим, що використовує дисперсійне оцінювання слів і словосполучень та визначає оцінку їх семантичної важливості в рубриці.

– Вперше розроблено інформаційну технологію автоматизованого формування семантичної структури інформаційного навчального матеріалу, яка забезпечує автоматизоване одержування семантичної структури інформаційного навчального матеріалу за цифровим файлом електронного документу.

Розроблений для дослідження практичної ефективності створеної інформаційної технології інформаційна система виконує наступні основні функції: зчитування та парсинг електронного ІНМ у форматі .docx; формування ієрархічної структури рубрикації інформаційного навчального матеріалу та обмеження відповідних рубрикам фрагментів контенту; пошук ключових термінів у рубриках інформаційного навчального матеріалу з використанням дисперсійного оцінювання слів і словосполучень та визначення оцінки їх

семантичної важливості в рубриках; побудова семантичної структури інформаційного навчального матеріалу.

Виконано експериментальну перевірку ІТ автоматизованого формування семантичної структури інформаційного навчального матеріалу шляхом розробки й використання відповідної ІС. Результати експериментального тестування запропонованої ІТ довели її спроможність розв'язувати поставлені задачі. При цьому, середня точність пошуку ключових термінів складає 73,2%, середня повнота пошуку – 69,7%. Знайдені ключові терміни містяться в середньому в 81,2% абзаців, або в 87,3% речень.

Вищенаведені положення дипломної роботи магістра автором висвітлено в 5 наукових *публікаціях* [35, 36, 37, 38, 39], в тому числі 1 стаття – в фаховому виданні [39], включеному в перелік МОН України. Прийнято участь у 4 конференціях: Конференція молодих вчених «Сучасні технології в механіці», Всеукраїнська науково-практична конференція «Інтелектуальний потенціал – 2018», XI Всеукраїнської науково-практична конференція «Актуальні проблеми комп'ютерних наук АПКН-2019», XII Всеукраїнської науково-практична конференція «Актуальні проблеми комп'ютерних наук АПКН-2020».

Перелік посилань

1. Освіта як важлива характеристика індивідуального пропонування праці [Електронний ресурс]. – Режим доступу: https://pidru4niki.com/15660212/ekonomika/osvita_vazhliva_harakteristika_individualnogo_proponuvannya_pratsi

2. Конституція України, Розділ II, Стаття 53 [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/254к/96-вр#Text

3. Дистанційна освіта [Електронний ресурс]. – Режим доступу: http://vnz.org.ua/dystantsijna-osvita/pro

4. Вікіпедія [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Вікіпедія#Мовні_розділи

5. Ільїна В. М. Інформаційні ресурси при використанні дистанційного навчання у післядипломному удосконаленні лікарів / В.М.Ільїна, І.В.Кочін, О.М.Акулова, О.О.Гайволя, Д.О.Трошин [Електронний ресурс]. – Режим доступу: https://zounb.zp.ua/node/1144

6. Коробов Є. Т. Навчальний матеріал та його структура [Електронний ресурс]. – Режим доступу: http://www.rusnauka.com/19_DSN_2010/Pedagogica/69745.doc.htm.

7. Бєляк О. М. Структурування навчальної інформації як складова підготовки студентів немовних спеціальностей / О. М. Бєляк // «Наука і освіта». – 2014. – № 3. – С. 12-15.

8. Коробов Є. Т. Структура начальної інформації [Електронний ресурс]. – Режим доступу: www.rusnauka.com/8_NMIV_2013/Pedagogica/5_131289.doc.htm

9. Мазурець О. В. Інформаційна технологія автоматизованого структурування навчальних матеріалів та створення тестів для адаптивного контролю рівня знань / Мазурець О. В. // Дис. канд. техн. наук: спец. 05.13.05 «Інформаційні технології». – Тернопільський національний економічний університет. – Тернопіль. – 2019. – 181 с.

10. Снитюк В. Е. Интеллектуальное управление оцениванием знаний / В. Е. Снитюк, К. Н. Юрченко // Черкассы, 2013. – 262 с.

11. Рабчевский Е. А. Автоматическое построение онтологий на основе лексико-синтаксических шаблонов для информационного поиска / Е.А. Рабчевский // Труды XI Всеросс. науч. конф. «Электронные библиотеки: перспективные методы и технологии, электронные коллекции». – Петрозаводск, 2009.

12. Орлова Є. В. Аналіз методів і моделей автоматичної побудови онтологій [Електронний ресурс]. – Режим доступу: http://masters.donntu.org/2012/iii /orlova /diss/indexu.htm#p83

13. Найханова Л. В. Методы и модели автоматического построения онтологий на основе генетического и автоматного программирования: Автореф. дис. докт. тех. наук. / Найханова Л. В. // Красноярск, 2008. – 36 с.

14. Термін у системі професійного мовлення [Електронний ресурс]. – Режим доступу: https://pidru4niki.com/1727101247606/dokumentoznavstvo/termin_sistemi_profesiynogo_movlennya

15. Ключове слово [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org /wiki/Ключове_слово

16. Term Frequency - Inverse Document Frequency [Електронний ресурс]. – Режим доступу: https://pahulpreet86.github.io/term-frequency-inverse-document-frequency

17. Demystify TF-IDF in Indexing and Ranking [Електронний ресурс] – Режим доступу: https://ted-mei.medium.com/demystify-tf-idf-in-indexing-and-ranking-5c3ae88c3fa0

18. Бармак О. В. Методи автоматизації визначення семантичних термінів у навчальних матеріалах / О. В. Бармак, О. В. Мазурець // Вісник Хмельницького національного університету. Технічні науки. - 2015. - № 2. - С. 209-213.

19. Rich Text Format [Електронний ресурс] – https://en.wikipedia.org/wiki/Rich_Text_Format

20. The Styles advantage in Word [Електронний ресурс]. – Режим доспуту: https://support.microsoft.com/en-us/office/the-styles-advantage-in-word-b4a6372f-188c-93cb-831b-c4dd0cb3a881

21. Insert a table of contents [Електронний ресурс]. – Режим доспуту: https://support.microsoft.com/en-us/office/insert-a-table-of-contents-882e8564-0edb-435e-84b5-1d8552ccf0c0

22. Use the Navigation pane in Word [Електронний ресурс]. – Режим доспуту: https://support.microsoft.com/en-us/office/use-the-navigation-pane-in-word-394787be-bca7-459b-894e-3f8511515e55

23. Considerations for server-side Automation of Office [Електронний ресурс]. – Режим доступу: https://support.microsoft.com/en-us/help/257757/considerations-for-server-side-automation-of-office

24. DocumentFormat.OpenXml [Електронний ресурс] – Режим доступу: https://www.nuget.org/packages/DocumentFormat.OpenXml/.

25. SpiteDoc [Електронний ресурс] – Режим доспуту: https://www.e-iceblue.com/Introduce/word-for-net-introduce.html

26. DocX for creates or modifies Microsoft Word files [Електронний ресурс]. – Режим доступу: https://github.com/xceedsoftware/DocX

27. Retrieve Style Names of all TextRanges in a Word Document in C#, VB.NET [Електронний ресурс]. – Режим доступу: https://www.e-iceblue.com/Tutorials/Spire.Doc/Spire.Doc-Program-Guide/Text/Retrieve

28. Word object model overview [Електронний ресурс]. – Режим доспуту: https://docs.microsoft.com/ru-ru/visualstudio/vsto/word-object-model-overview

29. Символ нового рядка [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Символ_нового_рядка

30. Типи речень [Електронний ресурс]. – Режим доспуту: https://pidru4niki.com/18800 41340571/dokumentoznavstvo/tipi_rechen

31. Частини мови та їх властивості [Електронний ресурс]. – Режим доспуту: https:// znoclub.com/mova-ta-literatura/546-chastini-movi.html -Style-Names-of-all-TextRanges-in-a-Word-Document-in-C-VB.NET.html

32. Comparison of regular-expression engines [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Comparison_of_regular-expression_engines

33. Рейтинг мов програмування 2020 [Електронний ресурс]. – Режим доступу: https://dou.ua/lenta/articles/language-rating-jan-2020/?from=doufp

34. .NET [Електронний ресурс]. – Режим доступу: https://dotnet.microsoft.com

35. Мазурець О. В. Використання спеціалізованих програмних розширень для автоматизації роботи з цифровими документами навчальних матеріалів / О. В. Мазурець, О. В. Ковальчук, В. О. Слободзян // Науковий журнал «Вісник Хмельницького національного університету» серія: Технічні науки. Хмельницький, 2018, №1. – С.61-69.

36. Слободзян В. О. Аналіз сучасних спеціалізованих програмних розширень для автоматизації роботи з цифровими документами навчальних матеріалів / В. О. Слободзян, О. В. Мазурець // Сучасні технології в механіці: Збірник наукових праць. Хмельницький – 2018. – С.184-191.

37. Ковальчук О. В. Використання програмного розширення spire.doc для автоматизації роботи з цифровими документами / О. В. Ковальчук, Г. А. Білоус, В. О. Слободзян // Збірник наукових праць за матеріалами XI Всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2019» – Хмельницький, 2019, Т.1. – С.116-122.

38. Мазурець О. В. Метод формального опису елементів моделей автоматизованого формування тестових завдань / О. В. Мазурець, О. В. Ковальчук, В. О. Слободзян, Г. А. Білоус // Збірник наукових праць за матеріалами Всеукраїнської науково-практичної конференції «Інтелектуальний потенціал – 2018». – Хмельницький, 2018, Ч.1. – С.51-56.

39. Слободзян В. О. Аналіз результатів автоматизованого пошуку ключових термінів у навчальних матеріалах / Слободзян В. О., Мазурець О. В. // Збірник наукових праць за матеріалами XII Всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2020». – Хмельницький, 2020 – С.269-274.

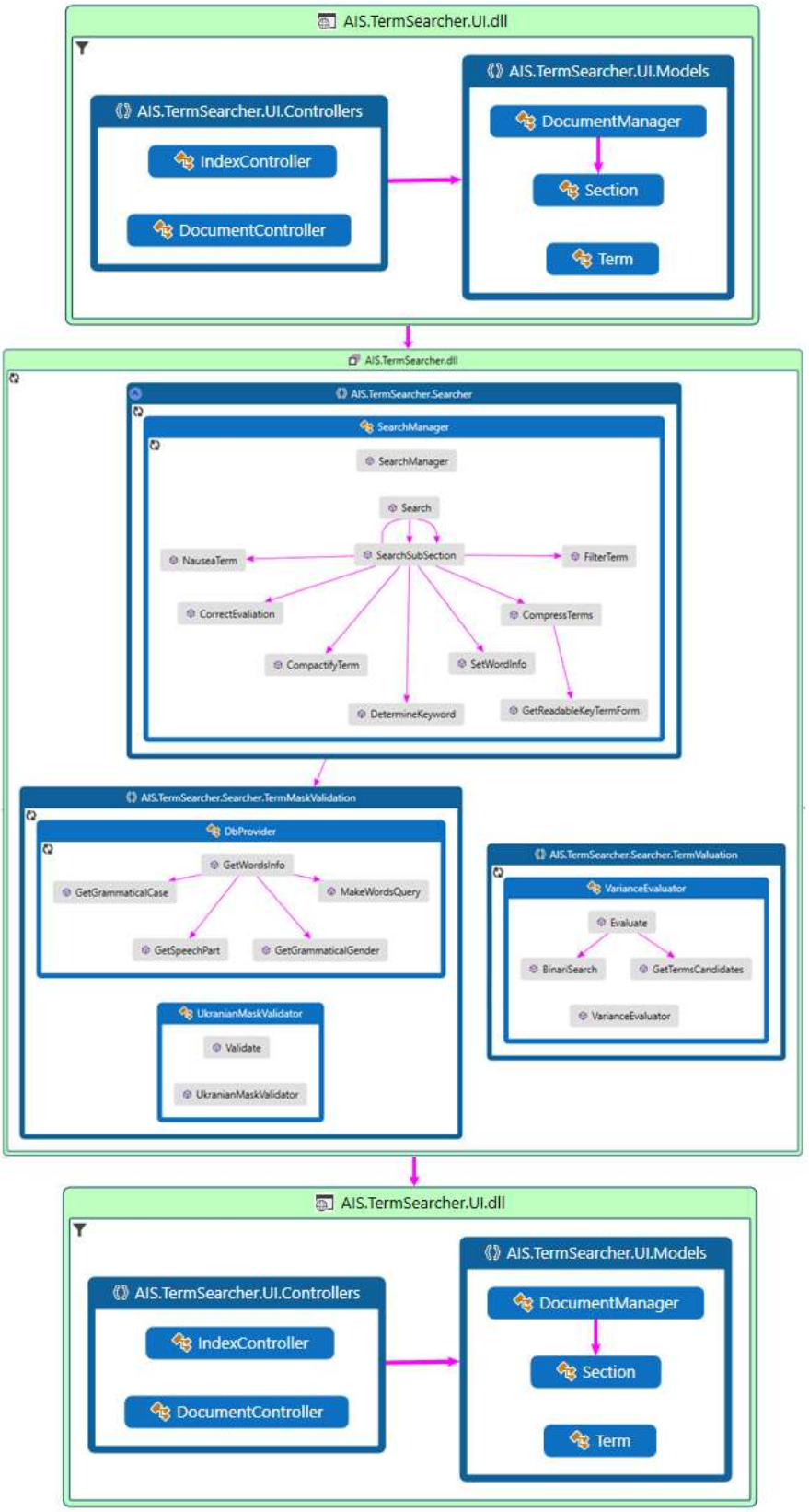
ДОДАТКИ

Додаток А

Розгорнута структура класів автоматизованої системи

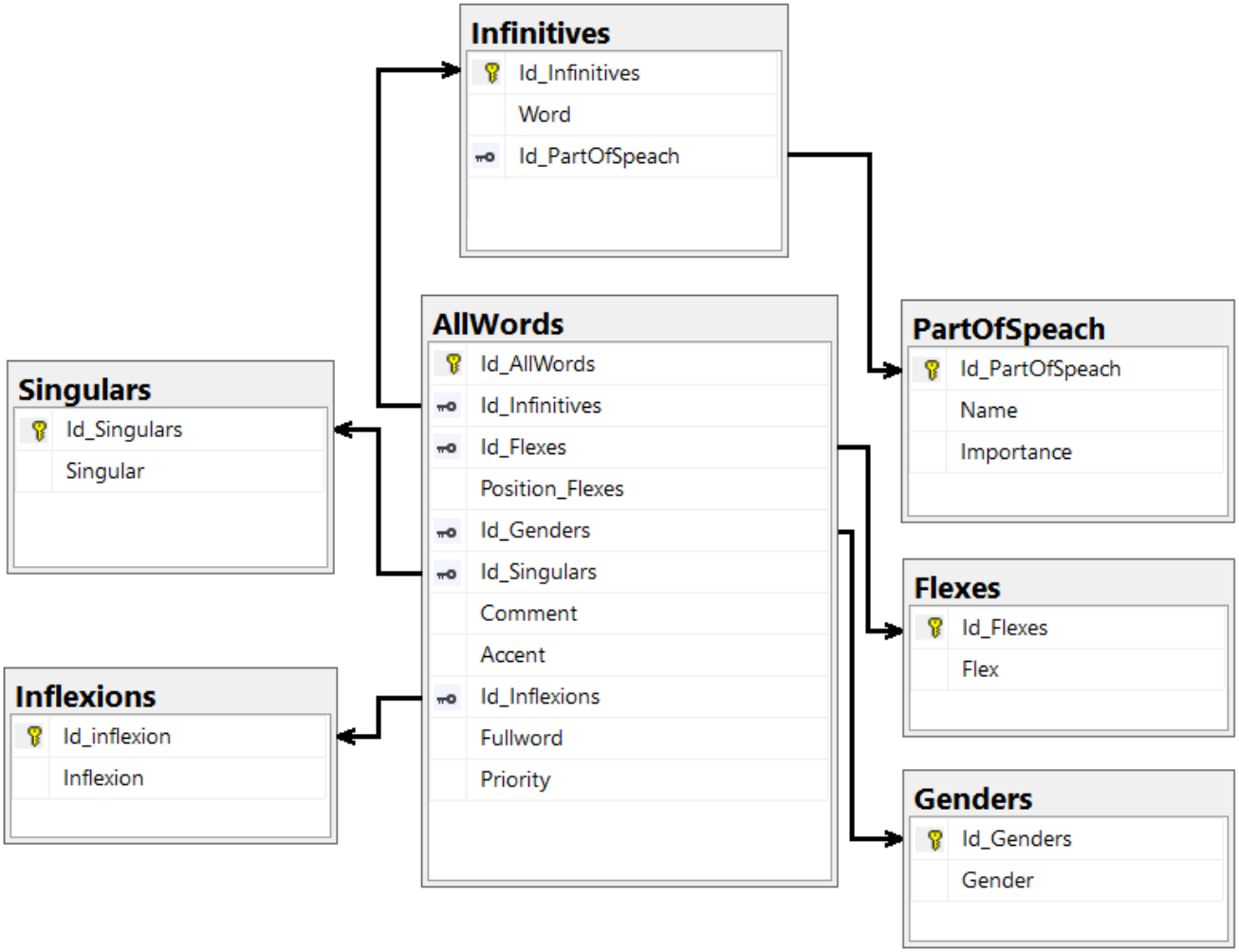
Додаток Б

Повна мапа коду інформаційної системи



Додаток В

Даталогічна модель бази даних корпусу українських слів



Додаток Г
Тестування інформаційної системи

Для дослідження було проведено декілька видів тестування, а саме: навантажувальне тестування, тестування стабільності та автоматизоване тестування інтерфейсу користувача.

Навантажувальне тестування (Load Testing) – це проста форма тестування продуктивності. Таке тестування здебільшого проводиться з метою оцінки поведінки ІС із визначеним очікуваним навантаженням. Для прикладу, таким навантаженням може бути розмір файлів, що обробляються ІС. Результатом навантажувального тестування може бути час відклику важливих операцій ІС.

Для даного типу тестування було обрано три документа різного об’єму.

ІНМ «Нейронна мережа перцептрон» з дисципліни «Розробка систем штучного інтелекту». Властивості документу представленні в таблиці Г.1.

Таблиця Г.1 – Властивості документу «Нейронна мережа перцептрон»

Назва документу	*Нейронна мережа перцептрон*
Кількість слів	*3068*
Кількість сторінок	*5*
Розмір файлу	*2.17 Мб*

Результат навантажувального тестування для досліджуваного документу представлений на рисунку Г.1.

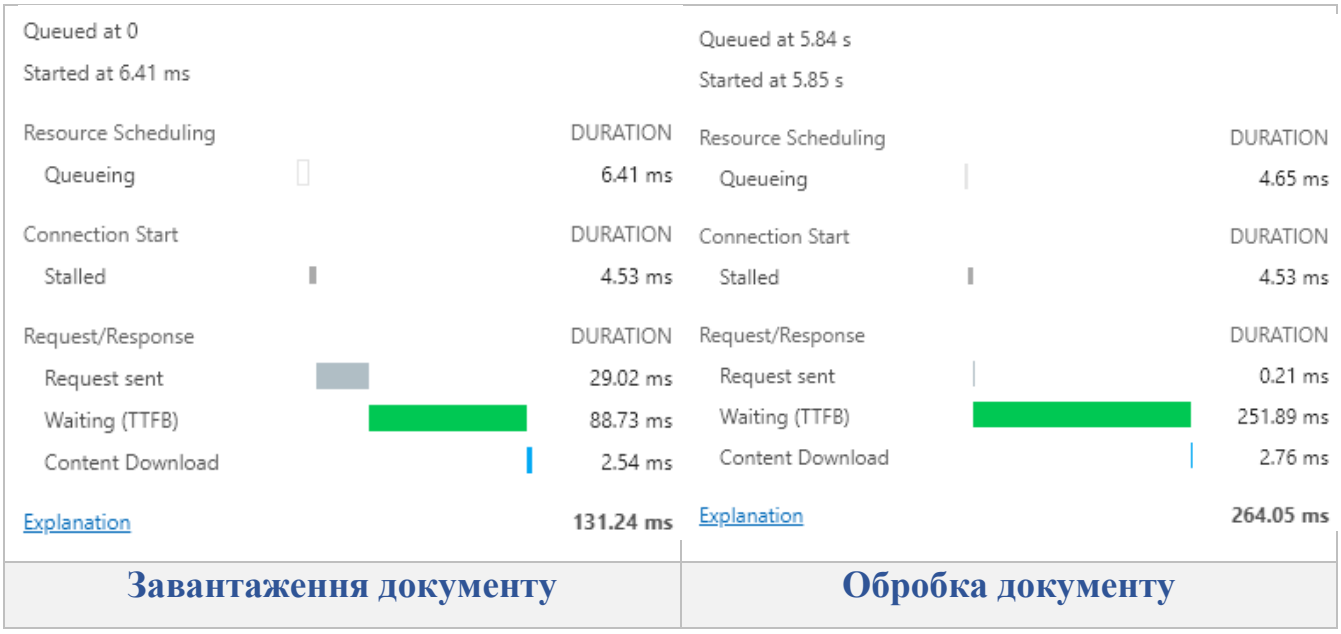


Рисунок Г.1 – Результати навантажувального тестування для ІНМ «Нейронна мережа перцептрон» з дисципліни «Розробка систем штучного інтелекту»

ІНМ «Історія та етапи розвитку баз даних» з дисципліни «Організація баз даних і знань». Властивості документу представленні в таблиці Г.2.

Таблиця Г.2 – Властивості документу «Історія та етапи розвитку баз даних»

Назва документу	*Історія та етапи розвитку баз даних*
Кількість слів	*15589*
Кількість сторінок	*9*
Розмір файлу	*0.45 Мб*

Результат навантажувального тестування для досліджуваного документу представлений на рисунку Г.2.

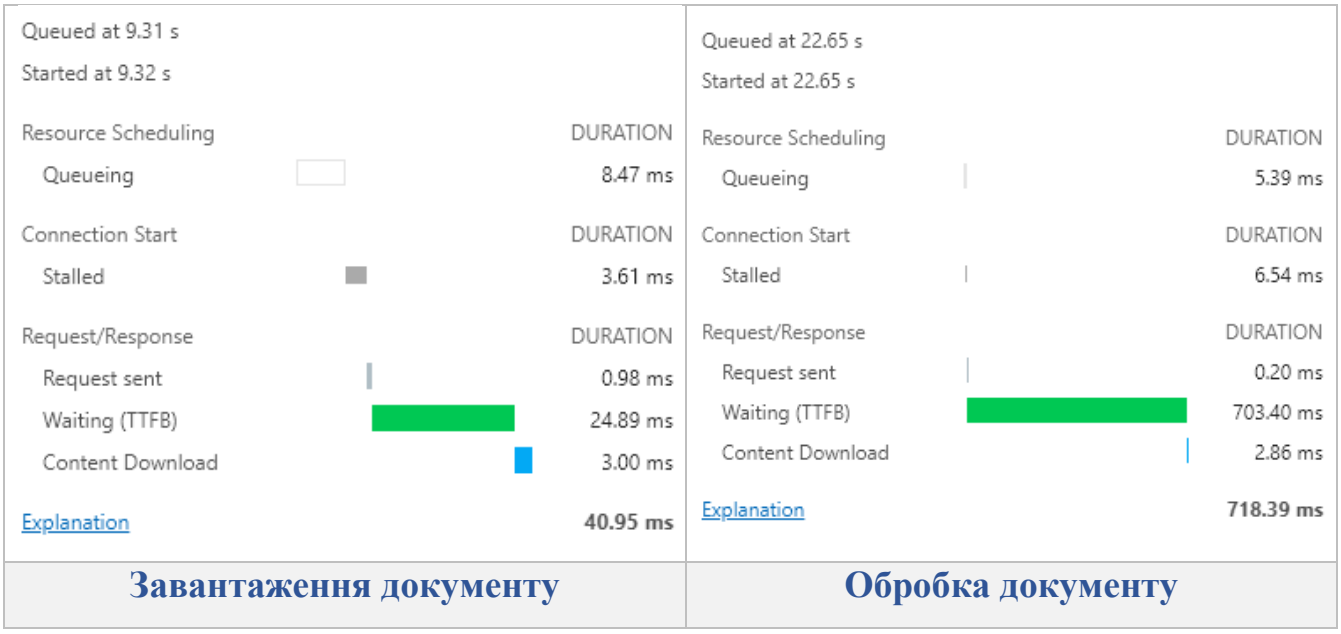


Рисунок Г.2 – Результати навантажувального тестування для ІНМ «Історія та етапи розвитку баз даних» з дисципліни «Організація баз даних і знань»

ІНМ повного курсу дисципліни «Організація баз даних і знань». Властивості документу представленні в таблиці Г.3.

Таблиця Г.3 – Властивості документу «Нейронна мережа перцептрон»

Назва документу	*Нейронна мережа перцептрон*
Кількість слів	*23194*
Кількість сторінок	*78*
Розмір файлу	*0.385 Мб*

Результат навантажувального тестування для досліджуваного документу представлений на рисунку Г.3.

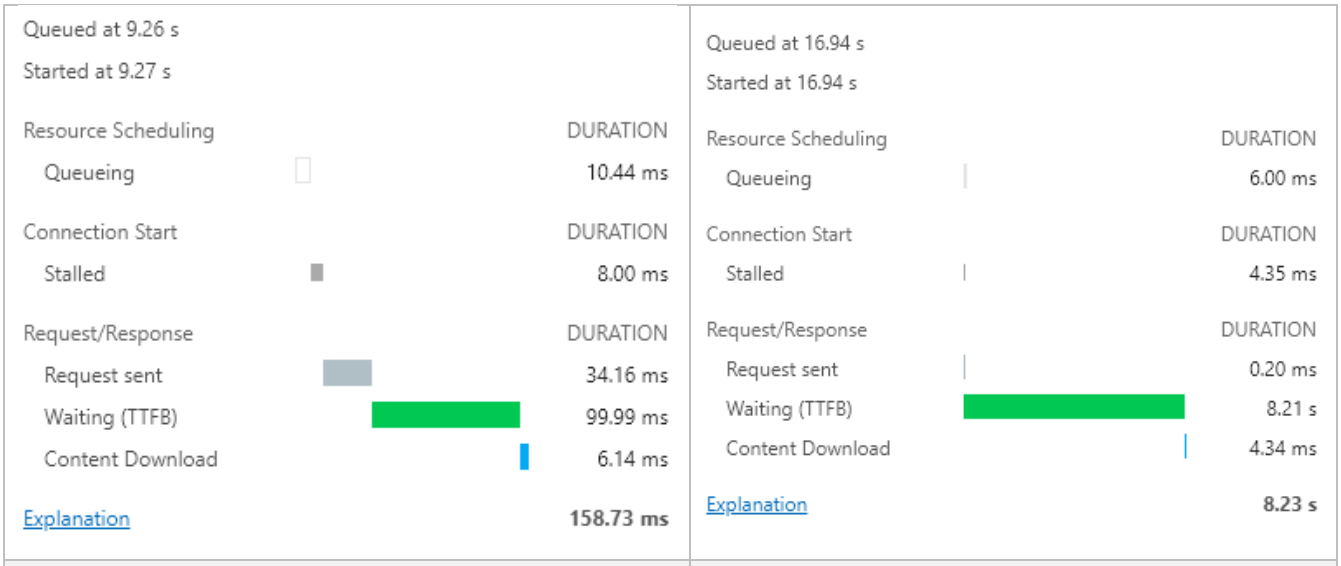


Рисунок Г.3 – Результати навантажувального тестування повного курсу дисципліни «Організація баз даних і знань»

Результати навантажувального тестування показали, що СІ здатна працювати достатньо швидко з документами різного об’єму. Час затрачений на завантаження документу є пропорційним його розміру.

Тестування стабільності (Stability Testing) – проводиться для того щоб переконатися, що ІС здатна працювати на протязі довгого часу з визначеним очікуваним навантаження. При проведенні тестування такого типу виконується спостерігання за використанням пам'яті програмою, для виявлення імовірних надлишкових втрат. До того ж, тестування стабільності визначає регрес продуктивності, що виражається в явному для користувача зменшені швидкодії ІС, тобто зниження швидкості опрацювання інформації та збільшенні часу відгуку ІС після довготривалої роботи порівнюючи з відповідними показниками, що були зафіксовані на початку даного тестування.

Для даного тестування використовувалися вище описані документи «Історія та етапи розвитку баз даних», «Нейронна мережа перцептрон» та документ повного курсу дисципліни «Організація баз даних і знань». Для тестування стабільності було проведено по 30 послідовних опрацювань документу. Результати тестування представлені на рисунках Г.4, Г.5 та Г.6.

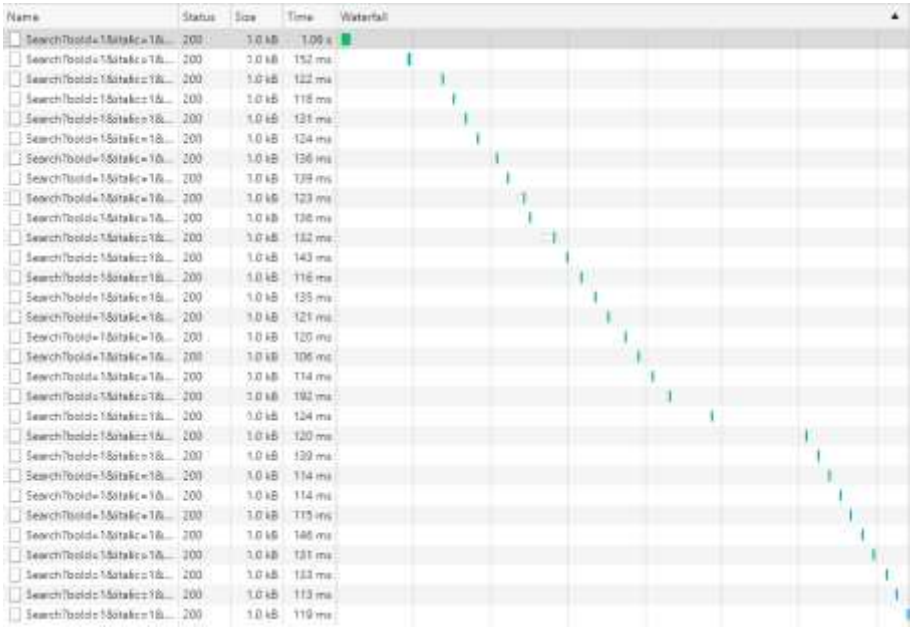


Рисунок Г.4 – Результати тестування стабільності для ІНМ «Нейронна мережа перцептрон» з дисципліни «Розробка систем штучного інтелекту»

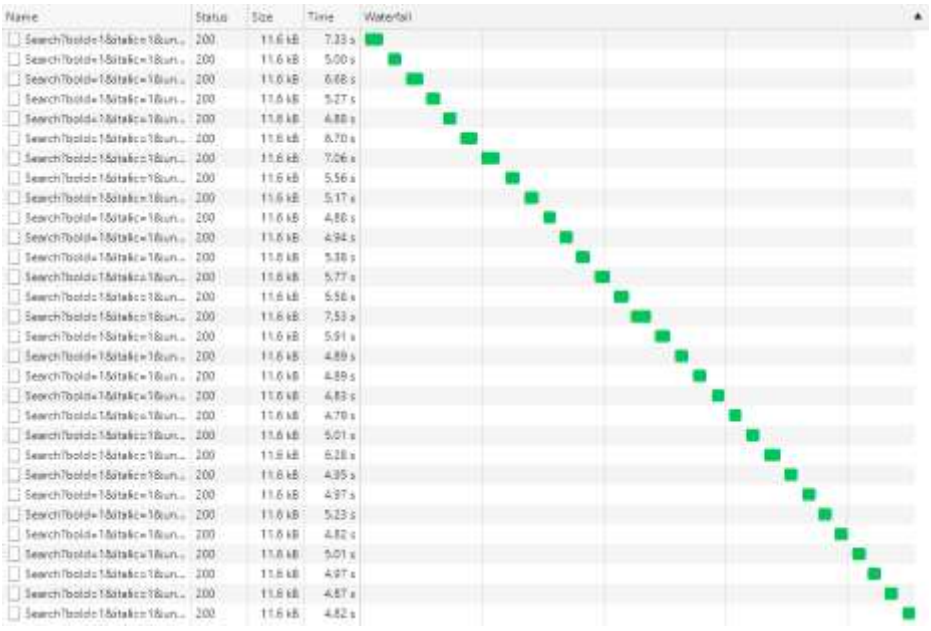


Рисунок Г.5 – Результати тестування стабільності для ІНМ «Історія та етапи розвитку баз даних» з дисципліни «Організація баз даних і знань»

Результати тестування стабільності показали, що CI здатна працювати на протязі довгого часу з визначеним очікуваним навантаження. Внаслідок

навантаження швидкодія ІС не погіршилася, а подекуди і покращилася внаслідок автоматичної оптимізації СКБД.

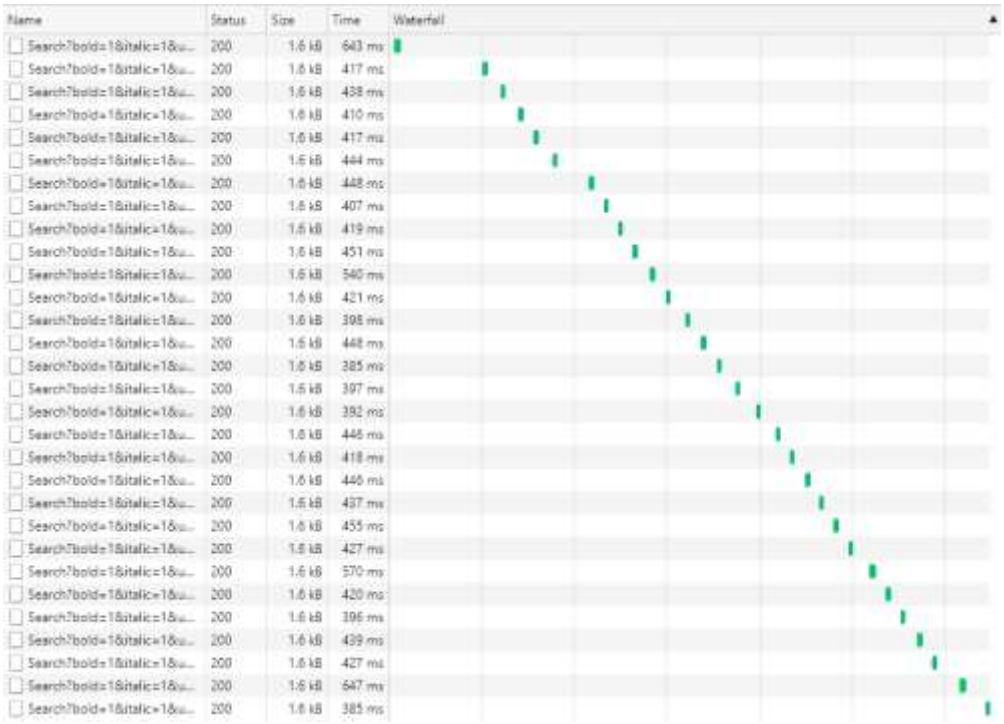


Рисунок Г.6 – Результати тестування стабільності для повного курсу дисципліни «Організація баз даних і знань»

Автоматизоване тестування (Test automation) – тестування, що відбувається в процесі розробки ІС. Таке тестування використовує програмні засоби для виконання тестів і перевірки результатів виконання, що допомагає зменшити час тестування ІС і спростити його процес. Є багато підходів до перевірки автоматизації, найбільш широко використовуваним є GUI-тестування під час якого програма тестування генерує події інтерфейсу користувача , такі як натискання клавіш і кліки миші, і дотримується змін, які призводять до інтерфейсу користувача, для перевірки правильності поведінки програми.

Для автоматизованого тестування роботи в web-браузері було розроблено проект ASI.TermSearcher.GUITests (рисунок Г.7) з використанням мови програмування C# та відкритих допоміжних засобів.

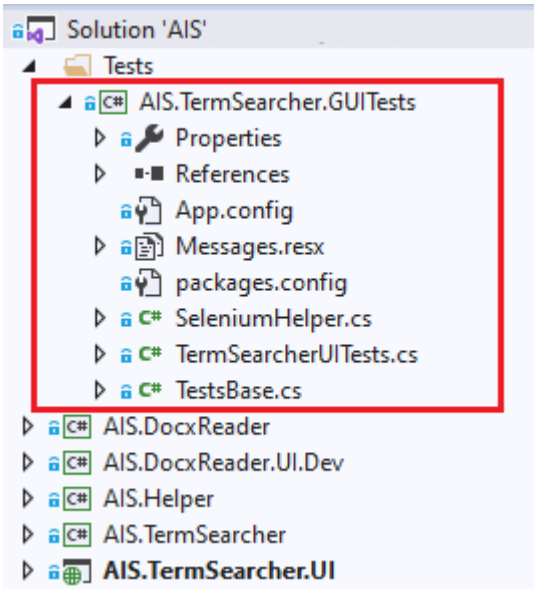


Рисунок Г.7 – Проект для тестування ASI.TermSearcher.GUITests

Використовувалися наступні допоміжні засоби:

– інструмент автоматизації Selenium з використанням Selenium Client API для мови програмування C# та Selenium.WebDriver.ChromeDriver;

– відкрите середовище модульного тестування NUnit3;

– розширення NUnit3TestAdapter, що дозволяє запускати тести в середовищі Visual Studio.

Проект потребує для роботи всього декілька класів, що зображені на мапі коду на рисунку Г.8.

TestsBase – базовий клас для тестів. Містить методи для налаштування середовища *SetUp* (запуск браузера, ініціалізація допоміжних ресурсів та їх налаштування) і дій після закінчення виконання тестів *TearDown* (закриття браузера, видалення процесів ChromeWebDriver).

TermSearcherUITests – основний клас для тестів. Ніяких функцій окрім того що містить всі тести більше не виконує.

SeleniumHelper – допоміжний клас, що містить методи які було б необхідно повторювати при написанні тестів (різні перевірки, наприклад, на присутність елемента чи їх кількість, тощо). Доданий для уникнення повторюваного коду.

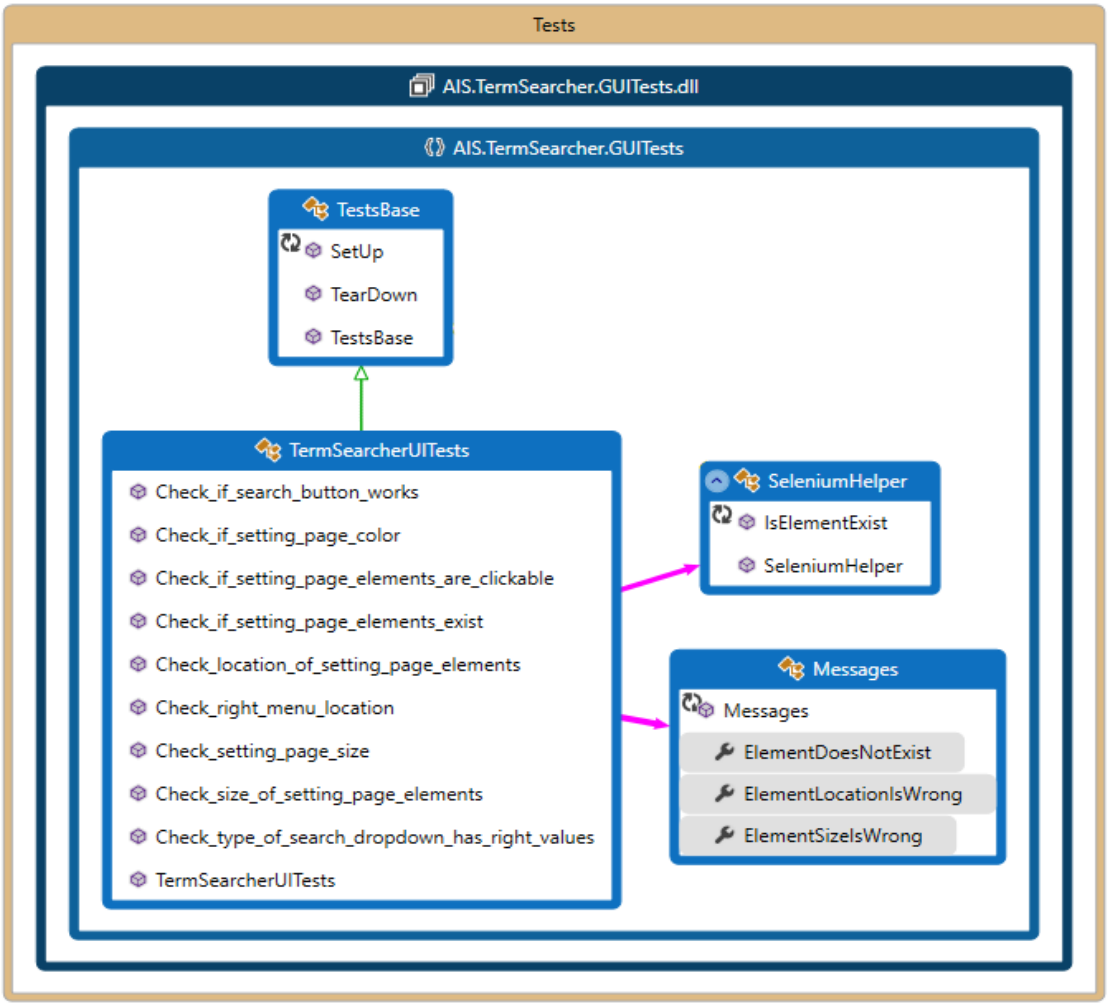


Рисунок Г.8 – Мапа коду проекту з автоматичними UI тестами

Messages – автозгенерований клас файлу ресурсів Messages.resx. Messages.resx файл містить в собі літеральні значення тексту повідомлень про помилки, тощо.

Перед початком тестування необхідно визначити перелік визначених потенційних проблем інтерфейсу користувача для яких є необхідність провести автоматизоване UI тестування.

Було проведено аналіз інтерфейсу користувача результатом якого було визначення потенційних проблем. Визначений перелік потенційних проблем та їх аргументація наведені далі:

1. В наслідок модифікації інтерфейсу користувача елементи керування можуть стати невидимими чи бути випадково видаленими, що може спричинити неможливість користуватися ІС.

2. Розташування елементів може змінитись в наслідок змін в коді чи конфлікту стилів, спричинивши неможливість користуватися ІС.

3. Розмір елементів може змінитись в наслідок змін в коді чи конфлікту стилів, спричинивши неможливість користуватися ІС.

4. Кнопки, що виконують відповідають за взаємодію з серверною частиною можуть не працювати через проблеми з сервером чи базою даних.

5. Елементи дерева структури документу можуть не розкриватися.

6. Елементи дерева структури документу можуть бути не активними в наслідок некоректного документу чи його стилів або з причин попереднього пункту.

7. Таблиця з результатами пошуку ключових термінів для певних рубрикацій може бути невидима чи зміщена в наслідок змін в коді чи конфлікту стилів, спричинивши неможливість користуватися ІС.

8. Фільтр результатів ключових термінів за назвою чи оцінкою для певних рубрикацій може не працювати, бути невидимим чи зміщена в наслідок змін в коді чи конфлікту стилів, спричинивши неможливість користуватися ІС.

9. Останній завантажений документ може відображатися некоректно в наслідок нестабільної роботи сервера.

Враховуючи вище перелічені потенційні проблеми було розроблено відповідні автоматичні тести та під час розробки ІС регулярно проводилося автоматизоване UI тестування (рисунок Г.9) задля забезпечення підтримки працездатності ІС.

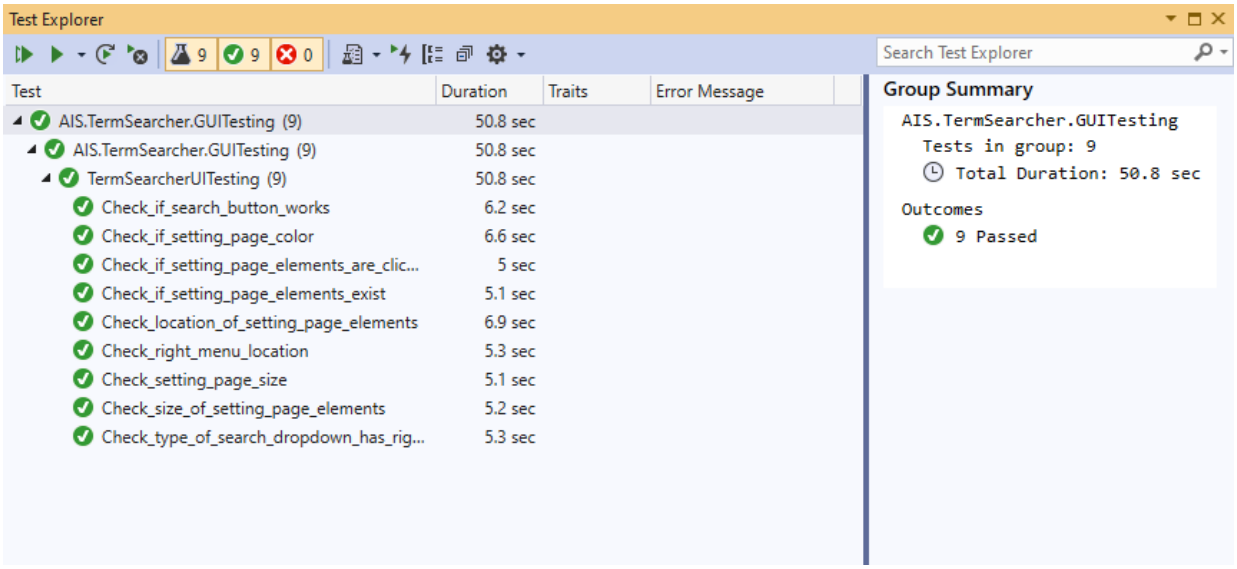


Рисунок Г.9 – Результат виконання тестів в інструменті для роботи з тестами NUnit3

Результати автоматизованого тестування з допомогою інструменту автоматизації Selenium показали, що інтерфейс користувача стабільний та виконує всі необхідні функції. Задля забезпечення підтримки працездатності ІС автоматизоване UI тестування необхідно проводити після будь яких змін в коді.

Таким чином, було проведено дослідження функціональності ІС з допомогою декількох видів тестування, а саме: навантажувальне тестування, тестування стабільності та автоматизоване тестування інтерфейсу користувача. Результати навантажувального тестування показали, що ІС здатна працювати достатньо швидко з документами різного об’єму. Результати тестування стабільності показали, що ІС здатна працювати на протязі довгого часу з визначеним очікуваним навантаження. Результати автоматизованого тестування з допомогою інструменту автоматизації Selenium показали, що інтерфейс користувача стабільний та виконує всі необхідні функції.

Додаток Д
Дослідження функціональності інформаційної системи

Результатом роботи ІТ є семантична структура ІНМ, яка може бути використана для розв’язання ряду задач в питаннях автоматизації роботи з навчальними матеріалами, серед яких наступні:

– допомога в формуванні ІНМ;

– перевірка ІНМ відповідності вимогам;

– автоматизація класифікації та організації ІНМ.

Згідно поставленого завдання розроблений для дослідження практичної ефективності створеної інформаційної технології програмний продукт виконує наступні функції:

– зчитування .docx файлу та витяг його текстового контенту;

– побудова дерева структури документу;

– вибір елемента дерева структури документу для аналізу;

– формування множини термінів до обраного елемента дерева структури документу;

– оцінка важливості термінів тексту методом дисперсійного оцінювання;

– сортування й обмеження результуючої множини термінів;

– побудова семантичної структуру інформаційного навчального матеріалу.

Перевірка роботи з ІТ представлена нижче.

Для початку роботи потрібно перейти на розроблений веб-сайт. На першій сторінці сайту користувача зустріне сторінка налаштувань (рисунок Д.1).

Необхідно обрати файл, ввести поправочні коефіцієнти та обрати етап пошуку (рисунок Д.2).

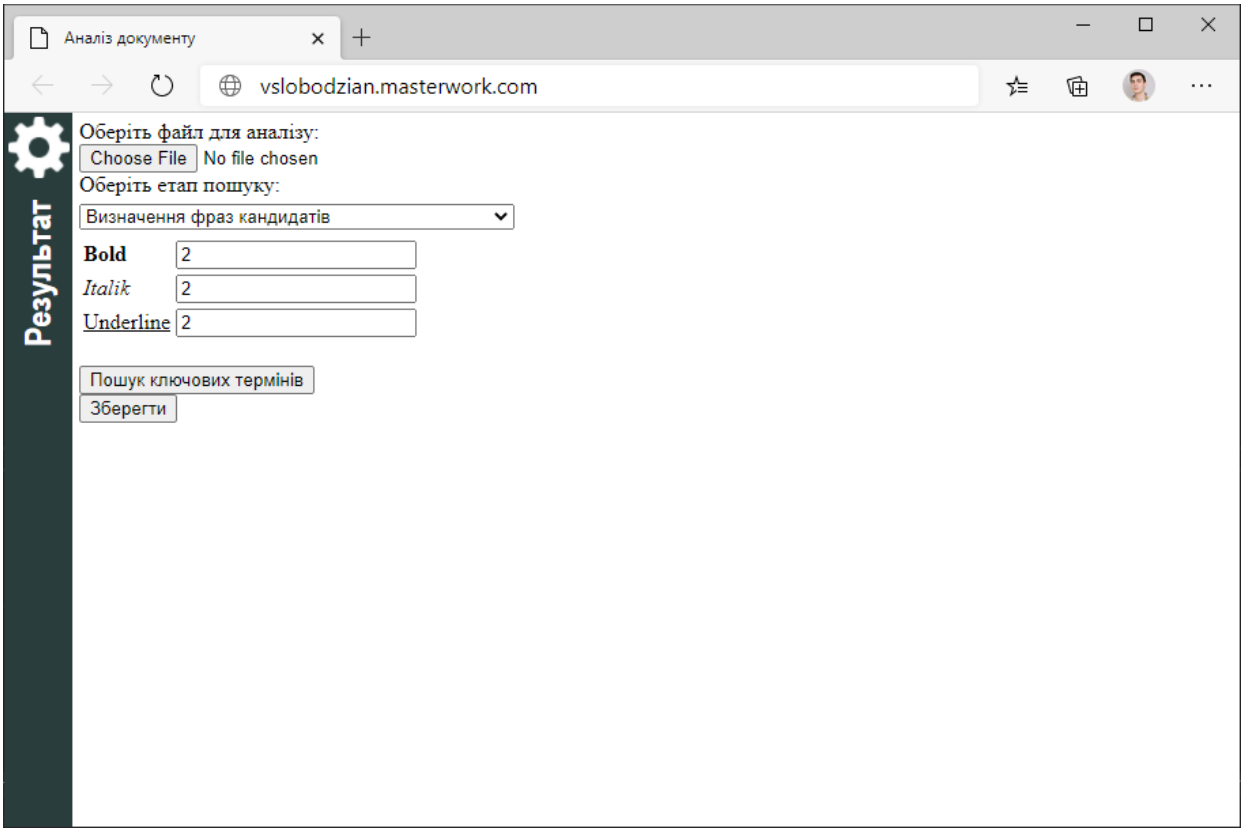


Рисунок Д.1 – Налаштування

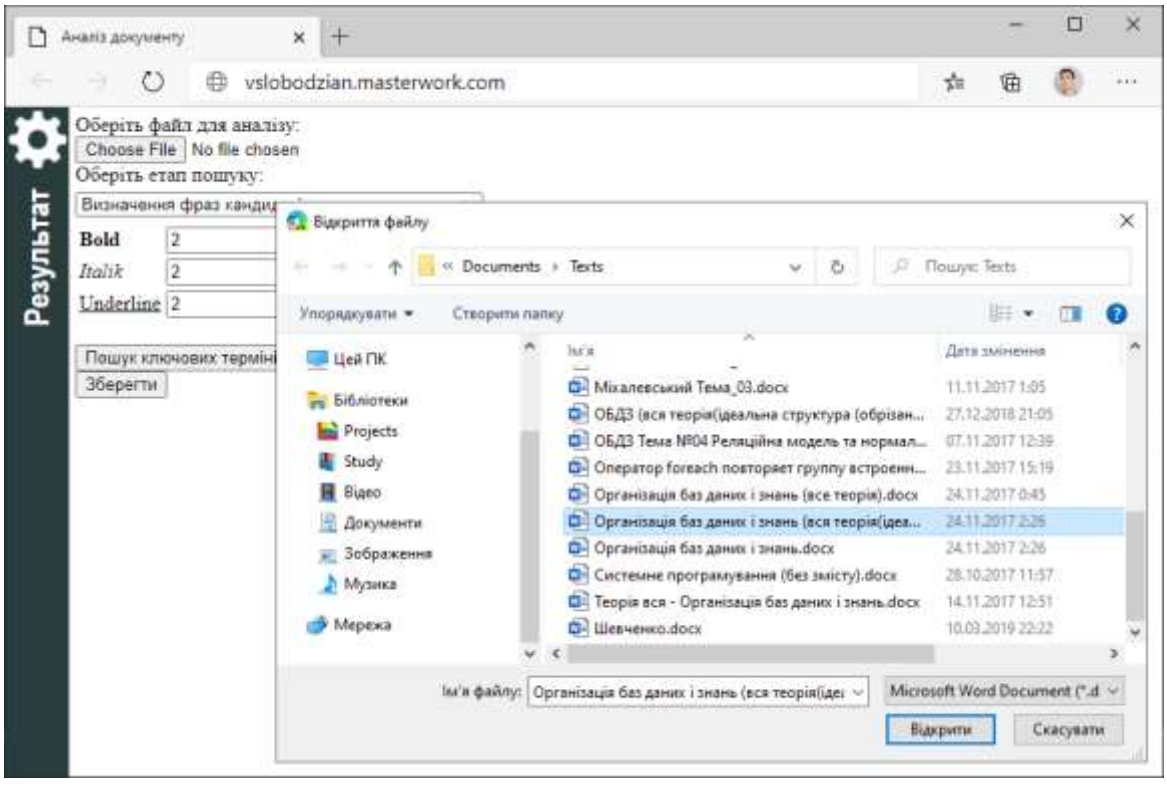


Рисунок Д.2 – Вибір файлу та встановлення поправочних коефіцієнтів

Після завантаження файлу ІС проінформує по це та закінчивши налаштування всіх необхідних параметрів можна розпочинати пошук ключових термінів натиснувши відповідну кнопку. Далі сайт автоматично перемкнеться на сторінку з результатами (рисунок Д.3).

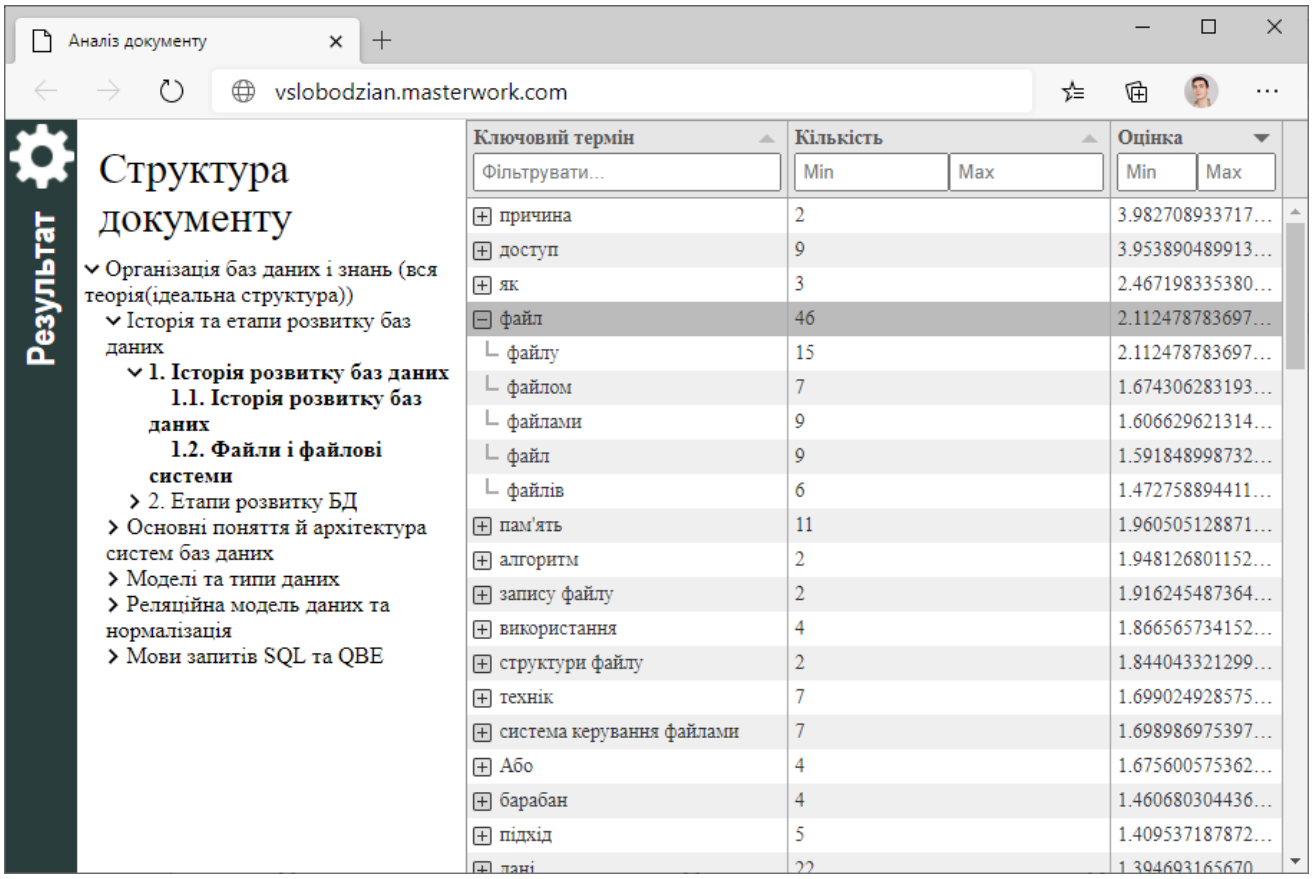


Рисунок Д.3 – Відображення результатів

Таким чином, розроблена ІС виконує всі необхідні функції для забезпечення повноцінної роботи ІТ.

Додаток Е

Програмні коди

Лістинг AIS.DocxReader\DocxReader.cs

```
using AIS.DocxReader.Entities.Document;
using AIS.DocxReader.Entities.Paragraph;
using AIS.DocxReader.Entities.Section;
using AIS.DocxReader.Entities.Sentences;
using AIS.DocxReader.Entities.TextContainer;
using AIS.DocxReader.Entities.Word;
using AIS.DocxReader.Resources;
using ASI.DocxReader.Entities.Sentences;
using DocumentFormat.OpenXml.Packaging;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text.RegularExpressions;
using Wordprocessing =
DocumentFormat.OpenXml.Wordprocessing;

namespace AIS.DocxReader
{
 public class DocxReader
 {
 private readonly string filePath;
 private readonly WordprocessingDocument
wordDocument;
 private readonly Regex regexWord;

 private string[] headingStyles;
 private string[] normalStyles;
 private string[] titleStyles;
 private string[] listStyles;

 private int lastParagraphIndex = 0;
 private int lastHeadingLevel = -1;

 public DocxReader(string filePath)
 {
 this.filePath = filePath;
 wordDocument =
WordprocessingDocument.Open(filePath, false);

 regexWord = new
Regex($"{RegexPatterns.Exclusion}|{RegexPattern
s.Date}|{RegexPatterns.Name}|{RegexPatterns.Wor
d}|{RegexPatterns.Separator}");

 ReadStyles();
 }

 public Document Read()
 {
 Document document = new Document()
 {
 Name =
Path.GetFileNameWithoutExtension(filePath),
 Sections = new List<ISection>()
 };

 List<IParagraph> paragraphs = new
List<IParagraph>();

 foreach (object wordParagraph in
wordDocument.MainDocumentPart.Document.ChildEle
ments[0].ChildElements)
 {
 if (wordParagraph is Wordprocessing.Paragraph
currentWordParagraph)
 {
 IParagraph paragraph =
GetParagraph(currentWordParagraph);

 if (paragraph.Sentences.Any())
 {
 paragraphs.Add(paragraph);
 }
 }
 }


document.Sections.Add(MakeStructure(paragraphs)
);

 return document;
 }

 private ISection
MakeStructure(List<IParagraph> paragraphs)
 {
 Section rootSection = new Section
 {
 Level = -1,
 Name =
Path.GetFileNameWithoutExtension(filePath),
 Paragraphs = new List<IParagraph>(),
 Sections = new List<ISection>(),
 Guid = Guid.NewGuid()
 };

 MakeSections(rootSection, paragraphs);

 return rootSection;
 }

 private void ReadStyles()
 {
 headingStyles = ParagraphTypes.Heading.Split('
');
 normalStyles = ParagraphTypes.Normal.Split('
');
 titleStyles = ParagraphTypes.Title.Split(' ');
 listStyles = ParagraphTypes.List.Split(' ');
 }

 private void MakeSections(ISection
currentSection, List<IParagraph> paragraphs)
 {
 while (lastParagraphIndex + 1 <
paragraphs.Count)
 {
 IParagraph currentParagraph =
paragraphs[lastParagraphIndex];

 if (currentParagraph.Level ==
currentSection.Level)
 {
 if (currentParagraph.Type ==
EParagraphType.Heading || currentParagraph.Type
== EParagraphType.Title)
 {
 break;
 }


currentSection.Paragraphs.Add(currentParagraph)
;
 lastParagraphIndex++;
 }
 else if (currentParagraph.Level >
currentSection.Level)
 {
 string headerText;

 if (currentParagraph.Type ==
EParagraphType.Heading || currentParagraph.Type
== EParagraphType.Title)
 {
 headerText = currentParagraph.Text;
 }
 else
 {
 headerText =
currentParagraph.Sentences[0].Text;
 }

 Section section = new Section
 {
 Sections = new List<ISection>(),
 Paragraphs = new List<IParagraph>(),
 Level = currentParagraph.Level,
 Guid = Guid.NewGuid(),
 Name = headerText
 };
 section.Paragraphs.Add(currentParagraph);

 lastParagraphIndex++;

 MakeSections(section, paragraphs);
 currentSection.Sections.Add(section);
 }
 else if (currentParagraph.Level <
currentSection.Level)
 {
 break;
 }
 }
 }

 private IParagraph
GetParagraph(Wordprocessing.Paragraph
wordParagraph)
 {
 Paragraph paragraph = new Paragraph
 {
 Sentences = GetSentences(wordParagraph),
 Type = GetParagraphType(wordParagraph),
 Text = wordParagraph.InnerText
 };

 SetParagraphLevel(paragraph, wordParagraph);

 return paragraph;
 }

 private List<ISentences>
GetSentences(Wordprocessing.Paragraph
wordParagraph)
 {
 List<IWord> words = new List<IWord>();
 List<ISentences> sentences = new
List<ISentences>();
 List<ITextContainer> textContainers = new
List<ITextContainer>();
 WordStyle lastWordStyle = new WordStyle();
 TextContainer textContainer = new
TextContainer()
 {
 Words = new List<IWord>(),
 StartSymbolType = ESymbolType.Start
 };

 int lastSentencesIndex = 0;
 int lastTextContainerIndex = 0;

 MatchCollection wordMatches =
regexWord.Matches(wordParagraph.InnerText);
 var wordMathesByStyle =
GetWordMathesByStyle(wordParagraph);

 foreach (Match wordMatch in wordMatches)
 {
 if (string.IsNullOrEmpty(wordMatch.Value) ||
string.IsNullOrWhiteSpace(wordMatch.Value))
 {
 continue;
 }

 if (wordMatch.Groups["separator"].Success)
 {
 ESentencesType sentencesType =
GetSentencesType(wordMatch.Value);
 ESymbolType symbolType =
GetSymbolType(wordMatch.Value);

 textContainer.EndSymbolType = symbolType;
 textContainer.EndSymbol = wordMatch.Value;
 textContainer.Text =
wordParagraph.InnerText.Substring(lastTextConta
inerIndex, wordMatch.Index -
lastTextContainerIndex).Trim();
 textContainers.Add(textContainer);
 textContainer = new TextContainer()
 {
 Words = new List<IWord>(),
 StartSymbolType = symbolType,
 StartSymbol = wordMatch.Value
 };

 lastTextContainerIndex = wordMatch.Index + 1;

 if (sentencesType == ESentencesType.None)
 {
 continue;
 }

 Sentences currentSentences = new Sentences()
 {
 Text =
wordParagraph.InnerText.Substring(lastSentences
Index, wordMatch.Index -
lastSentencesIndex).Trim(),
 Words = words,
 SentencesType = sentencesType,
 TextContainers = textContainers
 };

 sentences.Add(currentSentences);

 lastSentencesIndex = wordMatch.Index + 1;
 words = new List<IWord>();
 textContainers = new List<ITextContainer>();
 }
 else
 {
 IWord word = GetWord(wordMatch);
 WordStyle wordStyle = new WordStyle();
```

```
 if
(wordMathesByStyle.ContainsKey(wordMatch.Index)
)
 {
 wordStyle =
wordMathesByStyle[wordMatch.Index].Style;
 word.WordStyle = wordStyle;
 }
 else
 {
 word.WordStyle = wordStyle;
 }

 words.Add(word);

 if (lastWordStyle.Equals(wordStyle))
 {
 textContainer.Words.Add(word);
 }
 else
 {
 textContainer.EndSymbolType =
ESymbolType.None;
 textContainer.Text =
wordParagraph.InnerText.Substring(lastTextConta
inerIndex, wordMatch.Index -
lastTextContainerIndex).Trim();
 if (textContainer.Words.Any())
 {
 textContainers.Add(textContainer);
 }
 textContainer = new TextContainer()
 {
 Words = new List<IWord>() { word },
 StartSymbolType = ESymbolType.None
 };

 lastTextContainerIndex = wordMatch.Index;
 }

 lastWordStyle = wordStyle;
 }
 }

 if (words.Any())
 {
 textContainer.EndSymbolType =
ESymbolType.None;
 textContainer.Text =
wordParagraph.InnerText.Substring(lastTextConta
inerIndex, wordParagraph.InnerText.Length -
lastTextContainerIndex).Trim();
 textContainers.Add(textContainer);

 sentences.Add(new Sentences()
 {
 Text =
wordParagraph.InnerText.Substring(lastSentences
Index, wordParagraph.InnerText.Length -
lastSentencesIndex).Trim(),
 Words = words,
 SentencesType = ESentencesType.None,
 TextContainers = textContainers
 });
 }

 return sentences;
 }

 private ESymbolType GetSymbolType(string
symbolType)
 {
 if (SymbolTypes.FullStop.Contains(symbolType))
 return ESymbolType.FullStop;
 else if
(SymbolTypes.Comma.Contains(symbolType))
 return ESymbolType.Comma;
 else if
(SymbolTypes.OpenBracket.Contains(symbolType))
 return ESymbolType.OpenBracket;
 else if
(SymbolTypes.CloseBracket.Contains(symbolType))
 return ESymbolType.CloseBracket;
 else if
(SymbolTypes.Hyphen.Contains(symbolType))
 return ESymbolType.Hyphen;
 else if
(SymbolTypes.Quotation.Contains(symbolType))
 return ESymbolType.Quotation;
 else return ESymbolType.Other;
 }

 private Dictionary<int, (Match Match,
WordStyle Style)>
GetWordMathesByStyle(Wordprocessing.Paragraph
wordParagraph)
 {
 Dictionary<int, (Match, WordStyle)>
wordMatches = new Dictionary<int, (Match,
WordStyle)>();
 int lastIndex = 0;

 foreach (var paragraphChild in
wordParagraph.ChildElements)
 {
 if (paragraphChild is Wordprocessing.Run run)
 {
 WordStyle wordStyle = new WordStyle()
 {
 Bold = run.RunProperties?.Bold != null,
 Italic = run.RunProperties?.Italic != null,
 Underline = run.RunProperties?.Underline !=
null,
 };

 foreach (Match match in
regexWord.Matches(run.InnerText))
 {
 wordMatches.Add(lastIndex + match.Index,
(match, wordStyle));
 }

 lastIndex += run.InnerText.Length;
 }
 else if
(!string.IsNullOrWhiteSpace(paragraphChild.Inne
rText))
 {
 foreach (Match match in
regexWord.Matches(paragraphChild.InnerText))
 {
 wordMatches.Add(lastIndex + match.Index,
(match, new WordStyle()));
 }

 lastIndex += paragraphChild.InnerText.Length;
 }
 }

 return wordMatches;
 }

 private IWord GetWord(Match word)
 {
 if (word.Groups["name"].Success)
 {
 return new Name()
 {
 Text = word.Groups["name"].Value,
 Lastname = word.Groups["lastname"].Value,
 Initials = word.Groups["initials"].Value
 };
 }
 else if (word.Groups["year"].Success ||
word.Groups["day"].Success)
 {
 return new Date()
 {
 Text = word.Value
 };
 }
 else if (word.Groups["word"].Success ||
word.Groups["exclusion"].Success)
 {
 return new Word()
 {
 Text = word.Value
 };
 }
 //todo: check if the condition is possible
 else return null;
 }

 private ESentencesType GetSentencesType(string
separator)
 {
 switch (separator)
 {
 case ".":
 return ESentencesType.Declarative;
 case "?":
 return ESentencesType.Interrogative;
 case "!":
 return ESentencesType.Exclamatory;
 default:
 return ESentencesType.None;
 }
 }

 private void SetParagraphLevel(Paragraph
paragraph, Wordprocessing.Paragraph
wordParagraph)
 {
 if (paragraph.Type == EParagraphType.Heading)
 {
 if
(int.TryParse(wordParagraph.ParagraphProperties
.ParagraphStyleId.Val.InnerText, out int
level))
 {
 paragraph.Level = level;
 lastHeadingLevel = level;
 }
 else
 {
 paragraph.Level = 0;
 lastHeadingLevel = 0;
 }
 }
 else if (paragraph.Type ==
EParagraphType.Title)
 {
 paragraph.Level = 0;
 lastHeadingLevel = 0;
 }
 else
 {
 paragraph.Level = lastHeadingLevel;
 }
 }

 private EParagraphType
GetParagraphType(Wordprocessing.Paragraph
wordParagraph)
 {
 if
(wordParagraph.ParagraphProperties?.ParagraphSt
yleId == null)
 {
 return EParagraphType.Normal;
 }

 string styleId =
wordParagraph.ParagraphProperties.ParagraphStyl
eId.Val.InnerText;

 if (headingStyles.Contains(styleId) ||
styleId.Contains("Heading"))
 {
 return EParagraphType.Heading;
 }
 else if (titleStyles.Contains(styleId))
 {
 return EParagraphType.Title;
 }
 else if (listStyles.Contains(styleId))
 {
 return EParagraphType.List;
 }
 else if (normalStyles.Contains(styleId))
 {
 return EParagraphType.Normal;
 }
 else
 {
 return EParagraphType.None;
 }
 }

 ~DocxReader()
 {
 wordDocument.Dispose();
 }
 }
}
```

Лістинг
AIS.DocxReader\Entities\Document\Document.cs

```
using AIS.DocxReader.Entities.Section;
using System.Collections.Generic;

namespace AIS.DocxReader.Entities.Document
{
 public class Document : IDocument
 {
 public string Name { get; set; }

 public List<ISection> Sections { get; set; }
 }
}
```

Лістинг
AIS.DocxReader\Entities\Document\IDocument.cs

```
using AIS.DocxReader.Entities.Section;
using System.Collections.Generic;

namespace AIS.DocxReader.Entities.Document
{
 public interface IDocument
 {
 string Name { get; set; }

 List<ISection> Sections { get; set; }
 }
}
```

Лістинг
AIS.DocxReader\Entities\Paragraph\EParagraphTyp
e.cs

```
namespace AIS.DocxReader.Entities.Paragraph
```

```
{
 public enum EParagraphType : sbyte
 {
 None = -1,
 Normal,
 List,
 Table,
 Heading,
 Title
 }
}
```

Лістинг
AIS.DocxReader\Entities\Paragraph\IParagraph.cs

```
using ASI.DocxReader.Entities.Sentences;
using System.Collections.Generic;

namespace AIS.DocxReader.Entities.Paragraph
{
 public interface IParagraph
 {
 int Level { get; set; }

 string Text { get; set; }

 EParagraphType Type { get; set; }

 List<ISentences> Sentences { get; set; }
 }
}
```

Лістинг
AIS.DocxReader\Entities\Paragraph\Paragraph.cs

```
using ASI.DocxReader.Entities.Sentences;
using System.Collections.Generic;

namespace AIS.DocxReader.Entities.Paragraph
{
 public class Paragraph : IParagraph
 {
 public int Level { get; set; }

 public string Text { get; set; }

 public EParagraphType Type { get; set; }

 public List<ISentences> Sentences { get; set;
}
 }
}
```

Лістинг
AIS.DocxReader\Entities\Section\ISection.cs

```
using AIS.DocxReader.Entities.Paragraph;
using System;
using System.Collections.Generic;

namespace AIS.DocxReader.Entities.Section
{
 public interface ISection
 {
 int Level { get; set; }

 string Text { get; }

 string Name { get; set; }

 Guid Guid { get; set; }

 List<ISection> Sections { get; set; }

 List<IParagraph> Paragraphs { get; set; }

 List<IParagraph> AllParagraphs { get; }
 }
}
```

Лістинг
AIS.DocxReader\Entities\Section\Section.cs

```
using AIS.DocxReader.Entities.Paragraph;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace AIS.DocxReader.Entities.Section
{
 public class Section : ISection
 {
 private string text;
 private List<IParagraph> allParagraphs;

 public int Level { get; set; }

 public string Name { get; set; }

 public string Text
 {
 get
 {
 if (string.IsNullOrEmpty(text))
 {
 text = GetText();
 }

 return text;
 }
 protected set { text = value; }
 }

 public Guid Guid { get; set; }

 public List<ISection> Sections { get; set; }

 public List<IParagraph> Paragraphs { get; set;
}

 public List<IParagraph> AllParagraphs
 {
 get
 {
 if (allParagraphs == null)
 {
 allParagraphs = new List<IParagraph>();
 allParagraphs.AddRange(this.Paragraphs);
 allParagraphs.AddRange(from s in Sections
 from p in s.AllParagraphs
 select p);
 }

 return allParagraphs;
 }
 }

 private string GetText()
 {
 StringBuilder stringBuilder = new
StringBuilder();

 if (Paragraphs != null)
 {
 foreach (var paragraph in Paragraphs)
 {
 stringBuilder.AppendLine(paragraph.Text);
 }
 }

 if (Sections != null)
 {
 foreach (var section in Sections)
 {
 stringBuilder.AppendLine(section.Text);
 }
 }

 return stringBuilder.ToString();
 }
 }
}
```

Лістинг
AIS.DocxReader\Entities\Sentences\ESentencesTyp
e.cs

```
namespace AIS.DocxReader.Entities.Sentences
{
 public enum ESentencesType
 {
 /// <summary>
 /// Default value. It is used when a sentences
type impossible to know
 /// </summary>
 None,
 /// <summary>
 /// Ends with '.'
 /// </summary>
 Declarative,
 /// <summary>
 /// Ends with '?'
 /// </summary>
 Interrogative,
 /// <summary>
 /// Ends with '!'
 /// </summary>
 Exclamatory
 }
}
```

Лістинг
AIS.DocxReader\Entities\Sentences\ISentences.cs

```
using AIS.DocxReader.Entities.Sentences;
using AIS.DocxReader.Entities.TextContainer;
using AIS.DocxReader.Entities.Word;
using System.Collections.Generic;

namespace ASI.DocxReader.Entities.Sentences
{
 public interface ISentences
 {
 string Text { get; set; }

 List<IWord> Words { get; set; }

 List<ITextContainer> TextContainers { get;
set; }

 ESentencesType SentencesType { get; set; }
 }
}
```

Лістинг
AIS.DocxReader\Entities\Sentences\Sentences.cs

```
using AIS.DocxReader.Entities.TextContainer;
using AIS.DocxReader.Entities.Word;
using ASI.DocxReader.Entities.Sentences;
using System.Collections.Generic;

namespace AIS.DocxReader.Entities.Sentences
{
 public class Sentences : ISentences
 {
 public string Text { get; set; }

 public List<IWord> Words { get; set; }

 public List<ITextContainer> TextContainers {
get; set; }

 public ESentencesType SentencesType { get;
set; }
 }
}
```

Лістинг
AIS.DocxReader\Entities\TextContainer\ESymbolTy
pe.cs

```
namespace AIS.DocxReader.Entities.TextContainer
{
 public enum ESymbolType
 {
 None, //Style conteiner
 Comma, //,
 FullStop, //.!?…
 OpenBracket, //[({(
 CloseBracket, //])}
 Hyphen, //- – — ―
 Quotation, //‘’“”'"
 Start,
 End,
 Other
 }
}
```

Лістинг
AIS.DocxReader\Entities\TextContainer\ITextCont
ainer.cs

```
using AIS.DocxReader.Entities.Word;
using System.Collections.Generic;

namespace AIS.DocxReader.Entities.TextContainer
{
 public interface ITextContainer
 {
 string Text { get; set; }

 List<IWord> Words { get; set; }

 ESymbolType StartSymbolType { get; set; }

 ESymbolType EndSymbolType { get; set; }

 string EndSymbol { get; set; }

 string StartSymbol { get; set; }
 }
}
```

Лістинг
AIS.DocxReader\Entities\TextContainer\TextConta
iner.cs

```
using AIS.DocxReader.Entities.Word;
using System.Collections.Generic;

namespace AIS.DocxReader.Entities.TextContainer
{
 public class TextContainer : ITextContainer
 {
 public string Text { get; set; }

 public List<IWord> Words { get; set; }

 public ESymbolType StartSymbolType { get; set;
}
```

```
 public ESymbolType EndSymbolType { get; set; }

 public string EndSymbol { get; set; }

 public string StartSymbol { get; set; }
 }
}
```

Лістинг AIS.DocxReader\Entities\Word\Date.cs

```
namespace AIS.DocxReader.Entities.Word
{
 class Date : IWord
 {
 public IWordStyle WordStyle { get; set; }

 public string Text { get; set; }
 }
}
```

Лістинг AIS.DocxReader\Entities\Word\IWord.cs

```
namespace AIS.DocxReader.Entities.Word
{
 public interface IWord
 {
 IWordStyle WordStyle { get; set; }

 string Text { get; set; }
 }
}
```

Лістинг
AIS.DocxReader\Entities\Word\IWordStyle.cs

```
namespace AIS.DocxReader.Entities.Word
{
 public interface IWordStyle
 {
 bool Bold { get; set; }

 bool Italic { get; set; }

 bool Underline { get; set; }
 }
}
```

Лістинг AIS.DocxReader\Entities\Word\Name.cs

```
namespace AIS.DocxReader.Entities.Word
{
 class Name : IWord
 {
 public IWordStyle WordStyle { get; set; }

 public string Text { get; set; }

 public string Lastname { get; set; }

 public string Initials { get; set; }
 }
}
```

Лістинг AIS.DocxReader\Entities\Word\Word.cs

```
namespace AIS.DocxReader.Entities.Word
{
 public class Word : IWord
 {
 public IWordStyle WordStyle { get; set; }

 public string Text { get; set; }
 }
}
```

Лістинг
AIS.DocxReader\Entities\Word\WordStyle.cs

```
namespace AIS.DocxReader.Entities.Word
{
 public class WordStyle : IWordStyle
 {
 public bool Bold { get; set; }

 public bool Italic { get; set; }

 public bool Underline { get; set; }

 public override bool Equals(object obj)
 {
 if (!(obj is WordStyle style))
 {
 return false;
 }
 else if (this.Bold == style.Bold &&
 this.Italic == style.Italic &&
 this.Underline == style.Underline)
 {
 return true;
 }
 return false;
 }
 }
}
```

Лістинг
AIS.DocxReader\Properties\AssemblyInfo.cs

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// General Information about an assembly is
controlled through the following
// set of attributes. Change these attribute
values to modify the information
// associated with an assembly.
[assembly: AssemblyTitle("AIS.DocxReader")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("AIS.DocxReader")]
[assembly: AssemblyCopyright("Copyright ©
2018")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types
in this assembly not visible
// to COM components. If you need to access a
type in this assembly from
// COM, set the ComVisible attribute to true on
that type.
[assembly: ComVisible(false)]

// The following GUID is for the ID of the
typelib if this project is exposed to COM
[assembly: Guid("67b6752f-ada5-439c-82f6-
80d5546462ef")]

// Version information for an assembly consists
of the following four values:
//
// Major Version
// Minor Version
// Build Number
// Revision
//
// You can specify all the values or you can
default the Build and Revision Numbers
// by using the '*' as shown below:
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

Лістинг
AIS.DocxReader\Resources\ParagraphTypes.Designe
r.cs

```
//------------------------------------------
------------------------------
// <auto-generated>
// This code was generated by a tool.
// Runtime Version:4.0.30319.42000
//
// Changes to this file may cause incorrect
behavior and will be lost if
// the code is regenerated.
// </auto-generated>
//------------------------------------------
------------------------------

namespace AIS.DocxReader.Resources {
 using System;


 /// <summary>
 /// A strongly-typed resource class, for
looking up localized strings, etc.
 /// </summary>
 // This class was auto-generated by the
StronglyTypedResourceBuilder
 // class via a tool like ResGen or Visual
Studio.
 // To add or remove a member, edit your .ResX
file then rerun ResGen
 // with the /str option, or rebuild your VS
project.

[global::System.CodeDom.Compiler.GeneratedCodeA
ttribute("System.Resources.Tools.StronglyTypedR
esourceBuilder", "15.0.0.0")]

[global::System.Diagnostics.DebuggerNonUserCode
Attribute()]

[global::System.Runtime.CompilerServices.Compil
erGeneratedAttribute()]
 internal class ParagraphTypes {

 private static
global::System.Resources.ResourceManager
resourceMan;

 private static
global::System.Globalization.CultureInfo
resourceCulture;


[global::System.Diagnostics.CodeAnalysis.Suppre
ssMessageAttribute("Microsoft.Performance",
"CA1811:AvoidUncalledPrivateCode")]
 internal ParagraphTypes() {
 }

 /// <summary>
 /// Returns the cached ResourceManager
instance used by this class.
 /// </summary>

[global::System.ComponentModel.EditorBrowsableA
ttribute(global::System.ComponentModel.EditorBr
owsableState.Advanced)]
 internal static
global::System.Resources.ResourceManager
ResourceManager {
 get {
 if (object.ReferenceEquals(resourceMan, null))
{
 global::System.Resources.ResourceManager temp
= new
global::System.Resources.ResourceManager("AIS.D
ocxReader.Resources.ParagraphTypes",
typeof(ParagraphTypes).Assembly);
 resourceMan = temp;
 }
 return resourceMan;
 }
 }

 /// <summary>
 /// Overrides the current thread's
CurrentUICulture property for all
 /// resource lookups using this strongly typed
resource class.
 /// </summary>

[global::System.ComponentModel.EditorBrowsableA
ttribute(global::System.ComponentModel.EditorBr
owsableState.Advanced)]
 internal static
global::System.Globalization.CultureInfo
Culture {
 get {
 return resourceCulture;
 }
 set {
 resourceCulture = value;
 }
 }

 /// <summary>
 /// Looks up a localized string similar to 1 2
3 4 5 6 7 8 9.
 /// </summary>
 internal static string Heading {
 get {
 return ResourceManager.GetString("Heading",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to a5.
 /// </summary>
 internal static string List {
 get {
 return ResourceManager.GetString("List",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to n.
 /// </summary>
 internal static string Normal {
 get {
 return ResourceManager.GetString("Normal",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to a3.
 /// </summary>
 internal static string Title {
 get {
 return ResourceManager.GetString("Title",
resourceCulture);
 }
 }
```

```
 }
}
```

Лістинг AIS.DocxReader\Resources\RegexPatterns.Designer.cs

```
//------------------------------------------------------------------------------
// <auto-generated>
// This code was generated by a tool.
// Runtime Version:4.0.30319.42000
//
// Changes to this file may cause incorrect behavior and will be lost if
// the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace AIS.DocxReader.Resources {
 using System;


 /// <summary>
 /// A strongly-typed resource class, for looking up localized strings, etc.
 /// </summary>
 // This class was auto-generated by the StronglyTypedResourceBuilder
 // class via a tool like ResGen or Visual Studio.
 // To add or remove a member, edit your .ResX file then rerun ResGen
 // with the /str option, or rebuild your VS project.

[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyTypedResourceBuilder", "15.0.0.0")]

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]

[global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
 internal class RegexPatterns {

 private static global::System.Resources.ResourceManager resourceMan;

 private static global::System.Globalization.CultureInfo resourceCulture;


[global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance", "CA1811:AvoidUncalledPrivateCode")]
 internal RegexPatterns() {
 }

 /// <summary>
 /// Returns the cached ResourceManager instance used by this class.
 /// </summary>

[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
 internal static global::System.Resources.ResourceManager ResourceManager {
 get {
 if (object.ReferenceEquals(resourceMan, null)) {
 global::System.Resources.ResourceManager temp = new global::System.Resources.ResourceManager("AIS.DocxReader.Resources.RegexPatterns", typeof(RegexPatterns).Assembly);
 resourceMan = temp;
 }
 return resourceMan;
 }
 }

 /// <summary>
 /// Overrides the current thread's CurrentUICulture property for all
 /// resource lookups using this strongly typed resource class.
 /// </summary>

[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
 internal static global::System.Globalization.CultureInfo Culture {
 get {
 return resourceCulture;
 }
 set {
 resourceCulture = value;
 }
 }

 /// <summary>
 /// Looks up a localized string similar to (?&amp;lt;day&amp;gt;\d{1,2}(?&amp;gt;\s+|-\d{1,2}\s+)(?&amp;gt;січня|лютого|березня|квітня|травня|червня|липня|серпня|вересня|жовтня|листопада|грудня)\s+)|(?&amp;lt;year&amp;gt;\d{4}(?&amp;gt;\s*(?&amp;gt;р\.|рік|року|році))|\d{4}-\d{4}\s*(?&amp;gt;рр\.|років|роках)).
 /// </summary>
 internal static string Date {
 get {
 return ResourceManager.GetString("Date", resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to (?&amp;lt;exclusion&amp;gt;ін\.|тис\.|див\.|мал\.|рис\.|т\.д\.|т\.п\.).
 /// </summary>
 internal static string Exclusion {
 get {
 return ResourceManager.GetString("Exclusion", resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to (?&amp;lt;=\s|\W|^)(?&amp;lt;name&amp;gt;(?&amp;lt;initials&amp;gt;(?&amp;gt;(?&amp;gt;[A-ЯІЇЄ]|[A-Z])\.\s?(?&amp;gt;|$)){1,2})(?&amp;lt;lastname&amp;gt;(?&amp;gt;[A-ЯІЇЄ]|[A-Z])(?&amp;gt;(?&amp;gt;\w+\S)*\w+|\w+)(?&amp;gt;-(?&amp;gt;[A-ЯІЇЄ]|[A-Z])(?&amp;gt;\w+\S)*\w+|\w+)?)).
 /// </summary>
 internal static string Name {
 get {
 return ResourceManager.GetString("Name", resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to (?&amp;lt;separator&amp;gt;\S).
 /// </summary>
 internal static string Separator {
 get {
 return ResourceManager.GetString("Separator", resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to (?&amp;lt;word&amp;gt;(?&amp;gt;\w+\S)*\w+|\w+).
 /// </summary>
 internal static string Word {
 get {
 return ResourceManager.GetString("Word", resourceCulture);
 }
 }
 }
}
```

Лістинг AIS.DocxReader\Resources\SymbolTypes.Designer.cs

```
//------------------------------------------------------------------------------
// <auto-generated>
// This code was generated by a tool.
// Runtime Version:4.0.30319.42000
//
// Changes to this file may cause incorrect behavior and will be lost if
// the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace AIS.DocxReader.Resources {
 using System;


 /// <summary>
 /// A strongly-typed resource class, for looking up localized strings, etc.
 /// </summary>
 // This class was auto-generated by the StronglyTypedResourceBuilder
 // class via a tool like ResGen or Visual Studio.
 // To add or remove a member, edit your .ResX file then rerun ResGen
 // with the /str option, or rebuild your VS project.

[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyTypedResourceBuilder", "15.0.0.0")]

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]

[global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
 internal class SymbolTypes {

 private static global::System.Resources.ResourceManager resourceMan;

 private static global::System.Globalization.CultureInfo resourceCulture;


[global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance", "CA1811:AvoidUncalledPrivateCode")]
 internal SymbolTypes() {
 }

 /// <summary>
 /// Returns the cached ResourceManager instance used by this class.
 /// </summary>

[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
 internal static global::System.Resources.ResourceManager ResourceManager {
 get {
 if (object.ReferenceEquals(resourceMan, null)) {
 global::System.Resources.ResourceManager temp = new global::System.Resources.ResourceManager("AIS.DocxReader.Resources.SymbolTypes", typeof(SymbolTypes).Assembly);
 resourceMan = temp;
 }
 return resourceMan;
 }
 }

 /// <summary>
 /// Overrides the current thread's CurrentUICulture property for all
 /// resource lookups using this strongly typed resource class.
 /// </summary>

[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
 internal static global::System.Globalization.CultureInfo Culture {
 get {
 return resourceCulture;
 }
 set {
 resourceCulture = value;
 }
 }

 /// <summary>
 /// Looks up a localized string similar to ])}).
 /// </summary>
 internal static string CloseBracket {
 get {
 return ResourceManager.GetString("CloseBracket", resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to ,.
 /// </summary>
 internal static string Comma {
 get {
 return ResourceManager.GetString("Comma", resourceCulture);
 }
```

```
 }

 /// <summary>
 /// Looks up a localized string similar to
.!?….
 /// </summary>
 internal static string FullStop {
 get {
 return ResourceManager.GetString("FullStop",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to ---
—.
 /// </summary>
 internal static string Hyphen {
 get {
 return ResourceManager.GetString("Hyphen",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to
[({{.
 /// </summary>
 internal static string OpenBracket {
 get {
 return
ResourceManager.GetString("OpenBracket",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to
‘’“”&amp;apos;&amp;quot;.
 /// </summary>
 internal static string Quotation {
 get {
 return ResourceManager.GetString("Quotation",
resourceCulture);
 }
 }
 }
}
```

Лістинг AIS.Helper\Extention\String\StringExtentions.cs

```
using System.Linq;

namespace AIS.Helper.Extention.String
{
 public static class StringExtentions
 {
 public static string FirstCharToUpper(this
string currentString)
 {
 return char.ToUpper(currentString.First()) +
currentString.Substring(1);
 }

 public static string FirstCharToLower(this
string currentString)
 {
 return char.ToLower(currentString.First()) +
currentString.Substring(1);
 }
 }
}
```

Лістинг AIS.Helper\Properties\AssemblyInfo.cs

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// General Information about an assembly is
controlled through the following
// set of attributes. Change these attribute
values to modify the information
// associated with an assembly.
[assembly: AssemblyTitle("AIS.Helper")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("AIS.Helper")]
[assembly: AssemblyCopyright("Copyright ©
2019")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types
in this assembly not visible
// to COM components. If you need to access a
type in this assembly from
// COM, set the ComVisible attribute to true on
that type.
[assembly: ComVisible(false)]

// The following GUID is for the ID of the
typelib if this project is exposed to COM
[assembly: Guid("c38c988e-5e51-42e9-9a5b-
d14fe91e26b2")]

// Version information for an assembly consists
of the following four values:
//
// Major Version
// Minor Version
// Build Number
// Revision
//
// You can specify all the values or you can
default the Build and Revision Numbers
// by using the '*' as shown below:
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

Лістинг AIS.TermSearcher\Entities\EntitiesHelper.cs

```
using AIS.TermSearcher.Entities.Word;
using System;
using System.Collections.Generic;
using System.ComponentModel;

namespace AIS.TermSearcher.Entities
{
 public static class EntitiesHelper
 {
 private static Dictionary<string, ESpeechPart>
speechPartMapping;
 private static Dictionary<string,
EGrammaticalGender> grammaticalGenderMapping;
 private static Dictionary<string,
EGrammaticalCase> grammaticalCaseMapping;

 public static Dictionary<string, ESpeechPart>
SpeechPartMapping
 {
 get
 {
 if (speechPartMapping == null)
 {
 speechPartMapping = new Dictionary<string,
ESpeechPart>();

 foreach (ESpeechPart speechPart in
Enum.GetValues(typeof(ESpeechPart)))
 {

speechPartMapping.Add(speechPart.GetUkrainianVa
lue().ToLower(), speechPart);
 }

 return speechPartMapping;
 }
 else return speechPartMapping;
 }
 }

 public static Dictionary<string,
EGrammaticalGender> GrammaticalGenderMapping
 {
 get
 {
 if (grammaticalGenderMapping == null)
 {
 grammaticalGenderMapping = new
Dictionary<string, EGrammaticalGender>();

 foreach (EGrammaticalGender speechPart in
Enum.GetValues(typeof(EGrammaticalGender)))
 {

grammaticalGenderMapping.Add(speechPart.GetUkra
inianValue().ToLower(), speechPart);
 }
 }

 return grammaticalGenderMapping;
 }
 }

 public static Dictionary<string,
EGrammaticalCase> GrammaticalCaseMapping
 {
 get
 {
 if (grammaticalCaseMapping == null)
 {
 grammaticalCaseMapping = new
Dictionary<string, EGrammaticalCase>();

 foreach (EGrammaticalCase speechPart in
Enum.GetValues(typeof(EGrammaticalCase)))
 {

grammaticalCaseMapping.Add(speechPart.GetUkrain
ianValue().ToLower(), speechPart);
 }
 }

 return grammaticalCaseMapping;
 }
 }

 public static string GetUkrainianValue(this
ESpeechPart eSpeechPart)
 {
 var type = typeof(ESpeechPart);
 var memInfo =
type.GetMember(eSpeechPart.ToString());
 var attributes =
memInfo[0].GetCustomAttributes(typeof(Descripti
onAttribute), false);
 return
((DescriptionAttribute)attributes[0]).Descripti
on;
 }

 public static string GetUkrainianValue(this
EGrammaticalGender eGrammaticalGender)
 {
 var type = typeof(EGrammaticalGender);
 var memInfo =
type.GetMember(eGrammaticalGender.ToString());
 var attributes =
memInfo[0].GetCustomAttributes(typeof(Descripti
onAttribute), false);
 return
((DescriptionAttribute)attributes[0]).Descripti
on;
 }

 public static string GetUkrainianValue(this
EGrammaticalCase eGrammaticalCase)
 {
 var type = typeof(EGrammaticalCase);
 var memInfo =
type.GetMember(eGrammaticalCase.ToString());
 var attributes =
memInfo[0].GetCustomAttributes(typeof(Descripti
onAttribute), false);
 return
((DescriptionAttribute)attributes[0]).Descripti
on;
 }
 }
}
```

Лістинг AIS.TermSearcher\Entities\Section\INewSection.cs

```
using AIS.DocxReader.Entities.Section;
using System.Collections.Generic;

namespace AIS.TermSearcher.Entities.Section
{
 public interface INewSection : ISection
 {
 List<Term.Term> KeyTerms { get; set; }
 }
}
```

Лістинг AIS.TermSearcher\Entities\Section\NewSection.cs

```
using AIS.DocxReader.Entities.Section;
using System.Collections.Generic;
using System.Linq;

namespace AIS.TermSearcher.Entities.Section
{
 public class NewSection :
DocxReader.Entities.Section.Section,
INewSection
 {
 public List<Term.Term> KeyTerms { get; set; }

 public NewSection() { }

 public NewSection(ISection section)
 {
 Level = section.Level;
 Text = section.Text;
 Name = section.Name;
 Guid = section.Guid;
 Paragraphs = section.Paragraphs;
 Sections = new List<ISection>();

 if (section.Sections != null &&
section.Sections.Any())
 {
 foreach (var item in section.Sections)
 {
 Sections.Add(new NewSection(item));
```

```
 }
 }
 }
 }
}
```

Лістинг AIS.TermSearcher\Entities\Term\Term.cs

```
using System.Collections.Generic;
using System.Linq;

namespace AIS.TermSearcher.Entities.Term
{
 public class Term
 {
 private string text;
 private string infinitive;

 public string Text
 {
 get
 {
 if (string.IsNullOrEmpty(this.text))
 {
 this.text = string.Join(" ",
this.Words.Select(x => x.Text));
 }

 return this.text;
 }
 }

 public string Infinitive
 {
 get
 {
 if (string.IsNullOrEmpty(this.infinitive))
 {
 this.infinitive = string.Join(" ",
this.Words.Select(x => x.Infinitive));
 }

 return this.infinitive;
 }
 }

 public double Evaluation { get; set; }

 public List<Term> AllExistedForm { get; set; }

 public List<Word.Word> Words { get; set; }

 public int Count { get; set; }

 public int Position { get; set; }

 public Term()
 {

 }

 public Term(string text)
 {
 this.text = text;
 }

 }
}
```

Лістинг AIS.TermSearcher\Entities\Word\EGrammaticalCase.cs

```
using System.ComponentModel;

namespace AIS.TermSearcher.Entities.Word
{
 public enum EGrammaticalCase
 {
 /// <summary>Default</summary>
 [Description("None")]
 None,
 /// <summary>Називний</summary>
 [Description("Називний")]
 Nominative,
 /// <summary>Родовий</summary>
 [Description("Родовий")]
 Genitive,
 /// <summary>Давальний</summary>
 [Description("Давальний")]
 Dative,
 /// <summary>Знахідний </summary>
 [Description("Знахідний ")]
 Accusative,
 /// <summary>Орудний </summary>
 [Description("Орудний ")]
 Instrumental,
 /// <summary>Місцевий</summary>
 [Description("Місцевий")]
 Prepositional,
 /// <summary>Кличний</summary>
 [Description("Кличний")]
 Vocative
 }
}
```

Лістинг AIS.TermSearcher\Entities\Word\EGrammaticalGender.cs

```
using System.ComponentModel;

namespace AIS.TermSearcher.Entities.Word
{
 public enum EGrammaticalGender
 {
 /// <summary>Default</summary>
 [Description("Відсутній")]
 None,
 /// <summary>Чоловічий</summary>
 [Description("Чоловічий")]
 Masculine,
 /// <summary>Жіночий</summary>
 [Description("Жіночий")]
 Feminine,
 /// <summary>Середній</summary>
 [Description("Середній")]
 Neuter
 }
}
```

Лістинг AIS.TermSearcher\Entities\Word\ESpeechPart.cs

```
using System.ComponentModel;

namespace AIS.TermSearcher.Entities.Word
{
 public enum ESpeechPart
 {
 /// <summary>Default</summary>
 [Description("None")]
 None,
 /// <summary>Іменник</summary>
 [Description("Іменник")]
 Noun,
 /// <summary>Прикметник</summary>
 [Description("Прикметник")]
 Adjective,
 /// <summary>Числівник</summary>
 [Description("Числівник")]
 Numeral,
 /// <summary>Займенник</summary>
 [Description("Займенник")]
 Pronoun,
 /// <summary>Дієслово</summary>
 [Description("Дієслово")]
 Verb,
 /// <summary>Дієприкметник</summary>
 [Description("Дієприкметник")]
 Participle,
 /// <summary>Дієприслівник</summary>
 [Description("Дієприслівник")]
 Transgressive,
 /// <summary>Прислівник</summary>
 [Description("Прислівник")]
 Adverb,
 /// <summary>Сполучник</summary>
 [Description("Сполучник")]
 Conjunction,
 /// <summary>Прийменник</summary>
 [Description("Прийменник")]
 Preposition,
 /// <summary>Частка</summary>
 [Description("Частка")]
 Particle,
 }
}
```

Лістинг AIS.TermSearcher\Entities\Word\Word.cs

```
using AIS.DocxReader.Entities.Word;

namespace AIS.TermSearcher.Entities.Word
{
 public class Word : IWord
 {
 public string Text { get; set; }

 public string Infinitive { get; set; }

 public IWordStyle WordStyle { get; set; }

 public bool IsFromDb { get; set; }

 public int Position { get; set; }

 public EGrammaticalGender GrammaticalGender {
get; set; }

 public EGrammaticalCase GrammaticalCase { get;
set; }

 public ESpeechPart SpeechPart { get; set; }
 }
}
```

Лістинг AIS.TermSearcher\Properties\AssemblyInfo.cs

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// General Information about an assembly is
controlled through the following
// set of attributes. Change these attribute
values to modify the information
// associated with an assembly.
[assembly:
AssemblyTitle("AIS.KeyTermRetriever")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly:
AssemblyProduct("AIS.KeyTermRetriever")]
[assembly: AssemblyCopyright("Copyright ©
2018")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types
in this assembly not visible
// to COM components. If you need to access a
type in this assembly from
// COM, set the ComVisible attribute to true on
that type.
[assembly: ComVisible(false)]

// The following GUID is for the ID of the
typelib if this project is exposed to COM
[assembly: Guid("bfb9d0ee-c2a4-4b73-9421-
8fe674e766c2")]

// Version information for an assembly consists
of the following four values:
//
// Major Version
// Minor Version
// Build Number
// Revision
//
// You can specify all the values or you can
default the Build and Revision Numbers
// by using the '*' as shown below:
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

Лістинг AIS.TermSearcher\Searcher\ISearchManager.cs

```
using AIS.DocxReader.Entities.Document;
using AIS.TermSearcher.Entities.Term;
using AIS.TermSearcher.Entities.Word;
using System;
using System.Collections.Generic;

namespace AIS.TermSearcher.Searcher
{
 public interface ISearchManager
 {
 Dictionary<Guid, List<Term>> KeyTerms { get; }

 Dictionary<Guid, List<Term>> CompresedKeyTerms
{ get; }

 Dictionary<Guid, List<Term>>
KeyTermAfterAbsorption { get; }

 Dictionary<Guid, List<Term>> KeyTermMaskResult
{ get; }

 Dictionary<Guid, List<Term>> NauseaResult {
get; }

 List<Word> UnknownWords { get; }

 IDocument AnalizedDocument { get; }

 void Search();
 }
}
```

Лістинг AIS.TermSearcher\Searcher\SearchManager.cs

```
using AIS.DocxReader.Entities.Document;
using AIS.DocxReader.Entities.Section;
using AIS.TermSearcher.Entities.Section;
using AIS.TermSearcher.Entities.Term;
using AIS.TermSearcher.Entities.Word;
using AIS.TermSearcher.Searcher.Settings;
```

```
using
AIS.TermSearcher.Searcher.TermMaskValidation;
using AIS.TermSearcher.Searcher.TermValuation;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;

namespace AIS.TermSearcher.Searcher
{
 class SearchManager : ISearchManager
 {
 #region Fields and Properties
 private readonly IMaskValidator maskValidator;
 private readonly IEvaluator evaluator;
 private readonly SearchSetting searchSetting;
 private readonly int wordCountInTerm;
 private readonly IDocument document;
 private Dictionary<string, Word> wordsInfo;

 private Dictionary<Guid, List<Term>> keyTerms;
 private Dictionary<Guid, List<Term>>
keyTermMaskResult;
 private Dictionary<Guid, List<Term>>
keyTermAfterAbsorption;
 private Dictionary<Guid, List<Term>>
compresedKeyTerms;
 private Dictionary<Guid, List<Term>>
correctedKeyTerms;
 private Dictionary<Guid, List<Term>>
nauseaResult;

 public Dictionary<Guid, List<Term>> KeyTerms
 {
 get
 {
 return this.keyTerms;
 }
 }

 public Dictionary<Guid, List<Term>>
CompresedKeyTerms
 {
 get
 {
 return this.compresedKeyTerms;
 }
 }

 public Dictionary<Guid, List<Term>>
CorrectedKeyTerms
 {
 get
 {
 return this.correctedKeyTerms;
 }
 }

 public Dictionary<Guid, List<Term>>
KeyTermAfterAbsorption
 {
 get
 {
 return this.keyTermAfterAbsorption;
 }
 }

 public Dictionary<Guid, List<Term>>
KeyTermMaskResult
 {
 get
 {
 return this.keyTermMaskResult;
 }
 }

 public Dictionary<Guid, List<Term>>
NauseaResult
 {
 get
 {
 return this.nauseaResult;
 }
 }

 public IDocument AnalizedDocument =>
this.document;

 public List<Word> UnknownWords =>
this.wordsInfo.Values.Where(x =>
!x.IsFromDb).ToList();
 #endregion

 public SearchManager(IMaskValidator
maskValidator, IEvaluator evaluator, IDocument
document, SearchSetting searchSetting, int
wordCountInTerm)
 {
 if (document == null)
 throw new
ArgumentNullException(nameof(document));
 if (evaluator == null)
 throw new
ArgumentNullException(nameof(evaluator));
 if (maskValidator == null)
 throw new
ArgumentNullException(nameof(maskValidator));
 if (searchSetting == null)
 throw new
ArgumentNullException(nameof(searchSetting));

 this.evaluator = evaluator;
 this.searchSetting = searchSetting;
 this.wordCountInTerm = wordCountInTerm;
 this.maskValidator = maskValidator;
 this.wordsInfo = new Dictionary<string,
Word>();

 this.document = new Document
 {
 Name = document.Name,
 Sections = new List<ISection>()
 };

 foreach (var section in document.Sections)
 {
 this.document.Sections.Add(new
NewSection(section));
 }
 }

 public void Search()
 {
 this.keyTerms = new Dictionary<Guid,
List<Term>>();
 this.keyTermMaskResult = new Dictionary<Guid,
List<Term>>();
 this.compresedKeyTerms = new Dictionary<Guid,
List<Term>>();
 this.correctedKeyTerms = new Dictionary<Guid,
List<Term>>();
 this.keyTermAfterAbsorption = new
Dictionary<Guid, List<Term>>();
 this.nauseaResult = new Dictionary<Guid,
List<Term>>();

 foreach (var section in
this.document.Sections)
 {
 SearchSubSection(section as INewSection);
 }
 }

 private void SearchSubSection(INewSection
section)
 {
 foreach (var subSection in section.Sections)
 {
 SearchSubSection(subSection as INewSection);
 }

 DetermineKeyword(section);
 SetWordInfo(section);
 FilterTerm(section);
 CompressTerms(section);
 CompactifyTerm(section);
 CorrectEvaliation(section,
this.searchSetting.CorrectionSetting);
 NauseaTerm(section);
 }

 private void DetermineKeyword(INewSection
section)
 {
 List<Term> keyTerms = new List<Term>();
 for (int i = 1; i <= this.wordCountInTerm;
i++)
 {
 List<Term> result =
this.evaluator.Evaluate(section, i);
 keyTerms.AddRange(result);
 }

 this.keyTerms.Add(section.Guid, keyTerms);

 section.KeyTerms = keyTerms;
 }

 private void SetWordInfo(INewSection section)
 {
 List<Word> allWords = new List<Word>();
 foreach (var term in section.KeyTerms)
 {
 allWords.AddRange(term.Words);
 }

 List<string> uniqueWords = allWords
 .ToLookup(word => word.Text)
 .Select(a => a.First().Text)
 .ToList();

 using (DbProvider dbProvider = new
DbProvider())
 {
 Dictionary<string, Word> newWordsInfo =
dbProvider.GetWordsInfo(uniqueWords.Except(this
.wordsInfo.Keys).ToList());
 foreach (var word in newWordsInfo)
 {
 this.wordsInfo.Add(word.Key, word.Value);
 }
 }

 foreach (var term in section.KeyTerms)
 {
 foreach (var word in term.Words)
 {
 Word wordInfo = this.wordsInfo[word.Text];

 word.IsFromDb = true;
 word.Infinitive = wordInfo.Infinitive;
 word.GrammaticalGender =
wordInfo.GrammaticalGender;
 word.GrammaticalCase =
wordInfo.GrammaticalCase;
 word.SpeechPart = wordInfo.SpeechPart;
 }
 }
 }

 private void FilterTerm(INewSection section)
 {
 section.KeyTerms =
this.maskValidator.Validate(section.KeyTerms);
 this.keyTermMaskResult.Add(section.Guid,
section.KeyTerms);
 }

 private void CompressTerms(INewSection
section)
 {
 List<Term> compresedKeyTerm = new
List<Term>();
 List<string> uniqueInfinitive =
section.KeyTerms.Select(x =>
x.Infinitive).Distinct().ToList();

 foreach (var infinitive in uniqueInfinitive)
 {
 var keyTermForms = section.KeyTerms.Where(x =>
x.Infinitive == infinitive).ToList();
 double maxEvaluation = keyTermForms.Max(x =>
x.Evaluation);
 if (maxEvaluation == 0)
 {
 continue;
 }

 int count = keyTermForms.Sum(x => x.Count);
 Term readableForm =
GetReadableKeyTermForm(keyTermForms);
 if (readableForm == null)
 {
 readableForm = keyTermForms.First(x =>
x.Evaluation == maxEvaluation);
 }

 readableForm.AllExistedForm = keyTermForms;
 readableForm.Count = count;
 readableForm.Evaluation = maxEvaluation;
 readableForm.Words = readableForm.Words;

 compresedKeyTerm.Add(readableForm);
 }

 section.KeyTerms = compresedKeyTerm;
 this.compresedKeyTerms.Add(section.Guid,
compresedKeyTerm);
 }

 private void CompactifyTerm(INewSection
section)
 {
 List<Term> termsToSave = new List<Term>();

 for (int i = 1; i < this.wordCountInTerm; i++)
 {
 var smallTerms = section.KeyTerms.Where(t =>
t.Words.Count == i);
 var bigTerms = section.KeyTerms.Where(t =>
t.Words.Count == i + 1);

 foreach (var smallTerm in smallTerms)
 {
 var hasBigVersion = bigTerms.FirstOrDefault(t
=>
 new
Regex($"^{smallTerm.Infinitive}\\s|\\s{smallTer
m.Infinitive}$").IsMatch(t.Infinitive)
 && smallTerm.Count <= t.Count * 2);

 if (hasBigVersion == null)
```

```
 {
 termsToSave.Add(smallTerm);
 }
 }
 }

 section.KeyTerms = termsToSave;
 this.keyTermAfterAbsorption.Add(section.Guid,
termsToSave);
 }

 private void CorrectEvaliation(INewSection
section, CorrectionSetting correctionSetting)
 {
 List<Term> correctedKeyTerm = new
List<Term>();

 foreach (var term in section.KeyTerms)
 {
 bool isBold = term.Words.Any(word =>
word.WordStyle?.Bold != null);
 bool isItalic = term.Words.Any(word =>
word.WordStyle?.Italic != null);
 bool isUnderline = term.Words.Any(word =>
word.WordStyle?.Underline != null);

 if (isBold)
 term.Evaluation *=
correctionSetting.BoldFontCoefficient;

 if (isItalic)
 term.Evaluation *=
correctionSetting.ItalicFontCoefficient;

 if (isUnderline)
 term.Evaluation *=
correctionSetting.UnderlineFontCoefficient;

 correctedKeyTerm.Add(term);
 }

 this.correctedKeyTerms.Add(section.Guid,
correctedKeyTerm);
 }

 private void NauseaTerm(INewSection section)
 {
 int termCount = (int)(section.KeyTerms.Count *
0.15);

 List<Term> nausea = section.KeyTerms
 .OrderByDescending(x => x.Evaluation)
 .Take(termCount)
 .ToList();

 section.KeyTerms = nausea;
 this.nauseaResult.Add(section.Guid, nausea);
 }

 private Term GetReadableKeyTermForm(List<Term>
options)
 {
 Term firstTerm = options.First();

 if (firstTerm.Words.Count == 1)
 {
 return new Term(firstTerm.Infinitive)
 {
 Words = firstTerm.Words
 };
 }

 foreach (var keyTerm in options)
 {
 var selectedKeyTerm =
keyTerm.Words.FirstOrDefault(x => x.SpeechPart
== ESpeechPart.Noun && x.Text == x.Infinitive);
 if (selectedKeyTerm != null)
 {
 return keyTerm;
 }
 }

 return null;
 }
 }
}
```

Лістинг AIS.TermSearcher\Searcher\TermSearcherFactory.cs

```
using AIS.DocxReader.Entities.Document;
using AIS.TermSearcher.Searcher.Settings;
using
AIS.TermSearcher.Searcher.TermMaskValidation;
using AIS.TermSearcher.Searcher.TermValuation;

namespace AIS.TermSearcher.Searcher
{
 public static class TermSearcherFactory
 {
 public static ISearchManager
CreateSearchManager(IDocument document, int
wordInTermCount, SearchSetting searchSetting)
 {
 VarianceEvaluator varianceEvaluator = new
VarianceEvaluator();
 UkranianMaskValidator mask = new
UkranianMaskValidator();

 return new SearchManager(mask,
varianceEvaluator, document, searchSetting,
wordInTermCount);
 }
 }
}
```

Лістинг AIS.TermSearcher\Searcher\Settings\CorrectionSetting.cs

```
namespace AIS.TermSearcher.Searcher.Settings
{
 public class CorrectionSetting
 {
 public float BoldFontCoefficient { get; set; }

 public float ItalicFontCoefficient { get; set;
}

 public float UnderlineFontCoefficient { get;
set; }
 }
}
```

Лістинг AIS.TermSearcher\Searcher\Settings\SearchSetting.cs

```
using AIS.TermSearcher.Searcher.Settings;

namespace AIS.TermSearcher.Searcher.Settings
{
 public class SearchSetting
 {
 public CorrectionSetting CorrectionSetting {
get; set; }
 }
}
```

Лістинг AIS.TermSearcher\Searcher\TermMaskValidation\DbProvider.cs

```
using AIS.TermSearcher.Entities.Word;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using static
AIS.TermSearcher.Entities.EntitiesHelper;

namespace
AIS.TermSearcher.Searcher.TermMaskValidation
{
 public class DbProvider : IDisposable
 {
 private readonly SqlConnection connection;

 public DbProvider()
 {
 string connectionString = @"Data
Source=(local);Initial
Catalog=TESTWORDSSQL;Trusted_Connection=true";
 this.connection = new
SqlConnection(connectionString);
 this.connection.Open();
 }

 public Dictionary<string, Word>
GetWordsInfo(List<string> words)
 {
 Dictionary<string, Word> wordsInfo = new
Dictionary<string, Word>();

 if (words.Any())
 {
 string query = MakeWordsQuery(words);
 SqlCommand command = new SqlCommand(query,
this.connection);

 using (SqlDataReader reader =
command.ExecuteReader())
 {
 while (reader.Read())
 {
 string key =
reader["Word"].ToString().ToLower();
 if (wordsInfo.ContainsKey(key))
 {
 continue;
 }

 Word word = new Word
 {
 IsFromDb = true,
 Infinitive = reader["Infinitive"].ToString(),
 GrammaticalGender =
GetGrammaticalGender(reader["GrammaticalGender"
].ToString()),
 GrammaticalCase =
GetGrammaticalCase(reader["GrammaticalCase"].To
String()),
 SpeechPart =
GetSpeechPart(reader["SpeechPart"].ToString())
 };
 wordsInfo.Add(key, word);
 }
 }

 List<string> unknownWords =
words.Except(wordsInfo.Keys).ToList();
 foreach (var word in unknownWords)
 {
 wordsInfo.Add(word, new Word()
 {
 IsFromDb = false,
 Text = word,
 GrammaticalGender = EGrammaticalGender.None,
 GrammaticalCase = EGrammaticalCase.None,
 SpeechPart = ESpeechPart.None
 });
 }
 }

 return wordsInfo;
 }

 private string MakeWordsQuery(List<string>
words)
 {
 string wordsQuery =
string.Format(SqlQueryResources.Words,
words.First().ToLower().Replace("'", "''"));

 foreach (var word in words.Skip(1))
 {
 wordsQuery += " OR " +
string.Format(SqlQueryResources.Words,
word.ToLower().Replace("'", "''"));
 }

 return
string.Format(SqlQueryResources.GetWordInfo,
wordsQuery);
 }

 private EGrammaticalGender
GetGrammaticalGender(string grammaticalGender)
 {
 return
GrammaticalGenderMapping[grammaticalGender.ToLo
wer()];
 }

 private ESpeechPart GetSpeechPart(string
speechPart)
 {
 speechPart = speechPart.ToLower();

 if (SpeechPartMapping.ContainsKey(speechPart))
 {
 return SpeechPartMapping[speechPart];
 }
 else
 {
 return ESpeechPart.None;
 }
 }

 private EGrammaticalCase
GetGrammaticalCase(string grammaticalCase)
 {
 grammaticalCase = grammaticalCase.ToLower();

 if
(GrammaticalCaseMapping.ContainsKey(grammatical
Case))
 {
 return
GrammaticalCaseMapping[grammaticalCase];
 }
 else
 {
 return EGrammaticalCase.None;
 }
 }

 public void Dispose()
 {
 this.connection.Close();
 }
```

```
 }
}
```

Лістинг
AIS.TermSearcher\Searcher\TermMaskValidation\IMaskValidator.cs

```
using AIS.TermSearcher.Entities.Term;
using System.Collections.Generic;

namespace
AIS.TermSearcher.Searcher.TermMaskValidation
{
 public interface IMaskValidator
 {
 List<Term> Validate(List<Term> terms);
 }
}
```

Лістинг
AIS.TermSearcher\Searcher\TermMaskValidation\SqlQueryResources.Designer.cs

```
//------------------------------------------------
--------------------------------
// <auto-generated>
// This code was generated by a tool.
// Runtime Version:4.0.30319.42000
//
// Changes to this file may cause incorrect
behavior and will be lost if
// the code is regenerated.
// </auto-generated>
//------------------------------------------------
--------------------------------

namespace
AIS.TermSearcher.Searcher.TermMaskValidation {
 using System;


 /// <summary>
 /// A strongly-typed resource class, for
looking up localized strings, etc.
 /// </summary>
 // This class was auto-generated by the
StronglyTypedResourceBuilder
 // class via a tool like ResGen or Visual
Studio.
 // To add or remove a member, edit your .ResX
file then rerun ResGen
 // with the /str option, or rebuild your VS
project.

[global::System.CodeDom.Compiler.GeneratedCodeA
ttribute("System.Resources.Tools.StronglyTypedR
esourceBuilder", "15.0.0.0")]

[global::System.Diagnostics.DebuggerNonUserCode
Attribute()]

[global::System.Runtime.CompilerServices.Compil
erGeneratedAttribute()]
 internal class SqlQueryResources {

 private static
global::System.Resources.ResourceManager
resourceMan;

 private static
global::System.Globalization.CultureInfo
resourceCulture;


[global::System.Diagnostics.CodeAnalysis.Suppre
ssMessageAttribute("Microsoft.Performance",
"CA1811:AvoidUncalledPrivateCode")]
 internal SqlQueryResources() {
 }

 /// <summary>
 /// Returns the cached ResourceManager
instance used by this class.
 /// </summary>

[global::System.ComponentModel.EditorBrowsableA
ttribute(global::System.ComponentModel.EditorBr
owsableState.Advanced)]
 internal static
global::System.Resources.ResourceManager
ResourceManager {
 get {
 if (object.ReferenceEquals(resourceMan, null))
{
 global::System.Resources.ResourceManager temp
= new
global::System.Resources.ResourceManager("AIS.T
ermSearcher.Searcher.TermMaskValidation.SqlQuer
yResources",
typeof(SqlQueryResources).Assembly);
 resourceMan = temp;
 }
 return resourceMan;
 }
 }

 /// <summary>
 /// Overrides the current thread's
CurrentUICulture property for all
 /// resource lookups using this strongly typed
resource class.
 /// </summary>

[global::System.ComponentModel.EditorBrowsableA
ttribute(global::System.ComponentModel.EditorBr
owsableState.Advanced)]
 internal static
global::System.Globalization.CultureInfo
Culture {
 get {
 return resourceCulture;
 }
 set {
 resourceCulture = value;
 }
 }

 /// <summary>
 /// Looks up a localized string similar to
SELECT word.[Fullword] [Word],
 /// speechPart.[Name] [SpeechPart],
 /// inflexion.Inflexion [GrammaticalCase],
 /// gender.Gender [GrammaticalGender],
 /// infinitive.Word Infinitive,
 /// speechPart.Importance
 ///FROM Infinitives infinitive
 /// INNER JOIN AllWords word ON
infinitive.Id_Infinitives = word.Id_Infinitives
 /// INNER JOIN PartOfSpeach speechPart ON
speechPart.Id_PartOfSpeach =
infinitive.Id_PartOfSpeach
 /// INNER JOIN Inflexions inflexion
 /// ON word.Id_Inflexions =
inflexion.Id_inflexion
 /// OR inflexio [rest of string was
truncated]&amp;quot;;.
 /// </summary>
 internal static string GetWordInfo {
 get {
 return
ResourceManager.GetString("GetWordInfo",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to
word.Fullword = &amp;apos;{0}&amp;apos;.
 /// </summary>
 internal static string Words {
 get {
 return ResourceManager.GetString("Words",
resourceCulture);
 }
 }
 }
}
```

Лістинг
AIS.TermSearcher\Searcher\TermMaskValidation\UkranianMaskValidator.cs

```
using AIS.TermSearcher.Entities.Term;
using AIS.TermSearcher.Entities.Word;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;

namespace
AIS.TermSearcher.Searcher.TermMaskValidation
{
 public class UkranianMaskValidator :
IMaskValidator
 {
 public List<Term> Validate(List<Term> terms)
 {
 List<Term> resultTerm = new List<Term>();

 foreach (var term in terms)
 {
 int wordCount = term.Words.Count;

 if (!term.Words.Any(word => word.SpeechPart ==
ESpeechPart.Noun))
 continue;

 if (term.Words.First().SpeechPart !=
ESpeechPart.Noun &&
term.Words.First().SpeechPart !=
ESpeechPart.Adjective)
 continue;

 if (term.Words.Last().SpeechPart !=
ESpeechPart.Noun &&
term.Words.Last().SpeechPart !=
ESpeechPart.Adjective)
 continue;

 int n = 0;
 for (int i = 0; i < wordCount; i++)
 {
 if (term.Words[i].SpeechPart ==
ESpeechPart.Noun
 || term.Words[i].SpeechPart ==
ESpeechPart.Adjective
 || term.Words[i].SpeechPart ==
ESpeechPart.Conjunction
 || term.Words[i].SpeechPart ==
ESpeechPart.Particle)
 {
 n++;
 }
 }

 if (n == wordCount)
 {
 resultTerm.Add(term);
 }
 }

 return resultTerm;
 }
 }
}
```

Лістинг
AIS.TermSearcher\Searcher\TermValuation\IEvaluator.cs

```
using AIS.TermSearcher.Entities.Section;
using AIS.TermSearcher.Entities.Term;
using System.Collections.Generic;

namespace
AIS.TermSearcher.Searcher.TermValuation
{
 public interface IEvaluator
 {
 List<Term> Evaluate(INewSection section, int
wordCount);
 }
}
```

Лістинг
AIS.TermSearcher\Searcher\TermValuation\VarianceEvaluator.cs

```
using AIS.TermSearcher.Entities.Section;
using AIS.TermSearcher.Entities.Term;
using AIS.TermSearcher.Entities.Word;
using System;
using System.Collections.Generic;
using System.Linq;

namespace
AIS.TermSearcher.Searcher.TermValuation
{
 public class VarianceEvaluator : IEvaluator
 {
 public List<Term> Evaluate(INewSection
section, int wordCount)
 {
 List<Term> resultValuation = new List<Term>();
 List<Term> termsCondidates =
GetTermsCandidates(section, wordCount);

 if (!termsCondidates.Any())
 {
 return resultValuation;
 }

 int lastTermPosition =
termsCondidates.Last().Position;
 termsCondidates.Sort((term1, term2) =>
string.Compare(term1.Text, term2.Text));

 List<Term> uniqueTermsCondidates =
termsCondidates
 .ToLookup(term => term.Text)
 .Select(a => a.First())
 .ToList();

 foreach (var uniqueTerm in
uniqueTermsCondidates)
 {
 double evaluation;
 int sumDist = 0, sumDistSquer = 0;

 List<Term> selectedTerms =
BinariSearch(termsCondidates, uniqueTerm.Text);
 selectedTerms = selectedTerms.OrderBy(a =>
a.Position).ToList();
```

```
 int termCount = selectedTerms.Count;
 if (termCount == 1)
 {
 evaluation = 0;
 }
 else
 {
 for (int j = 0; j < selectedTerms.Count - 1;
j++)
 {
 sumDist += (selectedTerms[j + 1].Position -
selectedTerms[j].Position) - wordCount;
 sumDistSquer += (int)Math.Pow((selectedTerms[j
+ 1].Position - selectedTerms[j].Position) -
wordCount, 2);
 }

 sumDist += (lastTermPosition -
selectedTerms.Last().Position +
selectedTerms.First().Position) - wordCount;
 sumDistSquer +=
(int)Math.Pow((lastTermPosition -
selectedTerms.Last().Position +
selectedTerms.First().Position) - wordCount,
2);

 evaluation = Math.Sqrt(((double)sumDistSquer /
(selectedTerms.Count)) -
Math.Pow(((double)sumDist /
(selectedTerms.Count)), 2)) / ((double)sumDist
/ (selectedTerms.Count));
 }

 if (evaluation > 0)
 {
 resultValuation.Add(new Term()
 {
 Evaluation = evaluation,
 Count = selectedTerms.Count,
 Words = uniqueTerm.Words
 });
 }
 }

 return resultValuation.OrderByDescending(p =>
p.Evaluation).ToList();
 }

 public List<Term>
GetTermsCandidates(INewSection section, int
wordCount)
 {
 List<Word> words;
 List<Term> termsCondidates = new List<Term>();
 int position = 0;

 foreach (var paragraph in
section.AllParagraphs)
 {
 foreach (var sentences in paragraph.Sentences)
 {
 foreach (var textContainer in
sentences.TextContainers)
 {
 words = textContainer.Words.Select(word => new
Word()
 {
 Text = word.Text.ToLower(),
 WordStyle = word.WordStyle
 }).ToList();

 for (int i = 0; i < words.Count; i++)
 {
 Term term = new Term()
 {
 Words = new List<Word>(),
 Position = position
 };

 for (int j = 0; j < wordCount && i + j <
words.Count; j++)
 {
 term.Words.Add(words[i + j]);
 }

 if (term.Words.Count == wordCount)
 {
 termsCondidates.Add(term);
 }

 position++;
 }
 }
 }
 }

 return termsCondidates;
 }

 List<Term> BinariSearch(List<Term>
allKeywordOptions, string value)
 {
 List<Term> option = new List<Term>();
 int start = 0, end = allKeywordOptions.Count;
 while (start < end)
 {
 int center = (start + end) / 2;
 switch (String.Compare(value,
allKeywordOptions[center].Text))
 {
 case -1:
 end = center;
 break;
 case 0:
 int incriment = 0;
 bool left = true, rigth = true;
 while (left || rigth)
 {
 if (left && (center - incriment < 0 ||
allKeywordOptions[center - incriment].Text !=
value))
 left = false;
 if (rigth && (center + incriment + 1 >=
allKeywordOptions.Count ||
allKeywordOptions[center + incriment + 1].Text
!= value))
 rigth = false;
 if (left)
 option.Add(allKeywordOptions[center -
incriment]);
 if (rigth)
 option.Add(allKeywordOptions[center +
incriment + 1]);
 incriment++;
 }

 return option;
 case 1: start = center + 1; break;
 }
 }

 return option;
 }
 }
}
```

Лістинг
AIS.TermSearcher.GUITests\Messages.Designer.cs

```
//------------------------------------------------
--------------------------------
// <auto-generated>
// This code was generated by a tool.
// Runtime Version:4.0.30319.42000
//
// Changes to this file may cause incorrect
behavior and will be lost if
// the code is regenerated.
// </auto-generated>
//------------------------------------------------
--------------------------------

namespace AIS.TermSearcher.GUITests {
 using System;


 /// <summary>
 /// A strongly-typed resource class, for
looking up localized strings, etc.
 /// </summary>
 // This class was auto-generated by the
StronglyTypedResourceBuilder
 // class via a tool like ResGen or Visual
Studio.
 // To add or remove a member, edit your .ResX
file then rerun ResGen
 // with the /str option, or rebuild your VS
project.

[global::System.CodeDom.Compiler.GeneratedCodeA
ttribute("System.Resources.Tools.StronglyTypedR
esourceBuilder", "16.0.0.0")]

[global::System.Diagnostics.DebuggerNonUserCode
Attribute()]

[global::System.Runtime.CompilerServices.Compil
erGeneratedAttribute()]
 internal class Messages {

 private static
global::System.Resources.ResourceManager
resourceMan;

 private static
global::System.Globalization.CultureInfo
resourceCulture;

[global::System.Diagnostics.CodeAnalysis.Suppre
ssMessageAttribute("Microsoft.Performance",
"CA1811:AvoidUncalledPrivateCode")]
 internal Messages() {
 }

 /// <summary>
 /// Returns the cached ResourceManager
instance used by this class.
 /// </summary>

[global::System.ComponentModel.EditorBrowsableA
ttribute(global::System.ComponentModel.EditorBr
owsableState.Advanced)]
 internal static
global::System.Resources.ResourceManager
ResourceManager {
 get {
 if (object.ReferenceEquals(resourceMan, null))
{
 global::System.Resources.ResourceManager temp
= new
global::System.Resources.ResourceManager("AIS.T
ermSearcher.GUITests.Messages",
typeof(Messages).Assembly);
 resourceMan = temp;
 }
 return resourceMan;
 }
 }

 /// <summary>
 /// Overrides the current thread's
CurrentUICulture property for all
 /// resource lookups using this strongly typed
resource class.
 /// </summary>

[global::System.ComponentModel.EditorBrowsableA
ttribute(global::System.ComponentModel.EditorBr
owsableState.Advanced)]
 internal static
global::System.Globalization.CultureInfo
Culture {
 get {
 return resourceCulture;
 }
 set {
 resourceCulture = value;
 }
 }

 /// <summary>
 /// Looks up a localized string similar to The
element does not exist.
 /// </summary>
 internal static string ElementDoesNotExist {
 get {
 return
ResourceManager.GetString("ElementDoesNotExist"
, resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to The
element location is wrong.
 /// </summary>
 internal static string ElementLocationIsWrong
{
 get {
 return
ResourceManager.GetString("ElementLocationIsWro
ng", resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to The
element size is wrong.
 /// </summary>
 internal static string ElementSizeIsWrong {
 get {
 return
ResourceManager.GetString("ElementSizeIsWrong",
resourceCulture);
 }
 }
 }
}
```

Лістинг
AIS.TermSearcher.GUITests\SeleniumHelper.cs

```
using OpenQA.Selenium;

namespace AIS.TermSearcher.GUITests
{
 internal class SeleniumHelper :
NUnit.Framework.Assert
```

```
 {
 private readonly IWebDriver driver;

 public SeleniumHelper(IWebDriver driver)
 {
 this.driver = driver;
 }

 internal bool IsElementExist(By by)
 {
 var elements = driver.FindElements(by);

 return elements.Count > 0;
 }
 }
}
```

Лістинг AIS.TermSearcher.GUITests\TermSearcherUITests.cs

```
using NUnit.Framework;
using OpenQA.Selenium;
using System.Drawing;

namespace AIS.TermSearcher.GUITests
{
 internal class TermSearcherUITests : TestsBase
 {
 [Test]
 public void
Check_if_setting_page_elements_exist()
 {

Assert.IsTrue(SeleniumHelper.IsElementExist(By.
Id("setting-view")),
Messages.ElementDoesNotExist);

Assert.IsTrue(SeleniumHelper.IsElementExist(By.
Id("file-on-server")),
Messages.ElementDoesNotExist);

Assert.IsTrue(SeleniumHelper.IsElementExist(By.
Id("file-loader-input")),
Messages.ElementDoesNotExist);

Assert.IsTrue(SeleniumHelper.IsElementExist(By.
Id("search-stage")),
Messages.ElementDoesNotExist);

Assert.IsTrue(SeleniumHelper.IsElementExist(By.
Id("bold-value")),
Messages.ElementDoesNotExist);

Assert.IsTrue(SeleniumHelper.IsElementExist(By.
Id("italic-value")),
Messages.ElementDoesNotExist);

Assert.IsTrue(SeleniumHelper.IsElementExist(By.
Id("underline-value")),
Messages.ElementDoesNotExist);

Assert.IsTrue(SeleniumHelper.IsElementExist(By.
Id("searchButton")),
Messages.ElementDoesNotExist);
 }

 [Test]
 public void
Check_location_of_setting_page_elements()
 {
 var fileOnServerElement =
driver.FindElement(By.Id("file-on-server"));
 Assert.AreEqual(new Point(55, 5),
fileOnServerElement.Location,
Messages.ElementLocationIsWrong);

 var settingViewElement =
driver.FindElement(By.Id("setting-view"));
 Assert.AreEqual(new Point(50, 0),
settingViewElement.Location,
Messages.ElementLocationIsWrong);

 var fileLoaderInputElement =
driver.FindElement(By.Id("file-loader-input"));
 Assert.AreEqual(new Point(55, 23),
fileLoaderInputElement.Location,
Messages.ElementLocationIsWrong);

 var searchStageElement =
driver.FindElement(By.Id("search-stage"));
 Assert.AreEqual(new Point(60, 67),
searchStageElement.Location,
Messages.ElementLocationIsWrong);

 var boldValueElement =
driver.FindElement(By.Id("bold-value"));
 Assert.AreEqual(new Point(126, 94),
boldValueElement.Location,
Messages.ElementLocationIsWrong);

 var italicValueElement =
driver.FindElement(By.Id("italic-value"));
 Assert.AreEqual(new Point(126, 119),
italicValueElement.Location,
Messages.ElementLocationIsWrong);

 var underlineValueElement =
driver.FindElement(By.Id("underline-value"));
 Assert.AreEqual(new Point(126, 144),
underlineValueElement.Location,
Messages.ElementLocationIsWrong);

 var searchButtonElement =
driver.FindElement(By.Id("searchButton"));
 Assert.AreEqual(new Point(55, 186),
searchButtonElement.Location,
Messages.ElementLocationIsWrong);
 }

 [Test]
 public void
Check_size_of_setting_page_elements()
 {
 var settingViewElement =
driver.FindElement(By.Id("setting-view"));
 Assert.AreEqual(new Size(898, 212),
settingViewElement.Size,
Messages.ElementSizeIsWrong);

 var fileOnServerElement =
driver.FindElement(By.Id("file-on-server"));
 Assert.AreEqual(new Size(176, 17),
fileOnServerElement.Size,
Messages.ElementSizeIsWrong);

 var fileLoaderInputElement =
driver.FindElement(By.Id("file-loader-input"));
 Assert.AreEqual(new Size(253, 21),
fileLoaderInputElement.Size,
Messages.ElementSizeIsWrong);

 var searchStageElement =
driver.FindElement(By.Id("search-stage"));
 Assert.AreEqual(new Size(131, 19),
searchStageElement.Size,
Messages.ElementSizeIsWrong);

 var boldValueElement =
driver.FindElement(By.Id("bold-value"));
 Assert.AreEqual(new Size(177, 21),
boldValueElement.Size,
Messages.ElementSizeIsWrong);

 var italicValueElement =
driver.FindElement(By.Id("italic-value"));
 Assert.AreEqual(new Size(177, 21),
italicValueElement.Size,
Messages.ElementSizeIsWrong);

 var underlineValueElement =
driver.FindElement(By.Id("underline-value"));
 Assert.AreEqual(new Size(177, 21),
underlineValueElement.Size,
Messages.ElementSizeIsWrong);

 var searchButtonElement =
driver.FindElement(By.Id("searchButton"));
 Assert.AreEqual(new Size(173, 21),
searchButtonElement.Size,
Messages.ElementSizeIsWrong);
 }
 }
}
```

Лістинг AIS.TermSearcher.GUITests\TestsBase.cs

```
using NUnit.Framework;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using System;
using System.Configuration;

namespace AIS.TermSearcher.GUITests
{
 internal class TestsBase
 {
 protected IWebDriver driver;

 public SeleniumHelper SeleniumHelper { get;
private set; }

 [SetUp]
 public void SetUp()
 {
 driver = new ChromeDriver();
 SeleniumHelper = new SeleniumHelper(driver);

 if
(Uri.TryCreate(ConfigurationManager.AppSettings
["url"], UriKind.Absolute, out Uri uri))
 {
 driver.Url = uri.ToString();
 }
 else
 {
 throw new ArgumentException(nameof(uri));
 }
 }

 [TearDown]
 public void TearDown()
 {
 driver.Quit();
 }
 }
}
```

Лістинг AIS.TermSearcher.GUITests\Properties\AssemblyInfo.cs

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// General Information about an assembly is
controlled through the following
// set of attributes. Change these attribute
values to modify the information
// associated with an assembly.
[assembly:
AssemblyTitle("AIS.TermSearcher.GUITests")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly:
AssemblyProduct("AIS.TermSearcher.GUITests")]
[assembly: AssemblyCopyright("Copyright ©
2020")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types
in this assembly not visible
// to COM components. If you need to access a
type in this assembly from
// COM, set the ComVisible attribute to true on
that type.
[assembly: ComVisible(false)]

// The following GUID is for the ID of the
typelib if this project is exposed to COM
[assembly: Guid("89c72613-c0b4-45a7-81a2-
a96baeab3d9d")]

// Version information for an assembly consists
of the following four values:
//
// Major Version
// Minor Version
// Build Number
// Revision
//
// You can specify all the values or you can
default the Build and Revision Numbers
// by using the '*' as shown below:
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

Лістинг AIS.TermSearcher.UI\Global.asax.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace AIS.TermSearcher.UI
{
 public class MvcApplication :
System.Web.HttpApplication
 {
 protected void Application_Start()
 {
 AreaRegistration.RegisterAllAreas();
 RouteConfig.RegisterRoutes(RouteTable.Routes);
 }
 }
}
```

Лістинг AIS.TermSearcher.UI\App_Start\RouteConfig.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;
```

```
namespace AIS.TermSearcher.UI
{
 public class RouteConfig
 {
 public static void
RegisterRoutes(RouteCollection routes)
 {

routes.IgnoreRoute("{resource}.axd/{*pathInfo}"
);

 routes.MapRoute(
 name: "Default",
 url: "{controller}/{action}/{id}",
 defaults: new { controller = "Index", action =
"Index", id = UrlParameter.Optional }
 );
 }
 }
}
```

Лістинг AIS.TermSearcher.UI\Controllers\DocumentControl
ler.cs

```
using AIS.DocxReader.Entities.Document;
using AIS.TermSearcher.Searcher;
using AIS.TermSearcher.Searcher.Settings;
using AIS.TermSearcher.UI.Models;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Web.Mvc;

namespace AIS.TermSearcher.UI.Controllers
{
 public class DocumentController : Controller
 {
 public void UploadFile()
 {
 foreach (string upload in this.Request.Files)
 {
 if
(!string.IsNullOrWhiteSpace(this.Request.Files[
upload].FileName))
 {
 string directoryPath =
AppDomain.CurrentDomain.BaseDirectory +
"App_Data\\Uploads";
 string fileName =
Path.GetFileName(this.Request.Files[upload].Fil
eName);

 DocumentManager.FilePath =
Path.Combine(directoryPath, fileName);


this.Request.Files[upload].SaveAs(DocumentManag
er.FilePath);
 }
 }
 }

 public ActionResult Search(float bold, float
italic, float underline)
 {
 DocxReader.DocxReader reader = new
DocxReader.DocxReader(DocumentManager.FilePath)
;
 IDocument document = reader.Read();

 SearchSetting searchSetting = new
SearchSetting
 {
 CorrectionSetting = new CorrectionSetting()
 {
 BoldFontCoefficient = bold,
 ItalicFontCoefficient = italic,
 UnderlineFontCoefficient = underline
 }
 };

 DocumentManager.SearchManager =
TermSearcherFactory.CreateSearchManager(documen
t, 5, searchSetting);
 DocumentManager.SearchManager.Search();
 DocumentManager.Sections =
DocumentManager.CreateSections(document.Section
s);

 return Json(DocumentManager.Sections,
JsonRequestBehavior.AllowGet);
 }

 public ActionResult GetKeyTerms(string guid,
string stage)
 {
 Guid sectionGuid = Guid.Parse(guid);
 Dictionary<Guid, List<Entities.Term.Term>>
structureTerms = null;

 switch (stage)
 {
 case "candidates":
 structureTerms =
DocumentManager.SearchManager.KeyTerms;
 break;
 case "compressed":
 structureTerms =
DocumentManager.SearchManager.CompresedKeyTerms
;
 break;
 case "absorption":
 structureTerms =
DocumentManager.SearchManager.KeyTermAfterAbsor
ption;
 break;
 case "mask":
 structureTerms =
DocumentManager.SearchManager.KeyTermMaskResult
;
 break;
 case "nausea":
 default:
 structureTerms =
DocumentManager.SearchManager.NauseaResult;
 break;
 }

 if (structureTerms.ContainsKey(sectionGuid))
 {
 IEnumerable<Term> terms =
structureTerms[sectionGuid].Select(term => new
Term
 {
 Text = term.Text,
 Evaluation = term.Evaluation,
 Count = term.Count
 });

 return Json(terms,
JsonRequestBehavior.AllowGet);
 }

 return new
HttpStatusCodeResult(HttpStatusCode.NotFound);
 }

 public ActionResult GetFileName()
 {
 if
(string.IsNullOrWhiteSpace(DocumentManager.File
Path))
 {
 return new
HttpStatusCodeResult(HttpStatusCode.NotFound);
 }
 else
 {
 return Json(DocumentManager.FilePath,
JsonRequestBehavior.AllowGet);
 }
 }
 }
}
```

Лістинг
AIS.TermSearcher.UI\Controllers\IndexController
.cs

```
using System.Web.Mvc;

namespace AIS.TermSearcher.UI.Controllers
{
 public class IndexController : Controller
 {
 // GET: Index
 public ActionResult Index()
 {
 return View();
 }
 }
}
```

Лістинг
AIS.TermSearcher.UI\Models\DocumentManager.cs

```
using AIS.DocxReader.Entities.Document;
using AIS.DocxReader.Entities.Section;
using AIS.TermSearcher.Searcher;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace AIS.TermSearcher.UI.Models
{
 internal static class DocumentManager
 {
 internal static string FilePath { get; set; }
 internal static ISearchManager SearchManager {
get; set; }

 internal static List<Section> Sections { get;
set; }

 internal static List<Section>
CreateSections(List<ISection> sections)
 {
 List<Section> newSections = new
List<Section>();

 foreach (var section in sections)
 {
 newSections.Add(new Section()
 {
 Label = section.Name,
 Guid = section.Guid,
 Sections = CreateSections(section.Sections)
 });
 }

 return newSections;
 }
 }
}
```

Лістинг AIS.TermSearcher.UI\Models\Section.cs

```
using System;
using System.Collections.Generic;

namespace AIS.TermSearcher.UI.Models
{
 public class Section
 {
 public Guid Guid { get; set; }

 public string Label { get; set; }

 public List<Section> Sections { get; set; }
 }
}
```

Лістинг AIS.TermSearcher.UI\Models\Term.cs

```
namespace AIS.TermSearcher.UI.Models
{
 public class Term
 {
 public string Text { get; set; }

 public int Count { get; set; }

 public double Evaluation { get; set; }
 }
}
```

Лістинг
AIS.TermSearcher.UI\Properties\AssemblyInfo.cs

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// General Information about an assembly is
controlled through the following
// set of attributes. Change these attribute
values to modify the information
// associated with an assembly.
[assembly:
AssemblyTitle("AIS.TermSearcher.UI")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly:
AssemblyProduct("AIS.TermSearcher.UI")]
[assembly: AssemblyCopyright("Copyright ©
2019")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types
in this assembly not visible
// to COM components. If you need to access a
type in this assembly from
// COM, set the ComVisible attribute to true on
that type.
[assembly: ComVisible(false)]

// The following GUID is for the ID of the
typelib if this project is exposed to COM
[assembly: Guid("3f4a187a-52db-4802-b4e9-
6e269be1d269")]

// Version information for an assembly consists
of the following four values:
//
// Major Version
// Minor Version
// Build Number
// Revision
//
```

```
// You can specify all the values or you can
default the Revision and Build Numbers
// by using the '*' as shown below:
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

Лістинг
AIS.TermSearcher.UI\Content\MyTree\myTree.js

```
class myTree {
 treeHeaderTitle;
 root;

 constructor(conteinerElementId, title) {
 this.root = this.createRoot();
 this.treeConteinerElement =
this.createConteinerElement(conteinerElementId,
title);
 this.treeTitle = title;
 }

 get treeTitle() {
 return this.treeHeaderTitle.innerHTML;
 }

 set treeTitle(value) {
 this.treeHeaderTitle.innerHTML = value;
 }

 createRoot() {
 let root = new myNode('root', 'root');

 root.nodeLabelElement.style.display = 'none';
 root.nodeExpandElement.style.display = 'none';

root.nodeChildsElement.style.paddingInlineStart
= 0;

root.nodeChildsElement.classList.add('active');

 return root;
 }

 createConteinerElement(id, title) {
 let treeConteiner =
document.getElementById(id);
 document.createElement('div');
 treeConteiner.classList.add('myTree');
 treeConteiner.innerHTML = '';

 let treeHeaderTitle =
document.createElement('span');
 treeHeaderTitle.classList.add('tree-header-
title');
 treeHeaderTitle.innerHTML = title;

 let treeHeader =
document.createElement('div');
 treeHeader.classList.add('tree-header');
 treeHeader.appendChild(treeHeaderTitle);
 treeConteiner.appendChild(treeHeader);

 let treeBody = document.createElement('div');
 treeBody.classList.add('tree-body');
 treeBody.appendChild(this.root.nodeElement);
 treeConteiner.appendChild(treeBody);

 this.treeHeaderTitle = treeHeaderTitle;
 return treeConteiner;
 }

 addNode(id, label, parrentNodeId) {
 let parrentNode = this.getNode(this.root,
parrentNodeId);

 if (parrentNode === null) {
 throw { message: 'The tree does not contain a
node with id \'' + parrentNodeId + '\'' };
 }

 return parrentNode.addNode(id, label);
 }

 getNode(node, id) {
 if (node.id === id) {
 return node;
 }
 else if (node.childNodes !== null) {
 let i, result = null;
 for (i = 0; result === null && i <
node.childNodes.length; i++) {
 result = this.getNode(node.childNodes[i], id);
 }
 return result;
 }
 return null;
 }
}

class myNode {
 nodeExpandElement;
 nodeLabelElement;
 nodeChildsElement;

 constructor(id, label) {
 this.nodeElement = this.createNodeElement(id,
label);
 this.id = id;
 this.label = label;
 this.childNodes = [];
 this.parrentNode;
 }

 get label() {
 return this.nodeLabelElement.innerHTML;
 }

 set label(value) {
 this.nodeLabelElement.innerHTML = value;
 }

 createNodeElement(id) {
 let nodeElement;

 nodeElement = document.createElement('div');
 nodeElement.classList.add('node');
 nodeElement.id = id;

 this.nodeExpandElement =
document.createElement('span');

this.nodeExpandElement.classList.add('nodeExpan
d');

 this.nodeExpandElement.onclick = function () {
 let childList =
this.parentElement.querySelector('.nested');
 if(childList.children.length > 0){
 childList.classList.toggle('active');
 this.classList.toggle("caret-down");
 }
 };

 this.nodeLabelElement =
document.createElement('span');

this.nodeLabelElement.classList.add('nodeLabel'
);

 this.nodeChildsElement =
document.createElement('ul');

this.nodeChildsElement.classList.add('nested');

this.nodeChildsElement.classList.add('childs');

this.nodeChildsElement.classList.add('active');


nodeElement.appendChild(this.nodeExpandElement)
;

nodeElement.appendChild(this.nodeLabelElement);

nodeElement.appendChild(this.nodeChildsElement)
;

 return nodeElement;
 }

 addNode(id, label) {
 let newNode, child;

 newNode = new myNode(id, label);
 newNode.parrentNode = this;

 child = document.createElement('li');
 child.appendChild(newNode.nodeElement);
 this.nodeChildsElement.appendChild(child);
 this.nodeExpandElement.classList.add('caret');

 this.childNodes.push(newNode);

 return newNode;
 }
}
```

Лістинг AIS.TermSearcher.UI\Scripts\main.js

```
let tree, documentStucture, documentName;

function initEvents() {
 $('#file-loader-input').on('change', function
() {
 let data = new FormData();
 let files = $("#file-loader-
input").get(0).files;

 if (files.length > 0) {
 data.append("File", files[0]);
 documentName = files[0].name;

 let ajaxRequest = $.ajax({
 type: "POST",
 url: "Document/UploadFile",
 contentType: false,
 processData: false,
 data: data
 });

 ajaxRequest.done(function (data, textStatus) {
 console.log('loading: ' + textStatus);
 });
 }
 else throw { message: "Файл не обрано" };
 });

 $('#file-loader-input').ready(function () {
 let ajaxRequest = $.ajax({
 type: "GET",
 url: "Document/GetFileName"
 });

 ajaxRequest.done(function (filePath,
textStatus) {
 if (textStatus === 'success') {
 $('#file-on-server').text('Файл на сервері: '
+ getFileNameFromPath(filePath));
 }
 });
 });

 $('#searchButton').on('click', function () {
 let ajaxRequest = $.ajax({
 type: "GET",
 url: "Document/Search",
 data: {
 bold: $('#bold-value')[0].value,
 italic: $('#italic-value')[0].value,
 underline: $('#underline-value')[0].value
 }
 });

 ajaxRequest.done(function (data, textStatus) {
 initTree();
 console.log('search: ' + textStatus);
 addNodes(data);
 showView('#document-view');
 });
 });

 $('#navigation-document').on('click', function
() {
 showView('#document-view');
 });

 $('#navigation-setting').on('click', function
() {
 showView('#setting-view');
 });
}

function showView(selector) {
 $('.view').hide();
 $(selector).show();
}

function initTree() {
 tree = new myTree('structure-tree',
documentName);
 tree.treeTitle = "Структура документу";
}

function addNodes(data) {
 for (let i = 0; i < data.length; i++) {
 let currentNode = data[i];
 tree.addNode(currentNode.Guid,
currentNode.Label, 'root');
 onNodeClick(currentNode.Guid);

 addSubNodes(currentNode.Sections,
currentNode.Guid);
 }
}

function addSubNodes(nodes, parrentId) {
 for (let i = 0; i < nodes.length; i++) {
 let currentNode = nodes[i];
 tree.addNode(currentNode.Guid,
currentNode.Label, parrentId);
 onNodeClick(currentNode.Guid);

 if (currentNode.Sections.length > 0) {
 addSubNodes(currentNode.Sections,
currentNode.Guid);
 }
 }
}

function onNodeClick(guid) {
 $('#' + guid + ' .nodeLabel').on('click',
function (event) {
 let searchStage = $('#search-stage').val();
```

```
 let ajaxRequest = $.ajax({
 type: "GET",
 url: "Document/GetKeyTerms",
 data: { guid: event.target.parentElement.id,
stage: searchStage }
 });

 ajaxRequest.done(function (data, textStatus) {
 initTabel(data);
 });
 });
}

function initTabel(data) {
 let table = new Tabulator("#key-terms-list", {
 data: data,
 layout: "fitColumns",
 responsiveLayout: "hide",
 tooltips: true,
 addRowPos: "top",
 history: true,
 movableColumns: true,
 resizableRows: true,
 initialSort: [
 { column: "Text", dir: "asc" }
 ],
 columns: [
 {
 title: "Ключовий термін",
 field: "Text",
 headerFilter: "input",
 headerFilterPlaceholder: "Фільтрувати..."
 },
 {
 title: "Кількість",
 field: "Count",
 align: "left",
 headerFilter: minMaxFilterEditor,
 headerFilterFunc: minMaxFilterFunction
 },
 {
 title: "Оцінка",
 field: "Evaluation",
 align: "center",
 width: 120,
 headerFilter: minMaxFilterEditor,
 headerFilterFunc: minMaxFilterFunction
 }
 ]
 });
}

function getFileNameFromPath(path) {
 return
path.split('\\').pop().split('/').pop();
}

var minMaxFilterEditor = function (cell,
onRendered, success, cancel, editorParams) {

 var end;

 var container =
document.createElement("span");

 var start = document.createElement("input");
 start.setAttribute("type", "number");
 start.setAttribute("placeholder", "Min");
 start.setAttribute("min", 0);
 start.setAttribute("max", 100);
 start.style.padding = "4px";
 start.style.width = "50%";
 start.style.boxSizing = "border-box";

 start.value = cell.getValue();

 function buildValues() {
 success({
 start: start.value,
 end: end.value,
 });
 }

 function keypress(e) {
 if (e.keyCode === 13) {
 buildValues();
 }

 if (e.keyCode === 27) {
 cancel();
 }
 }

 end = start.cloneNode();
 end.setAttribute("placeholder", "Max");

 start.addEventListener("change", buildValues);
 start.addEventListener("blur", buildValues);
 start.addEventListener("keydown", keypress);

 end.addEventListener("change", buildValues);
 end.addEventListener("blur", buildValues);
 end.addEventListener("keydown", keypress);


 container.appendChild(start);
 container.appendChild(end);

 return container;
}

function minMaxFilterFunction(headerValue,
rowValue, rowData, filterParams) {

 if (rowValue) {
 if (headerValue.start !== "") {
 if (headerValue.end !== "") {
 return rowValue >= headerValue.start &&
rowValue <= headerValue.end;
 } else {
 return rowValue >= headerValue.start;
 }
 } else {
 if (headerValue.end !== "") {
 return rowValue <= headerValue.end;
 }
 }
 }

 return false;
}

initEvents();
initTree();
```

Лістинг
AIS.TermSearcher.UI\Views_ViewStart.cshtml

```
@{
 Layout = "~/Views/Shared/_Layout.cshtml";
}
```

Лістинг
AIS.TermSearcher.UI\Views\Index\Index.cshtml

```
<link href="~/Content/MyTree/myTree.css"
rel="stylesheet" />
<script
src="~/Content/MyTree/myTree.js"></script>
<script
src="https://ajax.aspnetcdn.com/ajax/jQuery/jqu
ery-3.3.1.min.js"></script>
<script
src="https://ajax.aspnetcdn.com/ajax/jquery.ui/
1.12.1/jquery-ui.min.js"></script>
<link
href="~/Content/Tabulator/css/tabulator.min.css
" rel="stylesheet" />
<script
src="~/Content/Tabulator/js/tabulator.min.js"><
/script>


@{
 ViewBag.Title = "Аналіз документу";
}

<div id="navigation">
 <div class="navigation-tab" id="navigation-
setting">
 <img src="~/Content/settings.png"
alt="Settings" />
 </div>
 <div class="navigation-tab" id="navigation-
document">
 <span class="navigation-text">Результат</span>
 </div>
</div>
<div id="content">
 <div id="setting-view" class="view">
 <span id="file-on-server">Оберіть файл для
аналізу:</span><br />
 <input type="file" id="file-loader-input">
 <br />
 <span>Оберіть етап пошуку:</span><br />
 <select id="search-stage">
 <option disabled>Оберіть етап</option>
 <option value="candidates">Фрази
кандидати</option>
 <option
value="compressed">Компактифікація</option>
 <option value="absorption">Поглинання</option>
 <option value="mask">Маска</option>
 <option value="nausea">Нудота</option>
 </select>
 <br />
 <table>
 <tr>
 <td><b>Bold</b></td>
 <td><input type="number" step="0.01" min="1"
value="1" id="bold-value"></td>
 </tr>
 <tr>
 <td><i>Italik</i></td>
 <td><input type="number" step="0.01" min="1"
value="1" id="italic-value"></td>
 </tr>
 <tr>
 <td><u>Underline</u></td>
 <td><input type="number" step="0.01" min="1"
value="1" id="underline-value"></td>
 </tr>
 </table>
 <br />
 <input type="button" name="searchButton"
value="Пошук ключових термінів"
id="searchButton" />
 </div>
 <div id="document-view" class="view">
 <div class="" id="document-tree">
 <div class="myTree" id="structure-tree"></div>
 </div>
 <div id="key-terms">
 <div id="key-terms-list"></div>
 </div>
 </div>
</div>

<script src="~/Scripts/main.js"></script>
```

Лістинг
AIS.TermSearcher.UI\Views\Shared_Layout.cshtml

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8" />
 <meta name="viewport" content="width=device-
width, initial-scale=1.0">
 <link href="~/Content/Styles/Site.css"
rel="stylesheet" type="text/css" />
 <link rel="shortcut icon" href="#">
 <title>@ViewBag.Title</title>
</head>
<body>
 @RenderBody()
</body>
</html>
```

Лістинг AIS.DocxReader\DocxReader.cs

```
using AIS.DocxReader.Entities.Document;
using AIS.DocxReader.Entities.Paragraph;
using AIS.DocxReader.Entities.Section;
using AIS.DocxReader.Entities.Sentences;
using AIS.DocxReader.Entities.TextContainer;
using AIS.DocxReader.Entities.Word;
using AIS.DocxReader.Resources;
using ASI.DocxReader.Entities.Sentences;
using DocumentFormat.OpenXml.Packaging;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text.RegularExpressions;
using Wordprocessing =
DocumentFormat.OpenXml.Wordprocessing;

namespace AIS.DocxReader
{
 public class DocxReader
 {
 private readonly string filePath;
 private readonly WordprocessingDocument
wordDocument;
 private readonly Regex regexWord;

 private string[] headingStyles;
 private string[] normalStyles;
 private string[] titleStyles;
 private string[] listStyles;

 private int lastParagraphIndex = 0;
 private int lastHeadingLevel = -1;

 public DocxReader(string filePath)
 {
 this.filePath = filePath;
 wordDocument =
WordprocessingDocument.Open(filePath, false);

 regexWord = new
Regex($"{RegexPatterns.Exclusion}|{RegexPattern
s.Date}|{RegexPatterns.Name}|{RegexPatterns.Wor
d}|{RegexPatterns.Separator}");

 ReadStyles();
 }

 public Document Read()
 {
 Document document = new Document()
 {
 Name =
Path.GetFileNameWithoutExtension(filePath),
 Sections = new List<ISection>()
```

```
 };

 List<IParagraph> paragraphs = new
List<IParagraph>();

 foreach (object wordParagraph in
wordDocument.MainDocumentPart.Document.ChildEle
ments[0].ChildElements)
 {
 if (wordParagraph is Wordprocessing.Paragraph
currentWordParagraph)
 {
 IParagraph paragraph =
GetParagraph(currentWordParagraph);

 if (paragraph.Sentences.Any())
 {
 paragraphs.Add(paragraph);
 }
 }
 }


document.Sections.Add(MakeStructure(paragraphs)
);

 return document;
 }

 private ISection
MakeStructure(List<IParagraph> paragraphs)
 {
 Section rootSection = new Section
 {
 Level = -1,
 Name =
Path.GetFileNameWithoutExtension(filePath),
 Paragraphs = new List<IParagraph>(),
 Sections = new List<ISection>(),
 Guid = Guid.NewGuid()
 };

 MakeSections(rootSection, paragraphs);

 return rootSection;
 }

 private void ReadStyles()
 {
 headingStyles = ParagraphTypes.Heading.Split('
');
 normalStyles = ParagraphTypes.Normal.Split('
');
 titleStyles = ParagraphTypes.Title.Split(' ');
 listStyles = ParagraphTypes.List.Split(' ');
 }

 private void MakeSections(ISection
currentSection, List<IParagraph> paragraphs)
 {
 while (lastParagraphIndex + 1 <
paragraphs.Count)
 {
 IParagraph currentParagraph =
paragraphs[lastParagraphIndex];

 if (currentParagraph.Level ==
currentSection.Level)
 {
 if (currentParagraph.Type ==
EParagraphType.Heading || currentParagraph.Type
== EParagraphType.Title)
 {
 break;
 }


currentSection.Paragraphs.Add(currentParagraph)
;
 lastParagraphIndex++;
 }
 else if (currentParagraph.Level >
currentSection.Level)
 {
 string headerText;

 if (currentParagraph.Type ==
EParagraphType.Heading || currentParagraph.Type
== EParagraphType.Title)
 {
 headerText = currentParagraph.Text;
 }
 else
 {
 headerText =
currentParagraph.Sentences[0].Text;
 }

 Section section = new Section
 {
 Sections = new List<ISection>(),
 Paragraphs = new List<IParagraph>(),
 Level = currentParagraph.Level,
 Guid = Guid.NewGuid(),
 Name = headerText
 };
 section.Paragraphs.Add(currentParagraph);

 lastParagraphIndex++;

 MakeSections(section, paragraphs);
 currentSection.Sections.Add(section);
 }
 else if (currentParagraph.Level <
currentSection.Level)
 {
 break;
 }
 }
 }

 private IParagraph
GetParagraph(Wordprocessing.Paragraph
wordParagraph)
 {
 Paragraph paragraph = new Paragraph
 {
 Sentences = GetSentences(wordParagraph),
 Type = GetParagraphType(wordParagraph),
 Text = wordParagraph.InnerText
 };

 SetParagraphLevel(paragraph, wordParagraph);

 return paragraph;
 }

 private List<ISentences>
GetSentences(Wordprocessing.Paragraph
wordParagraph)
 {
 List<IWord> words = new List<IWord>();
 List<ISentences> sentences = new
List<ISentences>();
 List<ITextContainer> textContainers = new
List<ITextContainer>();
 WordStyle lastWordStyle = new WordStyle();
 TextContainer textContainer = new
TextContainer()
 {
 Words = new List<IWord>(),
 StartSymbolType = ESymbolType.Start
 };

 int lastSentencesIndex = 0;
 int lastTextContainerIndex = 0;

 MatchCollection wordMatches =
regexWord.Matches(wordParagraph.InnerText);
 var wordMathesByStyle =
GetWordMathesByStyle(wordParagraph);

 foreach (Match wordMatch in wordMatches)
 {
 if (string.IsNullOrEmpty(wordMatch.Value) ||
string.IsNullOrWhiteSpace(wordMatch.Value))
 {
 continue;
 }

 if (wordMatch.Groups["separator"].Success)
 {
 ESentencesType sentencesType =
GetSentencesType(wordMatch.Value);
 ESymbolType symbolType =
GetSymbolType(wordMatch.Value);

 textContainer.EndSymbolType = symbolType;
 textContainer.EndSymbol = wordMatch.Value;
 textContainer.Text =
wordParagraph.InnerText.Substring(lastTextConta
inerIndex, wordMatch.Index -
lastTextContainerIndex).Trim();
 textContainers.Add(textContainer);
 textContainer = new TextContainer()
 {
 Words = new List<IWord>(),
 StartSymbolType = symbolType,
 StartSymbol = wordMatch.Value
 };

 lastTextContainerIndex = wordMatch.Index + 1;

 if (sentencesType == ESentencesType.None)
 {
 continue;
 }

 Sentences currentSentences = new Sentences()
 {
 Text =
wordParagraph.InnerText.Substring(lastSentences
Index, wordMatch.Index -
lastSentencesIndex).Trim(),
 Words = words,
 SentencesType = sentencesType,
 TextContainers = textContainers
 };

 sentences.Add(currentSentences);

 lastSentencesIndex = wordMatch.Index + 1;
 words = new List<IWord>();
 textContainers = new List<ITextContainer>();
 }
 else
 {
 IWord word = GetWord(wordMatch);
 WordStyle wordStyle = new WordStyle();

 if
(wordMathesByStyle.ContainsKey(wordMatch.Index)
)
 {
 wordStyle =
wordMathesByStyle[wordMatch.Index].Style;
 word.WordStyle = wordStyle;
 }
 else
 {
 word.WordStyle = wordStyle;
 }

 words.Add(word);

 if (lastWordStyle.Equals(wordStyle))
 {
 textContainer.Words.Add(word);
 }
 else
 {
 textContainer.EndSymbolType =
ESymbolType.None;
 textContainer.Text =
wordParagraph.InnerText.Substring(lastTextConta
inerIndex, wordMatch.Index -
lastTextContainerIndex).Trim();
 if (textContainer.Words.Any())
 {
 textContainers.Add(textContainer);
 }
 textContainer = new TextContainer()
 {
 Words = new List<IWord>() { word },
 StartSymbolType = ESymbolType.None
 };

 lastTextContainerIndex = wordMatch.Index;
 }

 lastWordStyle = wordStyle;
 }
 }

 if (words.Any())
 {
 textContainer.EndSymbolType =
ESymbolType.None;
 textContainer.Text =
wordParagraph.InnerText.Substring(lastTextConta
inerIndex, wordParagraph.InnerText.Length -
lastTextContainerIndex).Trim();
 textContainers.Add(textContainer);

 sentences.Add(new Sentences()
 {
 Text =
wordParagraph.InnerText.Substring(lastSentences
Index, wordParagraph.InnerText.Length -
lastSentencesIndex).Trim(),
 Words = words,
 SentencesType = ESentencesType.None,
 TextContainers = textContainers
 });
 }

 return sentences;
 }

 private ESymbolType GetSymbolType(string
symbolType)
 {
 if (SymbolTypes.FullStop.Contains(symbolType))
 return ESymbolType.FullStop;
 else if
(SymbolTypes.Comma.Contains(symbolType))
 return ESymbolType.Comma;
 else if
(SymbolTypes.OpenBracket.Contains(symbolType))
 return ESymbolType.OpenBracket;
 else if
(SymbolTypes.CloseBracket.Contains(symbolType))
 return ESymbolType.CloseBracket;
```

```
 else if
(SymbolTypes.Hyphen.Contains(symbolType))
 return ESymbolType.Hyphen;
 else if
(SymbolTypes.Quotation.Contains(symbolType))
 return ESymbolType.Quotation;
 else return ESymbolType.Other;
 }

 private Dictionary<int, (Match Match,
WordStyle Style)>
GetWordMathesByStyle(Wordprocessing.Paragraph
wordParagraph)
 {
 Dictionary<int, (Match, WordStyle)>
wordMatches = new Dictionary<int, (Match,
WordStyle)>();
 int lastIndex = 0;

 foreach (var paragraphChild in
wordParagraph.ChildElements)
 {
 if (paragraphChild is Wordprocessing.Run run)
 {
 WordStyle wordStyle = new WordStyle()
 {
 Bold = run.RunProperties?.Bold != null,
 Italic = run.RunProperties?.Italic != null,
 Underline = run.RunProperties?.Underline !=
null,
 };

 foreach (Match match in
regexWord.Matches(run.InnerText))
 {
 wordMatches.Add(lastIndex + match.Index,
(match, wordStyle));
 }

 lastIndex += run.InnerText.Length;
 }
 else if
(!string.IsNullOrWhiteSpace(paragraphChild.Inne
rText))
 {
 foreach (Match match in
regexWord.Matches(paragraphChild.InnerText))
 {
 wordMatches.Add(lastIndex + match.Index,
(match, new WordStyle()));
 }

 lastIndex += paragraphChild.InnerText.Length;
 }
 }

 return wordMatches;
 }

 private IWord GetWord(Match word)
 {
 if (word.Groups["name"].Success)
 {
 return new Name()
 {
 Text = word.Groups["name"].Value,
 Lastname = word.Groups["lastname"].Value,
 Initials = word.Groups["initials"].Value
 };
 }
 else if (word.Groups["year"].Success ||
word.Groups["day"].Success)
 {
 return new Date()
 {
 Text = word.Value
 };
 }
 else if (word.Groups["word"].Success ||
word.Groups["exclusion"].Success)
 {
 return new Word()
 {
 Text = word.Value
 };
 }
 //todo: check if the condition is possible
 else return null;
 }

 private ESentencesType GetSentencesType(string
separator)
 {
 switch (separator)
 {
 case ".":
 return ESentencesType.Declarative;
 case "?":
 return ESentencesType.Interrogative;
 case "!":
 return ESentencesType.Exclamatory;
 default:
 return ESentencesType.None;
 }
 }

 private void SetParagraphLevel(Paragraph
paragraph, Wordprocessing.Paragraph
wordParagraph)
 {
 if (paragraph.Type == EParagraphType.Heading)
 {
 if
(int.TryParse(wordParagraph.ParagraphProperties
.ParagraphStyleId.Val.InnerText, out int
level))
 {
 paragraph.Level = level;
 lastHeadingLevel = level;
 }
 else
 {
 paragraph.Level = 0;
 lastHeadingLevel = 0;
 }
 }
 else if (paragraph.Type ==
EParagraphType.Title)
 {
 paragraph.Level = 0;
 lastHeadingLevel = 0;
 }
 else
 {
 paragraph.Level = lastHeadingLevel;
 }
 }

 private EParagraphType
GetParagraphType(Wordprocessing.Paragraph
wordParagraph)
 {
 if
(wordParagraph.ParagraphProperties?.ParagraphSt
yleId == null)
 {
 return EParagraphType.Normal;
 }

 string styleId =
wordParagraph.ParagraphProperties.ParagraphStyl
eId.Val.InnerText;

 if (headingStyles.Contains(styleId) ||
styleId.Contains("Heading"))
 {
 return EParagraphType.Heading;
 }
 else if (titleStyles.Contains(styleId))
 {
 return EParagraphType.Title;
 }
 else if (listStyles.Contains(styleId))
 {
 return EParagraphType.List;
 }
 else if (normalStyles.Contains(styleId))
 {
 return EParagraphType.Normal;
 }
 else
 {
 return EParagraphType.None;
 }
 }

 ~DocxReader()
 {
 wordDocument.Dispose();
 }
 }
}
```

Лістинг AIS.DocxReader\Entities\Document\Document.cs

```
using AIS.DocxReader.Entities.Section;
using System.Collections.Generic;

namespace AIS.DocxReader.Entities.Document
{
 public class Document : IDocument
 {
 public string Name { get; set; }

 public List<ISection> Sections { get; set; }
 }
}
```

Лістинг AIS.DocxReader\Entities\Document\IDocument.cs

```
using AIS.DocxReader.Entities.Section;
using System.Collections.Generic;

namespace AIS.DocxReader.Entities.Document
{
 public interface IDocument
 {
 string Name { get; set; }

 List<ISection> Sections { get; set; }
 }
}
```

Лістинг AIS.DocxReader\Entities\Paragraph\EParagraphTyp e.cs

```
namespace AIS.DocxReader.Entities.Paragraph
{
 public enum EParagraphType : sbyte
 {
 None = -1,
 Normal,
 List,
 Table,
 Heading,
 Title
 }
}
```

Лістинг AIS.DocxReader\Entities\Paragraph\IParagraph.cs

```
using ASI.DocxReader.Entities.Sentences;
using System.Collections.Generic;

namespace AIS.DocxReader.Entities.Paragraph
{
 public interface IParagraph
 {
 int Level { get; set; }

 string Text { get; set; }

 EParagraphType Type { get; set; }

 List<ISentences> Sentences { get; set; }
 }
}
```

Лістинг AIS.DocxReader\Entities\Paragraph\Paragraph.cs

```
using ASI.DocxReader.Entities.Sentences;
using System.Collections.Generic;

namespace AIS.DocxReader.Entities.Paragraph
{
 public class Paragraph : IParagraph
 {
 public int Level { get; set; }

 public string Text { get; set; }

 public EParagraphType Type { get; set; }

 public List<ISentences> Sentences { get; set;
}
 }
}
```

Лістинг AIS.DocxReader\Entities\Section\ISection.cs

```
using AIS.DocxReader.Entities.Paragraph;
using System;
using System.Collections.Generic;

namespace AIS.DocxReader.Entities.Section
{
 public interface ISection
 {
 int Level { get; set; }

 string Text { get; }

 string Name { get; set; }

 Guid Guid { get; set; }

 List<ISection> Sections { get; set; }

 List<IParagraph> Paragraphs { get; set; }

 List<IParagraph> AllParagraphs { get; }
 }
}
```

Лістинг AIS.DocxReader\Entities\Section\Section.cs

```
using AIS.DocxReader.Entities.Paragraph;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace AIS.DocxReader.Entities.Section
{
 public class Section : ISection
 {
 private string text;
 private List<IParagraph> allParagraphs;

 public int Level { get; set; }

 public string Name { get; set; }

 public string Text
 {
 get
 {
 if (string.IsNullOrEmpty(text))
 {
 text = GetText();
 }

 return text;
 }
 protected set { text = value; }
 }

 public Guid Guid { get; set; }

 public List<ISection> Sections { get; set; }

 public List<IParagraph> Paragraphs { get; set;
}

 public List<IParagraph> AllParagraphs
 {
 get
 {
 if (allParagraphs == null)
 {
 allParagraphs = new List<IParagraph>();
 allParagraphs.AddRange(this.Paragraphs);
 allParagraphs.AddRange(from s in Sections
 from p in s.AllParagraphs
 select p);
 }

 return allParagraphs;
 }
 }

 private string GetText()
 {
 StringBuilder stringBuilder = new
StringBuilder();

 if (Paragraphs != null)
 {
 foreach (var paragraph in Paragraphs)
 {
 stringBuilder.AppendLine(paragraph.Text);
 }
 }

 if (Sections != null)
 {
 foreach (var section in Sections)
 {
 stringBuilder.AppendLine(section.Text);
 }
 }

 return stringBuilder.ToString();
 }
 }
}
```

Лістинг AIS.DocxReader\Entities\Sentences\ESentencesType.cs

```
namespace AIS.DocxReader.Entities.Sentences
{
 public enum ESentencesType
 {
 /// <summary>
 /// Default value. It is used when a sentences
type impossible to know
 /// </summary>
 None,
 /// <summary>
 /// Ends with '.'
 /// </summary>
 Declarative,
 /// <summary>
 /// Ends with '?'
 /// </summary>
 Interrogative,
 /// <summary>
 /// Ends with '!'
 /// </summary>
 Exclamatory
 }
}
```

Лістинг AIS.DocxReader\Entities\Sentences\ISentences.cs

```
using AIS.DocxReader.Entities.Sentences;
using AIS.DocxReader.Entities.TextContainer;
using AIS.DocxReader.Entities.Word;
using System.Collections.Generic;

namespace ASI.DocxReader.Entities.Sentences
{
 public interface ISentences
 {
 string Text { get; set; }

 List<IWord> Words { get; set; }

 List<ITextContainer> TextContainers { get;
set; }

 ESentencesType SentencesType { get; set; }
 }
}
```

Лістинг AIS.DocxReader\Entities\Sentences\Sentences.cs

```
using AIS.DocxReader.Entities.TextContainer;
using AIS.DocxReader.Entities.Word;
using ASI.DocxReader.Entities.Sentences;
using System.Collections.Generic;

namespace AIS.DocxReader.Entities.Sentences
{
 public class Sentences : ISentences
 {
 public string Text { get; set; }

 public List<IWord> Words { get; set; }

 public List<ITextContainer> TextContainers {
get; set; }

 public ESentencesType SentencesType { get;
set; }
 }
}
```

Лістинг AIS.DocxReader\Entities\TextContainer\ESymbolType.cs

```
namespace AIS.DocxReader.Entities.TextContainer
{
 public enum ESymbolType
 {
 None, //Style conteiner
 Comma, //,
 FullStop, //.!?…
 OpenBracket, //[{((
 CloseBracket, //])}
 Hyphen, //- – — ―
 Quotation, //‘’“”'"
 Start,
 End,
 Other
 }
}
```

Лістинг AIS.DocxReader\Entities\TextContainer\ITextContainer.cs

```
using AIS.DocxReader.Entities.Word;
using System.Collections.Generic;

namespace AIS.DocxReader.Entities.TextContainer
{
 public interface ITextContainer
 {
 string Text { get; set; }

 List<IWord> Words { get; set; }

 ESymbolType StartSymbolType { get; set; }

 ESymbolType EndSymbolType { get; set; }

 string EndSymbol { get; set; }

 string StartSymbol { get; set; }
 }
}
```

Лістинг AIS.DocxReader\Entities\TextContainer\TextContainer.cs

```
using AIS.DocxReader.Entities.Word;
using System.Collections.Generic;

namespace AIS.DocxReader.Entities.TextContainer
{
 public class TextContainer : ITextContainer
 {
 public string Text { get; set; }

 public List<IWord> Words { get; set; }

 public ESymbolType StartSymbolType { get; set;
}

 public ESymbolType EndSymbolType { get; set; }

 public string EndSymbol { get; set; }

 public string StartSymbol { get; set; }
 }
}
```

Лістинг AIS.DocxReader\Entities\Word\Date.cs

```
namespace AIS.DocxReader.Entities.Word
{
 class Date : IWord
 {
 public IWordStyle WordStyle { get; set; }

 public string Text { get; set; }
 }
}
```

Лістинг AIS.DocxReader\Entities\Word\IWord.cs

```
namespace AIS.DocxReader.Entities.Word
{
 public interface IWord
 {
 IWordStyle WordStyle { get; set; }

 string Text { get; set; }
 }
}
```

Лістинг AIS.DocxReader\Entities\Word\IWordStyle.cs

```
namespace AIS.DocxReader.Entities.Word
{
 public interface IWordStyle
 {
 bool Bold { get; set; }

 bool Italic { get; set; }

 bool Underline { get; set; }
 }
}
```

Лістинг AIS.DocxReader\Entities\Word\Name.cs

```
namespace AIS.DocxReader.Entities.Word
{
 class Name : IWord
 {
 public IWordStyle WordStyle { get; set; }

 public string Text { get; set; }

 public string Lastname { get; set; }

 public string Initials { get; set; }
 }
}
```

Лістинг AIS.DocxReader\Entities\Word\Word.cs

```
namespace AIS.DocxReader.Entities.Word
{
 public class Word : IWord
 {
 public IWordStyle WordStyle { get; set; }

 public string Text { get; set; }
 }
}
```

Лістинг AIS.DocxReader\Entities\Word\WordStyle.cs

```
namespace AIS.DocxReader.Entities.Word
{
 public class WordStyle : IWordStyle
 {
 public bool Bold { get; set; }
```

```
 public bool Italic { get; set; }

 public bool Underline { get; set; }

 public override bool Equals(object obj)
 {
 if (!(obj is WordStyle style))
 {
 return false;
 }
 else if (this.Bold == style.Bold &&
 this.Italic == style.Italic &&
 this.Underline == style.Underline)
 {
 return true;
 }

 return false;
 }
 }
}
```

Лістинг AIS.DocxReader\Properties\AssemblyInfo.cs

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// General Information about an assembly is
controlled through the following
// set of attributes. Change these attribute
values to modify the information
// associated with an assembly.
[assembly: AssemblyTitle("AIS.DocxReader")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("AIS.DocxReader")]
[assembly: AssemblyCopyright("Copyright ©
2018")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types
in this assembly not visible
// to COM components. If you need to access a
type in this assembly from
// COM, set the ComVisible attribute to true on
that type.
[assembly: ComVisible(false)]

// The following GUID is for the ID of the
typelib if this project is exposed to COM
[assembly: Guid("67b6752f-ada5-439c-82f6-
80d5546462ef")]

// Version information for an assembly consists
of the following four values:
//
// Major Version
// Minor Version
// Build Number
// Revision
//
// You can specify all the values or you can
default the Build and Revision Numbers
// by using the '*' as shown below:
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

Лістинг AIS.DocxReader\Resources\ParagraphTypes.Designer.cs

```
//------------------------------------------------
--------------------------------
// <auto-generated>
// This code was generated by a tool.
// Runtime Version:4.0.30319.42000
//
// Changes to this file may cause incorrect
behavior and will be lost if
// the code is regenerated.
// </auto-generated>
//------------------------------------------------
--------------------------------

namespace AIS.DocxReader.Resources {
 using System;


 /// <summary>
 /// A strongly-typed resource class, for
looking up localized strings, etc.
 /// </summary>
 // This class was auto-generated by the
StronglyTypedResourceBuilder
 // class via a tool like ResGen or Visual
Studio.
 // To add or remove a member, edit your .ResX
file then rerun ResGen
 // with the /str option, or rebuild your VS
project.

[global::System.CodeDom.Compiler.GeneratedCodeA
ttribute("System.Resources.Tools.StronglyTypedR
esourceBuilder", "15.0.0.0")]

[global::System.Diagnostics.DebuggerNonUserCode
Attribute()]

[global::System.Runtime.CompilerServices.Compil
erGeneratedAttribute()]
 internal class ParagraphTypes {

 private static
global::System.Resources.ResourceManager
resourceMan;

 private static
global::System.Globalization.CultureInfo
resourceCulture;


[global::System.Diagnostics.CodeAnalysis.Suppre
ssMessageAttribute("Microsoft.Performance",
"CA1811:AvoidUncalledPrivateCode")]
 internal ParagraphTypes() {
 }

 /// <summary>
 /// Returns the cached ResourceManager
instance used by this class.
 /// </summary>

[global::System.ComponentModel.EditorBrowsableA
ttribute(global::System.ComponentModel.EditorBr
owsableState.Advanced)]
 internal static
global::System.Resources.ResourceManager
ResourceManager {
 get {
 if (object.ReferenceEquals(resourceMan, null))
{
 global::System.Resources.ResourceManager temp
= new
global::System.Resources.ResourceManager("AIS.D
ocxReader.Resources.ParagraphTypes",
typeof(ParagraphTypes).Assembly);
 resourceMan = temp;
 }
 return resourceMan;
 }
 }

 /// <summary>
 /// Overrides the current thread's
CurrentUICulture property for all
 /// resource lookups using this strongly typed
resource class.
 /// </summary>

[global::System.ComponentModel.EditorBrowsableA
ttribute(global::System.ComponentModel.EditorBr
owsableState.Advanced)]
 internal static
global::System.Globalization.CultureInfo
Culture {
 get {
 return resourceCulture;
 }
 set {
 resourceCulture = value;
 }
 }

 /// <summary>
 /// Looks up a localized string similar to 1 2
3 4 5 6 7 8 9.
 /// </summary>
 internal static string Heading {
 get {
 return ResourceManager.GetString("Heading",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to a5.
 /// </summary>
 internal static string List {
 get {
 return ResourceManager.GetString("List",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to n.
 /// </summary>
 internal static string Normal {
 get {
 return ResourceManager.GetString("Normal",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to a3.
 /// </summary>
 internal static string Title {
 get {
 return ResourceManager.GetString("Title",
resourceCulture);
 }
 }
 }
}
```

Лістинг AIS.DocxReader\Resources\RegexPatterns.Designer.cs

```
//------------------------------------------------
--------------------------------
// <auto-generated>
// This code was generated by a tool.
// Runtime Version:4.0.30319.42000
//
// Changes to this file may cause incorrect
behavior and will be lost if
// the code is regenerated.
// </auto-generated>
//------------------------------------------------
--------------------------------

namespace AIS.DocxReader.Resources {
 using System;


 /// <summary>
 /// A strongly-typed resource class, for
looking up localized strings, etc.
 /// </summary>
 // This class was auto-generated by the
StronglyTypedResourceBuilder
 // class via a tool like ResGen or Visual
Studio.
 // To add or remove a member, edit your .ResX
file then rerun ResGen
 // with the /str option, or rebuild your VS
project.

[global::System.CodeDom.Compiler.GeneratedCodeA
ttribute("System.Resources.Tools.StronglyTypedR
esourceBuilder", "15.0.0.0")]

[global::System.Diagnostics.DebuggerNonUserCode
Attribute()]

[global::System.Runtime.CompilerServices.Compil
erGeneratedAttribute()]
 internal class RegexPatterns {

 private static
global::System.Resources.ResourceManager
resourceMan;

 private static
global::System.Globalization.CultureInfo
resourceCulture;


[global::System.Diagnostics.CodeAnalysis.Suppre
ssMessageAttribute("Microsoft.Performance",
"CA1811:AvoidUncalledPrivateCode")]
 internal RegexPatterns() {
 }

 /// <summary>
 /// Returns the cached ResourceManager
instance used by this class.
 /// </summary>

[global::System.ComponentModel.EditorBrowsableA
ttribute(global::System.ComponentModel.EditorBr
owsableState.Advanced)]
 internal static
global::System.Resources.ResourceManager
ResourceManager {
 get {
 if (object.ReferenceEquals(resourceMan, null))
{
 global::System.Resources.ResourceManager temp
= new
global::System.Resources.ResourceManager("AIS.D
ocxReader.Resources.RegexPatterns",
typeof(RegexPatterns).Assembly);
 resourceMan = temp;
```

```
 }
 return resourceMan;
 }
 }

 /// <summary>
 /// Overrides the current thread's
CurrentUICulture property for all
 /// resource lookups using this strongly typed
resource class.
 /// </summary>

[global::System.ComponentModel.EditorBrowsableA
ttribute(global::System.ComponentModel.EditorBr
owsableState.Advanced)]
 internal static
global::System.Globalization.CultureInfo
Culture {
 get {
 return resourceCulture;
 }
 set {
 resourceCulture = value;
 }
 }

 /// <summary>
 /// Looks up a localized string similar to
(?&lt;day&gt;\d{1,2}(?&gt;\s+|-
\d{1,2}\s+)(?&gt;січня|лютого|березня|квітня|тр
авня|червня|липня|серпня|вересня|жовтня|листопа
да|грудня)\s+)|(?&lt;year&gt;\d{4}(?&gt;\s*(?&g
t;р\.|рік|року|році))|\d{4}-
\d{4}\s*(?&gt;рр\.|років|роках)).
 /// </summary>
 internal static string Date {
 get {
 return ResourceManager.GetString("Date",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to
(?&lt;exclusion&gt;ін\.|тис\.|див\.|мал\.|рис\.
|т\.д\.|т\.п\.).
 /// </summary>
 internal static string Exclusion {
 get {
 return ResourceManager.GetString("Exclusion",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to
(?&lt;=\s|\W|^)(?&lt;name&gt;(?&lt;initials&gt;
(?&gt;(?&gt;[A-ЯІЇЄ]|[A-
Z])\.\s?(?&gt;|$)){1,2})(?&lt;lastname&gt;(?&gt
;[A-ЯІЇЄ]|[A-
Z])(?&gt;(?&gt;\w+\S)*\w+|\w+)(?&gt;-(?&gt;[A-
ЯІЇЄ]|[A-Z])(?&gt;\w+\S)*\w+|\w+)?)).
 /// </summary>
 internal static string Name {
 get {
 return ResourceManager.GetString("Name",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to
(?&lt;separator&gt;\S).
 /// </summary>
 internal static string Separator {
 get {
 return ResourceManager.GetString("Separator",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to
(?&lt;word&gt;(?&gt;\w+\S)*\w+|\w+).
 /// </summary>
 internal static string Word {
 get {
 return ResourceManager.GetString("Word",
resourceCulture);
 }
 }
 }
}
```

Лістинг AIS.DocxReader\Resources\SymbolTypes.Designer.cs

```
//------------------------------------------------
------------------------------
// <auto-generated>
// This code was generated by a tool.
// Runtime Version:4.0.30319.42000
//
// Changes to this file may cause incorrect
behavior and will be lost if
// the code is regenerated.
// </auto-generated>
//------------------------------------------------
------------------------------

namespace AIS.DocxReader.Resources {
 using System;


 /// <summary>
 /// A strongly-typed resource class, for
looking up localized strings, etc.
 /// </summary>
 // This class was auto-generated by the
StronglyTypedResourceBuilder
 // class via a tool like ResGen or Visual
Studio.
 // To add or remove a member, edit your .ResX
file then rerun ResGen
 // with the /str option, or rebuild your VS
project.

[global::System.CodeDom.Compiler.GeneratedCodeA
ttribute("System.Resources.Tools.StronglyTypedR
esourceBuilder", "15.0.0.0")]

[global::System.Diagnostics.DebuggerNonUserCode
Attribute()]

[global::System.Runtime.CompilerServices.Compil
erGeneratedAttribute()]
 internal class SymbolTypes {

 private static
global::System.Resources.ResourceManager
resourceMan;

 private static
global::System.Globalization.CultureInfo
resourceCulture;


[global::System.Diagnostics.CodeAnalysis.Suppre
ssMessageAttribute("Microsoft.Performance",
"CA1811:AvoidUncalledPrivateCode")]
 internal SymbolTypes() {
 }

 /// <summary>
 /// Returns the cached ResourceManager
instance used by this class.
 /// </summary>

[global::System.ComponentModel.EditorBrowsableA
ttribute(global::System.ComponentModel.EditorBr
owsableState.Advanced)]
 internal static
global::System.Resources.ResourceManager
ResourceManager {
 get {
 if (object.ReferenceEquals(resourceMan, null))
{
 global::System.Resources.ResourceManager temp
= new
global::System.Resources.ResourceManager("AIS.D
ocxReader.Resources.SymbolTypes",
typeof(SymbolTypes).Assembly);
 resourceMan = temp;
 }
 return resourceMan;
 }
 }

 /// <summary>
 /// Overrides the current thread's
CurrentUICulture property for all
 /// resource lookups using this strongly typed
resource class.
 /// </summary>

[global::System.ComponentModel.EditorBrowsableA
ttribute(global::System.ComponentModel.EditorBr
owsableState.Advanced)]
 internal static
global::System.Globalization.CultureInfo
Culture {
 get {
 return resourceCulture;
 }
 set {
 resourceCulture = value;
 }
 }

 /// <summary>
 /// Looks up a localized string similar to
])}).
 /// </summary>
 internal static string CloseBracket {
 get {
 return
ResourceManager.GetString("CloseBracket",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to ,.
 /// </summary>
 internal static string Comma {
 get {
 return ResourceManager.GetString("Comma",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to
.!?….
 /// </summary>
 internal static string FullStop {
 get {
 return ResourceManager.GetString("FullStop",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to ---
—.
 /// </summary>
 internal static string Hyphen {
 get {
 return ResourceManager.GetString("Hyphen",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to
[({{.
 /// </summary>
 internal static string OpenBracket {
 get {
 return
ResourceManager.GetString("OpenBracket",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to
‘’“”&amp;apos;&amp;quot;.
 /// </summary>
 internal static string Quotation {
 get {
 return ResourceManager.GetString("Quotation",
resourceCulture);
 }
 }
 }
}
```

Лістинг AIS.Helper\Extention\String\StringExtentions.cs

```
using System.Linq;

namespace AIS.Helper.Extention.String
{
 public static class StringExtentions
 {
 public static string FirstCharToUpper(this
string currentString)
 {
 return char.ToUpper(currentString.First()) +
currentString.Substring(1);
 }

 public static string FirstCharToLower(this
string currentString)
 {
 return char.ToLower(currentString.First()) +
currentString.Substring(1);
 }
 }
}
```

Лістинг AIS.Helper\Properties\AssemblyInfo.cs

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// General Information about an assembly is
controlled through the following
```

```
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[assembly: AssemblyTitle("AIS.Helper")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("AIS.Helper")]
[assembly: AssemblyCopyright("Copyright © 2019")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types in this assembly not visible
// to COM components. If you need to access a type in this assembly from
// COM, set the ComVisible attribute to true on that type.
[assembly: ComVisible(false)]

// The following GUID is for the ID of the typelib if this project is exposed to COM
[assembly: Guid("c38c988e-5e51-42e9-9a5b-d14fe91e26b2")]

// Version information for an assembly consists of the following four values:
//
// Major Version
// Minor Version
// Build Number
// Revision
//
// You can specify all the values or you can default the Build and Revision Numbers
// by using the '*' as shown below:
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

Лістинг AIS.TermSearcher\Entities\EntitiesHelper.cs

```
using AIS.TermSearcher.Entities.Word;
using System;
using System.Collections.Generic;
using System.ComponentModel;

namespace AIS.TermSearcher.Entities
{
 public static class EntitiesHelper
 {
 private static Dictionary<string, ESpeechPart> speechPartMapping;
 private static Dictionary<string, EGrammaticalGender> grammaticalGenderMapping;
 private static Dictionary<string, EGrammaticalCase> grammaticalCaseMapping;

 public static Dictionary<string, ESpeechPart> SpeechPartMapping
 {
 get
 {
 if (speechPartMapping == null)
 {
 speechPartMapping = new Dictionary<string, ESpeechPart>();

 foreach (ESpeechPart speechPart in Enum.GetValues(typeof(ESpeechPart)))
 {

speechPartMapping.Add(speechPart.GetUkrainianValue().ToLower(), speechPart);
 }

 return speechPartMapping;
 }
 else return speechPartMapping;
 }
 }

 public static Dictionary<string, EGrammaticalGender> GrammaticalGenderMapping
 {
 get
 {
 if (grammaticalGenderMapping == null)
 {
 grammaticalGenderMapping = new Dictionary<string, EGrammaticalGender>();

 foreach (EGrammaticalGender speechPart in Enum.GetValues(typeof(EGrammaticalGender)))
 {

grammaticalGenderMapping.Add(speechPart.GetUkrainianValue().ToLower(), speechPart);
 }

 return grammaticalGenderMapping;
 }
 }

 public static Dictionary<string, EGrammaticalCase> GrammaticalCaseMapping
 {
 get
 {
 if (grammaticalCaseMapping == null)
 {
 grammaticalCaseMapping = new Dictionary<string, EGrammaticalCase>();

 foreach (EGrammaticalCase speechPart in Enum.GetValues(typeof(EGrammaticalCase)))
 {

grammaticalCaseMapping.Add(speechPart.GetUkrainianValue().ToLower(), speechPart);
 }

 return grammaticalCaseMapping;
 }
 }

 public static string GetUkrainianValue(this ESpeechPart eSpeechPart)
 {
 var type = typeof(ESpeechPart);
 var memInfo = type.GetMember(eSpeechPart.ToString());
 var attributes = memInfo[0].GetCustomAttributes(typeof(DescriptionAttribute), false);
 return ((DescriptionAttribute)attributes[0]).Description;
 }

 public static string GetUkrainianValue(this EGrammaticalGender eGrammaticalGender)
 {
 var type = typeof(EGrammaticalGender);
 var memInfo = type.GetMember(eGrammaticalGender.ToString());
 var attributes = memInfo[0].GetCustomAttributes(typeof(DescriptionAttribute), false);
 return ((DescriptionAttribute)attributes[0]).Description;
 }

 public static string GetUkrainianValue(this EGrammaticalCase eGrammaticalCase)
 {
 var type = typeof(EGrammaticalCase);
 var memInfo = type.GetMember(eGrammaticalCase.ToString());
 var attributes = memInfo[0].GetCustomAttributes(typeof(DescriptionAttribute), false);
 return ((DescriptionAttribute)attributes[0]).Description;
 }
 }
}
```

Лістинг AIS.TermSearcher\Entities\Section\INewSection.cs

```
using AIS.DocxReader.Entities.Section;
using System.Collections.Generic;

namespace AIS.TermSearcher.Entities.Section
{
 public interface INewSection : ISection
 {
 List<Term.Term> KeyTerms { get; set; }
 }
}
```

Лістинг AIS.TermSearcher\Entities\Section\NewSection.cs

```
using AIS.DocxReader.Entities.Section;
using System.Collections.Generic;
using System.Linq;

namespace AIS.TermSearcher.Entities.Section
{
 public class NewSection : DocxReader.Entities.Section.Section, INewSection
 {
 public List<Term.Term> KeyTerms { get; set; }

 public NewSection() { }

 public NewSection(ISection section)
 {
 Level = section.Level;
 Text = section.Text;
 Name = section.Name;
 Guid = section.Guid;
 Paragraphs = section.Paragraphs;
 Sections = new List<ISection>();

 if (section.Sections != null && section.Sections.Any())
 {
 foreach (var item in section.Sections)
 {
 Sections.Add(new NewSection(item));
 }
 }
 }
 }
}
```

Лістинг AIS.TermSearcher\Entities\Term\Term.cs

```
using System.Collections.Generic;
using System.Linq;

namespace AIS.TermSearcher.Entities.Term
{
 public class Term
 {
 private string text;
 private string infinitive;

 public string Text
 {
 get
 {
 if (string.IsNullOrEmpty(this.text))
 {
 this.text = string.Join(" ", this.Words.Select(x => x.Text));
 }

 return this.text;
 }
 }

 public string Infinitive
 {
 get
 {
 if (string.IsNullOrEmpty(this.infinitive))
 {
 this.infinitive = string.Join(" ", this.Words.Select(x => x.Infinitive));
 }

 return this.infinitive;
 }
 }

 public double Evaluation { get; set; }

 public List<Term> AllExistedForm { get; set; }

 public List<Word.Word> Words { get; set; }

 public int Count { get; set; }

 public int Position { get; set; }

 public Term()
 {

 }

 public Term(string text)
 {
 this.text = text;
 }

 }
}
```

Лістинг AIS.TermSearcher\Entities\Word\EGrammaticalCase.cs

```
using System.ComponentModel;

namespace AIS.TermSearcher.Entities.Word
{
 public enum EGrammaticalCase
 {
 /// <summary>Default</summary>
 [Description("None")]
 None,
```

```
 /// <summary>Називний</summary>
 [Description("Називний")]
 Nominative,
 /// <summary>Родовий</summary>
 [Description("Родовий")]
 Genitive,
 /// <summary>Давальний</summary>
 [Description("Давальний")]
 Dative,
 /// <summary>Знахідний </summary>
 [Description("Знахідний ")]
 Accusative,
 /// <summary>Орудний </summary>
 [Description("Орудний ")]
 Instrumental,
 /// <summary>Місцевий</summary>
 [Description("Місцевий")]
 Prepositional,
 /// <summary>Кличний</summary>
 [Description("Кличний")]
 Vocative
 }
}
```

Лістинг AIS.TermSearcher\Entities\Word\EGrammaticalGender.cs

```
using System.ComponentModel;

namespace AIS.TermSearcher.Entities.Word
{
 public enum EGrammaticalGender
 {
 /// <summary>Default</summary>
 [Description("Відсутній")]
 None,
 /// <summary>Чоловічий</summary>
 [Description("Чоловічий")]
 Masculine,
 /// <summary>Жіночий</summary>
 [Description("Жіночий")]
 Feminine,
 /// <summary>Середній</summary>
 [Description("Середній")]
 Neuter
 }
}
```

Лістинг AIS.TermSearcher\Entities\Word\ESpeechPart.cs

```
using System.ComponentModel;

namespace AIS.TermSearcher.Entities.Word
{
 public enum ESpeechPart
 {
 /// <summary>Default</summary>
 [Description("None")]
 None,
 /// <summary>Іменник</summary>
 [Description("Іменник")]
 Noun,
 /// <summary>Прикметник</summary>
 [Description("Прикметник")]
 Adjective,
 /// <summary>Числівник</summary>
 [Description("Числівник")]
 Numeral,
 /// <summary>Займенник</summary>
 [Description("Займенник")]
 Pronoun,
 /// <summary>Дієслово</summary>
 [Description("Дієслово")]
 Verb,
 /// <summary>Дієприкметник</summary>
 [Description("Дієприкметник")]
 Participle,
 /// <summary>Дієприслівник</summary>
 [Description("Дієприслівник")]
 Transgressive,
 /// <summary>Прислівник</summary>
 [Description("Прислівник")]
 Adverb,
 /// <summary>Сполучник</summary>
 [Description("Сполучник")]
 Conjunction,
 /// <summary>Прийменник</summary>
 [Description("Прийменник")]
 Preposition,
 /// <summary>Частка</summary>
 [Description("Частка")]
 Particle,
 }
}
```

Лістинг AIS.TermSearcher\Entities\Word\Word.cs

```
using AIS.DocxReader.Entities.Word;

namespace AIS.TermSearcher.Entities.Word
{
 public class Word : IWord
 {
 public string Text { get; set; }

 public string Infinitive { get; set; }

 public IWordStyle WordStyle { get; set; }

 public bool IsFromDb { get; set; }

 public int Position { get; set; }

 public EGrammaticalGender GrammaticalGender {
get; set; }

 public EGrammaticalCase GrammaticalCase { get;
set; }

 public ESpeechPart SpeechPart { get; set; }
 }
}
```

Лістинг AIS.TermSearcher\Properties\AssemblyInfo.cs

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// General Information about an assembly is
controlled through the following
// set of attributes. Change these attribute
values to modify the information
// associated with an assembly.
[assembly:
AssemblyTitle("AIS.KeyTermRetriever")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly:
AssemblyProduct("AIS.KeyTermRetriever")]
[assembly: AssemblyCopyright("Copyright ©
2018")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types
in this assembly not visible
// to COM components. If you need to access a
type in this assembly from
// COM, set the ComVisible attribute to true on
that type.
[assembly: ComVisible(false)]

// The following GUID is for the ID of the
typelib if this project is exposed to COM
[assembly: Guid("bfb9d0ee-c2a4-4b73-9421-
8fe674e766c2")]

// Version information for an assembly consists
of the following four values:
//
// Major Version
// Minor Version
// Build Number
// Revision
//
// You can specify all the values or you can
default the Build and Revision Numbers
// by using the '*' as shown below:
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

Лістинг AIS.TermSearcher\Searcher\ISearchManager.cs

```
using AIS.DocxReader.Entities.Document;
using AIS.TermSearcher.Entities.Term;
using AIS.TermSearcher.Entities.Word;
using System;
using System.Collections.Generic;

namespace AIS.TermSearcher.Searcher
{
 public interface ISearchManager
 {
 Dictionary<Guid, List<Term>> KeyTerms { get; }

 Dictionary<Guid, List<Term>> CompresedKeyTerms
{ get; }

 Dictionary<Guid, List<Term>>
KeyTermAfterAbsorption { get; }

 Dictionary<Guid, List<Term>> KeyTermMaskResult
{ get; }

 Dictionary<Guid, List<Term>> NauseaResult {
get; }

 List<Word> UnknownWords { get; }

 IDocument AnalizedDocument { get; }

 void Search();
 }
}
```

Лістинг AIS.TermSearcher\Searcher\SearchManager.cs

```
using AIS.DocxReader.Entities.Document;
using AIS.DocxReader.Entities.Section;
using AIS.TermSearcher.Entities.Section;
using AIS.TermSearcher.Entities.Term;
using AIS.TermSearcher.Entities.Word;
using AIS.TermSearcher.Searcher.Settings;
using
AIS.TermSearcher.Searcher.TermMaskValidation;
using AIS.TermSearcher.Searcher.TermValuation;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;

namespace AIS.TermSearcher.Searcher
{
 class SearchManager : ISearchManager
 {
 #region Fields and Properties
 private readonly IMaskValidator maskValidator;
 private readonly IEvaluator evaluator;
 private readonly SearchSetting searchSetting;
 private readonly int wordCountInTerm;
 private readonly IDocument document;
 private Dictionary<string, Word> wordsInfo;

 private Dictionary<Guid, List<Term>> keyTerms;
 private Dictionary<Guid, List<Term>>
keyTermMaskResult;
 private Dictionary<Guid, List<Term>>
keyTermAfterAbsorption;
 private Dictionary<Guid, List<Term>>
compresedKeyTerms;
 private Dictionary<Guid, List<Term>>
correctedKeyTerms;
 private Dictionary<Guid, List<Term>>
nauseaResult;

 public Dictionary<Guid, List<Term>> KeyTerms
 {
 get
 {
 return this.keyTerms;
 }
 }

 public Dictionary<Guid, List<Term>>
CompresedKeyTerms
 {
 get
 {
 return this.compresedKeyTerms;
 }
 }

 public Dictionary<Guid, List<Term>>
CorrectedKeyTerms
 {
 get
 {
 return this.correctedKeyTerms;
 }
 }

 public Dictionary<Guid, List<Term>>
KeyTermAfterAbsorption
 {
 get
 {
 return this.keyTermAfterAbsorption;
 }
 }

 public Dictionary<Guid, List<Term>>
KeyTermMaskResult
 {
 get
 {
 return this.keyTermMaskResult;
 }
 }

 public Dictionary<Guid, List<Term>>
NauseaResult
 {
 get
 {
 return this.nauseaResult;
 }
```

```
 }

 public IDocument AnalizedDocument =>
this.document;

 public List<Word> UnknownWords =>
this.wordsInfo.Values.Where(x =>
!x.IsFromDb).ToList();
 #endregion

 public SearchManager(IMaskValidator
maskValidator, IEvaluator evaluator, IDocument
document, SearchSetting searchSetting, int
wordCountInTerm)
 {
 if (document == null)
 throw new
ArgumentNullException(nameof(document));
 if (evaluator == null)
 throw new
ArgumentNullException(nameof(evaluator));
 if (maskValidator == null)
 throw new
ArgumentNullException(nameof(maskValidator));
 if (searchSetting == null)
 throw new
ArgumentNullException(nameof(searchSetting));

 this.evaluator = evaluator;
 this.searchSetting = searchSetting;
 this.wordCountInTerm = wordCountInTerm;
 this.maskValidator = maskValidator;
 this.wordsInfo = new Dictionary<string,
Word>();

 this.document = new Document
 {
 Name = document.Name,
 Sections = new List<ISection>()
 };

 foreach (var section in document.Sections)
 {
 this.document.Sections.Add(new
NewSection(section));
 }
 }

 public void Search()
 {
 this.keyTerms = new Dictionary<Guid,
List<Term>>();
 this.keyTermMaskResult = new Dictionary<Guid,
List<Term>>();
 this.compresedKeyTerms = new Dictionary<Guid,
List<Term>>();
 this.correctedKeyTerms = new Dictionary<Guid,
List<Term>>();
 this.keyTermAfterAbsorption = new
Dictionary<Guid, List<Term>>();
 this.nauseaResult = new Dictionary<Guid,
List<Term>>();

 foreach (var section in
this.document.Sections)
 {
 SearchSubSection(section as INewSection);
 }
 }

 private void SearchSubSection(INewSection
section)
 {
 foreach (var subSection in section.Sections)
 {
 SearchSubSection(subSection as INewSection);
 }

 DetermineKeyword(section);
 SetWordInfo(section);
 FilterTerm(section);
 CompressTerms(section);
 CompactifyTerm(section);
 CorrectEvaliation(section,
this.searchSetting.CorrectionSetting);
 NauseaTerm(section);
 }

 private void DetermineKeyword(INewSection
section)
 {
 List<Term> keyTerms = new List<Term>();
 for (int i = 1; i <= this.wordCountInTerm;
i++)
 {
 List<Term> result =
this.evaluator.Evaluate(section, i);
 keyTerms.AddRange(result);
 }

 this.keyTerms.Add(section.Guid, keyTerms);
 section.KeyTerms = keyTerms;
 }

 private void SetWordInfo(INewSection section)
 {
 List<Word> allWords = new List<Word>();
 foreach (var term in section.KeyTerms)
 {
 allWords.AddRange(term.Words);
 }

 List<string> uniqueWords = allWords
 .ToLookup(word => word.Text)
 .Select(a => a.First().Text)
 .ToList();

 using (DbProvider dbProvider = new
DbProvider())
 {
 Dictionary<string, Word> newWordsInfo =
dbProvider.GetWordsInfo(uniqueWords.Except(this
.wordsInfo.Keys).ToList());
 foreach (var word in newWordsInfo)
 {
 this.wordsInfo.Add(word.Key, word.Value);
 }
 }

 foreach (var term in section.KeyTerms)
 {
 foreach (var word in term.Words)
 {
 Word wordInfo = this.wordsInfo[word.Text];

 word.IsFromDb = true;
 word.Infinitive = wordInfo.Infinitive;
 word.GrammaticalGender =
wordInfo.GrammaticalGender;
 word.GrammaticalCase =
wordInfo.GrammaticalCase;
 word.SpeechPart = wordInfo.SpeechPart;
 }
 }
 }

 private void FilterTerm(INewSection section)
 {
 section.KeyTerms =
this.maskValidator.Validate(section.KeyTerms);
 this.keyTermMaskResult.Add(section.Guid,
section.KeyTerms);
 }

 private void CompressTerms(INewSection
section)
 {
 List<Term> compresedKeyTerm = new
List<Term>();
 List<string> uniqueInfinitive =
section.KeyTerms.Select(x =>
x.Infinitive).Distinct().ToList();

 foreach (var infinitive in uniqueInfinitive)
 {
 var keyTermForms = section.KeyTerms.Where(x =>
x.Infinitive == infinitive).ToList();
 double maxEvaluation = keyTermForms.Max(x =>
x.Evaluation);
 if (maxEvaluation == 0)
 {
 continue;
 }

 int count = keyTermForms.Sum(x => x.Count);
 Term readableForm =
GetReadableKeyTermForm(keyTermForms);
 if (readableForm == null)
 {
 readableForm = keyTermForms.First(x =>
x.Evaluation == maxEvaluation);
 }

 readableForm.AllExistedForm = keyTermForms;
 readableForm.Count = count;
 readableForm.Evaluation = maxEvaluation;
 readableForm.Words = readableForm.Words;

 compresedKeyTerm.Add(readableForm);
 }

 section.KeyTerms = compresedKeyTerm;
 this.compresedKeyTerms.Add(section.Guid,
compresedKeyTerm);
 }

 private void CompactifyTerm(INewSection
section)
 {
 List<Term> termsToSave = new List<Term>();
 for (int i = 1; i < this.wordCountInTerm; i++)
 {
 var smallTerms = section.KeyTerms.Where(t =>
t.Words.Count == i);
 var bigTerms = section.KeyTerms.Where(t =>
t.Words.Count == i + 1);

 foreach (var smallTerm in smallTerms)
 {
 var hasBigVersion = bigTerms.FirstOrDefault(t
=>
 new
Regex($"^{smallTerm.Infinitive}\\s|\\s{smallTer
m.Infinitive}$").IsMatch(t.Infinitive)
 && smallTerm.Count <= t.Count * 2);

 if (hasBigVersion == null)
 {
 termsToSave.Add(smallTerm);
 }
 }
 }

 section.KeyTerms = termsToSave;
 this.keyTermAfterAbsorption.Add(section.Guid,
termsToSave);
 }

 private void CorrectEvaliation(INewSection
section, CorrectionSetting correctionSetting)
 {
 List<Term> correctedKeyTerm = new
List<Term>();

 foreach (var term in section.KeyTerms)
 {
 bool isBold = term.Words.Any(word =>
word.WordStyle?.Bold != null);
 bool isItalic = term.Words.Any(word =>
word.WordStyle?.Italic != null);
 bool isUnderline = term.Words.Any(word =>
word.WordStyle?.Underline != null);

 if (isBold)
 term.Evaluation *=
correctionSetting.BoldFontCoefficient;

 if (isItalic)
 term.Evaluation *=
correctionSetting.ItalicFontCoefficient;

 if (isUnderline)
 term.Evaluation *=
correctionSetting.UnderlineFontCoefficient;

 correctedKeyTerm.Add(term);
 }

 this.correctedKeyTerms.Add(section.Guid,
correctedKeyTerm);
 }

 private void NauseaTerm(INewSection section)
 {
 int termCount = (int)(section.KeyTerms.Count *
0.15);

 List<Term> nausea = section.KeyTerms
 .OrderByDescending(x => x.Evaluation)
 .Take(termCount)
 .ToList();

 section.KeyTerms = nausea;
 this.nauseaResult.Add(section.Guid, nausea);
 }

 private Term GetReadableKeyTermForm(List<Term>
options)
 {
 Term firstTerm = options.First();

 if (firstTerm.Words.Count == 1)
 {
 return new Term(firstTerm.Infinitive)
 {
 Words = firstTerm.Words
 };
 }

 foreach (var keyTerm in options)
 {
 var selectedKeyTerm =
keyTerm.Words.FirstOrDefault(x => x.SpeechPart
== ESpeechPart.Noun && x.Text == x.Infinitive);
 if (selectedKeyTerm != null)
 {
 return keyTerm;
 }
 }

 return null;
```

```
 }
 }
}
```

Лістинг AIS.TermSearcher\Searcher\TermSearcherFactory.cs

```
using AIS.DocxReader.Entities.Document;
using AIS.TermSearcher.Searcher.Settings;
using
AIS.TermSearcher.Searcher.TermMaskValidation;
using AIS.TermSearcher.Searcher.TermValuation;

namespace AIS.TermSearcher.Searcher
{
 public static class TermSearcherFactory
 {
 public static ISearchManager
CreateSearchManager(IDocument document, int
wordInTermCount, SearchSetting searchSetting)
 {
 VarianceEvaluator varianceEvaluator = new
VarianceEvaluator();
 UkranianMaskValidator mask = new
UkranianMaskValidator();

 return new SearchManager(mask,
varianceEvaluator, document, searchSetting,
wordInTermCount);
 }
 }
}
```

Лістинг AIS.TermSearcher\Searcher\Settings\CorrectionSetting.cs

```
namespace AIS.TermSearcher.Searcher.Settings
{
 public class CorrectionSetting
 {
 public float BoldFontCoefficient { get; set; }

 public float ItalicFontCoefficient { get; set;
}

 public float UnderlineFontCoefficient { get;
set; }
 }
}
```

Лістинг AIS.TermSearcher\Searcher\Settings\SearchSetting.cs

```
using AIS.TermSearcher.Searcher.Settings;

namespace AIS.TermSearcher.Searcher.Settings
{
 public class SearchSetting
 {
 public CorrectionSetting CorrectionSetting {
get; set; }
 }
}
```

Лістинг AIS.TermSearcher\Searcher\TermMaskValidation\DbProvider.cs

```
using AIS.TermSearcher.Entities.Word;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using static
AIS.TermSearcher.Entities.EntitiesHelper;

namespace
AIS.TermSearcher.Searcher.TermMaskValidation
{
 public class DbProvider : IDisposable
 {
 private readonly SqlConnection connection;

 public DbProvider()
 {
 string connectionString = @"Data
Source=(local);Initial
Catalog=TESTWORDSSQL;Trusted_Connection=true";
 this.connection = new
SqlConnection(connectionString);
 this.connection.Open();
 }

 public Dictionary<string, Word>
GetWordsInfo(List<string> words)
 {
 Dictionary<string, Word> wordsInfo = new
Dictionary<string, Word>();

 if (words.Any())
 {
 string query = MakeWordsQuery(words);
 SqlCommand command = new SqlCommand(query,
this.connection);

 using (SqlDataReader reader =
command.ExecuteReader())
 {
 while (reader.Read())
 {
 string key =
reader["Word"].ToString().ToLower();
 if (wordsInfo.ContainsKey(key))
 {
 continue;
 }

 Word word = new Word
 {
 IsFromDb = true,
 Infinitive = reader["Infinitive"].ToString(),
 GrammaticalGender =
GetGrammaticalGender(reader["GrammaticalGender"
].ToString()),
 GrammaticalCase =
GetGrammaticalCase(reader["GrammaticalCase"].To
String()),
 SpeechPart =
GetSpeechPart(reader["SpeechPart"].ToString())
 };
 wordsInfo.Add(key, word);
 }
 }

 List<string> unknownWords =
words.Except(wordsInfo.Keys).ToList();
 foreach (var word in unknownWords)
 {
 wordsInfo.Add(word, new Word()
 {
 IsFromDb = false,
 Text = word,
 GrammaticalGender = EGrammaticalGender.None,
 GrammaticalCase = EGrammaticalCase.None,
 SpeechPart = ESpeechPart.None
 });
 }
 }

 return wordsInfo;
 }

 private string MakeWordsQuery(List<string>
words)
 {
 string wordsQuery =
string.Format(SqlQueryResources.Words,
words.First().ToLower().Replace("'", "''"));

 foreach (var word in words.Skip(1))
 {
 wordsQuery += " OR " +
string.Format(SqlQueryResources.Words,
word.ToLower().Replace("'", "''"));
 }

 return
string.Format(SqlQueryResources.GetWordInfo,
wordsQuery);
 }

 private EGrammaticalGender
GetGrammaticalGender(string grammaticalGender)
 {
 return
GrammaticalGenderMapping[grammaticalGender.ToLo
wer()];
 }

 private ESpeechPart GetSpeechPart(string
speechPart)
 {
 speechPart = speechPart.ToLower();

 if (SpeechPartMapping.ContainsKey(speechPart))
 {
 return SpeechPartMapping[speechPart];
 }
 else
 {
 return ESpeechPart.None;
 }
 }

 private EGrammaticalCase
GetGrammaticalCase(string grammaticalCase)
 {
 grammaticalCase = grammaticalCase.ToLower();

 if
(GrammaticalCaseMapping.ContainsKey(grammatical
Case))
 {
 return
GrammaticalCaseMapping[grammaticalCase];
 }
 else
 {
 return EGrammaticalCase.None;
 }
 }

 public void Dispose()
 {
 this.connection.Close();
 }
 }
}
```

Лістинг AIS.TermSearcher\Searcher\TermMaskValidation\IMaskValidator.cs

```
using AIS.TermSearcher.Entities.Term;
using System.Collections.Generic;

namespace
AIS.TermSearcher.Searcher.TermMaskValidation
{
 public interface IMaskValidator
 {
 List<Term> Validate(List<Term> terms);
 }
}
```

Лістинг AIS.TermSearcher\Searcher\TermMaskValidation\SqlQueryResources.Designer.cs

```
//------------------------------------------
------------------------------
// <auto-generated>
// This code was generated by a tool.
// Runtime Version:4.0.30319.42000
//
// Changes to this file may cause incorrect
behavior and will be lost if
// the code is regenerated.
// </auto-generated>
//------------------------------------------
------------------------------

namespace
AIS.TermSearcher.Searcher.TermMaskValidation {
 using System;


 /// <summary>
 /// A strongly-typed resource class, for
looking up localized strings, etc.
 /// </summary>
 // This class was auto-generated by the
StronglyTypedResourceBuilder
 // class via a tool like ResGen or Visual
Studio.
 // To add or remove a member, edit your .ResX
file then rerun ResGen
 // with the /str option, or rebuild your VS
project.

[global::System.CodeDom.Compiler.GeneratedCodeA
ttribute("System.Resources.Tools.StronglyTypedR
esourceBuilder", "15.0.0.0")]

[global::System.Diagnostics.DebuggerNonUserCode
Attribute()]

[global::System.Runtime.CompilerServices.Compil
erGeneratedAttribute()]
 internal class SqlQueryResources {

 private static
global::System.Resources.ResourceManager
resourceMan;

 private static
global::System.Globalization.CultureInfo
resourceCulture;


[global::System.Diagnostics.CodeAnalysis.Suppre
ssMessageAttribute("Microsoft.Performance",
"CA1811:AvoidUncalledPrivateCode")]
 internal SqlQueryResources() {
 }

 /// <summary>
 /// Returns the cached ResourceManager
instance used by this class.
 /// </summary>
```

```
[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
 internal static global::System.Resources.ResourceManager ResourceManager {
 get {
 if (object.ReferenceEquals(resourceMan, null)) {
 global::System.Resources.ResourceManager temp = new global::System.Resources.ResourceManager("AIS.TermSearcher.Searcher.TermMaskValidation.SqlQueryResources", typeof(SqlQueryResources).Assembly);
 resourceMan = temp;
 }
 return resourceMan;
 }
 }

 /// <summary>
 /// Overrides the current thread's CurrentUICulture property for all
 /// resource lookups using this strongly typed resource class.
 /// </summary>

[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
 internal static global::System.Globalization.CultureInfo Culture {
 get {
 return resourceCulture;
 }
 set {
 resourceCulture = value;
 }
 }

 /// <summary>
 /// Looks up a localized string similar to SELECT word.[Fullword] [Word],
 /// speechPart.[Name] [SpeechPart],
 /// inflexion.Inflexion [GrammaticalCase],
 /// gender.Gender [GrammaticalGender],
 /// infinitive.Word Infinitive,
 /// speechPart.Importance
 ///FROM Infinitives infinitive
 /// INNER JOIN AllWords word ON infinitive.Id_Infinitives = word.Id_Infinitives
 /// INNER JOIN PartOfSpeach speechPart ON speechPart.Id_PartOfSpeach = infinitive.Id_PartOfSpeach
 /// INNER JOIN Inflexions inflexion
 /// ON word.Id_Inflexions = inflexion.Id_inflexion
 /// OR inflexio [rest of string was truncated]&amp;quot;;.
 /// </summary>
 internal static string GetWordInfo {
 get {
 return ResourceManager.GetString("GetWordInfo", resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to word.Fullword = &amp;apos;{0}&amp;apos;.
 /// </summary>
 internal static string Words {
 get {
 return ResourceManager.GetString("Words", resourceCulture);
 }
 }
 }
}
```

Лістинг AIS.TermSearcher\Searcher\TermMaskValidation\UkranianMaskValidator.cs

```
using AIS.TermSearcher.Entities.Term;
using AIS.TermSearcher.Entities.Word;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;

namespace AIS.TermSearcher.Searcher.TermMaskValidation
{
 public class UkranianMaskValidator : IMaskValidator
 {
 public List<Term> Validate(List<Term> terms)
 {
 List<Term> resultTerm = new List<Term>();

 foreach (var term in terms)
 {
 int wordCount = term.Words.Count;

 if (!term.Words.Any(word => word.SpeechPart == ESpeechPart.Noun))
 continue;

 if (term.Words.First().SpeechPart != ESpeechPart.Noun &&
 term.Words.First().SpeechPart != ESpeechPart.Adjective)
 continue;

 if (term.Words.Last().SpeechPart != ESpeechPart.Noun &&
 term.Words.Last().SpeechPart != ESpeechPart.Adjective)
 continue;

 int n = 0;
 for (int i = 0; i < wordCount; i++)
 {
 if (term.Words[i].SpeechPart == ESpeechPart.Noun
 || term.Words[i].SpeechPart == ESpeechPart.Adjective
 || term.Words[i].SpeechPart == ESpeechPart.Conjunction
 || term.Words[i].SpeechPart == ESpeechPart.Particle)
 {
 n++;
 }
 }

 if (n == wordCount)
 {
 resultTerm.Add(term);
 }
 }

 return resultTerm;
 }
 }
}
```

Лістинг AIS.TermSearcher\Searcher\TermValuation\IEvaluator.cs

```
using AIS.TermSearcher.Entities.Section;
using AIS.TermSearcher.Entities.Term;
using System.Collections.Generic;

namespace AIS.TermSearcher.Searcher.TermValuation
{
 public interface IEvaluator
 {
 List<Term> Evaluate(INewSection section, int wordCount);
 }
}
```

Лістинг AIS.TermSearcher\Searcher\TermValuation\VarianceEvaluator.cs

```
using AIS.TermSearcher.Entities.Section;
using AIS.TermSearcher.Entities.Term;
using AIS.TermSearcher.Entities.Word;
using System;
using System.Collections.Generic;
using System.Linq;

namespace AIS.TermSearcher.Searcher.TermValuation
{
 public class VarianceEvaluator : IEvaluator
 {
 public List<Term> Evaluate(INewSection section, int wordCount)
 {
 List<Term> resultValuation = new List<Term>();
 List<Term> termsCondidates = GetTermsCandidates(section, wordCount);

 if (!termsCondidates.Any())
 {
 return resultValuation;
 }

 int lastTermPosition = termsCondidates.Last().Position;
 termsCondidates.Sort((term1, term2) => string.Compare(term1.Text, term2.Text));

 List<Term> uniqueTermsCondidates = termsCondidates
 .ToLookup(term => term.Text)
 .Select(a => a.First())
 .ToList();

 foreach (var uniqueTerm in uniqueTermsCondidates)
 {
 double evaluation;
 int sumDist = 0, sumDistSquer = 0;

 List<Term> selectedTerms = BinariSearch(termsCondidates, uniqueTerm.Text);
 selectedTerms = selectedTerms.OrderBy(a => a.Position).ToList();

 int termCount = selectedTerms.Count;
 if (termCount == 1)
 {
 evaluation = 0;
 }
 else
 {
 for (int j = 0; j < selectedTerms.Count - 1; j++)
 {
 sumDist += (selectedTerms[j + 1].Position - selectedTerms[j].Position) - wordCount;
 sumDistSquer += (int)Math.Pow((selectedTerms[j + 1].Position - selectedTerms[j].Position) - wordCount, 2);
 }

 sumDist += (lastTermPosition - selectedTerms.Last().Position + selectedTerms.First().Position) - wordCount;
 sumDistSquer += (int)Math.Pow((lastTermPosition - selectedTerms.Last().Position + selectedTerms.First().Position) - wordCount, 2);

 evaluation = Math.Sqrt(((double)sumDistSquer / (selectedTerms.Count)) - Math.Pow(((double)sumDist / (selectedTerms.Count)), 2)) / ((double)sumDist / (selectedTerms.Count));
 }

 if (evaluation > 0)
 {
 resultValuation.Add(new Term()
 {
 Evaluation = evaluation,
 Count = selectedTerms.Count,
 Words = uniqueTerm.Words
 });
 }
 }

 return resultValuation.OrderByDescending(p => p.Evaluation).ToList();
 }

 public List<Term> GetTermsCandidates(INewSection section, int wordCount)
 {
 List<Word> words;
 List<Term> termsCondidates = new List<Term>();
 int position = 0;

 foreach (var paragraph in section.AllParagraphs)
 {
 foreach (var sentences in paragraph.Sentences)
 {
 foreach (var textContainer in sentences.TextContainers)
 {
 words = textContainer.Words.Select(word => new Word()
 {
 Text = word.Text.ToLower(),
 WordStyle = word.WordStyle
 }).ToList();

 for (int i = 0; i < words.Count; i++)
 {
 Term term = new Term()
 {
 Words = new List<Word>(),
 Position = position
 };

 for (int j = 0; j < wordCount && i + j < words.Count; j++)
 {
 term.Words.Add(words[i + j]);
```

```
 }

 if (term.Words.Count == wordCount)
 {
 termsCondidates.Add(term);
 }

 position++;
 }
 }
 }
 }

 return termsCondidates;
 }

 List<Term> BinariSearch(List<Term>
allKeywordOptions, string value)
 {
 List<Term> option = new List<Term>();
 int start = 0, end = allKeywordOptions.Count;
 while (start < end)
 {
 int center = (start + end) / 2;
 switch (String.Compare(value,
allKeywordOptions[center].Text))
 {
 case -1:
 end = center;
 break;
 case 0:
 int incriment = 0;
 bool left = true, rigth = true;
 while (left || rigth)
 {
 if (left && (center - incriment < 0 ||
allKeywordOptions[center - incriment].Text !=
value))
 left = false;
 if (rigth && (center + incriment + 1 >=
allKeywordOptions.Count ||
allKeywordOptions[center + incriment + 1].Text
!= value))
 rigth = false;
 if (left)
 option.Add(allKeywordOptions[center -
incriment]);
 if (rigth)
 option.Add(allKeywordOptions[center +
incriment + 1]);
 incriment++;
 }

 return option;
 case 1: start = center + 1; break;
 }
 }

 return option;
 }
 }
}
```

Лістинг
AIS.TermSearcher.GUITests\Messages.Designer.cs

```
//------------------------------------------------
------------------------------
// <auto-generated>
// This code was generated by a tool.
// Runtime Version:4.0.30319.42000
//
// Changes to this file may cause incorrect
behavior and will be lost if
// the code is regenerated.
// </auto-generated>
//------------------------------------------------
------------------------------

namespace AIS.TermSearcher.GUITests {
 using System;


 /// <summary>
 /// A strongly-typed resource class, for
looking up localized strings, etc.
 /// </summary>
 // This class was auto-generated by the
StronglyTypedResourceBuilder
 // class via a tool like ResGen or Visual
Studio.
 // To add or remove a member, edit your .ResX
file then rerun ResGen
 // with the /str option, or rebuild your VS
project.

[global::System.CodeDom.Compiler.GeneratedCodeA
ttribute("System.Resources.Tools.StronglyTypedR
esourceBuilder", "16.0.0.0")]

[global::System.Diagnostics.DebuggerNonUserCode
Attribute()]

[global::System.Runtime.CompilerServices.Compil
erGeneratedAttribute()]
 internal class Messages {

 private static
global::System.Resources.ResourceManager
resourceMan;

 private static
global::System.Globalization.CultureInfo
resourceCulture;


[global::System.Diagnostics.CodeAnalysis.Suppre
ssMessageAttribute("Microsoft.Performance",
"CA1811:AvoidUncalledPrivateCode")]
 internal Messages() {
 }

 /// <summary>
 /// Returns the cached ResourceManager
instance used by this class.
 /// </summary>

[global::System.ComponentModel.EditorBrowsableA
ttribute(global::System.ComponentModel.EditorBr
owsableState.Advanced)]
 internal static
global::System.Resources.ResourceManager
ResourceManager {
 get {
 if (object.ReferenceEquals(resourceMan, null))
{
 global::System.Resources.ResourceManager temp
= new
global::System.Resources.ResourceManager("AIS.T
ermSearcher.GUITests.Messages",
typeof(Messages).Assembly);
 resourceMan = temp;
 }
 return resourceMan;
 }
 }

 /// <summary>
 /// Overrides the current thread's
CurrentUICulture property for all
 /// resource lookups using this strongly typed
resource class.
 /// </summary>

[global::System.ComponentModel.EditorBrowsableA
ttribute(global::System.ComponentModel.EditorBr
owsableState.Advanced)]
 internal static
global::System.Globalization.CultureInfo
Culture {
 get {
 return resourceCulture;
 }
 set {
 resourceCulture = value;
 }
 }

 /// <summary>
 /// Looks up a localized string similar to The
element does not exist.
 /// </summary>
 internal static string ElementDoesNotExist {
 get {
 return
ResourceManager.GetString("ElementDoesNotExist"
, resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to The
element location is wrong.
 /// </summary>
 internal static string ElementLocationIsWrong
{
 get {
 return
ResourceManager.GetString("ElementLocationIsWro
ng", resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to The
element size is wrong.
 /// </summary>
 internal static string ElementSizeIsWrong {
 get {
 return
ResourceManager.GetString("ElementSizeIsWrong",
resourceCulture);
 }
 }
 }
}
```

Лістинг
AIS.TermSearcher.GUITests\SeleniumHelper.cs

```
using OpenQA.Selenium;

namespace AIS.TermSearcher.GUITests
{
 internal class SeleniumHelper :
NUnit.Framework.Assert
 {
 private readonly IWebDriver driver;

 public SeleniumHelper(IWebDriver driver)
 {
 this.driver = driver;
 }

 internal bool IsElementExist(By by)
 {
 var elements = driver.FindElements(by);

 return elements.Count > 0;
 }
 }
}
```

Лістинг
AIS.TermSearcher.GUITests\TermSearcherUITests.c
s

```
using NUnit.Framework;
using OpenQA.Selenium;
using System.Drawing;

namespace AIS.TermSearcher.GUITests
{
 internal class TermSearcherUITests : TestsBase
 {
 [Test]
 public void
Check_if_setting_page_elements_exist()
 {

Assert.IsTrue(SeleniumHelper.IsElementExist(By.
Id("setting-view")),
Messages.ElementDoesNotExist);

Assert.IsTrue(SeleniumHelper.IsElementExist(By.
Id("file-on-server")),
Messages.ElementDoesNotExist);

Assert.IsTrue(SeleniumHelper.IsElementExist(By.
Id("file-loader-input")),
Messages.ElementDoesNotExist);

Assert.IsTrue(SeleniumHelper.IsElementExist(By.
Id("search-stage")),
Messages.ElementDoesNotExist);

Assert.IsTrue(SeleniumHelper.IsElementExist(By.
Id("bold-value")),
Messages.ElementDoesNotExist);

Assert.IsTrue(SeleniumHelper.IsElementExist(By.
Id("italic-value")),
Messages.ElementDoesNotExist);

Assert.IsTrue(SeleniumHelper.IsElementExist(By.
Id("underline-value")),
Messages.ElementDoesNotExist);

Assert.IsTrue(SeleniumHelper.IsElementExist(By.
Id("searchButton")),
Messages.ElementDoesNotExist);
 }

 [Test]
 public void
Check_location_of_setting_page_elements()
 {
 var fileOnServerElement =
driver.FindElement(By.Id("file-on-server"));
 Assert.AreEqual(new Point(55, 5),
fileOnServerElement.Location,
Messages.ElementLocationIsWrong);

 var settingViewElement =
driver.FindElement(By.Id("setting-view"));
 Assert.AreEqual(new Point(50, 0),
settingViewElement.Location,
Messages.ElementLocationIsWrong);
```

```
 var fileLoaderInputElement =
driver.FindElement(By.Id("file-loader-input"));
 Assert.AreEqual(new Point(55, 23),
fileLoaderInputElement.Location,
Messages.ElementLocationIsWrong);

 var searchStageElement =
driver.FindElement(By.Id("search-stage"));
 Assert.AreEqual(new Point(60, 67),
searchStageElement.Location,
Messages.ElementLocationIsWrong);

 var boldValueElement =
driver.FindElement(By.Id("bold-value"));
 Assert.AreEqual(new Point(126, 94),
boldValueElement.Location,
Messages.ElementLocationIsWrong);

 var italicValueElement =
driver.FindElement(By.Id("italic-value"));
 Assert.AreEqual(new Point(126, 119),
italicValueElement.Location,
Messages.ElementLocationIsWrong);

 var underlineValueElement =
driver.FindElement(By.Id("underline-value"));
 Assert.AreEqual(new Point(126, 144),
underlineValueElement.Location,
Messages.ElementLocationIsWrong);

 var searchButtonElement =
driver.FindElement(By.Id("searchButton"));
 Assert.AreEqual(new Point(55, 186),
searchButtonElement.Location,
Messages.ElementLocationIsWrong);
 }

 [Test]
 public void
Check_size_of_setting_page_elements()
 {
 var settingViewElement =
driver.FindElement(By.Id("setting-view"));
 Assert.AreEqual(new Size(898, 212),
settingViewElement.Size,
Messages.ElementSizeIsWrong);

 var fileOnServerElement =
driver.FindElement(By.Id("file-on-server"));
 Assert.AreEqual(new Size(176, 17),
fileOnServerElement.Size,
Messages.ElementSizeIsWrong);

 var fileLoaderInputElement =
driver.FindElement(By.Id("file-loader-input"));
 Assert.AreEqual(new Size(253, 21),
fileLoaderInputElement.Size,
Messages.ElementSizeIsWrong);

 var searchStageElement =
driver.FindElement(By.Id("search-stage"));
 Assert.AreEqual(new Size(131, 19),
searchStageElement.Size,
Messages.ElementSizeIsWrong);

 var boldValueElement =
driver.FindElement(By.Id("bold-value"));
 Assert.AreEqual(new Size(177, 21),
boldValueElement.Size,
Messages.ElementSizeIsWrong);

 var italicValueElement =
driver.FindElement(By.Id("italic-value"));
 Assert.AreEqual(new Size(177, 21),
italicValueElement.Size,
Messages.ElementSizeIsWrong);

 var underlineValueElement =
driver.FindElement(By.Id("underline-value"));
 Assert.AreEqual(new Size(177, 21),
underlineValueElement.Size,
Messages.ElementSizeIsWrong);

 var searchButtonElement =
driver.FindElement(By.Id("searchButton"));
 Assert.AreEqual(new Size(173, 21),
searchButtonElement.Size,
Messages.ElementSizeIsWrong);
 }
 }
}
```

Лістинг AIS.TermSearcher.GUITests\TestsBase.cs

```
using NUnit.Framework;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using System;
using System.Configuration;

namespace AIS.TermSearcher.GUITests
{
 internal class TestsBase
 {
 protected IWebDriver driver;

 public SeleniumHelper SeleniumHelper { get;
private set; }

 [SetUp]
 public void SetUp()
 {
 driver = new ChromeDriver();
 SeleniumHelper = new SeleniumHelper(driver);

 if
(Uri.TryCreate(ConfigurationManager.AppSettings
["url"], UriKind.Absolute, out Uri uri))
 {
 driver.Url = uri.ToString();
 }
 else
 {
 throw new ArgumentException(nameof(uri));
 }
 }

 [TearDown]
 public void TearDown()
 {
 driver.Quit();
 }
 }
}
```

Лістинг AIS.TermSearcher.GUITests\Properties\AssemblyInfo.cs

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// General Information about an assembly is
controlled through the following
// set of attributes. Change these attribute
values to modify the information
// associated with an assembly.
[assembly:
AssemblyTitle("AIS.TermSearcher.GUITests")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly:
AssemblyProduct("AIS.TermSearcher.GUITests")]
[assembly: AssemblyCopyright("Copyright ©
2020")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types
in this assembly not visible
// to COM components. If you need to access a
type in this assembly from
// COM, set the ComVisible attribute to true on
that type.
[assembly: ComVisible(false)]

// The following GUID is for the ID of the
typelib if this project is exposed to COM
[assembly: Guid("89c72613-c0b4-45a7-81a2-
a96baeab3d9d")]

// Version information for an assembly consists
of the following four values:
//
// Major Version
// Minor Version
// Build Number
// Revision
//
// You can specify all the values or you can
default the Build and Revision Numbers
// by using the '*' as shown below:
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

Лістинг AIS.TermSearcher.UI\Global.asax.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace AIS.TermSearcher.UI
{
 public class MvcApplication :
System.Web.HttpApplication
 {
 protected void Application_Start()
 {
 AreaRegistration.RegisterAllAreas();
 RouteConfig.RegisterRoutes(RouteTable.Routes);
 }
 }
}
```

Лістинг AIS.TermSearcher.UI\App_Start\RouteConfig.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace AIS.TermSearcher.UI
{
 public class RouteConfig
 {
 public static void
RegisterRoutes(RouteCollection routes)
 {

routes.IgnoreRoute("{resource}.axd/{*pathInfo}"
);

 routes.MapRoute(
 name: "Default",
 url: "{controller}/{action}/{id}",
 defaults: new { controller = "Index", action =
"Index", id = UrlParameter.Optional }
 );
 }
 }
}
```

Лістинг AIS.TermSearcher.UI\Controllers\DocumentController.cs

```
using AIS.DocxReader.Entities.Document;
using AIS.TermSearcher.Searcher;
using AIS.TermSearcher.Searcher.Settings;
using AIS.TermSearcher.UI.Models;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Web.Mvc;

namespace AIS.TermSearcher.UI.Controllers
{
 public class DocumentController : Controller
 {
 public void UploadFile()
 {
 foreach (string upload in this.Request.Files)
 {
 if
(!string.IsNullOrWhiteSpace(this.Request.Files[
upload].FileName))
 {
 string directoryPath =
AppDomain.CurrentDomain.BaseDirectory +
"App_Data\\Uploads";
 string fileName =
Path.GetFileName(this.Request.Files[upload].Fil
eName);

 DocumentManager.FilePath =
Path.Combine(directoryPath, fileName);


this.Request.Files[upload].SaveAs(DocumentManag
er.FilePath);
 }
 }
 }

 public ActionResult Search(float bold, float
italic, float underline)
 {
 DocxReader.DocxReader reader = new
DocxReader.DocxReader(DocumentManager.FilePath)
;
 IDocument document = reader.Read();

 SearchSetting searchSetting = new
SearchSetting
 {
 CorrectionSetting = new CorrectionSetting()
 {
 BoldFontCoefficient = bold,
 ItalicFontCoefficient = italic,
 UnderlineFontCoefficient = underline
 }
 };
```

```
 DocumentManager.SearchManager =
TermSearcherFactory.CreateSearchManager(documen
t, 5, searchSetting);
 DocumentManager.SearchManager.Search();
 DocumentManager.Sections =
DocumentManager.CreateSections(document.Section
s);

 return Json(DocumentManager.Sections,
JsonRequestBehavior.AllowGet);
 }

 public ActionResult GetKeyTerms(string guid,
string stage)
 {
 Guid sectionGuid = Guid.Parse(guid);
 Dictionary<Guid, List<Entities.Term.Term>>
structureTerms = null;

 switch (stage)
 {
 case "candidates":
 structureTerms =
DocumentManager.SearchManager.KeyTerms;
 break;
 case "compressed":
 structureTerms =
DocumentManager.SearchManager.CompresedKeyTerms
;
 break;
 case "absorption":
 structureTerms =
DocumentManager.SearchManager.KeyTermAfterAbsor
ption;
 break;
 case "mask":
 structureTerms =
DocumentManager.SearchManager.KeyTermMaskResult
;
 break;
 case "nausea":
 default:
 structureTerms =
DocumentManager.SearchManager.NauseaResult;
 break;
 }

 if (structureTerms.ContainsKey(sectionGuid))
 {
 IEnumerable<Term> terms =
structureTerms[sectionGuid].Select(term => new
Term
 {
 Text = term.Text,
 Evaluation = term.Evaluation,
 Count = term.Count
 });

 return Json(terms,
JsonRequestBehavior.AllowGet);
 }

 return new
HttpStatusCodeResult(HttpStatusCode.NotFound);
 }

 public ActionResult GetFileName()
 {
 if
(string.IsNullOrWhiteSpace(DocumentManager.File
Path))
 {
 return new
HttpStatusCodeResult(HttpStatusCode.NotFound);
 }
 else
 {
 return Json(DocumentManager.FilePath,
JsonRequestBehavior.AllowGet);
 }
 }
 }
}
```

Лістинг AIS.TermSearcher.UI\Controllers\IndexController.cs

```
using System.Web.Mvc;

namespace AIS.TermSearcher.UI.Controllers
{
 public class IndexController : Controller
 {
 // GET: Index
 public ActionResult Index()
 {
 return View();
 }
 }
}
```

Лістинг AIS.TermSearcher.UI\Models\DocumentManager.cs

```
using AIS.DocxReader.Entities.Document;
using AIS.DocxReader.Entities.Section;
using AIS.TermSearcher.Searcher;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace AIS.TermSearcher.UI.Models
{
 internal static class DocumentManager
 {
 internal static string FilePath { get; set; }

 internal static ISearchManager SearchManager {
get; set; }

 internal static List<Section> Sections { get;
set; }

 internal static List<Section>
CreateSections(List<ISection> sections)
 {
 List<Section> newSections = new
List<Section>();

 foreach (var section in sections)
 {
 newSections.Add(new Section()
 {
 Label = section.Name,
 Guid = section.Guid,
 Sections = CreateSections(section.Sections)
 });
 }
 return newSections;
 }
 }
}
```

Лістинг AIS.TermSearcher.UI\Models\Section.cs

```
using System;
using System.Collections.Generic;

namespace AIS.TermSearcher.UI.Models
{
 public class Section
 {
 public Guid Guid { get; set; }

 public string Label { get; set; }

 public List<Section> Sections { get; set; }
 }
}
```

Лістинг AIS.TermSearcher.UI\Models\Term.cs

```
namespace AIS.TermSearcher.UI.Models
{
 public class Term
 {
 public string Text { get; set; }

 public int Count { get; set; }

 public double Evaluation { get; set; }
 }
}
```

Лістинг AIS.TermSearcher.UI\Properties\AssemblyInfo.cs

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// General Information about an assembly is
controlled through the following
// set of attributes. Change these attribute
values to modify the information
// associated with an assembly.
[assembly:
AssemblyTitle("AIS.TermSearcher.UI")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly:
AssemblyProduct("AIS.TermSearcher.UI")]
[assembly: AssemblyCopyright("Copyright ©
2019")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Version information for an assembly consists
of the following four values:
//
// Major Version
// Minor Version
// Build Number
// Revision
//
// You can specify all the values or you can
default the Revision and Build Numbers
// by using the '*' as shown below:
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

Додаток Ж

Ксерокопії наукових публікацій, виконаних при роботі над ДРМ

(ксерокопії титульної сторінки, сторінки змісту та всіх сторінок із публікацією)

Перелік наукових публікацій:

1. Мазурець О. В. Використання спеціалізованих програмних розширень для автоматизації роботи з цифровими документами навчальних матеріалів / О. В. Мазурець, О. В. Ковальчук, В. О. Слободзян // Науковий журнал «Вісник Хмельницького національного університету» серія: Технічні науки. Хмельницький, 2018, №1. – С.61-69.

2. Мазурець О. В. Метод формального опису елементів моделей автоматизованого формування тестових завдань / О. В. Мазурець, О. В. Ковальчук, В. О. Слободзян, Г. А. Білоус // Збірник наукових праць за матеріалами Всеукраїнської науково-практичної конференції «Інтелектуальний потенціал – 2018». Хмельницький, 2018, Ч.1. – С.51-56.

3. Слободзян В. О. Аналіз сучасних спеціалізованих програмних розширень для автоматизації роботи з цифровими документами навчальних матеріалів / В. О. Слободзян, О. В. Мазурець // Сучасні технології в механіці: Збірник наукових праць. Хмельницький – 2018. – С.184-191.

4. Ковальчук О. В. Використання програмного розширення spire.doc для автоматизації роботи з цифровими документами / О. В. Ковальчук, Г. А. Білоус, В. О. Слободзян // Збірник наукових праць за матеріалами XI Всеукраїнської науково-практичної конференції «Актуальні проблеми комп’ютерних наук АПКН-2019» – Хмельницький, 2019, Т.1. – С.116-122.

5. Слободзян В. О. Аналіз результатів автоматизованого пошуку ключових термінів у навчальних матеріалах / Слободзян В. О., Мазурець О. В. // Збірник наукових праць за матеріалами XII Всеукраїнської науково-практичної конференції «Актуальні проблеми комп’ютерних наук АПКН-2020». Хмельницький, 2020. – С.269-274.

ISSN 2307-5732

НАУКОВИЙ ЖУРНАЛ **1.2018**

ВІСНИК

Хмельницького національного університету

Технічні науки
Technical sciences

SCIENTIFIC JOURNAL
HERALD OF KHMELNYTSKYI NATIONAL UNIVERSITY

2018, Issue 1, Volume 257

Хмельницький

ВІСНИК
ХМЕЛЬНИЦЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ
серія: Технічні науки

Затверджений як фахове видання (перереєстрація)
Наказ МОН 04.07.2014 №793

Засновано в липні 1997 р. ***Виходить 6 разів на рік***

Хмельницький, 2018, № 1 (257)

Засновник і видавець: Хмельницький національний університет
(до 2005 р. – Технологічний університет Поділля, м. Хмельницький)

Включено до наукометричних баз:

Google Scholar	http://scholar.google.com.ua/citations?hl=uk&user=aIUP9OYAAAAJ
Index Copernicus	http://jml2012.indexcopernicus.com/passport.php?id=4538&id_lang=3
РИНЦ	http://elibrary.ru/title_about.asp?id=37650
Polish Scholarly Bibliography	https://pbn.nauka.gov.pl/journals/46221

Технічний редактор Горященко К. Л., к.т.н.
Редактор-коректор Броженко В. О.

Рекомендовано до друку рішенням вченої ради Хмельницького національного університету, протокол № 10 від 30.01.2018 р.

Адреса редакції: редакція журналу "Вісник Хмельницького національного університету"
Хмельницький національний університет
вул. Інститутська, 11, м. Хмельницький, Україна, 29016

☎ (038-2) 62-51-08 **web:** http://journals.khnu.km.ua/vestnik
e-mail: visnyk.khnu@gmail.com http://vestnik.ho.com.ua
http://lib.khnu.km.ua/visnyk_tup.htm

Зареєстровано Міністерством України у справах преси та інформації.
Свідоцтво про державну реєстрацію друкованого засобу масової інформації
Серія КВ № 9722 від 29 березня 2005 року

ЗМІСТ

РАДІОТЕХНІКА, ЕЛЕКТРОНІКА ТА ТЕЛЕКОМУНІКАЦІЇ

ТЕХНОЛОГІЇ ЛЕГКОЇ ПРОМИСЛОВОСТІ

МАШИНОЗНАВСТВО ТА ОБРОБКА МАТЕРІАЛІВ В МАШИНОБУДУВАННІ

УДК 004

О.В. МАЗУРЕЦЬ, О.В. КОВАЛЬЧУК, В.О. СЛОБОДЗЯН
Хмельницький національний університет

ВИКОРИСТАННЯ СПЕЦІАЛІЗОВАНИХ ПРОГРАМНИХ РОЗШИРЕНЬ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ З ЦИФРОВИМИ ДОКУМЕНТАМИ НАВЧАЛЬНИХ МАТЕРІАЛІВ

В статті досліджено проблему автоматизації роботи з електронними документами навчальних матеріалів у задачах визначення структури змістовних блоків у електронному документі навчального матеріалу та пошуку ключових термінів у контенті навчального матеріалу. Встановлено, що для реалізації програмної обробки електронних документів є доцільним використання спеціалізованих програмних комплексів, що надають необхідний інструментарій для програмної роботи з контентом відповідних файлів. З метою визначення найбільш ефективного програмного розширення проведено аналіз сучасних існуючих спеціалізованих програмних розширень для автоматизації обробки електронних документів: Microsoft.Office.Interap.Word.dll, DocumentFormat.OpenXml.dll та Spire.Doc.dll. За результатами аналізу встановлено, що бібліотека Microsoft.Office.Interap.Word.dll надає доступ до всіх функцій MS Office, так як працює безпосередньо з PIA, для її роботи потрібна ліцензія для MS Office на кожному клієнтському комп'ютері, MS Office завантажується у фоновому режимі, внаслідок чого займає значний обсяг оперативної пам'яті. При роботі з docx-файлом за допомогою DocumentFormat.OpenXml.dll не потрібна наявність MS Office та запуск в фоновому режимі, проте зростає складність програмного використання бібліотеки внаслідок необхідності регулярного співставлення ідентифікатора стиля з контейнером властивостей стилів. Перевагою Spire.Doc.dll визначено реалізацію функцій автоматичного співставлення стилів текстових блоків їх властивостям на рівні розширення. В результаті аналізу розглянутих бібліотек було визначено Spire.Doc.dll оптимальним варіантом для використання в автоматизації обробки електронних документів. Встановлено, що перенесення функцій автоматичного співставлення стилів текстових блоків їх властивостям з рівня функціоналу програмного коду застосунка на рівень функціоналу бібліотеки дозволяє спростити як роботу застосунка з електронним документом, так і процес програмування. Розглянуто практичні особливості використання розширення Spire.Doc.dll при роботі з електронними документами .DOCX. З метою визначення можливості застосування обраного інструменту в задачах роботи з електронними документами було проведено його практичну апробацію в складі програмної системи для вирішення прикладної задачі побудови структури електронного документу та пошуку ключових слів у його контенті. В результаті встановлено можливість та достатню ефективність використання розширення Spire.Doc.dll для роботи з електронними документами навчальних матеріалів.

Ключові слова: електронний документ, цифровий документ, навчальні матеріали, ключові терміни, Microsoft.Office.Interap.Word, DocumentFormat.OpenXml, Spire.Doc.

O.V. MAZURETS, O.V. KOVALCHYK, V.O. SLOBODZIAN
Khmelnytsky National University

USING SPECIALIZED SOFTWARE PACKEGES FOR AUTOMATION OF WORK WITH DIGITAL DOCUMENTS OF EDUCATIONAL MATERIALS

At article investigated a problem of the automation work with electronic documents of educational materials in the tasks for determining the structure of content blocks at electronic documents educational material and search of key terms in content of educational materials. It is established that for software processing of electronic documents need to use specialized program components that provide the necessary tools for work with content of files. Microsoft.Office.Interop.Word.dll, DocumentFormat.OpenXml.dll and Spire.Doc.dll were using for choose the most effective program components for automation work with electronic documents. In process audit established that Microsoft.Office.Interap.Word.dll provide access to all function of MS Office because it works with PIA. This library needs a license for MS Office at every computers of client and MS Office loaded in the background, in result it occupies many RAM. DocumentFormat.OpenXml.dll uses for work with docx file. This component does not need to install MS Office and work in the background but the complexity of use this library is increasing to the need to regularly match the style ID with the styles properties container. The advantage of Spire Doc.dll is the implementation of automatic functions for matching of style of text blocks to their properties. In result audit this libraries there was chosen Spire.Doc.dll for use to automation of processing of electronic documents. It is established that in result transfer of functions for automatic matching styles of text block to their properties from the level of the functional code of the application to the level of the functional library simplify work of the application with electronic document and programming process. There was considered practical features of Spire.Doc.dll for working with electric documents ".DOCX". There was practical use made this tool in program system for determine its the possibilities at using in the tasks for work with electronic documents. The system is designed to build the structure of an electronic document and search keywords in its content. Explored the possibility and effectiveness of using the Spire.Doc.dll library to work with electronic educational materials documents.

Keyword: digital document, electronic document, key terms, educational materials, Microsoft.Office.Interap.Word, DocumentFormat.OpenXml, Spire.Doc.

Постановка проблеми в загальному вигляді

Автоматизація вирішення ряду задач у галузі сучасної вищої освіти (оцінка відповідності навчальних матеріалів вимогам, оцінка відповідності наборів тестових завдань навчальним матеріалам, автоматизована генерація протопипів тестових завдань, реалізація гнучких алгоритмів тестування, допомога та контроль якості при формуванні навчальних матеріалів, допомога та контроль якості при формуванні тестів до навчальних матеріалів, автоматизація формування рефератів та анотацій до елементів навчальних матеріалів тощо) забезпечується через застосування онтології як методу формального опису знань, що містяться в навчальних матеріалах. Модель онтології навчального матеріалу може складатися з ключових

слів, ключових термінів, структури навчального матеріалу, атрибутів ключових слів та ключових термінів, що визначають їх властивості та забезпечують прив'язку до елементів структури навчального матеріалу [1]. За такої моделі, онтологія навчального матеріалу є методом виявлення сенсу навчального матеріалу та засобом вирішення наведеного ряду практичних задач.

Основними з етапів побудови онтології навчального матеріалу є пошук ключових термінів у контенті навчального матеріалу та побудова його логічної структури. Вхідними даними є електронний документ навчального матеріалу. Для автоматизації виконання обох наведених етапів потрібна програмна обробка відповідних цифрових файлів (зазвичай формату .DOCX). Для ефективної реалізації програмної обробки цифрових документів є доцільним використання спеціалізованих програмних комплексів, що надають об'єктно-орієнтований інструментарій для програмної роботи з контентом відповідних файлів.

Аналіз останніх досліджень

Проблему автоматизації побудови логічної структури навчального матеріалу (наприклад: Дисципліна / Розділ / Тема) пропонується вирішувати шляхом визначення ієрархії змістовних блоків у цифровому документі [1], таким чином формуючи верхній рівень вертикальної онтології відповідної навчальної дисципліни. Проблему пошуку ключових термінів у контенті навчального матеріалу пропонується вирішувати шляхом використання відповідної інформаційної технології на основі дисперсійної оцінки [2], таким чином формуючи нижній рівень онтології навчальної дисципліни.

Визначено, що онтологія навчального матеріалу розглянутої моделі може бути методом виявлення сенсу навчального матеріалу й ефективно використана в розробці інформаційних технологій для роботи з навчальними матеріалами [1]. Зокрема, з застосуванням такої моделі онтології навчального матеріалу було розв'язано ряд практичних задач: оцінка відповідності навчальних матеріалів вимогам [3], оцінка відповідності наборів тестових завдань навчальним матеріалам [4], автоматизована генерація прототипів тестових завдань [5], реалізація пучкових алгоритмів тестування [6], автоматизація формування рефератів та анотацій до елементів навчальних матеріалів [7] та інші.

Для реалізації програмної обробки цифрових документів є доцільним використання розглянутих в даній статті спеціалізованих програмних комплексів, що надають необхідний інструментарій для програмної роботи з контентом відповідних файлів, зокрема: Microsoft.Office.Interap.Word.dll [8], DocumentFormat.OpenXml.dll [9], Spire.Doc.dll [10].

Постановка задачі

Метою статті є аналіз сучасних існуючих спеціалізованих програмних розширень для автоматизації обробки цифрових документів і визначення найбільш ефективного програмного розширення. Методом визначення можливості застосування обраного інструменту в задачах роботи з цифровими документами є його практична апробація в складі програмної системи для вирішення відповідних прикладних задач.

Викладення основних матеріалів дослідження

На сучасному етапі для зберігання електронних документів навчальних матеріалів використовується файл з розширенням .DOCX (або створених на його основі .HTML, .PDF тощо). На відміну від файлів .DOC, які зберігають дані документа в одному бінарному файлі, файли .DOCX створюються за допомогою відкритого формату XML, в якому зберігаються документи як колекції окремих файлів і папок в стиснутому пакеті. .DOCX-файли містять .XML-файли і три папки, docProps, Word, і _rels (Рис. 1а), які містять властивості документа, його зміст (Рис. 1б) і відношення між файлами, тему (Рис. 1в) та включені файли (Рис. 1г). .DOCX-файли розроблені, щоб зробити вміст документів доступним і відкритим – так, текстовий документ чи зображення зберігаються як окремі прості файли в такій колекції у складі одного файлу .DOCX.

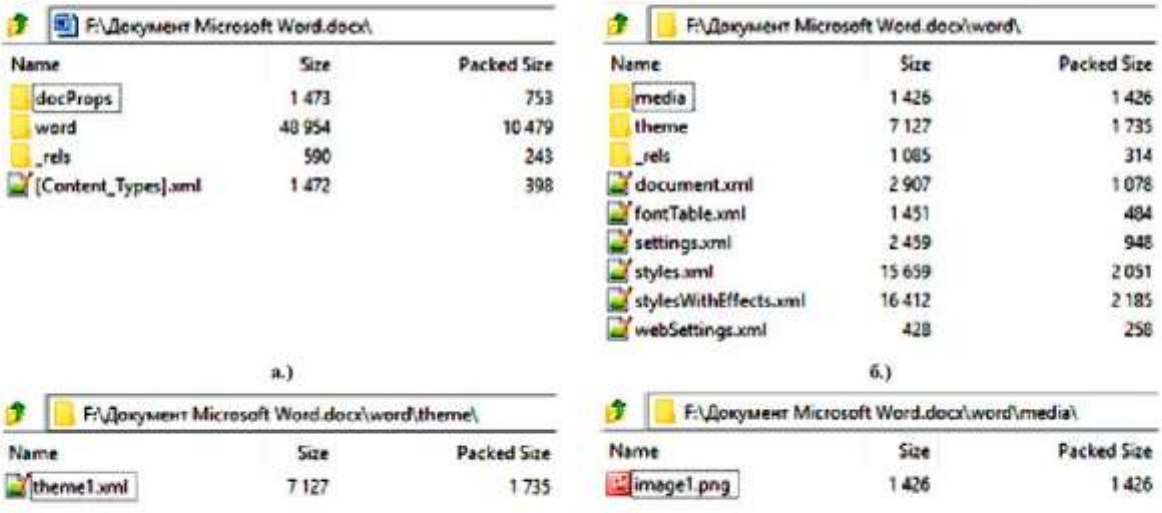


Рис. 1. Структура .DOCX-файлу

Office OpenXML є форматом електронних офісних документів, таблиць, презентацій та діаграм, розроблений компанією Microsoft. Даний формат є міжнародним стандартом, який затверджений в Міжнародній організації зі стандартизації (ISO) [11]. Він включає в себе такі первинні мови розмітки:

- WordprocessingML – для обробки текстових документів;
- SpeadsheetXL – електронних таблиць;
- PresentationML – для презентацій;
- DrawingML – для векторної графіки, діаграм.

Хоча Word реалізує свою функціональність через об'єктну модель, збереження даних відбувається у наборі XML-файлів, що ускладнює можливості для автоматизації прямого програмного парсингу [12]. Оскільки файл стилів та текст знаходяться в різних файлах у складі .XML, потрібно зчитувати одразу ряд файлів та на базі їх взаємозв'язків програмно формувати об'єктну модель документа, що є не найкращим рішенням. Тому для реалізації програмної обробки цифрових документів є доцільним використання спеціалізованих програмних комплексів, що надають об'єктно-орієнтований інструментарій для програмної роботи з контентом відповідних файлів. Найбільш відомими з таких спеціалізованих програмних комплексів є розширення Microsoft.Office.Interap.Word.dll [8], DocumentFormat.OpenXml.dll [9] та Spire.Doc.dll [10].

Microsoft.Office.Interap.Word.dll

Бібліотека Microsoft.Office.Interap.Word.dll [8] дозволяє керованому коду взаємодіяти з MS Office та об'єктною моделлю на базі COM.

За допомогою процесів автоматизації Office Automation програми, написані мовами, такими як Visual C# .NET, отримують можливість програмно керувати іншими програмами. Автоматизація Microsoft Word (далі Word) дозволяє виконувати відповідні дії, такі як створення нового документу, додавання даних у документ або створення таблиць. З програмами Word та іншими програмами Microsoft Office практично всі дії, які користувач може виконувати вручну за допомогою користувацького інтерфейсу, також можуть бути виконані програмним шляхом за допомогою автоматизації Office Automation.

Word реалізує цю програмну функціональність через об'єктну модель [13]. Об'єктна модель є набором класів і методів, які служать аналогами логічних компонентів Word. Наприклад, існує об'єкт Application, об'єкт Document та об'єкт Paragraph, кожен з яких містить функціональність цих частин Word (рис. 2).

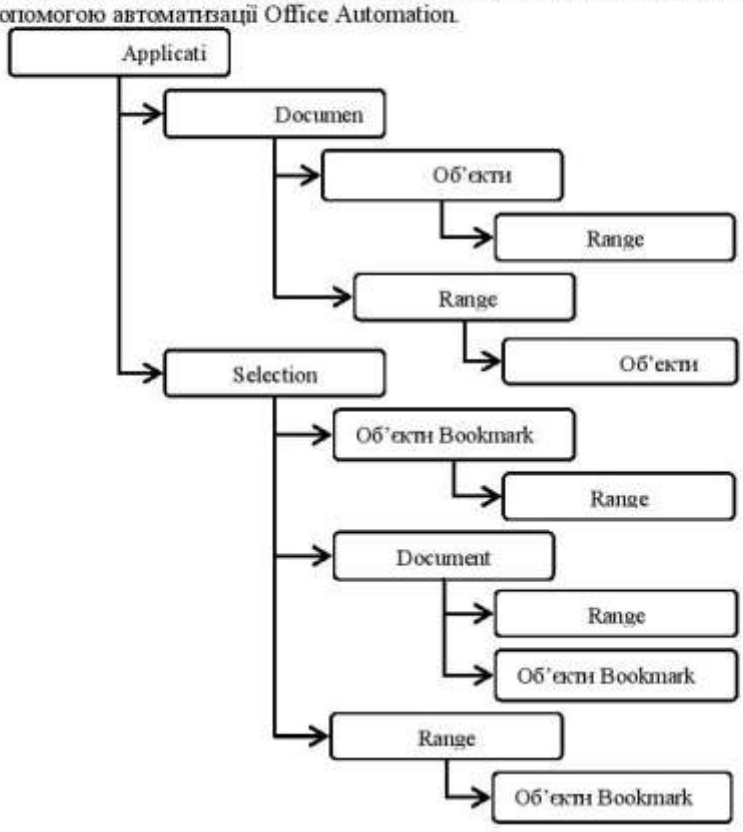


Рис. 2. Об'єктна модель Word

Щоб використовувати функції програми MS Office в проекті, можна використовувати первинний взаємозв'язок PIA (Primary Interop Assembly). PIA дозволяє керованому коду взаємодіяти з MS Office та об'єктною моделлю на базі COM. При створенні нового проекту Office, Visual Studio додає посилання на PIA, необхідний для побудови проекту. При цьому, у деяких сценаріях доводиться додавати посилання на додаткові PIA (наприклад, якщо необхідно використовувати функцію MS Office Word в проекті для MS Office Excel).

Для роботи з інтерфейсом Microsoft.Office.Interop.Word.dll необхідно встановити відповідну збірку, для чого в меню Tools потрібно обрати пункт NuGet Package Manager та запустити Package Manager Console та ввести наступну команду:

```
PM> Install-Package Microsoft.Office.Interop.Word
```

Після того як менеджер пакетів повідомить про успішне завершення встановлення, можна починати працювати з необхідними інтерфейсами.

DocumentFormat.OpenXml.dll

Бібліотека класів .NET DocumentFormat.OpenXml.dll (версії 2.0 / 2.5 / 2.8.1) [9] дає можливість розробникам програмного забезпечення працювати з пакетом Microsoft Office. Компанія надає дану бібліотеку безкоштовно в повному доступі, й її можна завантажити з офіційного сайту Microsoft або

засобами Visual Studio.

Для роботи з файлами формату .DOCX, який має зображену на рисунку 1 структуру, використовується мова розмітки WordprocessingML [14]. На рисунку 3 зображено приклад структури цієї мови розмітки.

```
<?xml version="1.0" encoding="utf-8"?>
<w:document xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main">
<w:body>
<w:p>
<w:r>
<w:t> Текст </w:t>
</w:r>
```

Рис. 3. Приклад WordprocessingML-розмітки .DOCX-документу

Теги, наведені на рис. 3, мають такі властивості:

- *document* – є основою частиною документа
- *body* – контейнер для збору структур рівня блоку, які складають основну історію;
- *p* – абзац (paragraph);
- *r* – контейнер який містить в собі текст з однаковими властивостями (run);
- *t* – власне текстовий елемент (text).

Всі інші теги, які можуть використовуватися, є допоміжними до вищенаведених і додають їм стильових властивостей.

Spire.Doc.dll

Spire.Doc.dll [10] для .NET є повністю незалежною бібліотекою класів .NET Word, спеціально створеною для розробників, й дозволяє швидко генерувати, відкривати, редагувати та зберігати документи Word різних версій. Конвертація функціональних можливостей дозволяє розробникам здійснювати перетворення Word в інші актуальні формати документів (.PDF, .EPub, .HTML, .RTF, .Image, .XML тощо). Можна створювати та обробляти вже існуючі документи Word динамічно.

Бібліотека Spire.Doc.dll забезпечує роботу з майже всіма елементами документа Word, а саме: сторінки, розділи, заголовки, колонтитули, абзаци, переліки, таблиці, текст, виноски, поля, гіперпосилання, закладки, коментарі, зображення, стилі, фонові параметри, функції друку, налаштування документа та захисту. Крім того, підтримуються графічні об'єкти, включаючи форми, текстові поля, зображення, OLE-об'єкти та елементи керування. Бібліотека Spire.Doc.dll підтримує функцію пошуку та заміни, вирівнювання, перерву сторінки, поле заповнення, об'єднання документів, копіювання документів, друк тощо.

Для роботи з бібліотекою Spire.Doc необхідно встановити відповідну збірку, для чого потрібно запустити Package Manager Console та виконати наступну команду:

```
PM> Install-Package Spire.Doc
```

Після завершення встановлення можна починати працювати з бібліотекою.

Дискусія

Хоч бібліотека Microsoft.Office.Interap.Word.dll надає доступ до всіх функцій MS Office, так як працює безпосередньо з PIA, для її роботи потрібна ліцензія для MS Office на кожному клієнтському комп'ютері. Крім того, при використанні Automation, MS Office завантажується у фоновому режимі, внаслідок чого займає деякий обсяг оперативної пам'яті та завантажує велику кількість файлів і DLL. Додатки MS Office були розроблені як додатки для користувацького інтерфейсу, і через це Microsoft.Office.Interop.Word.dll працює повільно. Microsoft не рекомендує використовувати Office Automation (або будь-який Office Interop) на сервері.

На відміну від Microsoft.Office.Interap.Word.dll, DocumentFormat.OpenXml.dll та Spire.Doc.dll не вимагають наявності MS Office на машині користувача та його запуску в фоновому режимі.

При роботі з docx-файлом за допомогою DocumentFormat.OpenXml.dll потрібно дотримуватися тієї ж ієрархії, що і сама структура розмітки тобто: *document-paragraph-run-text*. Також відповідно до розглянутих вище особливостей взаємозв'язків всередині документу, при присвоєнні параграфу деякого стилю, у властивостях параграфа надається лише ідентифікатор (ID) на даний стиль, натомість сам стиль описується окремо у файлі style.xml. Внаслідок необхідності регулярного співставлення ID стиля з контейнером style.xml для одержання характеристик стилів абзаців, зростає складність програмного використання бібліотеки.

Spire.Doc.dll не вимагає встановлення в систему кожного користувача MS Office, тобто є можливість повністю незалежної від нього роботи. Повна версія Spire.Doc.dll є платною, а безкоштовна версія FreeSpire.Doc має ряд обмежень (наприклад обробка не більше 500 абзаців і 25 таблиць) [15]. Перевагою Spire.Doc.dll визначено відсутність необхідності співставлення ID стиля з контейнером style.xml, оскільки дана функція реалізована на рівні бібліотеки.

Таким чином, в результаті аналізу сучасних існуючих спеціалізованих програмних розширень для автоматизації обробки цифрових документів, було визначено Spire.Doc.dll оптимальним варіантом для використання в автоматизації обробки цифрових документів. Основними перевагами Spire.Doc.dll встановлено відсутність необхідності наявності MS Office на машині користувача та запуску в фоновому режимі, а також реалізацію функцій автоматичного співставлення стилів текстових блоків їх властивостям на рівні розширення.

Перенесення функцій автоматичного співставлення стилів текстових блоків їх властивостям з рівня функціоналу програмного коду застосунка на рівень функціоналу бібліотеки дозволяє спростити як роботу застосунка з цифровим документом, так і процес програмування.

Програмна реалізація

При зчитуванні файлу Spire.Doc.dll перетворює документ в об'єктну модель. Ця модель має структуру, яка починається документом і закінчується об'єктом *TextRange* (рис. 4). Приклад використання такої об'єктної моделі документу MS Office показаний на рис. 5.

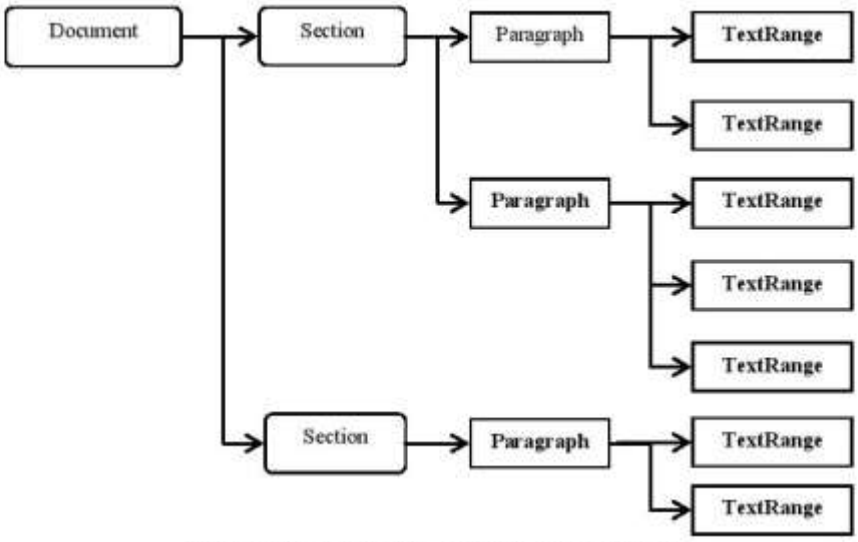


Рис. 4. Загальна структура об'єктної моделі документу

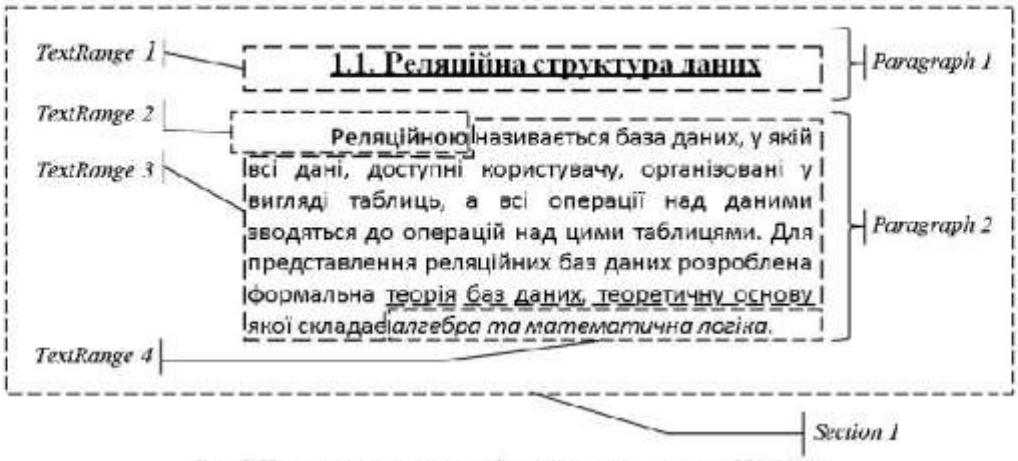


Рис. 5. Приклад використання об'єктної моделі документу MS Office

Відповідно до об'єктної моделі документу, MS Office використовує розділи (*Section*), щоб вказати частини документа, що мають різну орієнтацію сторінки, стовпці або заголовки та нижні колонтитули. Розбиття на Section дозволяє користувачеві вказати місце, де розпочинається і закінчується інше форматування. Тому Section використовуються коли потрібне використання заголовків, колонтитулів, схем, нумерації сторінок, розмірів аркушу, поля чи різні рівні захисту. Об'єкти Section містяться в об'єкті Document (рис. 5), в колекції Selections. А розділи, в свою чергу, містять в собі наступний елемент структури – абзаци (*Paragraph*).

Paragraph в MS Word визначає будь-який текст, який закінчується жорстким поверненням, яке формується, наприклад, при натисканні клавіші Enter. Формат абзацу дозволяє контролювати зовнішній вигляд фрагменту, якщо є окремі абзаци. Наприклад, можна змінити відстань між рядками або вирівнювання тексту по лівому краю на вирівнювання по центру. Можна додати індивідуальні стилі,

відступи для абзаців, або визначити заголовки і списки. В об'єктній моделі *Paragraph* знаходиться в *Document -> Sections -> Section -> Paragraphs* (рис. 6). Одержати відомості про стилі *Paragraph* можна за допомогою методу *GetStyle()* (рис. 7).

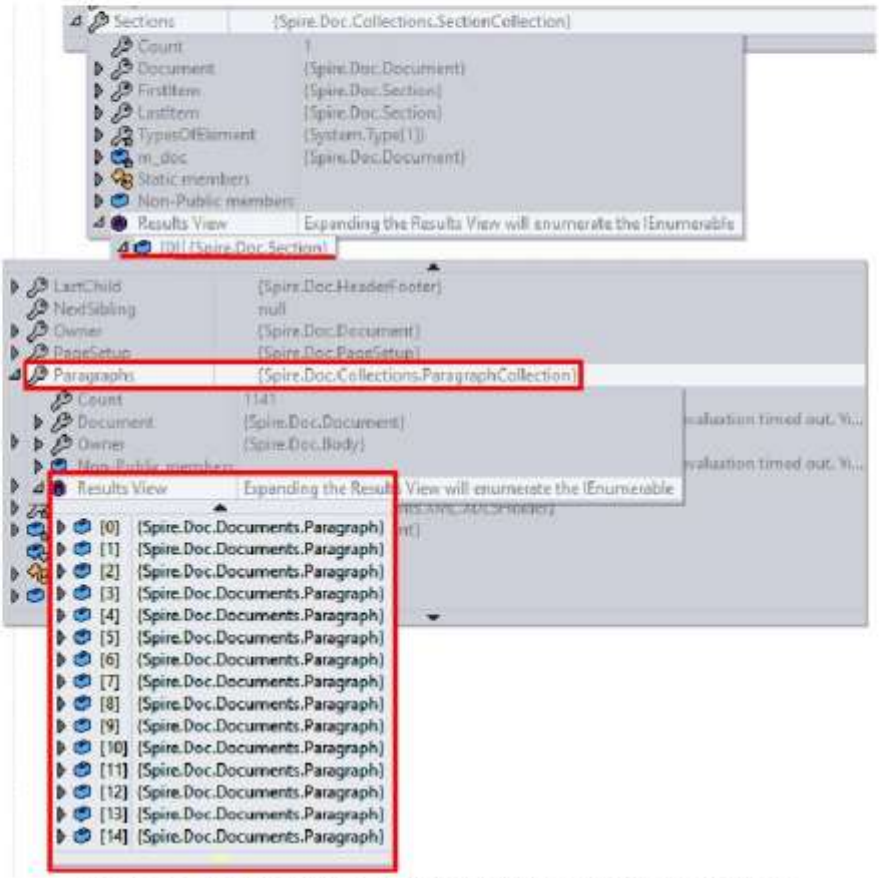


Рис. 6. Позиція визначеного об'єкта Paragraph в об'єктній моделі документа

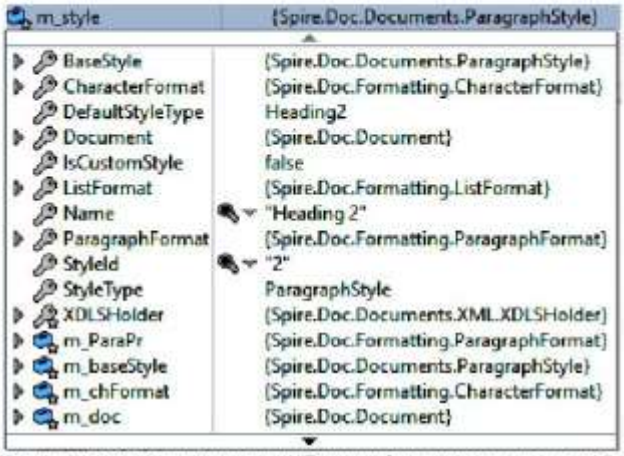


Рис. 7. Одержання відомостей про стилі Paragraph за допомогою методу GetStyle()

TextRange є найнижчим рівнем структури документу, що визначає фрагмент тексту однакового стилю. Тому для того, щоб при розв'язку прикладних задач отримати окремо кожне слово з власними властивостями стилів, потрібна розробка додаткового алгоритму для розбиття цих фрагментів *TextRange* на слова. *TextRange* знаходиться в *Paragraph -> Items*. Через роботу з відповідними властивостями, можна одержати текст (рис. 8) та стилі (рис. 9), що відносяться до визначеного *TextRange*.

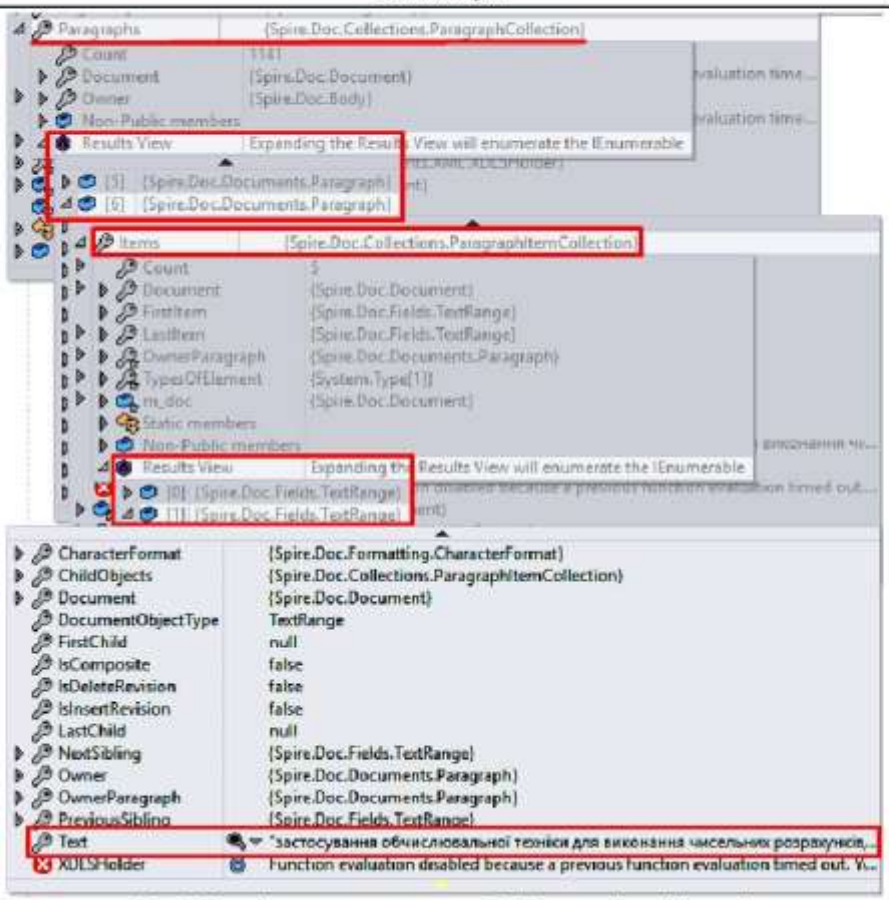


Рис. 8. Позиція визначеного тексту в TextRange у об'єктній моделі

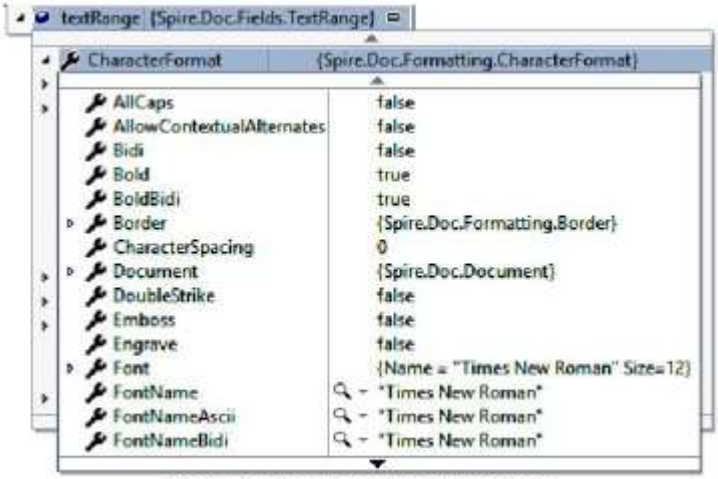


Рис. 9. Деякі стилі визначеного TextRange

Paragraph може містити лише один з кількох визначених форматів: *Level1, Level2, Level3, Level4, Level5, Level6, Level7, Level8, Level9, Body*. Відповідно до цього значення, можна визначити заголовки та їх рівні. Кожен похідний елемент має посилання на батьківський, і таким чином існує можливість визначити прив'язку певного елементу до певного модулю, заголовку чи підзаголовку при визначенні структури змістовних блоків у електронному документі навчального матеріалу.

З метою визначення можливості застосування розширення Spire.Doc.dll в задачах роботи з електронними документами, було проведено його практичну апробацію в рамках розробки програмної системи для вирішення прикладної задачі побудови структури електронного документу та пошуку ключових слів у його контенті. Зокрема, на рисунку 10 показано приклад обробки файлу «*Організація баз даних і знань.docx*» з автоматичним визначенням структури навчального матеріалу за допомогою аналізу системи заголовків та переліків ключових слів для контенту кожного елементу структури.

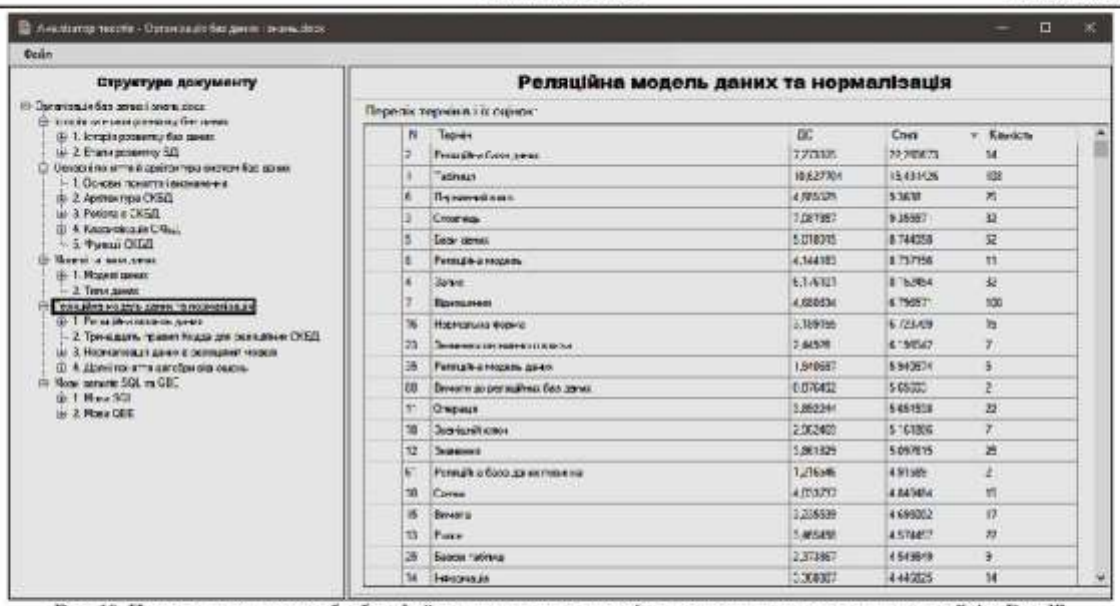


Рис. 10. Приклад результату обробки файлу навчального матеріалу з використанням розширення Spire.Doc.dll

Маючи стилі форматування *Paragraph* та *TextRange*, можна визначити заголовки, рівні заголовків, текст для аналізу чи ігнорування, прив'язку певного фрагменту тексту до заголовку.

При цьому, визначення властивостей *Paragraph* надало можливість для формування структури документу, а обробка властивостей елементів *TextRange* визначила необхідний для програмної обробки контент, що дозволило визначити множини ключових слів за допомогою методу дисперсійного оцінювання.

За результатами використання розширення Spire.Doc.dll в розробленому програмному комплексі встановлено, що дана бібліотека надає достатній інструментарій для роботи з цифровими документами навчальних матеріалів у рамках розглянутих актуальних задач, й може бути ефективно використана в реалізації інформаційних технологій для автоматизації роботи з навчальними матеріалами.

Висновки

Отже, у статті було досліджено проблему автоматизації роботи з цифровими документами навчальних матеріалів у задачах визначення структури змістовних блоків у цифровому документі навчального матеріалу та пошуку ключових термінів у контенті навчального матеріалу. Встановлено, що для реалізації програмної обробки цифрових документів є доцільним використання спеціалізованих програмних комплексів, що надають необхідний інструментарій для програмної роботи з контентом відповідних файлів.

З метою визначення найбільш ефективного програмного розширення було проведено аналіз сучасних існуючих спеціалізованих програмних розширень для автоматизації обробки цифрових документів: Microsoft.Office.Interap.Word.dll, DocumentFormat.OpenXml.dll та Spire.Doc.dll. В процесі аналізу встановлено, що бібліотека Spire.Doc.dll є оптимальним варіантом для використання при автоматизації обробки цифрових документів. Встановлено, що перенесення функцій автоматичного співставлення стилів текстових блоків їх властивостям з рівня функціоналу програмного коду застосунка на рівень функціоналу бібліотеки дозволяє спростити як роботу застосунка з цифровим документом, так і процес програмування.

Розглянуто практичні особливості використання розширення Spire.Doc.dll при роботі з цифровими документами .DOCX. З метою визначення можливості застосування обраного інструменту в задачах роботи з цифровими документами було проведено його практичну апробацію в складі програмної системи для вирішення прикладної задачі побудови структури цифрового документу та пошуку ключових слів у його контенті. В результаті встановлено можливість та достатню ефективність використання розширення Spire.Doc.dll для роботи з цифровими документами навчальних матеріалів.

Подальші дослідження спрямовані на впровадження розширення Spire.Doc.dll в розробку автоматизованих систем роботи з цифровими документами навчальних матеріалів для вирішення прикладних задач.

Література

1. Мазурець О. В. Онтологічний підхід до побудови семантичної моделі навчальних матеріалів / О. В. Мазурець // Вісник Хмельницького національного університету. Серія: Технічні науки. – 2017. – № 6. – С. 223–229.

2. Бармак О. В. Інформаційна технологія автоматизованого визначення термінів у навчальних матеріалах / О. В. Бармак, О. В. Мазурець // Міжнародний науково-технічний журнал «Вимірювальна та обчислювальна техніка в технологічних процесах». – Хмельницький, 2015. – № 2. – С. 94–102.

3. Коренчук О. В. Система інтелектуального аналізу відповідності лекційних матеріалів навчальних

дисциплін стандартам освіти / О. В. Коренчук, О. В. Мазурець // Збірник наукових праць за матеріалами восьмої міжнародної науково-технічної конференції «Актуальні проблеми комп'ютерних технологій 2014». – Хмельницький, 2014. – С. 191–201.

4. Поліщук А. О. Інформаційна технологія автоматизації аналізу відповідності тестових завдань лекційним матеріалам навчальних дисциплін / А. О. Поліщук, О. В. Мазурець // Збірник наукових праць за матеріалами дев'ятої міжнародної науково-технічної конференції «Актуальні проблеми комп'ютерних технологій 2015». – Хмельницький, 2015. – С. 207–218.

5. Бармак О. В. Інформаційна технологія автоматизованого формування тестових завдань / О. В. Бармак, О. В. Мазурець, В. І. Кліменко // Вісник Хмельницького національного університету. Серія: Технічні науки. – 2017. – № 5. – С. 93–103.

6. Бармак О. В. Застосування інформаційної технології пучкого тестування рівня знань у середовищі Moodle / О. В. Бармак, О. В. Мазурець, А. О. Матвійчук // Вісник Хмельницького національного університету. Серія: Технічні науки. – 2017. – № 3. – С. 103–115.

7. Бармак О. В. Інформаційна технологія автоматизованого анотування та реферування цифрових текстів / О. В. Бармак, О. В. Мазурець, А. В. Живілік // Вісник Хмельницького національного університету. Серія: Технічні науки. – 2017. – № 4. – С. 147–158.

8. Considerations for server-side Automation of Office [Електронний ресурс]. – Режим доступу : https://support.microsoft.com/en-us/help/257757/considerations-for-server-side-automation-of-office/

9. International Organization for Standardization - Open XML file formatsb [Електронний ресурс]. – Режим доступу : https://www.iso.org/search/x/query/Open%2520XML

10. Spire.Doc for .NET [Електронний ресурс]. – Режим доступу : https://www.nuget.org/packages/Spire.Doc/

11. International Organization for Standardization - Open XML file formatsb [Електронний ресурс]. – Режим доступу : https://www.iso.org/search/x/query/Open%2520XML

12. Office Open XML [Електронний ресурс]. – Режим доступу : https://uk.wikipedia.org/wiki/Office_Open_XML

13. How to automate Microsoft Excel from Microsoft Visual C#.NET [Електронний ресурс]. – Режим доступу : https://support.microsoft.com/en-us/help/302084/how-to-automate-microsoft-excel-from-microsoft-visual-c--net

14. DocX for creates or modifies Microsoft Word files [Електронний ресурс]. – Режим доступу : https://github.com/xceedsoftware/DocX

15. Free Spire.Doc for .NET [Електронний ресурс]. – Режим доступу : https://www.e-iceblue.com/Introduce/free-doc-component.html

References

1. Mazurets O. V. Ontologichiy pidhid do pobudovi semantichnoi modely navchalnih materialiv / O. V. Mazurets // Naukovyy zhurnal „Herald of Khmelnytskyi National University" seriya: Tekhnichni nauky. Khmelnytsky, 2017, № 6. – S. 223-229.

2. Barmak O. V., Mazurets O. V. Informatsiyna tekhnolohiya avtomatyzovanoho vyznachennya terminiv u navchalnykh materialakh / O. V. Barmak, O. V. Mazurets // Mizhnarodnyy naukovo-tekhnichnyy zhurnal «Vymiryuvalna ta obchyslyuvalna tekhnika v tekhnolohichnykh protsesakh» – Khmelnytsky, 2015. – №2. – C.94–102.

3. Korenchuk O. V., Mazurets O. V. Systema intelektualnoho analizu vidpovidnosti lektsiynykh materialiv navchalnykh dystsyplin standartam osvity / O. V. Korenchuk, O. V. Mazurets // Zbirnyk naukovykh prats za materialamy vosmoyi mizhnarodnoyi naukovo-tekhnichnoyi konferentsiyi «Aktualni problemy kompyuternykh tekhnolohiy 2014». Khmelnytsky – 2014. – S.191-201.

4. Polishchuk A. O., Mazurets O. V. Informatsiyna tekhnolohiya avtomatyzatsiyi analizu vidpovidnosti testovykh zavdan lektsiynym materialam navchalnykh dystsyplin / A. O. Polishchuk, O. V. Mazurets // Zbirnyk naukovykh prats za materialamy devyatoyi mizhnarodnoyi naukovo-tekhnichnoyi konferentsiyi «Aktualni problemy kompyuternykh tekhnolohiy 2015». Khmelnytsky – 2015. – S.207-218.

5. Barmak O. V., Mazurets O. V., Klimenko V. I. Informatsiyna tekhnolohiya avtomatyzovanoho formuvannya testovykh zavdan / O. V. Barmak, O. V. Mazurets, V. I. Klimenko // Naukovyy zhurnal „Herald of Khmelnytskyi National University" seriya: Tekhnichni nauky. Khmelnytsky, 2017, № 5. – S.93-103.

6. Barmak O. V., Mazurets O. V., Matviychuk A. O. Zastosuvannya informatsiynoyi tekhnolohiyi hnuchkoho testuvannya rivnya znan u seredovyshchi Moodle / O. V. Barmak, O. V. Mazurets, A. O. Matviychuk // Naukovyy zhurnal „Herald of Khmelnytskyi National University" seriya: Tekhnichni nauky. Khmelnytsky, 2017, №3. – S.103-115.

7. Barmak O. V., Mazurets O. V., Zhyvilik A. V. Informatsiyna tekhnolohiya avtomatyzovanoho anotuvannya ta referuvannya tsyfrovykh tekstiv / O. V. Barmak, O. V. Mazurets, A. V. Zhyvilik // Naukovyy zhurnal „Herald of Khmelnytskyi National University" seriya: Tekhnichni nauky. Khmelnytsky, 2017, № 4. – S. 147-158.

8. Considerations for server-side Automation of Office URL: https://support.microsoft.com/en-us/help/257757/considerations-for-server-side-automation-of-office/

9. International Organization for Standardization - Open XML file formatsb. URL: https://www.iso.org/search/x/query/Open%2520XML

10. Spire.Doc for .NET URL: https://www.nuget.org/packages/Spire.Doc/

11. International Organization for Standardization - Open XML file formatsb. URL: https://www.iso.org/search/x/query/Open%2520XML

12. Office Open XML. URL: https://uk.wikipedia.org/wiki/Office_Open_XML

13. How to automate Microsoft Excel from Microsoft Visual C#.NET. URL: https://support.microsoft.com/en-us/help/302084/how-to-automate-microsoft-excel-from-microsoft-visual-c--net

14. DocX for creates or modifies Microsoft Word files. URL: https://github.com/xceedsoftware/DocX

15. Free Spire.Doc for .NET. URL: https://www.e-iceblue.com/Introduce/free-doc-component.html

Рецензія/Peer review : 19.01.2018 р. Надрукована/Printed :28.01.2018 р.
Рецензент: стаття рецензована редакційною колегією

ББК 74.480.278
С.88

«Інтелектуальний потенціал – 2018» - збірник наукових праць молодих науковців і студентів з нагоди 30-річчя підготовки ІТ- фахівців в ХНУ/Колектив авторів – Хмельницький: ПВНЗ УЕП, 2018. – Ч1: Комп'ютерні науки та інформаційні технології проектування. – 132 с.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Хмельницький національний університет

Військовий інститут Київського національного університету ім.Тараса Шевченка

ПВНЗ "Університет економіки і підприємництва"

Тернопільський інститут агропромислового виробництва

Інтелектуальний потенціал - 2018

збірник наукових праць молодих науковців і студентів

Присвячується 30-річчю підготовки ІТ- фахівців в Хмельницькому національному університеті

сформовано за матеріалами
Всеукраїнської науково-практичної конференції
молодих науковців і студентів «Інтелектуальний потенціал – 2018»

14-16 листопада 2018р.

Частина 1
Комп'ютерні науки та інформаційні технології проектування

Хмельницький
2018

Метод формального опису елементів моделей автоматизованого формування тестових завдань

Мазурець О.В., Ковальчук О.В., Слободзян В.О., Білоус Г.А.
Хмельницький національний університет

Одним із основних способів контролю знань в навчальних інформаційних системах є комп'ютерне тестування, яке крім контрольної функції застосовується для навчання, тренінгу, розвитку здібностей. Інформаційні технології дають можливість суттєво зменшити трудові затрати на створення тестових завдань з можливістю їх постійного оновлення, що формує актуальний напрямок наукових досліджень.

Інформаційна технологія автоматизованого формування тестових завдань на основі контенту навчальних матеріалів [1] дозволяє на основі вхідних даних у вигляді контенту файлу формату .docx навчальних матеріалів або його визначеної частини автоматизовано формувати вихідні дані у вигляді файлу з тестами для імпорту у віртуальне навчальне середовище. Розв'язок задачі автоматизації формування тестових завдань містить ряд послідовних етапів перетворення інформації, до яких відносяться визначення семантичних термінів у контенті навчальних матеріалів [2], формування структури навчальних матеріалів [3], безпосередньо автоматизоване формування тестових завдань [1] й перенесення результатів у область взаємодії з кінцевим користувачем – підсистему тестування середовища Moodle [4].

Характерною рисою автоматизованого формування тестових завдань за даною інформаційною технологією є використання набору моделей тестових завдань для перетворення фрагментів контенту у тестові завдання.

Модель тестового завдання являє собою структуру, що складається з наступних функціональних елементів:

– маска для фрагменту тексту з поняттям, що призначена для ідентифікації фрагментів контенту із заданим терміном, до яких воно може бути застосовано;

– параметри моделі тестового завдання, що визначають особливості й ефективність його застосування: тип запитання, що може бути створено за цією моделлю; базова оцінка очікуваної якості застосування моделі у діапазоні (0;1), що одержується методом експертної оцінки результатів; набір правил корекції базової оцінки цього правила конвертації, що в залежності від вказаних особливостей фрагменту можуть знизити базову оцінку, при активації виступаючи множниками у діапазоні (0;1);

– маска для формування тестового завдання, що містить алгоритм перетворення даного фрагменту у елементи тестового завдання.

Тобто модель у даному випадку представляє собою структуру, в яку входять: набір з'єднувачів (слів або символів які вирізняють певний фрагмент тексту як релевантний відносно терміну); правило конвертації фрагментів з тексту у тестове завдання, та дані про словоформу

ЗМІСТ

Теги для формування тестів використовуються в масках формування й наведені у таблиці 2.

Таблиця 2 – Теги для формування тестів

Тег	Опис
{Header}	Візуальний визначник для тексту питання
{Answer}	Візуальний визначник для варіанту відповіді
{FALSE}	Вказівка на неправильний варіант відповіді
{TRUE}	Вказівка на правильний варіант відповіді

Нижче зображений приклад моделі формування тестового завдання одиничного вибору, де в якості тексту питання (Header) вказана теза, а у варіантах відповідей (Answer 1 – N) назви термінів:

```
Header -> [Caps][ThesisGroup]["-, - це"]
Answer 1 -> [TermGroup]            {TRUE}
Answer 2 -> [RandomTermGroup]      {FALSE}
Answer 3 -> [RandomTermGroup]      {FALSE}
Answer N -> [RandomTermGroup]      {FALSE}
```

Наведені теги можна віднести до елементів псевдокоду, або змінних, які під час роботи алгоритму мають різні значення, а іноді можуть не мають жодних (null). У випадках, коли тег може повернути порожнє значення, він вказується з додаванням знака дорівнює і нуля, наступним чином: [Tag = 0].

Наведений підхід дозволяє відкрите програмування алгоритму роботи різноманітних моделей генерування тестових завдань, що визначає множину формалізованих таким чином моделей як базу знань відповідної інформаційної системи для автоматизованого формування тестових завдань за навчальними матеріалами.

Для прикладу наведено процес формування тестового завдання типу «із введенням тексту» з використанням однієї з моделей формування тестових завдань для відповідного типу завдань.

Моделі формування завдань із введенням тексту полягають у формулюванні одного твердження з відсутнім ключовим словом та формуванні можливих варіантів відповідей, які не пропонуються користувачеві, а використовуються лише для перевірки введеного тексту. Відповідно, можлива модель, мета якої – забезпечити максимальну кількість правильних відповідей для коректної перевірки тексту, введеного користувачем.

При формуванні маски ідентифікації, за основу твердження береться певна пара [термін – теза], причому важливо, щоб термін був у початковій формі (називному відмінку однини), щоб виключити можливість ігнорування правильних відповідей через несхожість у закінченнях або словоформах. Далі відбувається формування набору правильних відповідей. Необхідним і (відмінок/рід). На вхід моделі завжди дається певна зв'язка [термін – слово маркер – теза], за якою здійснюється формування тестового завдання та підбір варіантів відповідей.

При використанні даного підходу спершу встановлюються вимоги до набору тестових завдань, визначаються актуальні моделі тестових завдань й на основі найбільш прийнятних із них формується набір тестових завдань. В результаті за кожною з обраних моделей тестових завдань формується одне тестове запитання у форматі .gift, що може бути конвертоване у середовище Moodle. Кінцевий результат зберігається в одному файлі тесту.

Таким чином, при формуванні тестових завдань різних типів використовуються моделі, наділені параметрами та критеріями, характерними лише для конкретного типу питання та будови терміну. В межах розглянутої інформаційної технології, з метою формування моделей тестових завдань є необхідною розробка набору тегів для формального опису елементів правил конвертації.

Метою роботи є розробка множини тегів як методу формального опису елементів моделей автоматизованого формування тестових завдань. Це дозволить проводити як зручне й ефективне створення нових чи редагування існуючих моделей формування тестових завдань, так і компактне зберігання розроблених моделей для подальшого використання.

Відповідно до розглянутих складових моделей формування тестових завдань, виділяються теги для ідентифікації елементів контенту та теги для формування тестів. Теги для ідентифікації елементів контенту використовуються в масках ідентифікації й наведені в таблиці 1.

Таблиця 1 – Теги для ідентифікації елементів контенту

Тег	Опис
[TermGroup]	Фрагмент тексту, який являє собою термін, що складається з набору слів
[ThesisGroup]	Фрагмент тексту (теза), який дає визначення терміну
[RandomTermGroup]	Випадково обраний інший термін
[RandomThesisGroup]	Випадково обране визначення іншого терміну
[Connector]	Слово або символ із тексту, що поєднує термін з тезою (– , – це, є, називається, тощо)
[BeginSentence]	Фрагмент тексту від початку речення до TermGroup або ThesisGroup (може бути null)
[Inflexion]	Тег повертає нормальну форму наступного елементу
[Caps]	Переведення першої букви до верхнього регістру
[ReCaps]	Переведення першої букви до нижнього регістру

Нижче наведений приклад для ідентифікації двох моделей з істинним твердженням, базової (Basic) та реверсивної (Reversed):

```
[Caps][TermGroup][connector][ThesisGroup]. {TRUE}
[Caps][ThesisGroup][connector][TermGroup]. {TRUE}
```

Відповідний сформований фрагмент GIFT-файлу, який сформований за даною моделлю формування тестового завдання типу «із введенням тексту» із відкритою відповіддю, наступний:

```
// question
Множина однотипних даних, що утворюють одну з граней гіперкуба це -
{
=%100% Вимірювання#
=%100% Dimension#
}
```

Отже, розроблені теги для моделей формування тестових завдань дозволяють формально описати процес формування тестових завдань із рахуванням всіх особливостей та параметрів, й забезпечити автоматизацію імпорту доступних тестових завдань у середовищі Moodle. Тег формального опису моделей є елементом псевдокоду, який призначений для формального опису структури моделі, її вхідних та вихідних параметрів.

За умови достатньої відповідності семантичним та структурним вимогам і коректного співвідношення між обсягом контенту навчального матеріалу та параметрів генерації набору тестових завдань, одержується репрезентативний тест, що може бути використаний як безпосередньо для тестування, так і як сировина для подальшої роботи розробника тестів.

Література

1. Бармак О. В., Мазурець О. В., Кліменко В. І. Інформаційна технологія автоматизованого формування тестових завдань / О. В. Бармак, О. В. Мазурець, В. І. Кліменко // Науковий журнал «Вісник Хмельницького національного університету» серія: Технічні науки. Хмельницький, 2017, №5. – С.93-103.
2. Мазурець О. В. Інформаційна технологія автоматизованого визначення семантичних термінів в елементах навчальних матеріалів / О. В. Мазурець // Науковий журнал «Вісник Хмельницького національного університету» серія: Технічні науки. Хмельницький, 2018, №3. – С.223-230
3. Мазурець О. В. Онтологічний підхід до побудови семантичної моделі навчальних матеріалів / О. В. Мазурець // Науковий журнал «Вісник Хмельницького національного університету» серія: Технічні науки. Хмельницький, 2017, №6. – С.223-229.
4. Moodle – Open-source learning platform. – [Електронний ресурс]. – Режим доступу: https://moodle.org/

достатнім є хоча б один варіант відповіді, якщо він повністю покриває можливі форми вживання даного терміну. Множина варіантів відповідей формуються на підставі параметрів моделі.

При формуванні правильних відповідей необхідно враховувати: абревіатуру терміну; скорочення; слова іншомовного походження (якщо є у тексті); відмінок/рід терміну.

Оскільки варіанти відповідей завжди сховані, функції цієї моделі не включають у себе підбір хибних термінів, а навпаки – модель має гарантувати, що всі можливі форми вживання цього поняття будуть розпізнані як правильна відповідь.

Отже, для сформованого на рисунку 1 тестового питання типу «Коротка відповідь» було використано термін «Вимірювання» та відповідну модель формування тестових завдань із відкритою відповіддю. Термін взятий з фрагменту тексту: «Вимірювання – це множина однотипних даних, що утворюють одну з граней гіперкуба» з теми «Моделі даних» навчального матеріалу «Організація баз даних та знань».

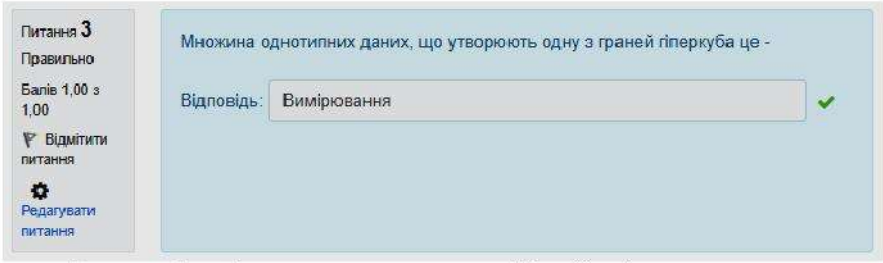


Рисунок 1 – Приклад використання у Moodle сформованого тестового завдання типу «Коротка відповідь»

Необхідно відмітити, що правильними відповідями є обидві з наведених, а для повної відповіді на питання достатньо вказати одну з них.

В даному випадку актуальний фрагмент тексту був знайдений наступною маскою ідентифікації:

```
[BeginSentence][ReCaps][TermGroup][Connector][ThesisGroup].
```

А генерація тестового завдання за актуалізованою моделлю відбулася шляхом використання відповідної маски формування тестового завдання із таким кодом:

```
Header -> [Caps][_____][connector][ThesisGroup]:
Answer 1 -> [Abbreviation] {TRUE}
Answer 2 -> [TermGroup1] {TRUE}
Answer N -> [TermGroupN] {TRUE}
```

Сучасні технології в механіці: Збірник наукових праць. / Укл.: Скиба М.Є., Олександренко В.П. Хмельницький: ФОП Мельник А.А., 2018. - 220 с.

Modern technologies in mechanical engineering: Collection of scientific works. / Com. Skyba M.E., Oleskandrenko V.P. Khmelnickiyi: sp. z o. o. Melnyk A.A. , 2018. - 220 c.

Редактор: Олександренко В.П. д.т.н., проф.
Відповідальні за випуск: Слащук В.О., Слащук О.О.

Editor: Oleksandrenko V.P. D.Sc., Prof.
Responsibility for the issue: Slashchuk V.O., Slashchuk O.O.

ISBN: 978-617-7600-17-5

Друк: "PolyLux" 29017, м. Хмельницький, вул. Зарічанська 22/3.
Тел.: 067 307-09-76. E-mail: polylux.ua@gmail.com

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

“СУЧАСНІ ТЕХНОЛОГІЇ В МЕХАНІЦІ”

ЗБІРНИК НАУКОВИХ ПРАЦЬ

19-21 КВІТНЯ 2018Р.

м. ХМЕЛЬНИЦЬКИЙ

Таблиця 1

Фазовий склад дифузійного шару сталі 40Х після АТР

Показники	Без пропану*	З додаванням пропану**
ε -$Fe_{2,3}N$, %	32	30
γ' -Fe_4N, %	56	53
α -Fe, %	12	17
* 833 К, 75 % N_2 + 25 % Ar – 4 год, 265 Па; ** 833 К, 75 % N_2 + 25 % Ar – 3 год і 90 % N_2 + 10 % пропану – 1 год, 265 Па		

Азотування в тліючому розряді за комбінованим режимом приводить до підвищення корозійної стійкості нітридної та дифузійної зон в 1,4...1,9 разів. Так, швидкість корозії в кислому середовищі для сталі 40Х поліпшеної становить 3,07 і 1,41 та 1,30 г/(м^2·год) при одностадійному та комбінованому азотуванні відповідно, що приблизно в 2,2...2,4 разів менше.

УДК 004.91

АНАЛІЗ СУЧАСНИХ СПЕЦІАЛІЗОВАНИХ ПРОГРАМНИХ РОЗШИРЕНЬ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ З ЦИФРОВИМИ ДОКУМЕНТАМИ НАВЧАЛЬНИХ МАТЕРІАЛІВ

Слободзян В.О., Мазурець О.В.
Хмельницький національний університет, Україна
E-mail: vitaliy.slobodzian@gmail.com

Вступ. Онтологія навчального матеріалу є методом виявлення сенсу навчального матеріалу та засобом вирішення ряду практичних задач. Модель онтології навчального матеріалу може складатися з ключових слів, ключових термінів, структури навчального матеріалу, атрибутів ключових слів та ключових термінів, що забезпечують їх прив'язку до елементів структури навчального матеріалу [1].

Основними з етапів побудови онтології навчального матеріалу є пошук ключових термінів у контенті навчального матеріалу та побудова його логічної структури. Вхідними даними є електронний документ навчального матеріалу. Для автоматизації виконання обох наведених етапів потрібна програмна обробка відповідних цифрових файлів (зазвичай формату .DOCX). Для ефективної реалізації програмної обробки цифрових документів є доцільним використання спеціалізованих програмних комплексів, що надають об'єктно-орієнтований інструментарій для програмної роботи з контентом відповідних файлів.

Метою роботи є аналіз сучасних існуючих спеціалізованих програмних розширень для автоматизації обробки цифрових документів і визначення найбільш ефективного програмного розширення.

Основна частина. На сучасному етапі для зберігання електронних документів навчальних матеріалів використовується файл з розширенням .DOCX (або створених на його основі .HTML, .PDF тощо). На відміну від файлів .DOC, які зберігають дані документа в одному бінарному файлі, файли .DOCX створюються за допомогою відкритого формату XML, в якому зберігаються документи як колекції окремих файлів і папок в стиснутому пакеті. .DOCX-файли містять .XML-файли і три папки, docProps, Word, і _rels (Рис. 1), які містять властивості документа, його зміст і відношення між файлами, тему та включені файли. .DOCX-файли розроблені, щоб зробити вміст документів доступним і відкритим – так, текстовий документ чи зображення зберігаються як окремі прості файли в такій колекції у складі одного файлу .DOCX.

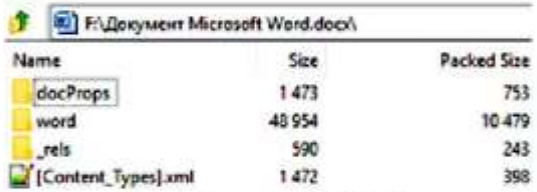


Рис. 1 – Структура .DOCX-файлу

Office OpenXML є форматом електронних офісних документів, таблиць, презентацій та діаграм, розроблений компанією Microsoft. Даний формат є міжнародним стандартом, який затверджений в Міжнародній організації зі стандартизації (ISO). Він включає в себе такі первинні мови розмітки:

- WordprocessingML – для обробки текстових документів;
- SpeadsheetXL – електронних таблиць;
- PresentationML – для презентацій;
- DrawingML – для векторної графіки, діаграм.

Хоча Word реалізує свою функціональність через об'єктну модель, збереження даних відбувається у наборі XML-файлів, що ускладнює можливості для автоматизації прямого програмного парсингу. Оскільки файл стилів та текст знаходяться в різних файлах у складі .XML, потрібно зчитувати одразу ряд файлів та на базі їх взаємозв'язків програмно формувати об'єктну модель документа, що є

не найкращим рішенням. Тому для реалізації програмної обробки цифрових документів є доцільним використання спеціалізованих програмних комплексів, що надають об'єктно-орієнтований інструментарій для програмної роботи з контентом відповідних файлів. Найбільш відомими з таких спеціалізованих програмних комплексів є розширення Microsoft.Office.Interap.Word.dll [2], DocumentFormat.OpenXml.dll [3] та Spire.Doc.dll [4].

Бібліотека ***Microsoft.Office.Interap.Word.dll*** [2] дозволяє керованому коду взаємодіяти з MS Office та об'єктною моделлю на базі COM.

За допомогою процесів автоматизації Office Automation програми, написані мовами, такими як Visual C# .NET, отримують можливість програмно керувати іншими програмами. Автоматизація Microsoft Word дозволяє виконувати відповідні дії, такі як створення нового документу, додавання даних у документ або створення таблиць. З програмами Word та іншими програмами Microsoft Office практично всі дії, які користувач може виконувати вручну за допомогою користувацького інтерфейсу, також можуть бути виконані програмним шляхом за допомогою автоматизації Office Automation.

Word реалізує цю програмну функціональність через об'єктну модель. Об'єктна модель є набором класів і методів, які служать аналогами логічних компонентів Word. Наприклад, існує об'єкт Application, об'єкт Document та об'єкт Paragraph, кожен з яких містить функціональність цих частин Word.

Щоб використовувати функції програми MS Office в проекті, можна використовувати первинний взаємозв'язок PIA (Primary Interop Assembly). PIA дозволяє керованому коду взаємодіяти з MS Office та об'єктною моделлю на базі COM. При створенні нового проекту Office, Visual Studio додає посилання на PIA, необхідний для побудови проекту. При цьому, у деяких сценаріях доводиться додавати посилання на додаткові PIA (наприклад, якщо необхідно використовувати функцію MS Office Word в проекті для MS Office Excel).

Бібліотека класів .NET ***DocumentFormat.OpenXml.dll*** [3] дає можливість розробникам програмного забезпечення працювати з пакетом Microsoft Office. Компанія надає дану бібліотеку безкоштовно в повному доступі, й її можна завантажити з офіційного сайту Microsoft або засобами Visual Studio.

Для роботи з файлами формату .DOCX, який має зображену на рисунку 1 структуру, використовується мова розмітки WordprocessingML. На рисунку 2 зображено приклад структури цієї мови розмітки.


```
<?xml version="1.0" encoding="utf-8"?>
<w:document
xmlns:w="http://schemas.openxmlformats.org/wordproc
essingml/2006/main">
  <w:body>
    <w:p>
      <w:r>
        <w:t> Текст </w:t>
      </w:r>
    </w:p>
  </w:body>
</w:document
```

Рис. 2 – Приклад WordprocessingML-розмітки .DOCX-документу

Теги, наведені на рисунку 2, мають такі властивості:

- *document* – є основою частиною документа
- *body* – контейнер для збору структур рівня блоку, які складають основну історію;
- *p* – абзац (paragraph);
- *r* – контейнер який містить в собі текст з однаковими властивостями (run);
- *t* – власне текстовий елемент (text).

Всі інші теги, які можуть використовуватися, є допоміжними до вищенаведених і додають їм стильових властивостей.

Spire.Doc.dll [4] для .NET є повністю незалежною бібліотекою класів .NET Word, спеціально створеною для розробників, й дозволяє швидко генерувати, відкривати, редагувати та зберігати документи Word різних версій. Конвертація функціональних можливостей дозволяє розробникам здійснювати перетворення Word в інші актуальні формати документів (.PDF, .EPub, .HTML, .RTF, .Image, .XML тощо). Можна створювати та обробляти вже існуючі документи Word динамічно.

Бібліотека Spire.Doc.dll забезпечує роботу з майже всіма елементами документа Word, а саме: сторінки, розділи, заголовки, колонтитули, абзаци, переліки, таблиці, текст, виноски, поля, гіперпосилання, закладки, коментарі, зображення, стилі, фонові параметри, функції друку, налаштування документа та захисту. Крім того, підтримуються графічні об'єкти, включаючи форми, текстові поля, зображення, OLE-об'єкти та елементи керування. Бібліотека Spire.Doc.dll підтримує функцію пошуку та заміни, вирівнювання,

перерву сторінки, поле заповнення, об'єднання документів, копіювання документів, друк тощо.

Аналіз. Хоч бібліотека Microsoft.Office.Interap.Word.dll надає доступ до всіх функцій MS Office, так як працює безпосередньо з PIA, для її роботи потрібна ліцензія для MS Office на кожному клієнтському комп'ютері. Крім того, при використанні Automation, MS Office завантажується у фоновому режимі, внаслідок чого займає деякий обсяг оперативної пам'яті та завантажує велику кількість файлів і DLL. Додатки MS Office були розроблені як додатки для користувацького інтерфейсу, і через це Microsoft.Office.Interop.Word.dll працює повільно. Microsoft не рекомендує використовувати Office Automation (або будь-який Office Interop) на сервері.

На відміну від Microsoft.Office.Interap.Word.dll, DocumentFormat.OpenXml.dll та Spire.Doc.dll не вимагають наявності MS Office на машині користувача та його запуску в фоновому режимі.

При роботі з docx-файлом за допомогою DocumentFormat.OpenXml.dll потрібно дотримуватися тієї ж ієрархії, що і сама структура розмітки тобто: *document-paragraph-run-text*. Також відповідно до розглянутих вище особливостей взаємозв'язків всередині документу, при присвоєнні параграфу деякого стилю, у властивостях параграфа надається лише ідентифікатор (ID) на даний стиль, натомість сам стиль описується окремо у файлі style.xml. Внаслідок необхідності регулярного співставлення ID стиля з контейнером style.xml для одержання характеристик стилів абзаців, зростає складність програмного використання бібліотеки.

Spire.Doc.dll не вимагає встановлення в систему кожного користувача MS Office, тобто є можливість повністю незалежної від нього роботи. Повна версія Spire.Doc.dll є платною, а безкоштовна версія FreeSpire.Doc має ряд обмежень (наприклад обробка не більше 500 абзаців і 25 таблиць). Перевагою Spire.Doc.dll визначено відсутність необхідності співставлення ID стиля з контейнером style.xml, оскільки дана функція реалізована на рівні бібліотеки.

Таким чином, в результаті аналізу сучасних існуючих спеціалізованих програмних розширень для автоматизації обробки цифрових документів, було визначено Spire.Doc.dll оптимальним варіантом для використання в автоматизації обробки цифрових документів. Основними перевагами Spire.Doc.dll встановлено відсутність необхідності наявності MS Office на машині користувача та запуску в фоновому режимі, а також реалізацію функцій автоматичного співставлення стилів текстових блоків їх властивостям на рівні розширення.

Перенесення функцій автоматичного співставлення стилів текстових блоків їх властивостям з рівня функціоналу програмного коду застосунка на рівень функціоналу бібліотеки дозволяє спростити як роботу застосунка з цифровим документом, так і процес програмування.

Практичне застосування. При зчитуванні файлу Spire.Doc.dll перетворює документ в об'єктну модель. Ця модель має структуру, яка починається документом і закінчується об'єктом *TextRange*.

Відповідно до об'єктної моделі документу, MS Office використовує розділи (*Section*), щоб вказати частини документа, що мають різну орієнтацію сторінки, стовпці або заголовки та нижні колонтитули. Розбиття на Section дозволяє користувачеві вказати місце, де розпочинається і закінчується інше форматування. Тому Section використовуються коли потрібне використання заголовків, колонтитулів, схем, нумерації сторінок, розмірів аркушу, поля чи різні рівні захисту. Об'єкти Section містяться в об'єкті Document, в колекції Selections. А розділи, в свою чергу, містять в собі наступний елемент структури – абзаци (*Paragraph*).

Paragraph в MS Word визначає будь-який текст, який закінчується жорстким поверненням, яке формується, наприклад, при натисканні клавіші Enter. Формат абзацу дозволяє контролювати зовнішній вигляд фрагменту, якщо є окремі абзаци. Наприклад, можна змінити відстань між рядками або вирівнювання тексту по лівому краю на вирівнювання по центру. Можна додати індивідуальні стилі, відступи для абзаців, або визначити заголовки і списки. В об'єктній моделі *Paragraph* знаходиться в *Document -> Sections -> Section -> Paragraphs* (Рис. 3). Одержати відомості про стилі *Paragraph* можна за допомогою методу *GetStyle()*.

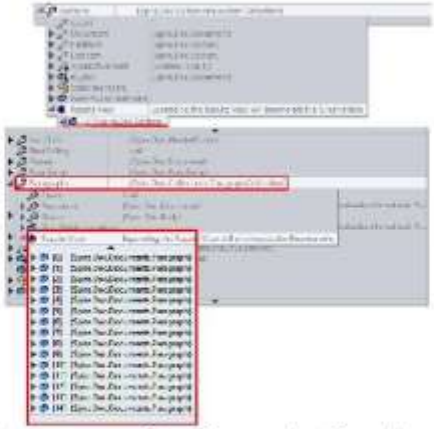


Рис. 3 – Позиція визначеного об'єкта Paragraph в об'єктній моделі документа

TextRange є найнижчим рівнем структури документу, що визначає фрагмент тексту однакового стилю. Тому для того, щоб при розв'язку прикладних задач отримати окремо кожне слово з власними властивостями стилів, потрібна розробка додаткового алгоритму для розбиття цих фрагментів *TextRange* на слова. *TextRange* знаходиться в *Paragraph -> Items*. Через роботу з відповідними властивостями, можна одержати текст та стилі (рис. 4), що відносяться до визначеного *TextRange*.

Рис. 4 – Деякі стилі визначеного TextRange

Маючи стилі форматування Paragraph та TextRange, можна визначити заголовки, рівні заголовків, текст для аналізу чи ігнорування, прив'язку певного фрагменту тексту до заголовку.

Paragraph може містити лише один з кількох визначених форматів: Level1, Level2, Level3, Level4, Level5, Level6, Level7, Level8, Level9, Body. Відповідно до цього значення, можна визначити заголовки та їх рівні. Кожен похідний елемент має посилання на

батьківський, і таким чином існує можливість визначити прив'язку певного елементу до певного модулю, заголовку чи підзаголовку при визначенні структури змістовних блоків у електронному документі навчального матеріалу.

З метою визначення можливості застосування розширення Spire.Doc.dll в задачах роботи з електронними документами, було проведено його практичну апробацію [5] в рамках розробки програмної системи для вирішення прикладної задачі побудови структури електронного документу та пошуку ключових слів у його контенті. При цьому, визначення властивостей Paragraph надало можливість для формування структури документу, а обробка властивостей елементів TextRange визначила необхідний для програмної обробки контент, що дозволило визначити множини ключових слів за допомогою методу дисперсійного оцінювання.

Висновки. За результатами використання розширення Spire.Doc.dll в розробленому програмному комплексі встановлено, що дана бібліотека надає достатній інструментарій для роботи з цифровими документами навчальних матеріалів у рамках розглянутих актуальних задач, й може бути ефективно використана в реалізації інформаційних технологій для автоматизації роботи з навчальними матеріалами.

Подальші дослідження спрямовані на впровадження розширення Spire.Doc.dll в розробку автоматизованих систем роботи з цифровими документами навчальних матеріалів для вирішення прикладних задач.

Література

1. Мазурець О. В. Онтологічний підхід до побудови семантичної моделі навчальних матеріалів / О. В. Мазурець // Науковий журнал «Вісник Хмельницького національного університету» серія: Технічні науки. Хмельницький, 2017, №6. – С.223-229.

2. Considerations for server-side Automation of Office [Електронний ресурс]. – Режим доступу: https://support.microsoft.com/en-us/help/257757/considerations-for-server-side-automation-of-office/

3. International Organization for Standardization - Open XML file formatsb [Електронний ресурс]. – Режим доступу: https://www.iso.org/search/x/query/Open%2520XML

4. Spire.Doc for .NET [Електронний ресурс]. – Режим доступу: https://www.nuget.org/packages/Spire.Doc/

5. Мазурець О. В., Ковальчук О. В., Слободян В. О. Використання спеціалізованих програмних розширень для автоматизації роботи з цифровими документами навчальних матеріалів / О. В. Мазурець, О. В. Ковальчук, В. О. Слободян //Науковий журнал «Вісник Хмельницького національного університету» серія: Технічні науки. Хмельницький, 2018, №1. – С.61-69.

Актуальні проблеми комп'ютерних наук. Збірник наукових праць за матеріалами XI всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2019» – Хмельницький: ХНУ, 2019, Т.1. – 248с.

У збірнику наукових праць представлені перспективні практичні розробки аспірантів, магістрів та здобувачів в області сучасних інформаційних технологій. Розглянуто актуальні проблеми комп'ютерних наук, прикладної математики й інформатики, приведено ряд робіт по впровадженню інформаційних технологій у виробництво та управління. Висвітлено перспективні розробки з сучасних систем пошуку й обробки інформації, САПР та математичного, комп'ютерного і нейромережевого моделювання.

Міністерство освіти і науки України
Хмельницький національний університет

АПКН 2019

АКТУАЛЬНІ ПРОБЛЕМИ КОМП'ЮТЕРНИХ НАУК

ЗБІРНИК НАУКОВИХ ПРАЦЬ
за матеріалами XI всеукраїнської науково-практичної конференції
«Актуальні проблеми комп'ютерних наук АПКН-2019»

14-15 листопада 2019

Том 1

Роботи студентів та молодих вчених Факультету програмування та комп'ютерних і телекомунікаційних систем ХНУ

Хмельницький 2019

ЗМІСТ

АКТУАЛЬНІ ПРОБЛЕМИ КОМП'ЮТЕРНИХ НАУК - 2019

XI всеукраїнська науково-практична конференція

Метою конференції є висвітлення актуальних проблем комп'ютерних наук, інформатики та інформаційних технологій.

Основні напрямки роботи конференції:

1. Прикладні інформаційні технології.
2. Сучасні системи пошуку, захисту і обробки інформації.
3. Математичне, комп'ютерне і нейромережеве моделювання.
4. САПР, математичні моделі і методи рішення інженерних задач.
5. Проблеми впровадження інформаційних технологій у виробництво та управління.

Робочі мови конференції: українська, англійська.

КЕРІВНИЦТВО ОРГКОМІТЕТУ:

СИНЮК О. М.
голова оргкомітету, проректор Хмельницького національного університету з наукової роботи, доктор технічних наук, професор

СОРОКАТИЙ Р. В.
заступник голови оргкомітету, доктор технічних наук, завідувач кафедри Комп'ютерних наук та інформаційних технологій ХНУ, професор

БАРМАК О. В.
заступник голови оргкомітету, доктор технічних наук, професор

СЕКРЕТАРІАТ КОНФЕРЕНЦІЇ:

Мазурець О. В.
секретар конференції, старший викладач каф. КНІТ ХНУ

КОНТАКТНА ІНФОРМАЦІЯ:
e-mail для листування: *apkt.khnu@gmail.com*

На сучасному етапі для зберігання електронних документів навчальних матеріалів використовується файл з розширенням .DOCX (або створених на його основі .HTML, .PDF тощо). На відміну від файлів .DOC, які зберігають дані документа в одному бінарному файлі, файли .DOCX створюються за допомогою відкритого формату XML, в якому зберігаються документи як колекції окремих файлів і папок в стиснутому пакеті. .DOCX-файли містять .XML-файли і три папки, docProps, Word, і _rels (Рис. 1), які містять властивості документа, його зміст і відношення між файлами, тему та включені файли. .DOCX-файли розроблені, щоб зробити вміст документів доступним і відкритим – так, текстовий документ чи зображення зберігаються як окремі прості файли в такій колекції у складі одного файлу .DOCX.

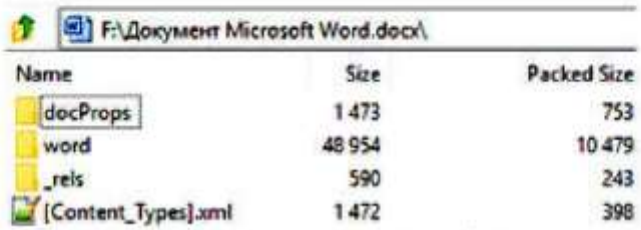


Рисунок 1 – Структура .DOCX-файлу

Office OpenXML є форматом електронних офісних документів, таблиць, презентацій та діаграм, розроблений компанією Microsoft. Даний формат є міжнародним стандартом, який затверджений в Міжнародній організації зі стандартизації (ISO) [5]. Він включає в себе такі первинні мови розмітки:

- WordprocessingML – для обробки текстових документів;
- SpeadsheetXL – електронних таблиць;
- PresentationML – для презентацій;
- DrawingML – для векторної графіки, діаграм.

Хоча Word реалізує свою функціональність через об'єктну модель, збереження даних відбувається у наборі XML-файлів, що ускладнює можливості для автоматизації прямого програмного парсингу [6]. Оскільки файл стилів та текст знаходяться в різних файлах у складі .XML, потрібно зчитувати одразу ряд файлів та на базі їх взаємозв'язків програмно формувати об'єктну модель документа, що є не найкращим рішенням.

Бібліотека ***Microsoft.Office.Interap.Word.dll*** [2] дозволяє керованому коду взаємодіяти з MS Office та об'єктною моделлю на базі COM.

УДК 004.4

Ковальчук О.В., Білоус Г.А., Слободзян В.О.

Хмельницький національний університет

ВИКОРИСТАННЯ ПРОГРАМНОГО РОЗШИРЕННЯ SPIRE.DOC ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ З ЦИФРОВИМИ ДОКУМЕНТАМИ

З метою визначення найбільш ефективного програмного розширення проведено аналіз сучасних існуючих спеціалізованих програмних розширень для автоматизації обробки електронних документів: Microsoft.Office.Interap.Word.dll, DocumentFormat.OpenXml.dll та Spire.Doc.dll. В результаті аналізу розглянутих бібліотек, було визначено Spire.Doc.dll оптимальним варіантом для використання в автоматизації обробки електронних документів. Встановлено, що перенесення функцій автоматичного співставлення стилів текстових блоків їх властивостям з рівня функціоналу програмного коду застосунка на рівень функціоналу бібліотеки дозволяє спростити як роботу застосунка з електронним документом, так і процес програмування.

Microsoft.Office.Interop.Word.dll, DocumentFormat.OpenXml.dll and Spire.Doc.dll were useing for chooce the most effective program components for automation work with electronic documents. In result audit this libraries there was chosen Spire.Doc.dll for use to automation of processing of electronic documents. It is established that in result transfer of functions for automatic matching styles of text block to their properties from the level of the functional code of the application to the level of the functional library simplify work of the application with electronic document and programming process.

Для ефективної реалізації програмної обробки цифрових документів є доцільним використання спеціалізованих програмних комплексів, що надають об'єктно-орієнтований інструментарій для програмної роботи з контентом відповідних файлів [1]. Для реалізації програмної обробки цифрових документів є доцільним використання розглянутих в даній статті спеціалізованих програмних комплексів, що надають необхідний інструментарій для програмної роботи з контентом відповідних файлів, зокрема: Microsoft.Office.Interap.Word.dll [2], DocumentFormat.OpenXml.dll [3], Spire.Doc.dll [4].

Метою роботи є аналіз сучасних існуючих спеціалізованих програмних розширень для автоматизації обробки цифрових документів і визначення найбільш ефективного програмного розширення.

```
<?xml version="1.0" encoding="utf-8"?>
<w:document
xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2
006/main">
  <w:body>
    <w:p>
      <w:r>
        <w:t> Текст </w:t>
      </w:r>
    </w:p>
  </w:body>
</w:document
```

Рисунок 2 – Приклад WordprocessingML-розмітки .DOCX-документу

Spire.Doc.dll [4] для .NET є повністю незалежною бібліотекою класів .NET Word, спеціально створеною для розробників, й дозволяє швидко генерувати, відкривати, редагувати та зберігати документи Word різних версій. Конвертація функціональних можливостей дозволяє розробникам здійснювати перетворення Word в інші актуальні формати документів (.PDF, .EPub, .HTML, .RTF, .Image, .XML тощо). Можна створювати та обробляти вже існуючі документи Word динамічно.

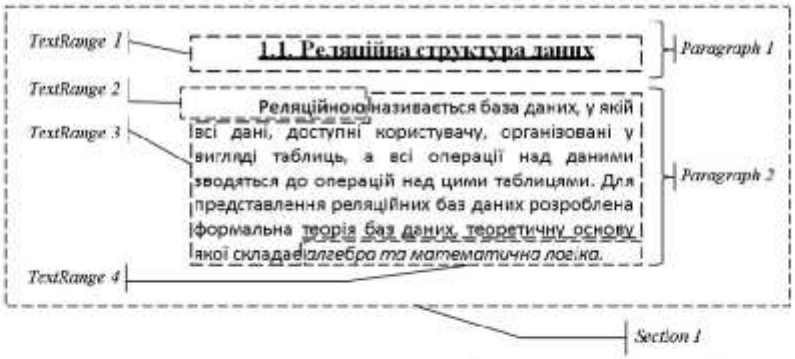


Рисунок 3 – Приклад використання об'єктної моделі MS Office

Бібліотека Spire.Doc.dll забезпечує роботу з майже всіма елементами документа Word, а саме: сторінки, розділи, заголовки, колонтитули, абзаци, переліки, таблиці, текст, виноски, поля, гіперпосилання, закладки, коментарі, зображення, стилі, фонові параметри, функції друку, налаштування документа та захисту. Крім того, підтримуються графічні об'єкти, включаючи форми, текстові поля,

За допомогою процесів автоматизації Office Automation програми, написані мовами, такими як Visual C# .NET, отримують можливість програмно керувати іншими програмами. Автоматизація Microsoft Word (далі Word) дозволяє виконувати відповідні дії, такі як створення нового документу, додавання даних у документ або створення таблиць. З програмами Word та іншими програмами Microsoft Office практично всі дії, які користувач може виконувати вручну за допомогою користувацького інтерфейсу, також можуть бути виконані програмним шляхом за допомогою автоматизації Office Automation.

Word реалізує цю програмну функціональність через об'єктну модель [7]. Об'єктна модель є набором класів і методів, які служать аналогами логічних компонентів Word. Наприклад, існує об'єкт Application, об'єкт Document та об'єкт Paragraph, кожен з яких містить функціональність цих частин Word.

Щоб використовувати функції програми MS Office в проекті, можна використовувати первинний взаємозв'язок PIA (Primary Interop Assembly). PIA дозволяє керованому коду взаємодіяти з MS Office та об'єктною моделлю на базі COM. При створенні нового проекту Office, Visual Studio додає посилання на PIA, необхідний для побудови проекту. При цьому, у деяких сценаріях доводиться додавати посилання на додаткові PIA (наприклад, якщо необхідно використовувати функцію MS Office Word в проекті для MS Office Excel).

DocumentFormat.OpenXml.dll

Бібліотека класів .NET DocumentFormat.OpenXml.dll (версії 2.0 / 2.5 / 2.8.1) [3] дає можливість розробникам програмного забезпечення працювати з пакетом Microsoft Office. Компанія надає дану бібліотеку безкоштовно в повному доступі, й її можна завантажити з офіційного сайту Microsoft або засобами Visual Studio.

Для роботи з файлами формату .DOCX, використовується мова розмітки WordprocessingML [8]. На рисунку 2 зображено приклад структури цієї мови розмітки.

Теги, наведені на рисунку 2, мають такі властивості:

- *document* – є основою частиною документа;
- *body* – контейнер для збору структур рівня блоку, які складають основну історію;
- *p* – абзац (paragraph);
- *r* – контейнер який містить в собі текст з однаковими властивостями (run);
- *t* – власне текстовий елемент (text).

Всі інші теги, які можуть використовуватися, є допоміжними до вищенаведених і додають їм стильових властивостей.

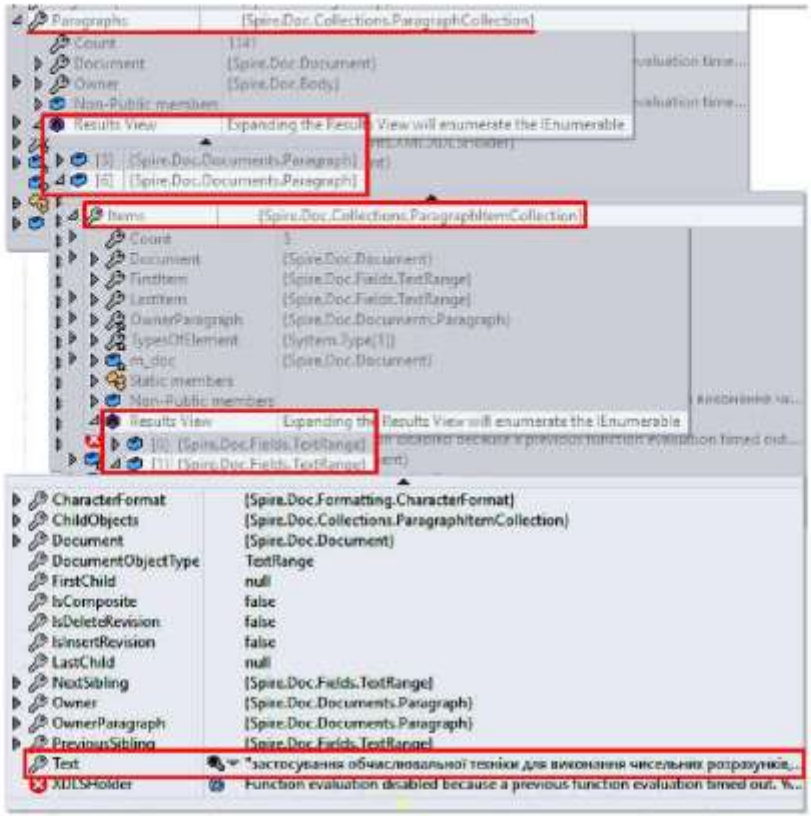


Рисунок 5 – Позиція визначеного тексту в TextRange у об'єктній моделі

На відміну від Microsoft.Office.Interap.Word.dll, DocumentFormat.OpenXml.dll та Spire.Doc.dll не вимагають наявності MS Office на машині користувача та його запуску в фоновому режимі.

При роботі з docx-файлом за допомогою DocumentFormat.OpenXml.dll потрібно дотримуватися тієї ж ієрархії, що і сама структура розмітки тобто: *document-paragraph-run-text*. Також відповідно до розглянутих вище особливостей взаємозв'язків всередині документу, при присвоєнні параграфу деякого стилю, у властивостях параграфа надається лише ідентифікатор (ID) на даний стиль, натомість сам стиль описується окремо у файлі style.xml. Внаслідок необхідності регулярного співставлення ID стиля з контейнером style.xml для одержання характеристик стилів абзаців, зростає складність програмного використання бібліотеки. Перевагою Spire.Doc.dll визначено відсутність

зображення, OLE-об'єкти та елементи керування. Бібліотека Spire.Doc.dll підтримує функцію пошуку та заміни, вирівнювання, перерву сторінки, поле заповнення, об'єднання документів, копіювання документів, друк тощо.

При зчитуванні файлу Spire.Doc.dll перетворює документ в об'єктну модель. Ця модель має структуру, яка починається документом і закінчується об'єктом *TextRange*. Приклад використання такої об'єктної моделі документу MS Office показаний на Рисунку 3.

Одержати відомості про стилі *Paragraph* можна за допомогою методу *GetStyle()* (Рис. 4). Через роботу з відповідними властивостями, можна одержати текст та стилі (Рис. 5), що відносяться до визначеного *TextRange*.

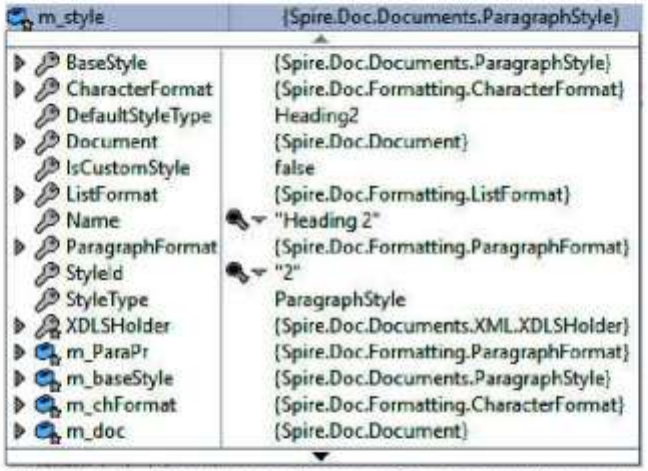


Рисунок 4 – Одержання відомостей про стилі Paragraph

Хоч бібліотека Microsoft.Office.Interap.Word.dll надає доступ до всіх функцій MS Office, так як працює безпосередньо з PIA, для її роботи потрібна ліцензія для MS Office на кожному клієнтському комп'ютері. Крім того, при використанні Automation, MS Office завантажується у фоновому режимі, внаслідок чого займає деякий обсяг оперативної пам'яті та завантажує велику кількість файлів і DLL. Додатки MS Office були розроблені як додатки для користувацького інтерфейсу, і через це Microsoft.Office.Interop.Word.dll працює повільно. Microsoft не рекомендує використовувати Office Automation (або будь-який Office Interop) на сервері.

необхідності співставлення ID стиля з контейнером style.xml, оскільки дана функція реалізована на рівні бібліотеки.

Таким чином, в результаті аналізу сучасних існуючих спеціалізованих програмних розширень для автоматизації обробки цифрових документів, було визначено Spire.Doc.dll оптимальним варіантом для використання в автоматизації обробки цифрових документів. Основними перевагами Spire.Doc.dll встановлено відсутність необхідності наявності MS Office на машині користувача та запуску в фоновому режимі, а також реалізацію функцій автоматичного співставлення стилів текстових блоків їх властивостям на рівні розширення.

Перенесення функцій автоматичного співставлення стилів текстових блоків їх властивостям з рівня функціоналу програмного коду застосунка на рівень функціоналу бібліотеки дозволяє спростити як роботу застосунка з цифровим документом, так і процес програмування.

Перелік посилань

1. Мазурець О. В. Використання спеціалізованих програмних розширень для автоматизації роботи з цифровими документами навчальних матеріалів / О. В. Мазурець, О. В. Ковальчук, В. О. Слободзян // Науковий журнал «Вісник Хмельницького національного університету» серія: Технічні науки. Хмельницький, 2018, №1. – С.61-69.
2. Considerations for server-side Automation of Office [Електронний ресурс]. – Режим доступу: https://support.microsoft.com/en-us/help/257757/considerations-for-server-side-automation-of-office/
3. International Organization for Standardization - Open XML file formatsb [Електронний ресурс]. – Режим доступу: https://www.iso.org/search/x/query/Open%2520XML
4. Spire.Doc for .NET [Електронний ресурс]. – Режим доступу: https://www.nuget.org/packages/Spire.Doc/
5. International Organization for Standardization - Open XML file formatsb [Електронний ресурс]. – Режим доступу: https://www.iso.org/search/x/query/Open%2520XML
6. Office Open XML [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Office_Open_XML
7. How to automate Microsoft Excel from Microsoft Visual C#.NET [Електронний ресурс]. – Режим доступу: https://support.microsoft.com/en-us/help/302084/how-to-automate-microsoft-excel-from-microsoft-visual-c--net
8. DocX for creates or modifies Microsoft Word files [Електронний ресурс]. – Режим доступу: https://github.com/xceedsoftware/DocX
9. Free Spire.Doc for .NET [Електронний ресурс]. – Режим доступу: https://www.e-iceblue.com/Introduce/free-doc-component.html

УДК 004:37:001:62

Збірник наукових праць за матеріалами XII всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2020». Хмельницький – 2020. – 365с.

У збірнику наукових праць подані перспективні практичні розробки аспірантів, студентів та здобувачів в області сучасних інформаційних технологій. Розглянуто актуальні проблеми комп'ютерних наук, комп'ютерної інженерії, прикладної математики й інженерії програмного забезпечення, приведено ряд робіт по впровадженню інформаційних технологій у виробництво та управління. Висвітлено перспективні розробки сучасних систем пошуку, обробки й захисту інформації, медійних та комунікаційних системи.

УДК 004:37:001:62

Матеріали конференції відтворені з авторських оригіналів. Оргкомітет конференції висловлює подяку учасникам конференції та сподівається на подальшу співпрацю.

З питань проведення конференції та подальшого обміну інформацією звертатись на e-mail конференції: *apkt.khnu@gmail.com*

Міністерство освіти і науки України
Хмельницький національний університет

ЗБІРНИК НАУКОВИХ ПРАЦЬ
за матеріалами XII всеукраїнської науково-практичної конференції
«Актуальні проблеми комп'ютерних наук АПКН-2020»

9-10 листопада 2020

Хмельницький 2020

ЗМІСТ

АКТУАЛЬНІ ПРОБЛЕМИ КОМП'ЮТЕРНИХ НАУК - 2020

XII Всеукраїнська науково-практична конференція

Метою конференції є висвітлення актуальних проблем комп'ютерних наук, інформатики та інформаційних технологій.

СЕКЦІЇ КОНФЕРЕНЦІЇ:

1. Комп'ютерні науки та прикладні інформаційні технології.
2. Комп'ютерна інженерія та системи захисту інформації.
3. Математичне моделювання та інженерія програмного забезпечення
4. Телерадіокомунікації, медійні та комунікаційні системи.
5. Проблеми впровадження інформаційних технологій у виробництво та управління.

Робочі мови конференції: українська, англійська

ОРГКОМІТЕТ:

СИНЮК О. М. голова оргкомітету, проректор Хмельницького національного університету з наукової роботи, доктор технічних наук, професор

СОРОКАТИЙ Р. В. заступник голови оргкомітету, завідувач кафедри Комп'ютерних наук та інформаційних технологій ХНУ, доктор технічних наук, професор

БАРМАК О. В. заступник голови оргкомітету, доктор технічних наук, професор

САВЕНКО О. С. декан Факультету програмування та комп'ютерних і телекомунікаційних систем ХНУ, доктор технічних наук, професор

ВИСОЦЬКА О. В. доктор технічних наук, завідувач кафедри радіоелектронних та біомедичних комп'ютеризованих засобів і технологій Національного аерокосмічного університету ім. М. Є. Жуковського «Харківський авіаційний інститут», професор

ЛАВРОВ Є. А. доктор технічних наук, професор (Сумський державний університет)

ТІМОФЄЄВА Л. В. відповідальна за студентську науково-дослідну роботу ХНУ

МАЗУРЕЦЬ О. В. секретар конференції, старший викладач кафедри комп'ютерних наук та інформаційних технологій ХНУ

КОНТАКТНА ІНФОРМАЦІЯ:

e-mail для листування: *apkt.khnu@gmail.com*

УДК 004.92

Слободзян В. О., Мазурець О. В.

Хмельницький національний університет

АНАЛІЗ РЕЗУЛЬТАТІВ АВТОМАТИЗОВАНОГО ПОШУКУ КЛЮЧОВИХ ТЕРМІНІВ У НАВЧАЛЬНИХ МАТЕРІАЛАХ

В роботі наведено результати досліджень, призначених для визначення ефективності інформаційної технології автоматизованого формування семантичної структури інформаційного навчального матеріалу. Дослідження проводилось з метою визначити частку контенту інформаційного навчально матеріалу, що містить автоматизовано знайдені ключові терміни.

Дослідження ефективності пошуку ключових термінів у контенті елементів інформаційних навчальних матеріалів підтвердили можливість ефективно автоматизовано формувати множини ключових семантичних термінів до контенту елементів інформаційних навчальних матеріалів. При цьому встановлено, що за значення показника щільності 11% ключові терміни з автоматизовано одержаної множини містяться в середньому в 87,3% речень, тобто 90,1% текстового контенту. Це надає підставу для використання автоматизовано сформованих за розглянутою ІТ множин ключових термінів для автоматизованої побудови семантичної структури навчального матеріалу, автоматизованої перевірки відповідності навчальних матеріалів вимогам та автоматизованої побудови тестових завдань.

The paper presents the results of research dedicated to determining the effectiveness of information technology for the automated generation of the semantic structure of teaching-learning material. The research was conducted to determine the proportion of teaching-learning material content that contains automatically found key terms.

Research of the effectiveness of the search of key terms in the content of elements of informational learning materials has confirmed the possibility of effectively automated generation of sets of key semantic terms to the content of elements of informational educational materials. It was found that for the value of the density index of 11%, the key terms from the automatically generated set are contained in an average of 87.3% of sentences, and 90.1% of text content. This provides a basis for the use of automatically generated by the considered IT sets of key terms for automated generation of the semantic structure of teaching-learning material, automated verification of compliance of educational materials with the requirements, and automated generation of test tasks.

Інформаційний навчальний матеріал (ІНМ) являє собою складну систему, яка має свою структуру зі специфічними елементами й зв'язками між ними. Він містить всю інформацію, яка подана для засвоєння й сприяє засвоєнню. Контент ІНМ зазвичай включає тексти, малюнки, таблиці та інші форми графічного вираження інформації. За характером інформації ІНМ подається у вигляді

підручника, навчального посібника, конспекту лекцій тощо. Контент ІНМ містить слабо структуровану символьну (переважно текстову) інформацію, засвоєння якої забезпечує набуття відповідних знань та вмінь суб'єктом, що вивчає навчальний курс.

У попередніх роботах було розглянуто інформаційну модель семантичної структури навчального курсу [1], з використанням якої розглянута інформаційна технологія [2] дозволяє автоматизовано формувати семантичну структуру ІНМ. Така структура, подана у вигляді ряду множин, може бути використана для вирішення ряду задач – визначення відповідності ІНМ вимогам, допомога при розробці ІНМ, оцінка відповідності наборів тестових завдань до ІНМ, допомога при створенні тестів й інших.

Дане дослідження проводилось з метою визначити частку контенту інформаційних навчальних матеріалів (ІНМ), що містить автоматизовано знайдені ключові терміни. Для пошуку ключових термінів було використано тестове програмне забезпечення (рис. 1), створене відповідно до інформаційної технології автоматизованого формування семантичної структури ІНМ [2]. Наведена інформаційна технологія дозволяє за результатом парсингу та обробки документів з ІНМ визначати їх семантичну структуру, що містить елементи рубрикації та співвіднесені з елементами структури ранжовані множини ключових термінів. Створені множини ключових термінів у структурі ІНМ можуть бути використані для автоматизованої побудови семантичної структури навчального матеріалу, автоматизованої перевірки відповідності навчальних матеріалів вимогам, а також автоматизованого створення тестових завдань [1].

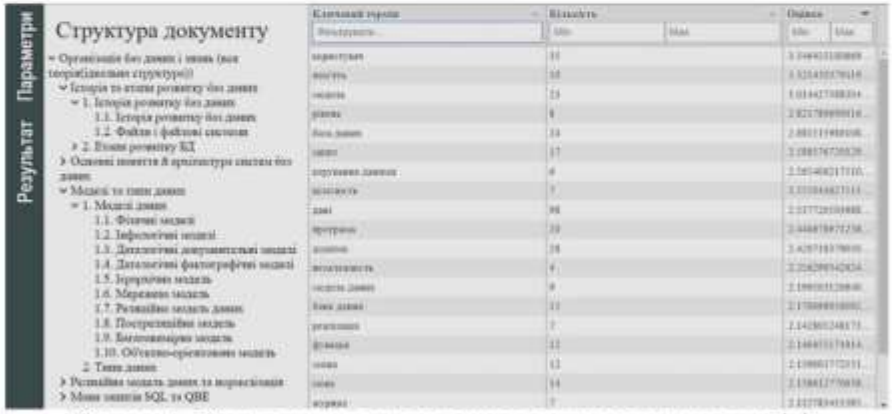


Рисунок 1 – Результати автоматизованого пошуку ключових термінів

За матеріал дослідження було використано розділи навчальних дисциплін тестової вибірки, яка складається з множини всіх фахових дисциплін освітнього

рівнів «бакалавр» та «магістр» спеціальності «122 – Комп'ютерні науки», що використовуються в навчальному процесі на кафедрі Комп'ютерних наук та інформаційних технологій Хмельницького національного університету, й розміщені в системі Модульного середовища для навчання ХНУ [3] і Навчального центру заочно-дистанційної освіти ХНУ [4]. Вибірка має наступні структурні параметри: кількість дисциплін – 40, кількість заголовків 1 та 2 рівня (іменованих одиниць структури, придатних для локальних пошуку ключових термінів та створення множин тестових завдань) – 203, загальна кількість тестових завдань у вибірці – 2421.

Структура ІНМ як електронних документів регламентується мовами розмітки документів (наприклад, WordprocessingML для XML чи Open Document Format) й реалізується через систему заголовків. По відношенню до онтології ІНМ навчальної дисципліни, заголовки співставленні її відповідним рівням. Використання текстових процесорів для форматування та структурування електронних документів значно полегшує роботу та дозволяє дотримуватися єдиного стилю в всіх документах та уникнути стильових суперечностей. Одержання доступу до вмісту і форматуванню цифрових документів з ІНМ за допомогою спеціалізованих програмних комплексів дозволяє автоматизовано визначати структуру ІНМ та відкриває можливості для пошуку ключових слів у них [5].

Для зчитування електронного ІНМ було використано бібліотеку Spire.Doc.dll, так як в процесі дослідження [6, 7] було встановлено, що дана бібліотека є оптимальним варіантом для використання при автоматизації обробки цифрових документів. Встановлено, що перенесення функцій автоматичного співставлення стилів текстових блоків їх властивостям з рівня функціоналу програмного коду застосунка на рівень функціоналу бібліотеки дозволяє спростити як роботу застосунка з цифровим документом, так і процес програмування.

За одиницю виміру повноти покриття контенту ІНМ ключовими термінами взято відношення K_H, K_P і K_S кількості семантичних блоків (відповідно підрубрик N_H, абзаців N_P і речень N_S), що містять знайдені ключові терміни, до загальної кількості таких блоків (відповідно N_{HH}, N_{PP} і N_{SS}) у контенті кожного ІНМ:

$$K_H = \frac{N_H}{N_{HH}}, \quad K_P = \frac{N_P}{N_{PP}}, \quad K_S = \frac{N_S}{N_{SS}}, \tag{1}$$

$$\overline{K_H} = \frac{\sum_{i=1}^{h} K_{H,i}}{h}, \quad \overline{K_P} = \frac{\sum_{i=1}^{h} K_{P,i}}{h}, \quad \overline{K_S} = \frac{\sum_{i=1}^{h} K_{S,i}}{h}, \tag{2}$$

де $\overline{K_H}$, $\overline{K_P}$ та $\overline{K_S}$ – середні значення для відповідно K_H, K_P та K_S для тестової вибірки; h – кількість оброблюваних документів ІНМ у вибірці.

З метою дослідження характеру семантичних блоків, що розглядаються, було використано похідні показники: L_H, L_P та L_S для відношення кількості слів у семантичних блоках (відповідно підрубрик, абзаців і речень), що містять знайдені ключові терміни, до загальної кількості слів у ІНМ F_W. Значення показників L_H, L_P та L_S й їх середні значення $\overline{L_H}$, $\overline{L_P}$ та $\overline{L_S}$ за тестовою вибіркою обраховуються наступним чином:

$$L_H = \frac{F_H}{F_W}, \quad L_P = \frac{F_P}{F_W}, \quad L_S = \frac{F_S}{F_W}, \tag{3}$$

$$\overline{L_H} = \frac{\sum_{i=1}^{h} L_{H,i}}{h}, \quad \overline{L_P} = \frac{\sum_{i=1}^{h} L_{P,i}}{h}, \quad \overline{L_S} = \frac{\sum_{i=1}^{h} L_{S,i}}{h}, \tag{4}$$

де F_H, F_P та F_S – кількість слів відповідно у підрубриках, абзацах та реченнях, в яких знайдені ключові терміни.

Наприклад, у результаті обробки розділу «Моделювання даних та нормалізація» дисципліни «Організація баз даних та знань» з значеннями максимальної кількості слів у терміні $n = 5$ та граничної щільності ключових термінів $Q = 11\%$, одержано множину ключових термінів, наведену у таблиці 1, до якої включено наступні унікальні терміни: база даних, таблиця, первинний ключ, запит, нормальна форма, нормалізація, відношення, кортеж, дані.

В даному випадку показники покриття контенту ключовими термінами за (1) склали: $K_H = 100\%$, $K_P = 73{,}3\%$, $K_S = 76{,}1\%$. При цьому, відповідні показники покриття контенту за кількістю слів відповідно до (3) одержано наступні: $L_H = 100\%$, $L_P = 79{,}8\%$, $L_S = 85{,}9\%$.

Таблиця 1 – Приклад результату пошуку ключових термінів

№ п/п	Ключовий термін T	Кількість у тексті k	Частота TF	Дисперсійна оцінка σ
1	бази даних	64	0,0169	3,3720
2	таблиця	73	0,0175	2,9294
3	первинного ключа	55	0,0162	3,3720
4	запитом	47	0,0110	2,9294
5	нормальна форма	49	0,0165	2,6444
6	нормалізації	41	0,0165	2,5320
7	відношення	34	0,0110	2,3817
8	кортеж	40	0,0137	2,2732
9	баз даних	67	0,0220	2,1460
10	нормальної форми	34	0,0302	1,9918
11	запит	52	0,0302	1,9917
12	даних	79	0,0302	1,9915

Загалом по тестовій вибірці №2 з $h = 203$ документів ІНМ за (2) одержано середні показники покриття контенту $\overline{K_H} = 100\%$, $\overline{K_P} = 81{,}2\%$ та $\overline{K_S} = 87{,}3\%$. Граничні значення цих показників склали: $K_{H,min} = 100\%$, $K_{H,max} = 100\%$, $K_{P,min} = 60{,}1\%$, $K_{P,max} = 100\%$, $K_{S,min} = 66{,}5\%$, $K_{S,max} = 100\%$. Аналогічні середні показники покриття контенту за кількістю слів відповідно до (4) одержано $\overline{L_H} = 100\%$, $\overline{L_P} = 88{,}1\%$ та $\overline{L_S} = 90{,}1\%$, при наступних граничних значеннях: $L_{H,min} = 100\%$, $L_{H,max} = 100\%$, $L_{P,min} = 67{,}6\%$, $L_{P,max} = 100\%$, $L_{S,min} = 69{,}7\%$, $L_{S,max} = 100\%$. Візуальне подання результатів аналізу покриття контенту ключовими термінами зображено на рисунку 2.

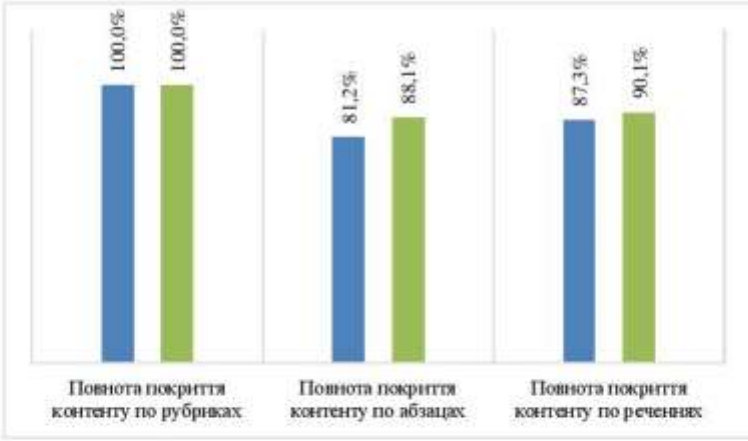


Рисунок 2 – Середні значення покриття контенту ключовими термінами в складі рубрик, абзаців та речень

Таким чином, за значення Q = 11% ключові терміни з автоматизовано одержаної множини містяться в усіх підрубриках ІНМ, зокрема в середньому в 81,2% абзаців (що містять 88,1% текстового контенту), або в 87,3% речень (що містять 90,1% текстового контенту). Збільшення значення параметру граничної щільності ключових термінів Q дозволить збільшити наведені значення, проте призведе до включення до результуючої множини мало важливих та похідних термінів, перевіряти знання яких за допомогою тестів не потрібно.

Оскільки одержано $\overline{K_S} > \overline{K_P}$, то можна зробити висновок, що ключові терміни не присутні переважно в абзацах, що складаються з невеликої кількості речень. Наприклад, «Далі наведено основні властивості цих класів.». Результат $\overline{L_S} > \overline{K_S}$ свідчить, що ключові терміни не присутні переважно в коротких реченнях. Ці факти пояснюються загальновідомими синтаксичними особливостями побудови текстів, коли деякі речення (а іноді й абзаци) мають не семантичну, а зв'язувальну важливість.

Суттєве зниження показників повноти покриття контенту ($K_{P,min}$ = 60,1% та $K_{S,min}$ = 66,5%, $L_{P,min}$ = 67,6% та $L_{S,min}$ = 69,7%) спричиняє поява додаткових суб'єктивних факторів, які характеризуються включенням спеціальних фрагментів контенту, зокрема програмних кодів та практичних прикладів. Наявність таких включень характерна для операційного навчального матеріалу (методичні вказівки, практикуми тощо), а для перевірки одержаних знань та вмінь потребує створення тестів за допомогою параметричного методу. Включення цих елементів до ІНМ хоча й не забороняється, проте не потребує тестової перевірки рівня засвоєння.

Отже, метою роботи було поставлено дослідження повноти покриття контенту ключовими термінами, які знайдені автоматизовано з допомогою інформаційної технології автоматизованого пошуку ключових термінів у навчальних матеріалах. Результати дослідження ефективності пошуку ключових термінів у контенті елементів ІНМ підтвердили можливість ефективно автоматизовано формувати множини ключових семантичних термінів до контенту елементів ІНМ. При цьому встановлено, що за значення показника щільності 11% ключові терміни з автоматизовано одержаної множини містяться в середньому в 87,3% речень, тобто 90,1% текстового контенту. Це надає підставу для використання автоматизовано сформованих за розглянутою інформаційною технологією множин ключових термінів у задачах автоматизованої побудови семантичної структури навчального матеріалу, автоматизованої перевірки відповідності навчальних матеріалів вимогам, автоматизованої побудови тестових завдань тощо.

Перелік посилань

1. Крак Ю. В. Інформаційна модель семантичної структури навчального курсу для генерації тестових завдань / Ю. В. Крак, О. В. Бармак, О. В. Мазурець // Матеріали XIX Міжнародної науково-практичної конференції «Моделювання та дослідження стійкості динамічних систем DSMSI-2019». Київ – 2019. – С.365-367.
2. Krak I. The practice investigation of the information technology efficiency for automated definition of terms in the semantic content of educational materials / I. Krak, O. Barmak, O. Mazurets // CEUR Workshop Proceedings. – 2016. – Vol. 1631. – P. 237-245.
3. Модульне середовище для навчання ХНУ [Електронний ресурс]. – Режим доступу: https://msn.khnu.km.ua/
4. Навчальний центр заочно-дистанційної освіти ХНУ [Електронний ресурс]. – Режим доступу: https://de.khnu.km.ua/p.aspx.
5. Мазурець О. В. Метод формального опису елементів моделей автоматизованого формування тестових завдань / О. В. Мазурець, О. В. Ковальчук, В. О. Слободзян, Г. А. Білоус // Збірник наукових праць за матеріалами Всеукраїнської науково-практичної конференції «Інтелектуальний потенціал – 2018». Хмельницький, 2018, Ч.1. – С.51-56.
6. Слободзян В. О. Аналіз сучасних спеціалізованих програмних розширень для автоматизації роботи з цифровими документами навчальних матеріалів / В. О. Слободзян, О. В. Мазурець // Сучасні технології в механіці: Збірник наукових праць. Хмельницький – 2018. – С.184-191.
7. Мазурець О. В. Використання спеціалізованих програмних розширень для автоматизації роботи з цифровими документами навчальних матеріалів / О. В. Мазурець, О. В. Ковальчук, В. О. Слободзян // Науковий журнал «Вісник Хмельницького національного університету» серія: Технічні науки. Хмельницький, 2018, №1. – С.61-69.

Додаток З

Презентаційний матеріал

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ
АВТОМАТИЗОВАНОГО ФОРМУВАННЯ СЕМАНТИЧНОЇ СТРУКТУРИ
ІНФОРМАЦІЙНОГО НАВЧАЛЬНОГО МАТЕРІАЛУ

ДИПЛОМНА РОБОТА МАГІСТРА

Виконав: студент II курсу групи КНм-19-1
Слободзян Віталій Олександрович

Керівник: старший викладач кафедри КНІТ
Мазурець Олександр Вікторович

АКТУАЛЬНІСТЬ ТЕМИ

Одержана семантична структура інформаційних навчальних матеріалів (ІНМ) має бути використана для розв'язання ряду задач в питаннях автоматизації роботи з навчальними матеріалами

- Допомога при розробці ІНМ
- Визначення відповідності ІНМ вимогам
- Оцінка відповідності наборів тестових завдань до ІНМ
- Допомога при створенні тестових завдань

МЕТА ДИПЛОМНОЇ РОБОТИ

Розробити інформаційну технологію автоматизованого формування семантичної структури інформаційного навчального матеріалу.

Для досягнення поставленої мети визначенні наступні задачі дослідження:

- *Проаналізувати* сучасні рішення з автоматизованого моделювання семантичної структури інформаційного навчального матеріалу
- *Удосконалити* інформаційну модель семантичної структури інформаційного навчального матеріалу
- *Удосконалити* метод формування рубрикації інформаційного навчального матеріалу
- *Удосконалити* метод пошуку ключових термінів у рубриках інформаційного навчального матеріалу
- *Розробити* інформаційну технологію автоматизованого формування семантичної структури інформаційного навчального матеріалу
- *Дослідити* практичну ефективності розробленої інформаційної технології шляхом створення відповідної експериментальної інформаційної системи та її тестування

2

НАУКОВІ ТА ІННОВАЦІЙНІ ПОЛОЖЕННЯ, ЩО ВИНОСЯТЬСЯ НА ЗАХИСТ

- Удосконалено інформаційну модель семантичної структури інформаційного навчального матеріалу, що містить елементи та атрибути семантичної структури інформаційного навчального матеріалу
- Удосконалено метод формування рубрикації інформаційного навчального матеріалу
- Удосконалено метод пошуку ключових термінів у рубриках інформаційного навчального матеріалу
- Вперше розроблено інформаційну технологію автоматизованого формування семантичної структури інформаційного навчального матеріалу

3

МОДЕЛЬ СЕМАНТИЧНОЇ СТРУКТУРИ ІНФОРМАЦІЙНОГО НАВЧАЛЬНОГО МАТЕРІАЛУ

$$\{M_{Heading} \cup M_{Par} \cup M_{Sen} \cup M_{Term} \cup M_{Word} \cup M_{Rel}\} \subset IEM \subset EC$$

- $M_{Heading}$ – множина заголовків;
- M_{Par} – множина абзаців;
- M_{Sen} – множина речень;
- M_{Term} – множина термінів;
- M_{Word} – множина слів;
- M_{Rel} – множина зв'язків.

Множину зв'язків M_{Rel} можна розділити на підмножини, відповідно до множин, елементи яких зв'язують між собою її елементи:

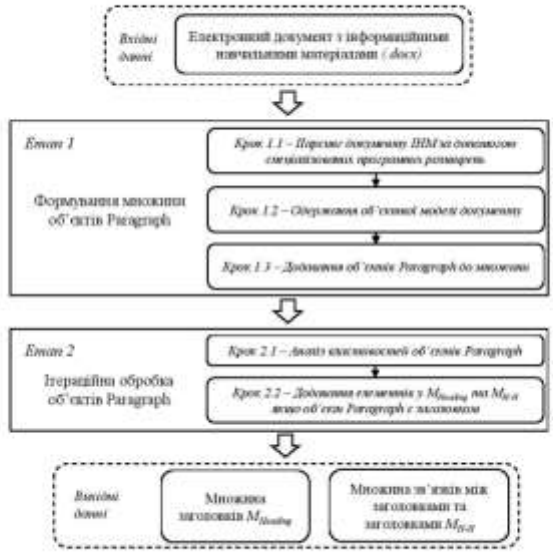
- множина зв'язків між заголовками та заголовками $M_{H\text{-}H}$;
- множина зв'язків між заголовками і абзацами $M_{H\text{-}P}$;
- множина зв'язків між абзацами і реченнями $M_{P\text{-}S}$;
- множина зв'язків між реченнями і термінами $M_{S\text{-}T}$;
- множина зв'язків між термінами та словами $M_{T\text{-}W}$.

$$IEM = \{ x | x \in M_{Heading} \vee x \in M_{Par} \vee x \in M_{Sen} \vee x \in M_{Term} \vee x \in M_{Word} \vee x \in M_{Rel} \}$$

4

МЕТОД ФОРМУВАННЯ РУБРИКАЦІЇ ІНФОРМАЦІЙНОГО НАВЧАЛЬНОГО МАТЕРІАЛУ

Схема методу формування рубрикації інформаційного навчального матеріалу



5

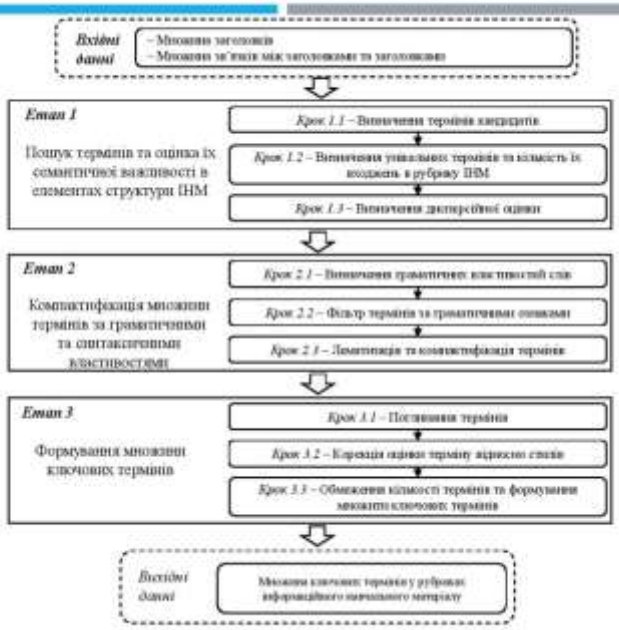
МЕТОД ФОРМУВАННЯ РУБРИКАЦІЇ ІНФОРМАЦІЙНОГО НАВЧАЛЬНОГО МАТЕРІАЛУ

Алгоритмічна схема обробки об'єктів Paragraph

6

МЕТОД ПОШУКУ КЛЮЧОВИХ ТЕРМІНІВ У РУБРИКАХ ІНФОРМАЦІЙНОГО НАВЧАЛЬНОГО МАТЕРІАЛУ

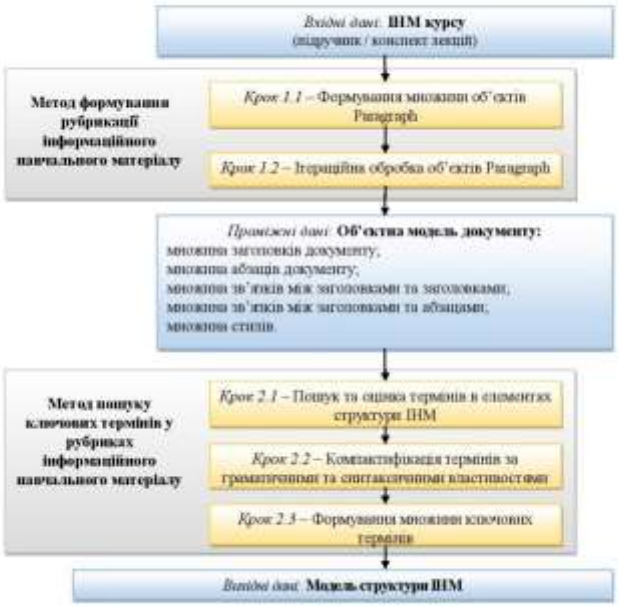
Схема етапів методу пошуку ключових термінів у рубриках інформаційного навчального матеріалу



7

СХЕМА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ АВТОМАТИЗОВАНОГО ФОРМУВАННЯ СЕМАНТИЧНОЇ СТРУКТУРИ ІНФОРМАЦІЙНОГО НАВЧАЛЬНОГО МАТЕРІАЛУ

Послідовність кроків інформаційної технології



8

ФУНКЦІОНАЛЬНЕ ПОДАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ЯК ІНФОРМАЦІЙНОЇ СИСТЕМИ

Функції користувача:

- *Обов'язкові активні функції користувача*, без яких робота системи є неможливою, до яких належить вибір файлу документу для обробки.
- *Необов'язкові активні функції користувача*, які впливають на кінцевий результат роботи системи, до яких належать:
  - встановлення етапу пошуку ключових слів;
  - встановлення коефіцієнтів впливу стильових ознак;
  - встановлення параметрів пошуку.

9

ДОСЛІДЖЕННЯ
АЛГОРИТМІВ ПОШУКУ КЛЮЧОВИХ ТЕРМІНІВ

- Частотна оцінка $TF_i = \frac{n(i)}{N}$

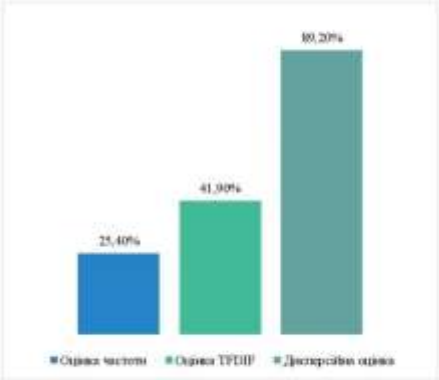
де $n(i)$ – кількість появ слова i у тексті, N – загальна кількість слів у тексті.

- Оцінка TF-IDF $TFIDF = TF * IDF, \ IDF_i = \log \frac{D}{d_i}$

де D – кількість альтернативних документів для порівняння або фрагментів, на які розбивається текст при аналізі; d_i – кількість документів або фрагментів, у яких дане слово присутнє.

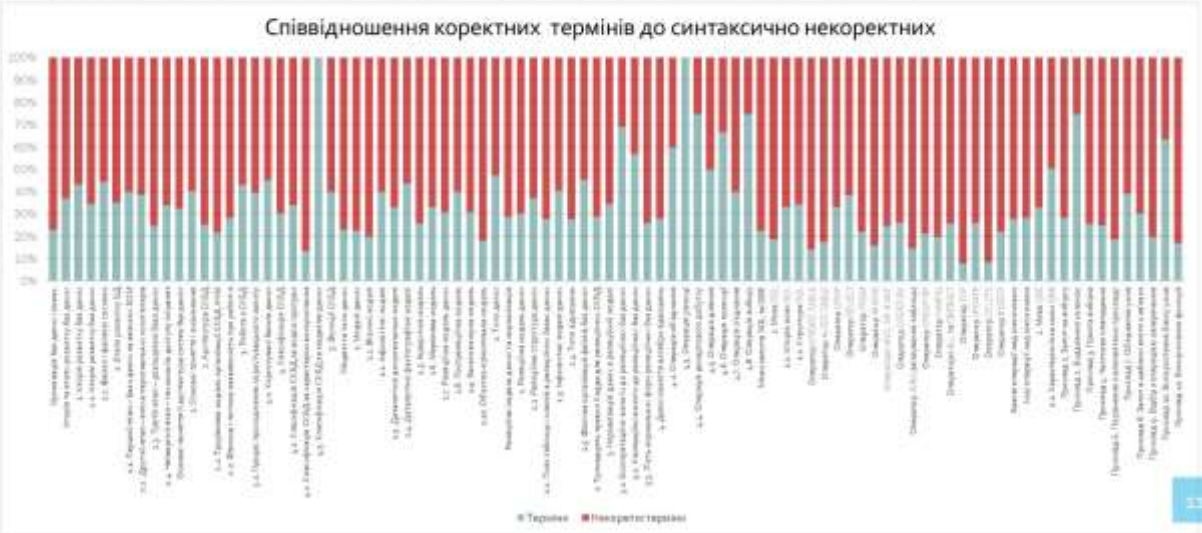
- Дисперсійна оцінка $DE = \frac{\sqrt{\langle \Delta A^2 \rangle - \langle \Delta A \rangle^2}}{\langle \Delta A \rangle}$

де $\langle \Delta A \rangle$ – середнє значення послідовності $\Delta A_1, \Delta A_2, \Delta A_3$; $\langle \Delta A^2 \rangle$ – послідовності A_1^2, A_2^2, A_3^2; K – кількість появ слова A в тексті.



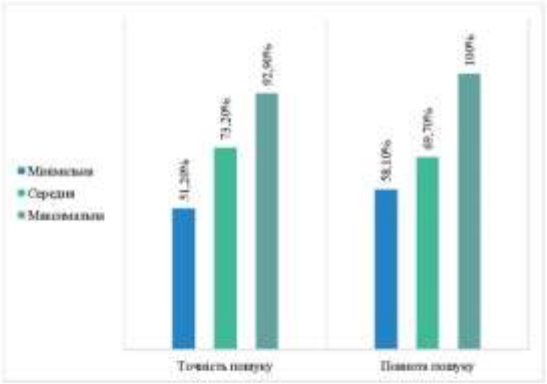
10

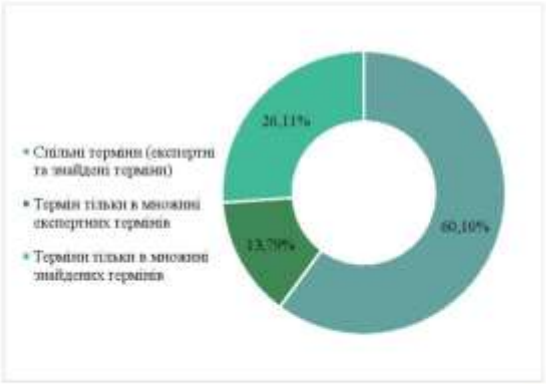
ДОСЛІДЖЕННЯ
СИНТАКСИЧНИХ ПАРАМЕТРІВ КЛЮЧОВИХ ТЕРМІНІВ



11

ДОСЛІДЖЕННЯ ПОВНОТИ І ТОЧНОСТІ ПОШУКУ КЛЮЧОВИХ ТЕРМІНІВ





12

ВИМОГИ ДО ВХІДНИХ ДАНИХ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

Обов'язкові вимоги	Інформаційний навчальний матеріал являється подання тестового матеріалу у вигляді ASCII символів
Додаткові вимоги	Використання єдиної назви терміну
	Використання стилів форматування тексту
	Формування рубрик
	Коректне використання розділових знаків та регістру літер

13

ВИСНОВКИ

За результатами проведеної роботи було досягнуто мету дипломної роботи магістра – розроблено **інформаційну технологію автоматизованого формування семантичної структури інформаційного навчального матеріалу.**

Було розроблено експериментальну інформаційну систему для тестування розробленої інформаційної технології.

Було проведено наступні дослідження:

- Дослідження функціональності інформаційної системи
- Дослідження синтаксичних параметрів ключових термінів
- Дослідження з вибору алгоритму пошуку ключових термінів
- Дослідження повноти і точності пошуку ключових термінів
- Дослідження повноти покриття контенту ключовими термінами

Виконано експериментальну перевірку **інформаційної технології** автоматизованого формування семантичної структури інформаційного навчального матеріалу шляхом розробки й використання відповідної ІС. Результати експериментального тестування запропонованої ІТ довели її спроможність розв'язувати поставлені задачі. При цьому, середня точність пошуку ключових термінів складає 73,2%, середня повнота пошуку – 69,7%. Знайдені ключові терміни містяться в середньому в 81,2% абзаців, або в 87,3% речень.

14

НАУКОВІ ТА ІННОВАЦІЙНІ ПОЛОЖЕННЯ, ЩО ВИНОСЯТЬСЯ НА ЗАХИСТ

- Удосконалено інформаційну модель семантичної структури інформаційного навчального матеріалу, що містить елементи та атрибути семантичної структури інформаційного навчального матеріалу
- Удосконалено метод формування рубрикації інформаційного навчального матеріалу
- Удосконалено метод пошуку ключових термінів у рубриках інформаційного навчального матеріалу
- Вперше розроблено інформаційну технологію автоматизованого формування семантичної структури інформаційного навчального матеріалу

15

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДИПЛОМНОЇ РОБОТИ МАГІСТРА ТА ПУБЛІКАЦІЇ

Основні наукові та практичні результати *доповідалися* на конференціях:

- Доповідь на тему «Інформаційна система для автоматизованого формування тестових завдань за навчальними матеріалами» на Міжнародній конференції молодих вчених «Сучасні технології в механіці» (Хмельницький, 18-21 квітня 2018 року, Хмельницький національний університет)
- Доповідь на тему «Інформаційна система для автоматизованого формування тестових завдань за навчальними матеріалами» на Всеукраїнській науково-практичній конференції «Інтелектуальний потенціал – 2018» (Хмельницький, 14-16 листопада 2018 року, Університет економіки і підприємництва)
- Доповідь на тему «Використання програмного розширення spire.doc для автоматизації роботи з цифровими документами» на Всеукраїнській науково-практичній конференції «Актуальні проблеми комп'ютерних наук АПКН-2019» (Хмельницький, 14-15 листопада 2019 року, Хмельницький національний університет)
- Доповідь на тему «Аналіз результатів автоматизованого пошуку ключових термінів у навчальних матеріалах» на Всеукраїнській науково-практичній конференції «Актуальні проблеми комп'ютерних наук АПКН-2020» (Хмельницький, 9-10 листопада 2020 року, Хмельницький національний університет)

16

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДИПЛОМНОЇ РОБОТИ МАГІСТРА ТА ПУБЛІКАЦІЇ

За темою дипломної роботи магістра автором виконано п'ять *наукових публікацій*, одна з яких у фаховому виданні, яке включено у перелік МОН України.

- Мазурець О. В. Використання спеціалізованих програмних розширень для автоматизації роботи з цифровими документами навчальних матеріалів / О. В. Мазурець, О. В. Ковальчук, В. О. Слободзян // Науковий журнал «Вісник Хмельницького національного університету» серія: Технічні науки. Хмельницький, 2018, №1. – С.61-69.
- Мазурець О. В. Метод формального опису елементів моделей автоматизованого формування тестових завдань / О. В. Мазурець, О. В. Ковальчук, В. О. Слободзян, Г. А. Білоус // Збірник наукових праць за матеріалами Всеукраїнської науково-практичної конференції «Інтелектуальний потенціал – 2018». Хмельницький, 2018, Ч.1. – С.51-56.
- Слободзян В. О. Аналіз сучасних спеціалізованих програмних розширень для автоматизації роботи з цифровими документами навчальних матеріалів / В. О. Слободзян, О. В. Мазурець // Сучасні технології в механіці: Збірник наукових праць. Хмельницький – 2018. – С.184-191.
- Ковальчук О. В. Використання програмного розширення spire.doc для автоматизації роботи з цифровими документами / О. В. Ковальчук, Г. А. Білоус, В. О. Слободзян // Збірник наукових праць за матеріалами XI всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2019» – Хмельницький, 2019, Т.1. – С.116-122.
- Слободзян В. О. Аналіз результатів автоматизованого пошуку ключових термінів у навчальних матеріалах / Слободзян В. О., Мазурець О. В. // Збірник наукових праць за матеріалами XII всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2020». Хмельницький, 2020. – С.269-274.

17

Wed Dec 02 11:44:42 EET 2020, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 2.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 10%**

	Документ		Сумарний збіг по Базі Даних	
ID: 82019 Назва: Інформаційна технологія автоматизованого формування семантичної структури інформаційного навчального матеріалу Додано в БД: 2020-12-02 Автора: Слободзян Віталій Олександрович Керівники: Мазурець О.В. Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	111305	871	4491 (4%)	46 (5%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

РІШЕННЯ КАФЕДРИ
КОМП'ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: **Інформаційна технологія автоматизованого формування семантичної структури інформаційного навчального матеріалу**
Автор: **Слободзян Віталій Олександрович**
Спеціальність: **122 Комп'ютерні науки**
Науковий керівник: **ст. викладач Мазурець Олександр Вікторович**

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних). Робота приймається до захисту.	**відповідає**
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	-
3	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	-
4	Інше:	-

Підтвердження: Показник збігів є незначним і складає 11.4% (найбільша схожість 2.64% з одним джерелом). Виявлені в роботі запозичення є законними і не є плагіатом, оскільки стосуються огляду існуючих рішень, мають відповідні посилання на джерела у Переліку посилань і розміщені в розділах, які не описують безпосередньо авторське дослідження. Випадки віднесення до плагіату елементів бібліографічного опису джерел у Переліку посилань, а також збіги із публікаціями автора за даною тематикою не є плагіатом. Робота приймається до захисту.

02.12.2020
Дата

Підпис керівника

Підпис завідувача кафедри

Хмельницький національний університет

Кафедра комп'ютерних наук та інформаційних технологій

ВІДГУК ОПОНЕНТА

на дипломну роботу магістра

Магістра *гр. КНм-19-1 Слободзяна Віталія Олександровича*

На тему: *Інформаційна технологія автоматизованого формування семантичної структури інформаційного навчального матеріалу.*

1. Актуальність і значення теми

Враховуючи великі обсяги електронних документів, задля пришвидшення роботи з ними інформаційними навчальними матеріалами та їх структуризації можна використовувати семантичну структуру. Для повноцінного подання семантичної структури доцільно сформувати систему рубрикацій та наповнити її ключовими термінами. Вадами ж існуючих методів є неповна інформаційна модель семантичної структури та неточні методи пошуку ключових слів. Тому є доцільною розробка інформаційної технології автоматизованого формування семантичної структури інформаційного навчального матеріалу, яка може бути використана для розв'язання ряду задач в питаннях автоматизації роботи з навчальними матеріалами

2. Оцінка якості та достовірності проведених досліджень.

Проведені в роботі дослідження підтвердили високу ефективність запропонованої інформаційної технології та її складових. Отримані результати відповідають вимогам, що висуваються до їх подальшого використання.

3. Оцінка запропонованих заходів та пропозицій, практичної цінності та ефективності.

Розроблену інформаційну технологію й її компоненти можна використовувати при вирішенні ряду задач, таких як визначення відповідності інформаційних навчальних матеріалів вимогам, допомога при розробці інформаційних навчальних матеріалів, допомога при створенні тестів тощо.

4. Загальний висновок та оцінка

Робота виконана в повному обсязі. Пояснювальна записка оформлена в відповідності з вимогами. Відмічені недоліки не знижують практичної та наукової цінності дипломної роботи. За своєю структурою, поставленій меті та вирішеними задачами робота відповідає вимогам вищої школи і вимогам, що пред'являються до освітньо-кваліфікаційного рівня «магістр», а її автор Слободзян В. О. заслуговує присвоєння кваліфікації магістра з комп'ютерних наук.

Робота заслуговує на оцінку «відмінно».

Опонент Мартинюк В.В., 18.01.21, 17:09