МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри

_____Литвиненко О.Є.
«_____»_____ 2020 р.

ДИПЛОМНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ МАГІСТР

ЗА НАПРЯМОМ ПІДГОТОВКИ 123 «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ»

Тема: «Технологія автоматизованої побудови логіко-лінгвістичних моделей»

Виконавець: студент, група СП-235М, Динько Андрій Юрійович
(студент, група, прізвище, ім'я, по-батькові)

Керівник: д.т.н., доцент Вавіленкова Анастасія Ігорівна
(науковий ступінь, вчене звання, прізвище, ім'я, по-батькові)

Нормоконтролер: _____ Тупота Є.В.
(підпис)

КИЇВ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет Кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютеризованих систем управління
Напрям (спеціальність) 123 «Комп'ютера інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____О.Є. Литвиненко
«_____» _____ 2020 р.

ЗАВДАННЯ
на виконання дипломної роботи

Динька Андрія Юрійовича
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема дипломної роботи «Технологія автоматизованої побудови логіко-лінгвістичних моделей» затверджена наказом ректора від «27» серпня 2020р. № 1203/ст

2. Термін виконання проекту: з 5 жовтня 2020 р. по 13 грудня 2020 р.

3. Вихідні дані до роботи: логіко-лінгвістичні моделі, правила формування словосполучень, електронний словник термінів української мови, типи побудови логіко-лінгвістичних моделей

4. Зміст пояснювальної записки:

1) Механізми екстракції знань;

2) Метод автоматизованої побудови логіко-лінгвістичних моделей;

3) Автоматизована побудова логіко лінгвістичних моделей.

5. Перелік обов'язкового графічного (ілюстративного) матеріалу:

1) Схема алгоритму технології автоматизованої побудови логіко-лінгвістичних моделей;

2) Структура програмного модуля формування логіко-лінгвістичних моделей;

3) База даних слів української мови;

4) Матриця кодування речення;

5) Схема алгоритму екстракції словосполучення «Прикметник+Іменник»;

6) Результат роботи технології автоматизованої побудови логіко-лінгвістичних моделей.

6. Календарний план-графік

№ з/п	Завдання	Термін виконання	Примітка
1.	Аналіз джерел за темою дослідження. Розроблення плану дипломного проекту.	5.10.2020-10.10.2020	
2.	Затвердження плану дипломного проекту та проведення консультацій з науковим керівником, щодо наповнення пояснювальної записки.	12.10.2020-24.10.2020	
3.	Дослідження структури систем здійснення автоматичного синтаксичного аналізу	26.10.2020-29.10.2020	
4.	Розробка алгоритму функціонування технології автоматизованої побудови логіко-лінгвістичних моделей	30.10.2020-13.11.2020	
5.	Створення програмного реалізації технології автоматизованої побудови логіко-лінгвістичних моделей	16.11.2020-27.11.2020	
6.	Написання пояснювальної записки	30.11.2020-9.12.2020	
7.	Підготовка демонстраційного матеріалу	10.12.2020 13.12.2020	

7. Дата видачі завдання: « 5 » жовтня 2020 р.

Керівник дипломного проекту _____ д.т.н., доцент, Вавіленкова А.І.
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Динько А.Ю.
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Технологія автоматизованої побудови логіко-лінгвістичних моделей»: 85 сторінок, 33 рисунків, 1 таблиця, 25 літературних джерел, 1 додаток.

ЗНАННЯ, ЕКСТРАКЦІЯ ЗНАНЬ, ЛОГІКО-ЛІНГВІСТИЧНІ МОДЕЛІ, МОРФОЛОГІЧНИЙ АНАЛІЗ, СИНТАКСИЧНИЙ АНАЛІЗ, *WINDOWS FORMS*.

Об’єктом дослідження даної дипломної роботи є процес автоматичної побудови логіко-лінгвістичних моделей простих речень природної мови.

Предметом дослідження є системи аналітичної обробки текстової інформації.

Метою даного дипломного проекту є розробка технології автоматичного формування логіко-лінгвістичних моделей речень природної мови.

Методи дослідження: технології створення аналітичних систем, порівняльний аналіз, методи обробки текстової інформації, методи представлення знань, методи синтаксичного аналізу, методи об’єктно-орієнтованого програмування.

Здійснено огляд існуючих аналітичних систем синтаксичного аналізу текстової інформації; досліджено моделі представлення знань; проаналізовано етапи обробки текстової інформації; вивчено алгоритми формування логіко-лінгвістичних моделей текстової інформації; удосконалено алгоритм автоматичного формування логіко-лінгвістичних моделей речень природної мови; програмно реалізовано технологію автоматизованої побудови логіко-лінгвістичних моделей.

Матеріали дипломного проекту рекомендується використовувати при проведенні наукових досліджень в області комп’ютерної обробки текстової інформації, створенні багатофункціонального програмного забезпечення для аналізу тексту, у навчальному процесі фахівців з системного програмування.

Прогнозні припущення про розвиток об’єкту та предмету дослідження – застосування для удосконалення способів змістовного аналізу текстової інформації.

ЗМІСТ

ВСТУП

З розвитком інформаційних технологій кількість текстової інформації щороку зростає в геометричній прогресії. Даний процес в свою чергу спричиняє потребу в автоматизованому сортуванні і збереженні необхідного для користувачів. Велика кількість фізичних носіїв для зберігання потребує великих фінансових витрат на їхнє утримування і експлуатацію. З'являється проблема, в процесі пошуку користувачем, знаходження потрібної інформації. Ці всі фактори слугують розвитку галузі аналітичної обробки текстової інформації, метою якої є швидкий і зручний для користувача аналіз тексту і видача необхідної інформації.

Постає питання про те як комп'ютерній системі зрозуміти природну мову і могти оперувати з нею і здійснювати екстракцію знань. Оскільки машини розуміють мову формул, то доцільним буде подати текстову інформацію у такому вигляді. Логіко-лінгвістичне моделювання забезпечує дану функцію, адже будує предикат сформований на основі синтаксичного аналізу речення природньої мови і дозволяє в подальшому здійснювати різного роду маніпуляції, при цьому зберігаючи зміст.

Сьомого липня 2019 року був прийнятий Закону України «Про забезпечення функціонування української мови як державної», котрий передбачає використання української мови в галузі освіти та науки, медицини, інтернет-сайтах та інших. Це в свою чергу збільшує кількість текстової інформації українською мовою, котра потребує текстових аналізаторів, що вміють обробляти дану природну мову.

Метою даної дипломної роботи є розробка технології формування логіко-лінгвістичних моделей простих речень природної мови. Це передбачає вивчення механізмів та засобів вилучення знань із текстової інформації, моделей представлення знань, етапів обробки текстової інформації, способів формування логіко-лінгвістичних моделей та розробці програмної реалізації технології автоматизованої побудови логіко-лінгвістичних моделей шести типів.

Об'єктом дослідження даної дипломної роботи є процес автоматичної побудови логіко-лінгвістичних моделей простих речень природної мови.

Предметом дослідження є системи аналітичної обробки текстової інформації.

Методи дослідження, які були використані для досягнення поставленої мети: технології створення аналітичних систем використовувалися для окреслення деталей розроблюваної технології, порівняльний аналіз для здійснення морфологічгого розбору, методи обробки текстової інформації для екстракції знань, методи представлення знань для надання знанням вигляду зрозумілому для комп'ютерних систем, методи синтаксичного аналізу для визначення елементів предикату логіко-лінгвістичної моделі, методи об'єктно-орієнтованого програмування для створення програмної реалізації розробленої тнхнології.

Новизна дипломної роботи полягає в застосуванні формалізованих правил української мови для створення технології автоматизованої побудови логіко-лінгвістичних моделей із речень українською мовою. Кожне речення закодоване за допомогою використанням логіко-лінгвістичного моделювання на основі морфологічного аналізу, екстракції словосполучень та синтаксичного аналізу. Було додано новий механізм кодування речення – формування матриці кодування речення, який містить в собі морфологічні та синтаксичні характеристики слів речення, пунктуація, що в свою чергу необхідно для побудови предикату, елементи якого є синтаксичні одиниці - члени речення.

Практичне значення отриманих результатів – створено та програмно реалізовано технологію автоматизованої побудови логіко-лінгвістичних моделей речень української мови. Програмний модуль містить текстовий редактор, котрий може відкривати файли з текстом для подальшої побудови формул на основі його вмісту, редагувати вміст файлу, зберігати зміни. Також доступна можливість зберегти результати побудови логіко-лінгвістичних моделей речень українською мовою в окремому файлі. У вікні застосунку після формування моделей також відображаються сформовані словосполучення наступних типів: «Прикметник + Іменник», «Порядковий числівник + Іменник», «Займенник + Іменник», «Дієприкметник та Іменник», Дієслово + Іменник», «Прикметник + Дієслово».

Особистий внесок випускника – розробка власного програмного модуля для автоматичної побудови логіко-лінгвістичних моделей речень українською мовою на

основі сформованих словосполучень, котрі в свою чергу побудовані завдяки морфологічному аналізу слів речення, та синтаксичного аналізу.

Апробація отриманих результатів. Взято участь у міжнародній технічній конференції «Інтелектуальні технології лінгвістичного аналізу» 2018 р.

Публікації. У міжнародній технічній конференції, «Інтелектуальні технології лінгвістичного аналізу» 2018 р., запропоновано внести зміни до функціоналу програмного забезпечення пошуку плагіату, а саме: перевірка на відповідність символів тій чи іншій природній мові, перевірка на прозорість символів, використання онлайн перекладачів для порівняння з іншими мовами, додавання бази даних синонімів.

Прогнозні припущення про розвиток об’єкту та предмету дослідження – застосування у системах аналізу текстової інформації за змістом, можливе застосування як компонент семантичного аналізатора на основі систем штучного інтелекту, що в свою чергу збільшить точність роботи.

РОЗДІЛ 1
ЗАСОБИ ВИЛУЧЕННЯ ЗНАНЬ З ТЕКСТОВОЇ ІНФОРМАЦІЇ

1.1. Інструменти екстракції знань

Вилучення знань з різних джерел, у тому числі із об'єктів природної мови, включає: аналіз вихідної інформації, формалізацію якісних знань й інтеграцію знань. Аналіз і формалізація пов'язані зі створенням методів, що дозволяють переходити від знань, виражених, в тому числі природною мовою, до їх аналогів, придатних для введення їх в пам'ять знання-орієнтованої інформаційної системи. Далі відбувається інтеграція знань, добутих від різних джерел, в деяку взаємопов'язану і несуперечливу систему знань про предметну область.

Знання, що містяться в джерелах інформації, відчужених від фахівця, як правило, недостатньо. Значну частину професійного досвіду ці фахівці не можуть висловити словесно (професійне вміння або інтуїція). Тому для того, щоб добути такі знання, потрібні спеціальні прийоми і методи. Вони використовуються в інструментальних системах добування знань, створення яких – одне із завдань інженерії знань. Отримані від експертів знання потрібно оцінити з точки зору їх відповідності раніше накопичених знань і формалізувати для введення в пам'ять системи. Крім того, знання, отримані від різних експертів, необхідно узгодити між собою, тому що нерідкі випадки, коли ці знання виявлялися зовні несумісними і навіть суперечливими. Розглянутий підпроцес включає: організацію роботи з експертами, оцінку і формалізація знань і узгодження знань.

Процес представлення знань передбачає розробку формальної наукової теорії, що включає побудову моделі знань, системи подання знань і бази знань. В інженерії знань системи подання знань включають сукупність процедур, необхідних для запису знань, вилучення їх з пам'яті і підтримки збереження знань

в робочому стані. Системи подання знань оформлюються як бази знань, є природним розвитком баз даних . Представлення знань – це угода про те, як і в якій формальній теорії описувати реальний світ. У природничих і технічних науках прийнятий наступний традиційний спосіб представлення знань. Природною мовою вводяться основні поняття і відношення між ними. Але при цьому використовуються раніше визначені поняття і відношення, зміст яких вже відомий. Далі встановлюється відповідність між характеристиками (найчастіше кількісними) понять і підходящої математичної моделі.

Основна мета подання знань – будувати математичні моделі реального світу і його частин. Системою подання знань називають сукупність засобів, що дозволяють:

- описувати знання про предметну область за допомогою мови представлення знань;
- організувати зберігання знань у системі (накопичення, аналіз, структурне узагальнення та організація знань);
- виводити нові знання з існуючих і об’єднувати їх;
- знаходити необхідні знання;
- поновлювати знання;
- здійснюватиінтерфейс між системою і користувачем.

До процесу маніпулювання знаннями відносяться такі розділи як поповнення знань, класифікація знань, узагальнення знань, виводи на знаннях (резолюційні методи, квазіаксіоматичні системи і системи правдоподібного виводу), міркування за допомогою знань, пояснення на знаннях, вирішення прикладних задач. Нові знання, які надходять у базу знань, повинні разом з тими відомостями, які вже були раніше записані в неї, сформувати розширення знань, які надійшли. Серед цих процедур особливе місце займають псевдофізичні логіки (часу, простору, дій та ін.), які, спираючись на закони зовнішнього світу, поповнюють базу знань інформацією, що надійшла. Знання знання-орієнтованої інформаційної системи утворюють впорядковані структури, що полегшує пошук необхідних знань і підтримку дієздатності бази хнань. Для цього використовуються різноманітні класифікаційні процедури. Типи класифікацій можуть бути різними: родо-видові, «частинаціле»,

ситуативні (коли в одну множину об’єднуються знання, релевантні деякій типовій ситуації). Зазначені процедури будуються на основі теорії класифікацій. У процесі класифікації часто відбувається абстрагування від окремих елементів описів (фрагментів знань про об’єкти або явища), з’являються узагальнені знання, що призводить, врешті-решт, до абстрактних знань, для яких немає прямого прообразу в зовнішньому світі. Вивід на знаннях залежить від моделі, яка використовується для їх представлення. Якщо в основі представлення використовуються логікові системи або продукції, то вивід на знаннях стає близьким до стандартного логічного виведення. Це ж відбувається при поданні знань у каузальній формі. У всіх цих випадках в знання-орієнтованої інформаційної системи використовуються методи виведення, що спираються на ідеї методу резолюцій. Можливість появи в пам’яті знання-орієнтованої інформаційної системи нових фактів і відомостей приводить до того, що починає порушуватися принцип монотонності, що лежить в основі функціонування всіх систем, що вивчаються традиційною математичною логікою. Немонотонність виведення у відкритих системах викликає немалі труднощі. В останні роки прихильники логічних методів у штучному інтелекті роблять спроби побудувати нові логічні системи, в рамках яких можна було б забезпечити немонотонне виведення. По суті, системи, за допомогою яких представляються знання про предметні області, не є строго аксіоматичними, як класичні логічні числення. Тому й системи, які виникають при таких умовах, слід називати квазіаксіоматичними. У таких системах цілком можлива зміна вихідних аксіом в процесі виведення. Ще одна особливість виведення на знаннях – неповнота відомостей про предметні області та процеси, що в ній протікають, неточність вхідної інформації. А це означає, що виводи в знання-орієнтованої інформаційної системи носять не абсолютно достовірний характер, як у традиційних логічних системах, а наближений, правдоподібний характер. Такі виводи вимагають розвиненого апарату обчислення оцінок правдоподібності і методів оперування ними. Оскільки знання-орієнтованої інформаційної системи приймають рішення, спираючись на знання, які можуть бути невідомі користувачеві, що вирішує своє завдання за допомогою цієї системи, то він може засумніватися в правильності отриманого рішення. Знання-

орієнтованої інформаційної системи повинна володіти засобами, які можуть сформувати користувачеві необхідні пояснення. Пояснення можуть стосуватися процесу отримання рішень, підстав, які були для цього використані, способів відсікання альтернативних варіантів та ін. Все це вимагає розвиненої теорії пояснення. На основі проведеного вище аналізу на рис. 1.1 синтезовано узагальнений процес введення і обробки вхідних даних, семантичної переробки інформації і роботи із знаннями. В ній також відображені зв'язки між відповідними процесами, послідовності яких складають інформаційні технології та обробки даних, текст-процесингу, роботи із знаннями і логікоінформаційного підходу [1-5], що становлять основу методології проектування знання-орієнтованих інформаційних систем з обробкою об'єктів природної мови і формалізованих знань.

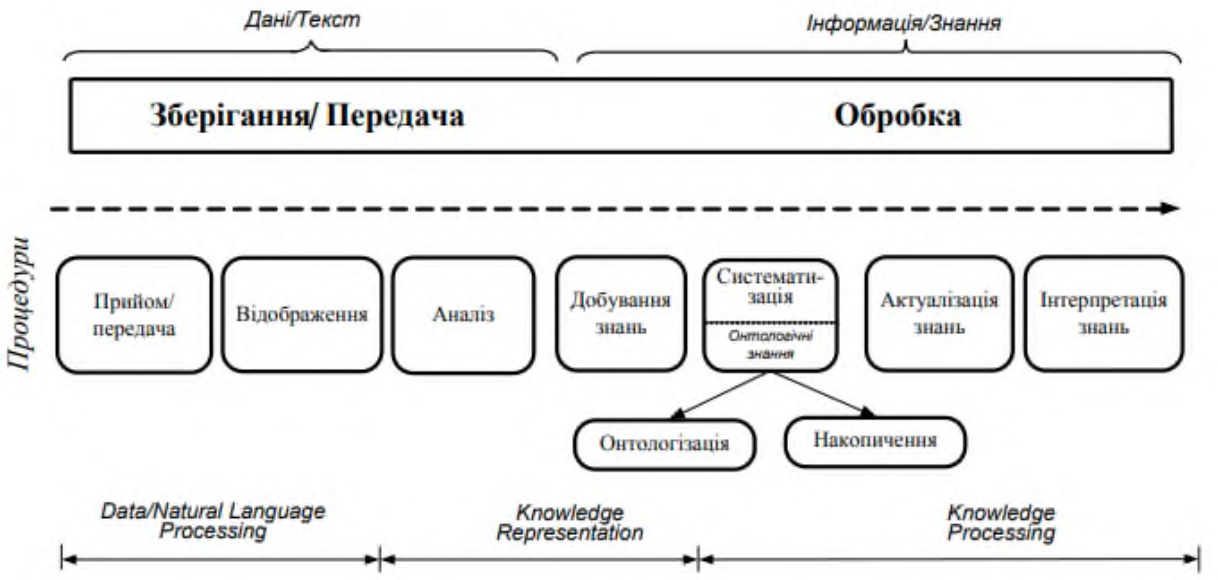

Рис. 1.1. Процес роботи із знаннями

Концептуальна парадигма роботи зі знаннями утвердилася у зв'язку і в міру розвитку концепції інтелектуальних інформаційних систем, згідно з якою головною інформаційною одиницею комп'ютерної обробки стало «знання». Незалежно і в рамках різних областей предметних знань розвиваються інформаційні технології: автоматичної обробки природної мови – в комп'ютерній лінгвістиці, представлення знань – в математичній логіці. Почали з'являтися наукові розробки, що пропонують конструктивні підходи до інтеграції зазначених технологій[6].

На сьогоднішній день практично не існує повноцінних рекомендацій і порад про те, як ефективно справлятися із завданнями обробки текстової інформації, при цьому розглядають рішення цих задач з основ.

Основні етапи обробки текстової інформації подано нижче[7]:

– Збір даних. Будь-яке завдання машинного навчання починається з даних будь то список адрес електронної пошти, постів або твітів. Для того щоб провести обробку чогось треба щось мати.
– Очистка даних. Чистий текст дозволить моделі вивчити важливі ознаки і не перенавчитися на нерелевантному шумі. За допомогою чеклиста проводиться очистка даних: видалити всі нерелевантні символи, розділити текст на окремі слова, видалити нерелевантні слова, перевести всі символи в нижній регістр(для того, щоб слова «привіт», «Привіт» та «ПРИВІТ» рахувалися як одне і те ж слово), врахувати можливість сумісності слів, які написані з помилками, або мають альтернативне написання(наприклад, «круто»/ «круть»/ «крууто»), розглянути можливість існування різних форм одного слова(наприклад, «машина» замість «машиною», «на машині», «машинами»). Після виконання чеклисту і виконання перевірки на додаткові помилки, можна починати використовувати чисті, помічені даніі для навчання моделей.
– Вибір представлення даних. Текс представляє собою список речень, для того щоб алгоритм міг видобути патерни з даних, необхідно представити його таким чином, щоб наш алгоритм міг його зрозуміти. Зазвичай відображення тексту в комп’ютерах являє собою кодування кожного символу індивідуально у вигляді числа. Проте кращим рішенням буде побудувати словник усіх унікальних слів у тексті та присвоїти унікальний індекс кожному слову в словнику. Тоді кожне речення можна буде відобразити вектором, довжина якого дорівнює числу унікальних слів у словнику, а в кожен індекс в цьому векторі буде зберігатися стільки разів, скільки разів це слово зустрічається в реченні. Дана модель називається «Мішок слів» (*Bag of Words*). Приклад моделі «Мішок слів» зображений на рис. 1.2.

Рис. 1.2. Модель «Мішка слів»

– Класифікація. Найбільш поширеним способом поділу є логістична регресія(статистична модель, яка використовується для прогнозування імовірності появи деякої події шляхом підгонки даних під логістичну криву) через свою універсальність та легкість тлумачення. Дуже проста для машинного навчання та її результати можна інтерпретувати, оскільки наявна можливість з легкістю видобути всі найважливіші коефіцієнти з моделі.
– Інспектування. Для початку необхідно визначити які типи помилок здійснює модель та які види помилок повинні з'являтися найрідше. Хорошим способом візуалізації даної інформації є використання матриці помилок(*Confusion matrix*), яка порівнює сподівання зроблені моделлю з реальними мітками. В ідеалі, дана матриця буде представлена як діагональна лінія, яка рухається з верхнього лівого кута до нижнього правого. Зображення матриці помилок подане на рис. 1.3.
– Врахування структури словника. Щоб модель сфокусувалася на значимих словах, використовується оцінка важливості слів у контексті *TF-IDF*(*Term Frequency, Inverse Document Frequency*). *TF-IDF* зважує слова на основі рідкості в тексті, знижуючи у відсотках слова, які зустрічаються занадто часто і просто додають шум.

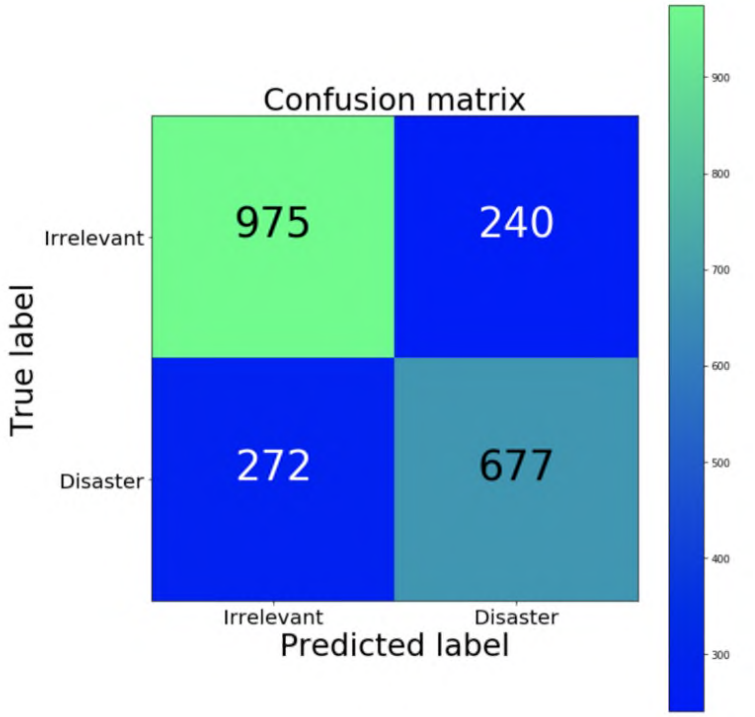

Рис. 1.3. Матриця помилок

– Врахування структури словника. Щоб модель сфокусувалася на значимих словах, використовується оцінка важливості слів у контексті *TF-IDF*(*Term Frequency, Inverse Document Frequency*). *TF-IDF* зважує слова на основі рідкості в тексті, знижуючи у відсотках слова, які зустрічаються занадто часто і просто додають шум.
– Застосування семантики. Семантика вивчає значення слів і їх складових частин, словосполучень і фразеологізмів. До цього етапу обробки текстової інформації наявна можливість лише видобувати слова, що мають найбільше значення. Проте не слід забувати про слова, які не зустрічалися у вибірці і нездатність моделі точно класифікувати ці слова. Для вирішення даного питання необхідно враховувати семантичне значення слів для того щоб модель розуміла, наприклад, що слова «хороший» і «позитивний» ближчі між собою чим слова «абрикос» і «континент». Використовують інструмент *Word2Vec*, який порівнює значення слів. *Word2Vec* навчається протягом великої кількості тексту із запам’ятовуванням того, яке слово виникає у схожих контекстах. Після навчання на достатній кількості даних, *Word2Vec*

генерує вектор із трьохсот вимірювань для кожного слова в словнику, в якому слова зі схожим значенням розташовуються ближче один до одного. Результат даної генерації зображено на рис. 1.4.

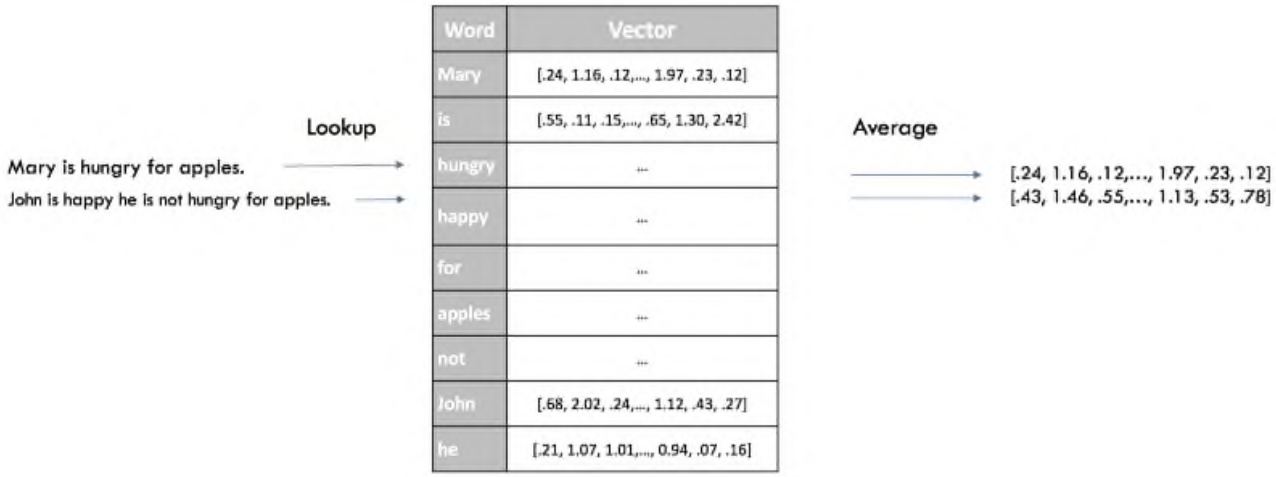

Рис. 1.4. Векторне представлення речень у *Word2Vec*

– Використання синтаксису при застосуванні «*endtoend*» підходів. Синтаксис вивчає граматичну будову словосполучень та речень у мові. Опускаючи порядок слів, відкидається вся синтаксична інформація з речень. Необхідне використання більш складної моделі, яка приймає цілі вирази як введення і пророкує мітки, без необхідності побудови проміжного представлення. Поширений для цього спосіб полягає в розгляді речень як послідовностей індивідуальних векторів слів з використанням або *Word2Vec*, або більш свіжих підходів. Згорткові нейронні мережі для класифікації речень(*CNNs for Sentence Classification*) навчаються дуже швидко і можуть відмінно служити в якості вхідного рівня в архітектурі глибокого навчання. Ця модель зберігає порядок слів і навчається цінної інформації про те, які послідовності слів служать пророкуванням наших цільових класів.

До найбільш актуальних напрямків вилучення знань з тексту на сьогоднішній день відносяться[8]:

– аналітична обробка фактів, ведення досьє;
– вилучення та структурування фактографічної інформації;

– пошук інформації за запитами на природній мові з використанням словників;
– напрямки пошуку інформації, об'єктів у сховищі документів, у добірці документів;
– анотування документів, побудова дайджестів по об'єктах;
– проведення тематичного аналізу документів (кластеризація і поділ на рубрики);
– побудова і динамічний аналіз семантичної структури текстів;
– виділення ключових тем та інформаційних об'єктів;
– дослідження частотних характеристик текстів.

Розглядаючи знання як щось об’єктивне, феноменальне, що відображає вихідну, перетворюючу і кінцеву складові процесу пізнання можна зробити висновок, що сукупність наукових дисциплін, які мають безпосереднє відношення до процесів мислення, розуміння, усвідомлення та здобуття нових знань, не може обходитися без вироблення своїх власних понять, певним способом фіксуючих властивості і закономірності її об’єктів. З таких основоположних понять, як смисл, знання, (знакова) система, текст, об’єкт, ставлення, предмет, мова, структура, зв’язок, розглянемо тільки перші два поняття, які виражають собою найбільшу проблемність досліджень в області побудови систем обробки знань, в тому числі і знання-орієнтовані інформаційні системи, що оперують об’єктами природньої мови.

Поняття є цілісна сукупність суджень, в яких щось стверджується про відмітні ознаки досліджуваної сутності, ядром якої є судження (або твердження) про найбільш загальні і в той же час істотні ознаки цієї сутності. Кожне поняття характеризується об’ємом і змістом. Об’єм і зміст поняття – дві взаємопов’язані сторони поняття. Об’єм – клас узагальнених в понятті предметів, зміст – сукупність (зазвичай істотних) ознак, за якими вироблено узагальнення і виділення предметів в даному понятті. Об’єм поняття є визначальним при формуванні ієрархічної структури відповідного онтографа , а зміст – при аксіоматизації його онтографа вершин[9].

Відомо, що теорії смислів в завершеному вигляді не існує, а тому і загальноприйнятого визначення поняття «смисл» також немає, як немає

однозначного розуміння і з точки зору його формального подання і перетворення в комп'ютерній системі. Ось найбільш поширені визначення з точки зору проектування комп'ютерних систем:

- Смисл – це є те, що робить знакові системи текстом[10].
- Смисл – деякий безперервний, невербальний конструкт, осмислення – інтерпретація в індивідуальній концептуальній системі[11].
- Смисл – це є мережа значень у певних позиціях і оперативний алгоритм для вирішення проблем[12].

Відомо [13, 14], що знання характеризуються рядом властивостей, що відрізняють їх від традиційних моделей даних. До таких властивостей можна віднести:

- Внутрішня інтерпретація. При зберіганні знань у пам'яті інтелектуальної інформаційної системи поряд з традиційними елементами даних зберігаються інформаційні структури, що дозволяють інтерпретувати вміст відповідних чарунок пам'яті.
- Структурованість. Знання складаються з окремих інформаційних одиниць, між якими можна встановити класифікуючі відношення: рід – вид, клас – елемент, тип – підтип, частина – ціле і т. п.
- Зв'язність. Між інформаційними одиницями передбачаються зв'язки різного типу: причина – наслідок, одночасно, бути поруч та ін. Дані зв'язки визначають семантику і прагматику предметної області.
- Семантична метрика. На множині інформаційних одиниць, що зберігаються в пам'яті, вводяться деякі шкали, що дозволяють оцінити їх семантичну близькість. Це дозволяє знаходити в інформаційній базі знання, близькі до вже знайдених.
- Активність. За допомогою даної властивості підкреслюється принципова відмінність знань від даних. Виконання тих чи інших дій в інтелектуальній інформаційній системі ініціюється станом бази знань. При цьому передбачається, що поява нових фактів і зв'язків може активізувати систему.

Проблема вилучення знань з тексту представляється не тільки не тривіальною, але і вельми складною, незважаючи на безсумнівні досягнення інформаційних технологій в цій галузі. Як справедливо зазначено в [15], «Текст є знакова конструкція і часто містить знання. Але текст є не знання, а тільки його джерело. Знання з тексту ще треба отримати. Людині або комп'ютерній системі». Виконаний аналіз дозволяє синтезувати деякий узагальнений процес еволюції знань, представлений на рис. 1.5. Зауважимо при цьому, що в мові даного звіту словоформи термінів «знання», «свідомість» і «пізнання» мають спільний корінь, а самі терміни також утворюють тріаду, компоненти якої фіксують об'єкт, суб'єкт і процес пізнання.

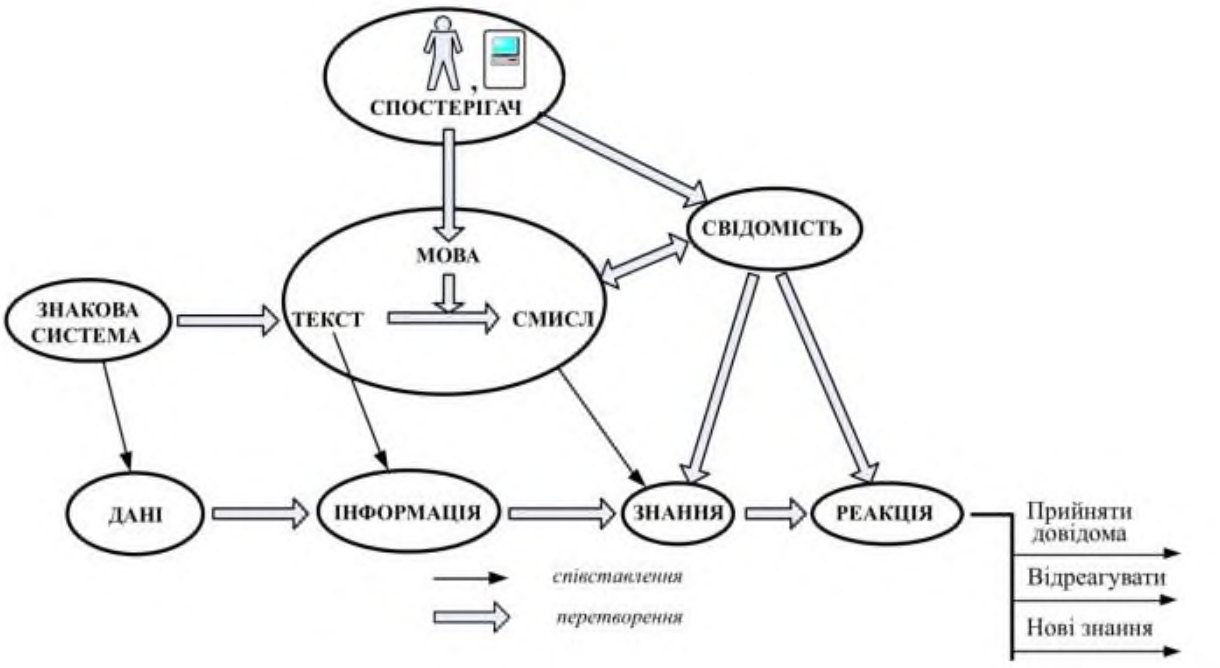

Рис. 1.5. Процес еволюції знань

Виділяють[16] два основних джерела знань – це експерти, фахівці у своїй предметній області і лінгвістичний корпус текстів (або множина об'єктів природної мови). Якщо для першого джерела методи придбання знань досить добре вивчені і опрацьовані, а також відомі відповідні промислові експертні системи, то для другого – розроблені лише окремі методи, не пов'язані в єдину інтегровану технологію, а відповідні інформаційні системи носять експериментальний характер і не досконалі. Актуальне удосконалення відомих і розроблення нових, більш ефективних формальних підходів і моделей обробки, починаючи від пошуку релевантної

інформації, її аналізу, усунення різного роду неоднозначностей, формальнологічного, онтолого-семантичного, інформаційно-кодового представлень, переходу до формалізованого опису знань, розробці процедур роботи із знаннями і відповідних їм алгоритмів, закінчуючи отриманням конкретних результатів користувачем.

Загальний процес комп'ютерної обробки знань, що містяться в об'єктах природньої мови, представлена на рис. 1.6[17]. Тут у загальному вигляді відображена інформаційна модель обробки знань, починаючи з пошуку на всьому інформаційному просторі відповідних текстових даних, які відповідають запиту користувача (можливо на природній мові) і подальшого їх перетворення (в загальному випадку) в прості, ситуаційні і нові (формальні) знання.

Під простими знаннями розуміється інформація, що надійшла про деякі сутності реального світу, якій відповідає реакція людини – «взяти до відома».

Під ситуаційними знаннями розуміється опис деякої ситуації, яку необхідно розпізнати і відповідним чином «відреагувати».

Під новими знаннями розуміється розпізнавання збільшення ΔS і поповнення ними бази знань заданої предметної області в деякому формалізованому вигляді. По суті, процес на рис. 1.6. представляє собою концептуальну модель засобів, методів і технологій обробки знань, що містяться в об'єктах природньої мови.

Аксіоматизація наукових знань визначається наступними структурними елементами:

- аксіоми або схеми аксіом (як положення, що в даній системі не доводяться);
- вихідні (початкові) поняття (явно не можуть бути визначені в системі);
- правила виводу та побудови системи знань;
- визначення як правила введення в теорію нових термінів, абстрактних об'єктів; теореми як висловлювання, що виводяться;
- завдання, наслідки, положення, що виводяться з теорії, леми, логіко-методологічні принципи теорії, що аксіоматизується.

Всі вищеназвані аспекти складають базову основу формального наукового дослідження. Поряд з ними науковий підхід необхідно передбачає ще одну важливу компоненту – інтерпретацію. Зрозуміло, що науковий смисл мають лише ті

абстракції, які завідомо можуть бути включені до чого-небудь і які можна зіставити істині, особливо в тих випадках, коли це практично важливо й можливо. Механізмом такого зіставлення істини, пошуку логічних можливостей її встановлення і виступає інтерпретація як логічний прийом щодо встановлення значення термінологічних виразів теорій, знакової (формальної) системи [18].

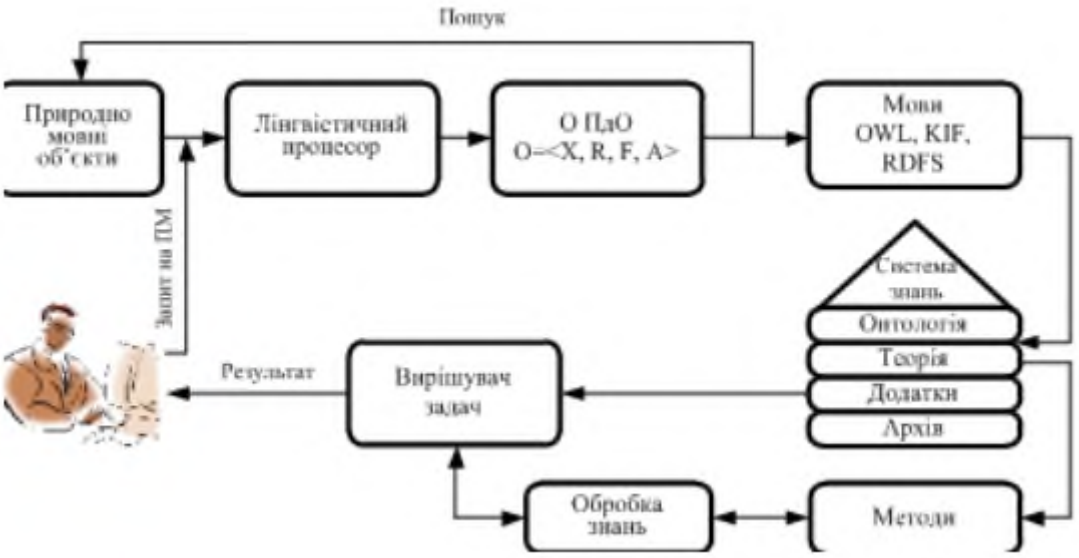

Рис. 1.6. Комп'ютерна обробка знань

Кожному реченню, логічно правильно побудованому, в інтерпретації надається деяке значення, наприклад, речення може бути істинним або хибним. Питання ж про те, яка з цих можливостей реалізована, не вирішується інтерпретацією. Тому одна і та ж формальна система передбачає безліч інтерпретацій і тільки одну істину. Інтерпретація гарантує несуперечність теорії, але не забезпечує змістовної істинності даної теорії. Інтерпретація має свої гносеологічні передумови. Суть їх полягає в тому, що в уявному процесі об'єктивна дійсність не просто пасивно відбивається, але певним чином реконструюється, перебудовується. В пізнанні відбувається безперервний процес утворення абстракцій і їх виключення, переклад одного рівня абстракцій на інший. Інтерпретація як раз і виступає логічним засобом переходу від одного рівня знань до іншого, засобом встановлення логічних зв'язків між цими рівнями [19].

Слід зазначити, що існуючі механізми вилучення нових знань оперують тільки знаннями силогістичної типу. Перехід до формування нових знань більш високого рівня абстракції методами штучного інтелекту, якщо взяти викладену в[10] ймовірнісну модель людського мислення, відповідає рівню передсвідомості Спостерігача. На цьому рівні знання організовані в складну онтологічну структуру з повним набором зв'язків між концептами. Зі сказаного випливає, що в даний час автоматична побудова бази знань для широких предметних областей представляє складну наукову проблему.

Моделі представлення знань можна умовно розділити на декларативні і процедурні. У декларативних моделях знання представляються у вигляді описів об'єктів і відносин між об'єктами без вказівки в явному вигляді, як ці знання обробляти. Такі моделі припускають відділення описів (декларацій) інформаційних структур від механізму вивід в, що оперує цими структурами. У процедурних моделях значення представляються алгоритмами (процедурами), що містять необхідний опис інформаційних елементів і одночасно визначають способи їх обробки. Конкретні моделі, застосовувані на практиці, являють собою комбінацію декларативних і процедурних представлень. Найбільш розповсюдженими є наступні моделі представлення знань:

– логічні моделі;
– продукційні моделі;
– семантичні мережі;
– онтології;
– фреймові моделі.

Логічні моделі реалізуються засобами логіки предикатів. У цьому випадку знання про предметну область представляються у вигляді сукупності логічних формул. Тотожні перетворення формул дозволяють одержувати нові знання. Перевагами логічних моделей представлення знань є наявність чіткого синтаксису і широко прийнятої формальної семантики, а також теоретично обґрунтованих процедур автоматичного вивід в. Основним недоліком даних моделей є неможливість одержання висновків в областях, де вимагаються правдоподібні висновки, коли

результат виходить з визначеною оцінкою впевненості в його істинності. Крім цього, такі моделі характеризуються монотонним характером вивід в, тобто в базу знань додаються тільки істинні твердження, що виключає можливість протиріч. На практиці часто зустрічаються немонотонні міркування, що важко реалізувати в рамках логічної моделі. Проте, логічні моделі виступають у якості теоретичної основи опису самої системи представлення знань і поступово розширюють свої можливості. Тому надалі цим моделям приділяється значна увага.

У продукційних моделях знання представляються набором правил виду "Якщо *A*, то *B*", де умова правила *A* є твердженням про вміст бази фактів, а наслідок *B* говорить про те, що треба робити, коли дане продукційне правило активізоване. Продукційні моделі представлення знань завдяки природній модульності правил, наочності і простоті їхнього створення широко застосовуються в інтелектуальних системах. Продукційні системи були запропоновані американським математиком Е. Постом (1943) і розглядалися як модель організації обчислюваного процесу. При цьому правило-продукція трактувалась як оператор заміни одного ланцюжка літер у деякому слові на інший ланцюжок. Систему продукцій можна розглядати як формальну систему з алфавітом $A = \{a_1, a_2, \dots, a_n\}$, кінцевою множинучю аксіом і кінцевою множинучю правил виводу, що мають вигляд:

$$P_i \in : \alpha_i s \rightarrow s\beta_i, i = 1,2, \dots, k,$$

де α_i, β_i, s – слова алфавіту A.

Застосування даного правила до деякого ланцюжка літер, що складається зі слів α_i, і s , означає викреслення α_i, і приписування β_i. Зазначена заміна відповідає підстановці, що є основним оператором нормальних алгоритмів Маркова.

Широке застосування продукційних моделей визначається наступними основними перевагами:

– універсальністю, практично будь-яка область знань може бути представлена в продукційній формі;

– модульностью, кожна продукція являє собою елемент знань про предметну область, видалення одних і додавання інших продукцій виконується незалежно;
– декларативністю, продукції визначають ситуації, предметної області, а не механізму керування;
– природністю процесу виводу, що багато в чому аналогічний процесу міркувань експерта;
– асинхронностю і природним паралелізмом, що; робить їх дуже перспективними для реалізації на рівнобіжних ЕОМ.

Однак продукційні моделі також мають недоліки:

– процес виводу має низьку ефективність, тому що при великому числі продукцій значна частина часу затрачається на невиробничу перевірку умов застосування правил;
– перевірка несуперечності системи продукцій стає дуже складною через недетермінованності вибору виконуваної продукції з конфліктної множини.

Семантичні мережі не є однорідним класом моделей представлення знань. Часто загальною основою віднесення схеми представлення знань до семантичної мережі є те, що вона представляється у вигляді спрямованого графа, вершини якого відповідають об’єктам (поняттям, сутностям) предметної області, а дуги – відносинам (зв’язкам) між ними. І вузли, і дуги, як правило, мають мітки (імена), Імена вершин і дуг звичайно збігаються з іменами відповідних об’єктів і відносин предметної області. Об’єкти предметної області, що відображаються в семантичній мережі, можна умовно розділити на три групи: узагальнені, індивідуальні (конкретні) і агрегатні об’єкти. Загалиний об’єкт відповідає деякій збірній абстракції реально існуючого об’єкта, чи процесу явища предметної області. Наприклад, “виріб”, “підприємство”, “співробітник” і т.д. Узагальнені об’єкти фактично представляють визначені класи предметної області. Семантичні мережі є винятковим випадком мережних моделей представлення знань. Формально мережні моделі задаються у вигляді:

$$H = < I, C_1, C_2, \dots, C_n, Q >,$$

де I – множину інформаційних елементів, що зберігаються у вузлах мережі;

$C_1, C_2, \dots, C_n$– типи зв'язків між інформаційними елементами;

Q – відображення, що встановлює відповідність між множиною типів зв'язків і множиною інформаційних елементів мережі.

Вперше семантичні мережі з'явилися в області машинної лінгвистики як засіб аналізу змісту (семантики) природної мови. У 1967 році М.Р. Куілліаном була розроблена програма, що фіксує зміст англійських слів, побудована за принципом тлумачного словника. У цій програмі деяке слово англійської мови визначається в термінах, що виражаються іншими словами. Кожне слово визначається множиною покажчиків на інші слова. Сукупність вказівників утворить мережу, "подорожуючи" у якій можна з'ясувати зміст того чи іншого слова. У семантичній мережі Куілліана вершини відповідають обумовленому поняттю і забезпечуються вказівниками на інші слова, що розкривають дане поняття. База знань організована у вигляді сторінок (площин), на кожній з який представляється граф, що визначає одне слово.

Мережні моделі представлення знань розрізняються між собою типами зв'язків (відносин), що використовуються. Якщо в мережі використовуються ієрархічні зв'язки (клас-підклас, рід-вид і т.п.), то мережа називається класифікованою. Якщо зв'язки між інформаційними елементами представляються функціональними відносинами, що дозволяють обчислювати значення одних інформаційних елементів за значеннями інших, то мережі називають функціональними (обчислювальними). Якщо в мережі допускаються зв'язки різного типу, то її називають семантичною мережею. Семантична мережа являє собою спрямований граф, в якому вершинам відповідають об'єкти (сутності) предметної області, а дугам – відносини, у яких знаходяться ці об'єкти. Вивід в семантичних мережах може виконуватися на основі алгоритмів співставлення, шляхом виглядіlення підграфів з визначеними властивостями. До переваг семантичних мереж відносять: велику виразну здатність, наочність графічного представлення, схожість структури мережі до семантичної структури фраз природної мови. Недоліком представлення знань у вигляді семантичних мереж є відсутність єдиної термінології. Дана модель представлення знань знаходить різне втілення в різних дослідників.

Фреймові моделі представлення знань використовують теорію організації пам’яті, розуміння і навчання, запропоновану М.Мінським. Фрейм (від англ. *frame* – рамка, каркас, кістяк) – структура даних, призначена для представлення стереотипних ситуацій. Фрейм складається зі слотів (*slot* – гніздо, щілинка, паз). Значенням слота можуть бути числа, вираження, тексти, програми, посилання на інші фрейми. Сукупності фреймів утворюють ієрархічні структури, побудовані на родовидових ознаках, що дозволяє успадковувати значення слотів. Така властивість фреймів забезпечує ощадливе розміщення бази знань у пам’яті. Крім цього, значення слотів можуть обчислюватися за допомогою різних процедур, тобто фрейми комбінують у собі декларативні і процедурні представлення знань. Фреймові моделі можна розуміти як мережні моделі уявлення знань, коли фрагмент мережі представляється фреймом з відповідними слотами і значеннями. З фреймовими моделями зв’язані моделі представлення знань на основі сценаріїв і об’єктів. Знання, якими оперує людина, часто мають якісну природу, характеризуються неповнотою, неточністю, нечіткістю.

Проблема отримання знань з об’єктів природньої мови, їх представлення та обробка відносяться до області штучного інтелекту. У загальній проблемі аналізу та розуміння об’єктів природньої мови, у всьому різноманітті робіт, методів, методологій та термінології явно проглядаються наступні завдання:

– синтактико-семантичний аналіз об’єктів природньої мови;
– аналіз об’єктів природньої мови з метою вибору підходящого формально-логічного представлення в деякій підсистемі логіки та подання в пам’яті комп’ютера і подальшої ефективної обробки;
– розробка методів добування знань з об’єктів природньої мови.

Проблема аналізу об’єктів природньої мови пов’язана з операційною і денотаційною семантикою мови. У системах штучного інтелекту і, зокрема, в системах аналізу текстів природньої мови переважає операційна семантика, оскільки очевидно переважання інтересу до засобів представлення знань (як?), а не до їх змісту (що?), яке підлягає формалізації [20].

Мови подання знань визначають конструкції, які підтримуються комп'ютером і забезпечують ефективність організації процедур, і як наслідок – максимальну уніфікацію мовних засобів. На інформаційно-логічному рівні переслідується зовсім інша мета – забезпечення відображення змісту об'єктів реального світу в мовних конструкціях, що узгоджуються зі сприйняттям, властивим людині. Для користувача такої системи (схеми) комп'ютерне подання незручне, оскільки він спілкується (або, принаймні, повинен) з системою на природній мові. Проте, для системного аналітика, що використовує мову формалізації для побудови моделі предметної області і створює для цього орієнтовані на цю предметну область словники понятійних одиниць і на їх основі – інтелектуальні системи, наповнені конкретними знаннями, комп'ютерне представлення грає ключову роль. Аналітику доводиться мати справу з конкретною множиною термінів та конструкцій природньої мови, саме їм давати тлумачення в термінах мови формалізації, встановлювати між ними інформаційно значимі зв'язки. Тому, в разі розбіжності мови формалізації зі складом мислення фахівця, реалізація системи обробки даних у конкретному смисловому матеріалі може стати занадто складною або взагалі нерозв'язаною проблемою. Найбільш прийнятним на сьогоднішній день є мови логічного типу, в яких поняття «зв'язок за змістом» формалізується з максимальною повнотою. Одним з головних питань при цьому є питання про рівень абстрактності зафіксованих категорій логічної мови. Якщо нас цікавить лише загальна схема логічного слідування, то досить розглядати лише числення предикатів першого порядку з правилом резолюцій в якості правила виводу, не конкретизуючи нічого, крім списку аксіом і логічних констант. Однак при більш багатих змістом предметних областях повинно бути вирішене питання про типи термових констант і вигляді нелогічних аксіом (власних аксіом), що використовуються при формалізації знань про емпіричні об'єкти. Викладене свідчить про актуальність проблеми розробки мови логічного типу для представлення знань, що містяться в тексті. Вочевидь, що ця мова має бути розширюваною у бік здатності більш повного опису об'єктів, здатності робити передбачення (гіпотези), які враховують контекст [20].

1.2. Огляд систем здійснення автоматичного синтаксичного аналізу

Існує два підходи щодо дослідження процесу автоматичного синтаксичного аналізу. Мову можна уявити у вигляді кібернетичної системи, на вході якої є сума речень, а на виході - класи мовних одиниць і правила їх сполучуваності. Або навпаки: на вході системи - породжувані цією системою речення. Ці два підходи пізнання структури мови лежать в основі побудови синтаксичних моделей (індуктивних та дедуктивних) та розробляються у методиці структурних лінгвістичних досліджень.

Метод моделювання змусив переглянути існуючі синтаксичні теорії, а також точніше визначити основні поняття синтаксису, розробляти нові методи його вивчення. Заново були поставлені основні проблеми синтаксису: проблема його об'єкта, співвідношення із семантикою й морфологією; проблема слова, групи, фрази як синтаксичних одиниць, а також проблема основних понять синтаксису: відношення (зв'язку), функції, структури, формальних показників.

Чимало цікавих ідей, використаних для розробки автоматичного синтаксичного аналізу, висловили представники дескриптивної школи структурної лінгвістики: із суми спостережень над текстом лінгвіст здобуває первісну уяву про спосіб організації тексту й у вигляді чітких процедур - правил алгоритму - повідомляє автомату свої дії, а потім за його допомогою одержує на більшому матеріалі дані, що цікавлять дослідника.

У роботах з автоматичного синтаксичного аналізу прийнято два способи опису синтаксичної структури:

– опис за безпосередніми складниками;

– опис за допомогою дерев залежностей, які називають деревами синтаксичного підпорядкування.

Ці два способи допомагають описати синтаксичну структуру на двох рівнях:

– за допомогою безпосередніх складників описуються в явному вигляді словосполучення, але не розпізнається "хазяїн" і "слуга";

– дерева залежностей дають можливість розрізнити характер зв'язків між словами. Якщо в результаті роботи алгоритму автоматичного синтаксичного аналізу встановлюються зв'язки, які більш-менш відповідають інтуїтивним уявленням носіїв мови, значить синтаксична структура речення "визначена" правильно.

Завдання автоматичного синтаксичного аналізу полягає у тому, щоб, використовуючи морфологічну інформацію про словоформи, одержану на попередньому морфологічному етапі, побудувати синтаксичну структуру вхідного речення. Об'єктом аналізу є речення, яке до моменту синтаксичного аналізу подається у вигляді інформаційних ланцюжків до словоформ. Виконувати синтаксичний аналіз повинен алгоритм синтаксичного аналізу, тобто інструкція, яка складається зі стандартних елементів, що здійснюють певну послідовність операцій над словоформами. Результатом аналізу є синтаксична структура речення, представлена як сукупність даних про синтаксичні зв'язки між його одиницями.

На сьогоднішній день існує велика кількість систем здійснення автоматичного синтаксичного аналізу. Нижче розглянуті деякі з них.

Система «*ABBYY Compreno*» дозволяє отримати необхідну для користувача інформацію з масивів даних. Унікальними можливостями даної системи є: визначення суті багатозначних слів, відновлення пропущених слів, визначення зв’язків між об’єктом, вираженим іменником, і заміняє його займенником.

Сфери застосування *ABBYY Compreno*:

– Видобування даних з неструктурованих документів(оптимізація вводу документів, підтримка приняття рішень з рахунок збору і аналізу значущих фактів і подій, зниження проектних ризиків шляхом аналізу документації).
– Інтелектуальний пошук в інформаційній системі(пошук за змістом, а не за ключовими словами для повного збору даних, швидкий доступ до документу за рахунок пошуку по його атрибутам).
– Класифікація потоку документів(оптимізація процесу обробки вхідної документації, пошук по категоріям).

Переваги даної системи:

– Швидкість і висока точність розпізнавання.
– Підтримка більшості мов світу(розпізнає документи, написані однією або декількома зі 190 мов, у тому числі арабською, в'єтнамською, корейською, китайською, японською, тайською та мовою іврит. У програму вбудовано функцію автоматичного визначення мови документа).
– Простий і зрозумілий інтерфейс
– Швидке цитування
– Розпізнавання сфотографованих документів
– Збереження документів у різних форматах, а також надсилання в хмарні сховища.

Приклад використання *ABBYY Compreno* зображено на рис. 1.7. на якому показано формування дерева на основі вхідних даних – текстової інформації природньою мовою.

Рис. 1.7. Приклад роботи *ABBYY Compreno*

Система «ВААЛ», робота над якою ведеться з 1992 року, дозволяє прогнозувати ефект неусвідомлюваного впливу текстів на масову аудиторію, аналізувати тексти з точки зору такого впливу, складати тексти із заданим вектором впливу, виявляти особистісно-психологічні якості авторів тексту, проводити поглиблений контент-аналіз текстів і робити багато іншого.

Області можливого застосування:

– Складання текстів виступів з наперед заданими характеристиками впливу на потенційну аудиторію.

– Активне формування емоційного ставлення до політичного діяча з боку різних соціальних груп.
– Складання емоційно забарвлених рекламних статей.
– Пошук найбільш вдалих назв і торгових марок.
– Психо та гіпнотерапія.
– Неявне психологічне тестування і експрес-діагностика.
– Створення легких в засвоєнні навчальних матеріалів.
– Наукові дослідження в області психолінгвістики і суміжних з нею дисциплінах.
– Журналістика та інші сфери діяльності, які використовують в якості інструменту «СЛОВО» .
– Соціологічні та соціолінгвістичні дослідження.
– Інформаційні війни.
– Контент-аналіз текстів.
– Моніторинг ЗМІ.

Система дозволяє:

– Оцінювати неусвідомлюване емоційний вплив фонетичної структури текстів та окремих слів на підсвідомість людини.
– Генерувати слова із заданими фоносемантичними характеристиками.
– Оцінювати неусвідомлюване емоційний вплив фонетичної структури текстів на підсвідомість людини.
– Задавати характеристики бажаного впливу і цілеспрямовано коригувати тексти за обраними параметрами з метою досягнення необхідного ефекту впливу.
– Оцінювати звуко-колірні характеристики слів і текстів.
– Виробляти словниковий аналіз текстів.
– Здійснювати повноцінний контент-аналіз тексту по великому числу спеціально складених вбудованих категорій і категорій, що задаються самим користувачем.

– Виробляти виділення тем, що порушувалися в текстах, і здійснювати на основі цього автоматичну категоризацію.
– Виробляти емоційно-лексичний аналіз текстів.
– Налаштовуватися на різні соціальні та професійні групи людей, які можуть бути виділені по використовуваної ними лексиці.
– Виробляти вторинний аналіз даних шляхом їх візуалізації, факторного і кореляційного аналізу.

Система реалізована у вигляді набору *DLL*-бібліотек, які підключаються до найбільш популярного текстового процесору *Word for Windows*. Просто в головному меню з'являється новий пункт. Такий спосіб реалізації дозволяє зберегти для користувача звично зручне середовище створення документів і максимально полегшує освоєння системи ВААЛ.

Результат фоносемантичної оцінки тексту за допомогою системи ВААЛ подано на рис. 1.8.

Рис. 1.8. Приклад роботи ВААЛ

Система «АОТ» базуються на багаторівневому поданні Зрозумілі мови, яке, в свою чергу, було запозичене у системи ФРАП. Компоненти, що становлять мовну

модель - лінгвістичні процесори, які один за одним обробляють вхідний текст. Вхід одного процесора є виходом іншого. Виділяються наступні компоненти:

- Графематичний аналіз. Виділення слів, цифрових комплексів, формул і т.д.
- Морфологічний аналіз. Побудова морфологічної інтерпретації слів вхідного тексту.
- Синтаксичний аналіз. Побудова дерева залежностей всього речення.
- Семантичний аналіз. Побудова семантичного графа тексту.

Для кожного рівня розроблялася своя мова вираження. Мова вираження складається з констант і правил їх комбінування. На графематіческом рівні константами були графематичні дескриптори (ЛЕ – лексема, ЦК – цифровий комплекс і т.д.) на морфологічному рівні – грамеми (рд – родовий відмінок, мн - множина). На синтаксичному – назва відносин і груп (ПОДЛ – відношення між підметом і присудком, ПГ - прийменникова група). На семантичному – семантичні категорії і відношення.

З кожного рівня вираження можна зробити перехід до такого ж подання іншою природною мовою, що дозволяє здійснювати переклад, навіть якщо семантичний аналізатор не зміг обробити текст. Основою для побудови рівнів служили результати роботи попередніх етапів, але наступні аналізатори також могли поліпшити подання попередніх. Текст оброблюється за технологією зображеною на рис. 1.9.

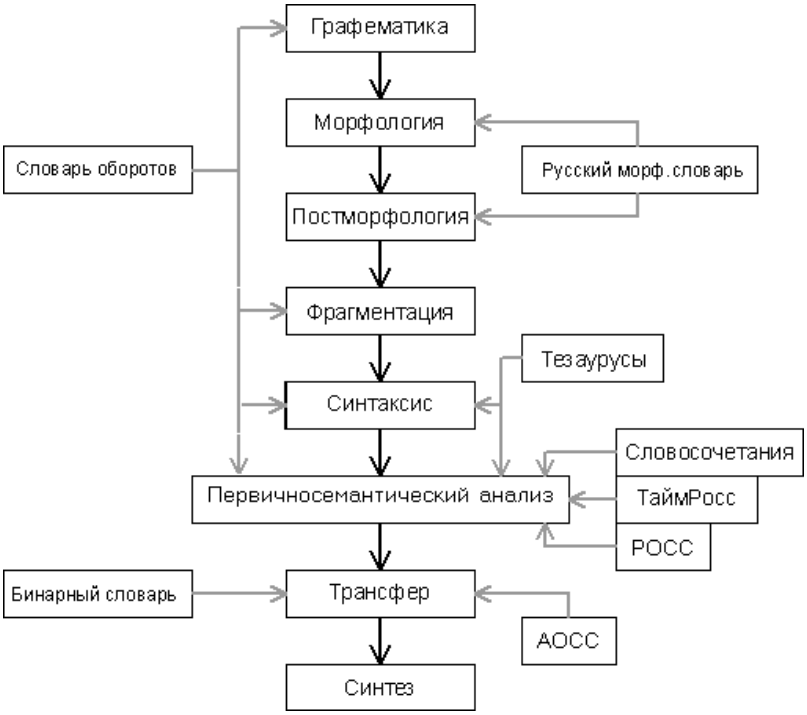

Рис. 1.9. Процес обробки тексту

Система АОТ може здійснювати наступне:

– Морфологія. Користувач вводить російську, англійську чи німецьку словоформу і отримує нормальну форму і морфологічні атрибути або, за бажанням, всю парадигму слова.
– Синтаксис. Користувач вводить речення російською мовою і отримує результати аналізу поверхневого синтаксису у вигляді системи складових.
– Граф. По одному реченню російською мовою будується поверхнево семантичний граф.
– Переклад. Переклад з російської мови на англійську. Використовуються результати графематичного, морфологічного і синтаксичного аналізаторів.
– Пошук по масиву. Лінгвістичний пошук по розміченому морфологічним аналізатором масиву. Можна шукати по частині мови і по морфологічним характеристикам. Розмічений корпус складається з 680 млн слів.
– Пошук по біграмам . Пошук по лемним біграмам (54 млн.). Леми отримані за допомогою програми *Trigram*.

Приклад роботи системи АОТ зображено а рис. 1.10.

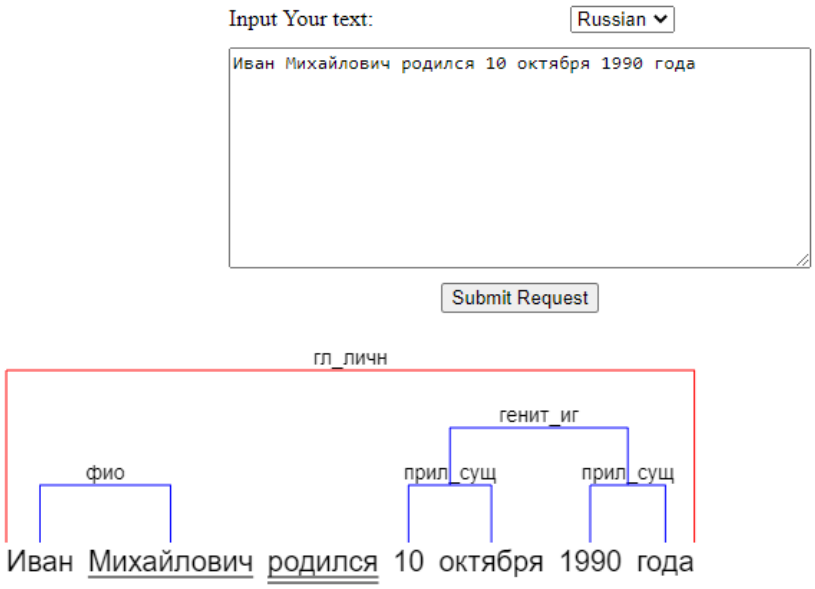

Рис. 1.10. Синтаксичний аналіз речення системою АОТ

Синтаксичний аналізатор «*Cognitive Dwarf 2.0*» здійснює частковий синтаксичний розбір пропозицій, вирішуючи практичну задачу побудови кращої синтаксичної інтерпретації для довільної вхідний ланцюжка слів. Розрізняють два способи подання пропозицій у вигляді ієрархічної структури граматичних конструкцій: дерево безпосередніх складових і дерево залежностей.

У дереві безпосередніх складових, складові лінгвістичні конструкції утворюються з набору більш простих лінійно непересічних відрізків, які називаються безпосередніми складовими цієї конструкції. При цьому в дереві розрізняють термінальні вузли, що відповідають окремим словам, і нетермінальні — відповідні конструкціям. Існують загальноприйняті найменування нетермінальних вузлів(наприклад *S* — речення, *NP* — іменна група, *VP* — дієслівна група).

Дерево залежностей будується з бінарних відносин безпосереднього підпорядкування залежного слова головному. При цьому не вводяться нетермінальні вузли, а представником конструкції завжди є її головне слово. Дерево безпосередніх складових можна перетворити в дерево залежностей, і навпаки.

Синтаксичний аналізатор *Cognitive Dwarf* дотримується синтаксичної теорії, при якій вершиною пропозиції є присудок, а структура пропозиції багато в чому визначається можливим набором актантів даного предиката (суб'ктом, об'ктом та

ін.). Є особливості в розборі конструкцій, де предикат управляє іншими дієслівними формами(например інфінітивом), в розборі прийменникових груп, а також в розборі конструкцій з однорідними членами.

Результат обробки текстової інформації природньою мовою системою *Cognitive Dwarf* зображено на рис. 1.11.

result : таблица

word_id	form	norm	parent_id	link	attr
1	*Top*	*Top*	1	_	/
2	тося	тося	6	subj	/nn/sg/msc/anm/nom/trd/heu/
3	договорился	договориться	6	auxs	/vrb/sg/msc/fin/fst/sec/trd/pst/ind/act/
4	с	с	5	prep	/prp/ins/
5	хубертом	хуберт	6	prepnp	/nn/sg/msc/anm/ins/trd/heu/
6	выкупить	выкупить	1	sent	/vrb/inf/act/
7	свое	свой	8	adj	/prn/sg/neu/acc/trd/
8	кольцо	кольцо	6	acc	/nn/sg/neu/acc/trd/
9	,	,	10	misc	/pnt/
10	и	и	16	conj	/cnj/
11	деньги	деньги	16	acc	/nn/pl/fem/acc/trd/
12	для	для	13	prep	/prp/gen/
13	этого	этот	16	prepnp	/prn/sg/msc/gen/trd/
14	должен	должен	16	auxs	/adj/sg/msc/fst/sec/trd/sht/
15	буду	быть	14	auxs	/vrb/sg/fem/msc/neu/fin/fst/prs/ind/act/
16	дать	дать	1	sent	/vrb/inf/act/
17	я	я	16	subj	/prn/sg/fem/msc/neu/nom/fst/
18	.	.	1	misc	/pnt/

Запись: 1 из 18

Рис. 1.11. Приклад роботи системи *Cognitive Dwarf*

Оскільки вищезгадані системи не працюють з текстовою інформацією українською мовою було прийняте рішення про створення власної технології автоматизованої побудови логіко-лінгвістичних, яка зможе в майбутньому удосконалюватися і використовуватися для автоматичного аналізу тексту, оскільки міститиме в собі морфологічні та синтаксичні характеристики.

1.3. Висновки до розділу

Для вилучення знань з текстової інформації замало лише її прочитати, оскільки знання це не текст, а переосмислений і засвоєний людиною чи інформаційною системою його вміст. Тому постає проблема у важкості реалізації того процесу осмислення машиною зчитаної інформації і виділення основних чи необхідних, в певний момент, аспектів зчитаного.

Для представлення знань створюють моделі знань, системи подання знань і бази знань. Вони повинні описувати знання про предметну область за допомогою мови представлення знань, зберігати їх, об'єднуватися таким чином створюючи нові знання, здійснювати інтерфейс між користувачем та знаннями.

Для екстракції знань з текстової інформації слід спочатку здійснити обробку текстової інформації а саме: збір даних, очистка даних, вибір представлення даних, класифікацію, перевірку на помилки, вирахувати семантичне значення слів, визначити синтаксичну інформацію.

Знання характеризуються рядом властивостей: внутрішня інтерпретація, структурованість, зв'язність, семантична метрика, активність.

Перехід до формування нових знань більш високого рівня абстракції можливо за використання методами штучного інтелекту, якщо взяти за основу модель людського мислення, котра відповідає рівню свідомості спостерігача. На цьому рівні знання організовані в складну онтологічну структуру з повним набором зв'язків між концептами. Зі сказаного випливає, що в даний час автоматична побудова бази знань для широких предметних областей представляє складну наукову проблему.

Моделі представлення знань можна умовно розділити на декларативні і процедурні. Моделі, застосовувані на практиці, являють собою комбінацію декларативних і процедурних представлень. Найбільш розповсюдженими є: логічні моделі, продукційні моделі, мережні моделі та фреймові моделі.

Розглянуто принцип роботи систем здійснення автоматичного синтексичного аналізу таких як: *ABBYY Compreno*, ВААЛ, АОТ, *Cognitive Dwarf 2.0*.

На основі вищесказаного було зроблено висновок, що тільки поєднання морфологічного, синтаксично-семантичного та частотного аналізу може дати необхідний результат в проблемі екстракції знань з текстової інформації природньої мови і подальшій їхній обробці.

РОЗДІЛ 2
МЕТОД АВТОМАТИЗОВАНОЇ ПОБУДОВИ ЛОГІКО -ЛІНГВІСТИЧНИХ МОДЕЛЕЙ

2.1. Логіко-лінгвістичні моделі представлення знань

Логіко-лінгвістична модель представлення знань дозволяє представити довільне речення природної мови у вигляді кон'юнкції атомарних предикатів, кожен із яких описує неподільну змістовну частину речення:

$$L^S = \bigwedge_{p \in P^S} \bigwedge_{h \in H_p^S} L_p^S(h), \tag{2.1}$$

$$L_p^S(h) = \bigwedge_{x \in X_p^S(h)} \bigwedge_{g \in G_p^S(x,h)} L_p^S(x,g,h), \tag{2.2}$$

$$L_p^S(x,g,h) = \bigwedge_{y \in Y_p^S(x,g,h)} \bigwedge_{q \in Q_p^S(x,g,y,h)} L_p^S(x,g,y,q,h), \tag{2.3}$$

$$L_p^S(x,g,y,q,h) = \bigwedge_{z \in Z_p^S(x,g,y,q,h)} \bigwedge_{r \in R_p^S(x,g,y,q,z,h)} L_p^S(x,g,y,q,z,r,h), \tag{2.4}$$

де S – речення природної мови;

p – відношення, що пов'язує суб'єкти, об'єкти та предмети відношень у реченні S, $p \in P^S$ належить множині відношень, що входять до речення S;

h – характеристика p-го відношення речення S, $h \in H_p^S$ (множина характеристик p-го відношення у реченні S);

$L_p^S(h)$ – предикат (предикативний вираз), який описує p-е відношення з h-ю характеристикою і пов'язує суб'єкти, об'єкти та предмети відношення p у реченні S;

x – суб'єкт речення S;

$x \in X_p^S(h)$ – множина суб'єктів, що пов'язані з об'єктами речення S p-м відношенням, яке володіє h-ю характеристикою;

g – характеристика суб'єкта x речення S, $g \in G_p^S(x,h)$ (множина характеристик суб'єкта $x \in X_p^S(h)$);

$L_p^S(x,g,h)$ – предикат (предикативний вираз), який описує p -е відношення з h характеристикою між суб'єктом $x \in X_p^S(h)$ з характеристикою $g \in G_p^S(x,h)$, об'єктами та предметами відношення p у реченні S;

y – об'єкт речення S;

$y \in Y_p^S(x,g,h)$ – множина об'єктів, що пов'язані з суб'єктами речення S p – м відношенням, яке володіє h -ю характеристикою;

q – характеристика об'єкта y речення S, $q \in Q_p^S(x,g,y,h)$ – множина характеристик об'єкта $y \in Y_p^S(x,g,h)$;

$L_p^S(x,g,y,q,h)$ – предикат (предикативний вираз), який описує p-е відношення з h -ю характеристикою між суб'єктом $x \in X_p^S(h)$ з характеристикою $g \in G_p^S(x,h)$ і об'єктом $y \in Y_p^S(x,g,h)$ з характеристикою $q \in Q_p^S(x,g,y,h)$ та предмети p -го відношення у реченні S;

z – предмет p -го відношення речення S;

$z \in Z_p^S(x,g,y,q,h)$ – множина предметів p -го відношення, яке володіє h-ю характеристикою, між суб'єктом $x \in X_p^S(h)$ з характеристикою $g \in G_p^S(x,h)$ та об'єктом $y \in Y_p^S(x,g,h)$ з характеристикою $q \in Q_p^S(x,g,y,h)$;

r – характеристика предмета p -го відношення речення S;

$r \in R_p^S(x,g,y,q,z,h)$ – множина характеристик предмета $z \in Z_p^S(x,g,y,q,h)$;

$L_p^S(x,g,y,q,z,r,h)$ – простий предикат, який описує частину речення, що має закінчений зміст та відображає у реченні S p -е відношення з h -ю характеристикою між суб'єктом $x \in X_p^S(h)$ з характеристикою $g \in G_p^S(x,h)$ і об'єктом $y \in Y_p^S(x,g,h)$ з характеристикою $q \in Q_p^S(x,g,y,h)$, предмет якого $z \in Z_p^S(x,g,y,q,h)$ володіє характеристикою $r \in R_p^S(x,g,y,q,z,h)$ [11].

Логіко-лінгвістична модель L^S речення S відображається сукупністю формул (2.1) – (2.4) та формально представляє собою послідовність восьми кон'юнкцій, що входять до цих формул. Перехід від загальної формули (2.1) до предикату $L_p^S(x, g, y, q, z, r, h)$ є декомпозицією проблеми формального опису довільного речення природної мови та відображає системний підхід до її вирішення.

Побудова логіко-лінгвістичних моделей речень природної мови є підґрунтям для створення загальної форми логіко-лінгвістичної моделі електронного текстового документу та розробки математичного апарату для здійснення порівняльного аналізу електронних текстових документів за змістом.

Формалізація опису різноманітних за структурою речень природної мови спонукає до здійснення класифікації окремих форм уніфікованої логіко-лінгвістичної моделі (2.1) – (2.4). Усі окремі форми адаптовані до певного типу речення природної мови.

У дипломній роботі передбачається побудова логіко-лінгвістичних моделей лише для простих речень природної мови:

– Тип 1. Описує просте речення природної мови, що містить лише одну частину, яка має закінчений зміст:

$$L^S = L_p^S(x, g, y, q, z, r, h). \quad (2.5)$$

Логіко-лінгвістична модель(2.5) для конкретних значень елементів множин, що формують просте, не ускладнене речення природної мов S_i, має вигляд:

$$L^{S_i} = p_i(x_i, g_i,\ y_i, q_i, z_i, r_i, h_i). \quad (2.6)$$

Наприклад, для простого речення української мови *«Прекрасні яблука регулярно подають до обіднього столу»* буде побудована логіко-лінгвістична модель типу 1 і буде мати вигляд:

$L^{S_1} = p_1(x_1, g_1,\ y_1, q_1, z_1, r_1, h_1) = p_1(0,0,\ y_1, q_1, z_1, r_1, h_1)$,

L^{S_1} =*подають (0, 0, яблука, прекрасні, столу, обіднього, регулярно).*

Для простого речення російської мови *«Прекрастные яблоки регулярно подают к обеденному столу»* логіко-лінгвістична модель також буде типу 1 і матиме вигляд:

$L^{S_1} = p_1(x_1, g_1,\ y_1, q_1, z_1, r_1, h_1) = p_1(0,0,\ y_1, q_1, z_1, r_1, h_1)$,

L^{S_1} =*подают (0, 0, яблоки, прекрастные, столу, обеденному, подают).*

Для простого речення англійської мови *«New disciplines are regularly included in curricula»* логіко-лінгвістична модель також типу 1 і має вигляд:

$L^{S_1} = p_{11}\&p_{12}(x_1, g_1,\ y_1, q_1, z_1, r_1, h_1) = p_{11}\&p_{12}(0,0,\ y_1, q_1, z_1, 0, h_1)$,

L^{S_1} = *are&included (0, 0, disciplines, new, curricula, 0, regularly).*

– Тип 2. Описує просте речення з однорідними присудками, внаслідок чого кількість частин, що мають закінчений зміст, дорівнює потужності множини відношень:

$$L^S = \underset{p \in P^S}{\wedge} L_p^S(x, g, y, q, z, r, h). \quad (2.7)$$

Логіко-лінгвістична модель (2.7) конкретного речення S_i з двома однорідними присудками має вигляд:

$$L^{S_i} = p_{i1}(x_i, g_i,\ y_i, q_i, z_i, r_i, h_i)\&p_{i2}(x_i, g_i,\ y_i, q_i, z_i, r_i, h_i). \quad (2.8)$$

Наприклад, для речення української мови *«Свіжі яблука регулярно приносять та подають до обіднього столу»* буде побудована логіко-лінгвістична модель типу 2, що матиме вигляд:

$L^{S_1} = p_{11}(x_1, g_1,\ y_1, q_1, z_1, r_1, h_1)\&p_{12}(x_1, g_1,\ y_1, q_1, z_1, r_1, h_1) =$

$p_{11}(0,0,\ y_1, q_1, z_1, r_1, h_1)\&p_{12}(0,0,\ y_1, q_1, z_1, r_1, h_1)$,

L^{S_1} = *приносять (0, 0, яблука, свіжі, стоу, обіднього, регулярно)&*

подають (0, 0, яблука, свіжі, столу, обіднього, регулярно).

Для речення російської мови *« Свежые яблоки регулярно приносят и подают к обеденному столу»* буде побудована логіко-лінгвістична модель типу 2, що матиме вигляд:

$L^{S_1} = p_{11}(x_1, g_1,\ y_1, q_1, z_1, r_1, h_1)\&p_{12}(x_1, g_1,\ y_1, q_1, z_1, r_1, h_1) =$

$p_{11}(0,0,\ y_1, q_1, z_1, r_1, h_1)\&p_{12}(0,0,\ y_1, q_1, z_1, r_1, h_1)$,

L^{S_1} = *приносят (0, 0, яблоки, свежые, столу, обеденомк, регулярно)&*

подают (0, 0, дисциплины, новые, планы, учебные, регулярно).

Для речення англійської мови *«New disciplines are regularly considered and included in curricula»* буде побудована логіко-лінгвістична модель типу 2, що матиме вигляд:

$L^{S_1} = p_{11}\&p_{12}(x_1, g_1,\ y_1, q_1, z_1, r_1, h_1)\&p_{11}\&p_{13}(x_1, g_1,\ y_1, q_1, z_1, r_1, h_1) =$

$p_{11}\&p_{12}(0,0,\ y_1, q_1, z_1, 0, h_1)\&p_{11}\&p_{13}(0,0,\ y_1, q_1, z_1, 0, h_1),$

$L^{S_1} =$ *are&considered (0, 0, disciplines, new, curricula, 0, regularly)&*

are&included (0, 0, disciplines, new, curricula, 0, regularly).

– Тип 3. Описує просте речення з однорідними обставинами, внаслідок чого кількість частин, що мають закінчений зміст, дорівнює потужності множини характеристик відношення:

$$L^S = \bigwedge_{h \in H_p^S} L_p^S(x, g, y, q, z, r, h). \qquad (2.9)$$

Логіко-лінгвістична модель (2.9) конкретного речення S_i з двома однорідними обставинами має вигляд:

$$L^{S_i} = p_i(x_i, g_i,\ y_i, q_i, z_i, r_i, h_{i1})\&p_i(x_i, g_i,\ y_i, q_i, z_i, r_i, h_{i2}) \qquad (2.10)$$

Наприклад, для речення природної мови *«Свіжі яблука регулярно та швидко подають до обіднього столу»* буде побудована логіко-лінгвістична модель типу 3, що матиме вигляд:

$L^{S_1} = p_1(x_1, g_1,\ y_1, q_1, z_1, r_1, h_{11})\&p_1(x_1, g_1,\ y_1, q_1, z_1, r_1, h_{12}) =$

$p_1(0,0,\ y_1, q_1, z_1, r_1, h_{11})\&p_1(0,0,\ y_1, q_1, z_1, r_1, h_{12}),$

$L^{S_1} =$ *подають (0, 0, яблука, свіжі, столу, обіднього, регулярно)&*

подають (0, 0, , яблука, свіжі, столу, обіднього, швидко).

Для речення російської мови *« Свежые яблоки регулярно и быстро подают к обеденному столу»* буде побудована логіко-лінгвістична модель типу 3, що матиме вигляд:

$L^{S_1} = p_1(x_1, g_1,\ y_1, q_1, z_1, r_1, h_{11})\&p_1(x_1, g_1,\ y_1, q_1, z_1, r_1, h_{12}) =$

$p_1(0,0,\ y_1, q_1, z_1, r_1, h_{11})\&p_1(0,0,\ y_1, q_1, z_1, r_1, h_{12}),$

$L^{S_1} =$ *подают (0, 0, яблоки, свежые, столу, обеденному, регулярно) &*

подают (0, 0, яблоки, свежые, столу, обеденному, быстро).

Для речення англійської мови *«New disciplines are regularly and scrupulously included in curricula»* буде побудована логіко-лінгвістична модель типу 3, що матиме вигляд:

$L^{S_1} = p_{11}\&p_{12}(x_1, g_1,\ y_1, q_1, z_1, r_1, h_{11})\&$

$p_{11}\&p_{12}(x_1, g_1,\ y_1, q_1, z_1, r_1, h_{12})p_{11}\&p_{12}(0,0,\ y_1, q_1, z_1, 0, h_{11})\&$

$p_{11}\&p_{12}(0,0,\ y_1, q_1, z_1, 0, h_{12}),$

$L^{S_1} =$ *are&included (0, 0, disciplines, new, curricula, 0, regularly)&*

are&included (0, 0, disciplines, new, curricula, 0, scrupulously).

– Тип 4. Описує просте речення з однорідними підметами, внаслідок чого кількість частин, що мають закінчений зміст, дорівнює потужності множини суб'єктів:

$$L^S = \underset{x \in X_p^S(h)}{\wedge} L_p^S(x, g, y, q, z, r, h). \qquad (2.11)$$

Логіко-лінгвістична модель (2.11) конкретного речення S_i з двома однорідними підметами має вигляд:

$$L^{S_i} = p_i(x_{i1}, g_i,\ y_i, q_i, z_i, r_i, h_i)\&p_i(x_{i2}, g_i,\ y_i, q_i, z_i, r_i, h_i). \qquad (2.12)$$

Наприклад, для речення української мови *«Сусіди та друзі регулярно приносять свіжі яблука до обіднього столу»* буде побудована логіко-лінгвістична модель типу 4 вигляду:

$L^{S_1} = p_1(x_{11}, g_1,\ y_1, q_1, z_1, r_1, h_1)\&p_1(x_{12}, g_1,\ y_1, q_1, z_1, r_1, h_1) =$

$p_1(x_{11}, g_1,\ y_1, q_1, z_1, r_1, h_1)\&p_1(x_{12}, 0,\ y_1, q_1, z_1, r_1, h_1),$

$L^{S_1} =$ *приносять(сусіди, 0, яблука, свіжі, столу, обіднього, регулярно)& приносять(друзі, 0, яблука, свіжі, столу, обіднього, регулярно).*

Для речення російської мови *«Свежые яблоки регулярно подают соседи и друзья к обеденному столу»* буде побудована логіко-лінгвістична модель типу 4, що матиме вигляд:

$L^{S_1} = p_1(x_{11}, g_1,\ y_1, q_1, z_1, r_1, h_1)\&p_1(x_{12}, g_1,\ y_1, q_1, z_1, r_1, h_1) =$

$p_1(x_{11}, g_1,\ y_1, q_1, z_1, r_1, h_1)\&p_1(x_{12}, 0,\ y_1, q_1, z_1, r_1, h_1),$

$L^{S_1} =$ *подаються (соседи, 0, яблоки, свежые, столу, обеденному, регулярно) & подаються (друзья, 0, яблоки, свежые, столу, обеденному, регулярно).*

Для речення англійської мови *«Leading teachers and methodologists regularly include new disciplines in curricula»* буде побудована логіко-лінгвістична модель типу 4, що матиме вигляд:

$L^{S_1} = p_1(x_{11}, g_1,\ y_1, q_1, z_1, r_1, h_1)\&p_1(x_{12}, g_1,\ y_1, q_1, z_1, r_1, h_1) =$

$p_1(x_{11}, g_1,\ y_1, q_1, z_1, 0, h_1)\&p_1(x_{12}, 0,\ y_1, q_1, z_1, 0, h_1),$

$L^{S_1} =$ *include (teachers, leading, disciplines, new, curricula, 0, regularly)& include (methodologists, 0, disciplines, new, curricula, 0, regularly).*

– Тип 5. Описує просте речення з однорідними означеннями, що характеризують підмет, внаслідок чого кількість частин, які мають закінчений зміст, дорівнює потужності множини характеристик суб'єкта відношення:

$$L^S = \underset{g \in G_p^S(x,h)}{\wedge} L_p^S(x, g, y, q, z, r, h). \qquad (2.13)$$

Логіко-лінгвістична модель (2.13) конкретного речення S_i з двома однорідними означеннями має вигляд:

$$L^{S_i} = p_i(x_i, g_{i1},\ y_i, q_i, z_i, r_i, h_i)\&p_i(x_i, g_{i2},\ y_i, q_i, z_i, r_i, h_i). \qquad (2.14)$$

Наприклад, для речення природної мови *«Свіжі яблука регулярно подають дабрі та чуйні сусіди до обіднього столу»* буде побудована логіко-лінгвістична модель типу 5, що матиме вигляд:

$L^{S_1} = p_1(x_1, g_{11},\ y_1, q_1, z_1, r_1, h_1)\&p_1(x_1, g_{12},\ y_1, q_1, z_1, r_1, h_1),$

$L^{S_1} =$ *подають(сусіди, чуйні, яблука, свіжі, столу, обіднього, регулярно)& подають(сусіди, добрі, яблука, свіжі, столу, обіднього, регулярно)*.

Для речення російської мови *«Свежые яблоки регулярно подают добрые и близкие соседи к обеденному столу»* буде побудована логіко-лінгвістична модель типу 5, що матиме вигляд:

$L^{S_1} = p_1(x_1, g_{11},\ y_1, q_1, z_1, r_1, h_1)\&p_1(x_1, g_{12},\ y_1, q_1, z_1, r_1, h_1),$

$L^{S_1} =$ *подают (соседи, добрые, яблоки, свежые, столу, обеденному, регулярно) & подают (соседи, близкие, яблоки, свежые, столу, обеденному, регулярно).*

Для речення англійської мови *«Leading teachers highly qualified regularly include new disciplines in curricula»* буде побудована логіко-лінгвістична модель типу 5, що матиме вигляд:

$L^{S_1} = p_1(x_1, g_{11},\ y_1, q_1, z_1, r_1, h_1)\&p_1(x_1, g_{12}\&g_{13},\ y_1, q_1, z_1, r_1, h_1) =$

$p_1(x_1, g_{11},\ y_1, q_1, z_1, 0, h_1)\&p_1(x_1, g_{12}\&g_{13},\ y_1, q_1, z_1, 0, h_1),$

$L^{S_1} =$ *include (teachers, leading, disciplines, new, curricula, 0, regularly)& include (teachers, highly&qualified, disciplines, new, curricula, 0, regularly).*

– Тип 6. Описує просте речення з однорідними додатками, внаслідок чого кількість частин, що мають закінчений зміст, дорівнює потужності множини об'єктів:

$$L^S = \overset{\wedge}{y \in Y_p^S}(x, g, h) L_p^S(x, g, y, q, z, r, h). \quad (2.15)$$

Логіко-лінгвістична модель (2.15) конкретного речення S_i з двома однорідними додатками має вигляд:

$$L^{S_i} = p_i(x_i, g_i,\ y_{i1}, q_i, z_i, r_i, h_i)\&p_i(x_i, g_i,\ y_{i2}, q_i, z_i, r_i, h_i). \quad (2.16)$$

Наприклад, для речення української мови *«Свіжі яблука та соковиті груші регулярно подають до обіднього столу»* буде побудована логіко-лінгвістична модель типу 6, що матиме вигляд:

$L^{S_1} = p_1(x_1, g_1,\ y_{11}, q_{11}, z_1, r_1, h_1)\&p_1(x_1, g_1,\ y_{12}, q_{12}, z_1, r_1, h_1) =$

$p_1(0,0,\ y_{11}, q_{11}, z_1, r_1, h_1)\&p_1(0,0,\ y_{12}, q_{12}, z_1, r_1, h_1),$

$L^{S_1} =$ *подають (0, 0, яблука, свіжі, столу, обіднього, регулярно)& подають (0, 0, груші, соковиті, столу, обіднього, регулярно).*

У розглянутому реченні потужність множини характеристик об'єкту також дорівнює двом, проте характеристики за своїм значенням не однакові, тому тип окремої форми логіко-лінгвістичної моделі – 6.

Для речення російської мови *«Свежые яблоки и сочние грушки регулярно подают к обеденному столу»* буде побудована логіко-лінгвістична модель типу 6, що матиме вигляд:

$L^{S_1} = p_1(x_1, g_1,\ y_{11}, q_{11}, z_1, r_1, h_1)\&p_1(x_1, g_1,\ y_{12}, q_{12}, z_1, r_1, h_1) =$

$p_1(0,0,\ y_{11}, q_{11}, z_1, r_1, h_1)\&p_1(0,0,\ y_{12}, q_{12}, z_1, r_1, h_1),$

L^{S_1} = *подают (0, 0, яблоки, свежые, столу, обеденому, регулярно)&*
подают (0, 0, грушки, сочние, столу, обеденому, регулярно).

Для речення англійської мови *«New disciplines and updated loans are regularly considered in curricula»* буде побудована логіко-лінгвістична модель типу 6, що матиме вигляд:

$L^{S_1} = p_{11}\&p_{12}(x_1, g_1,\ y_{11}, q_{11}, z_1, r_1, h_1)\&p_{11}\&$
$p_{12}(x_1, g_1,\ y_{12}, q_{12}, z_1, r_1, h_1) = p_{11}\&p_{12}(0,0,\ y_{11}, q_{11}, z_1, 0, h_1)\&$
$p_{11}\&p_{12}(0,0,\ y_{12}, q_{12}, z_1, 0, h_1),$

L^{S_1} = *are&considered (0, 0, disciplines, new, curricula, 0, regularly)&are &considered (0, 0, loans, updated, curricula, 0, regularly).*

– Тип 7. Описує просте речення з однорідними означеннями, що характеризують додаток, внаслідок чого кількість частин, які мають закінчений зміст, дорівнює потужності множини характеристик об'єкта відношення:

$$L^S = \bigwedge_{q \in Q_p^S(x,g,y,h)} L_p^S(x, g, y, q, z, r, h). \qquad (2.17)$$

Логіко-лінгвістична модель (2.17) конкретного речення S_i з двома однорідними означеннями має вигляд:

$$L^{S_i} = p_i(x_i, g_i,\ y_i, q_{i1}, z_i, r_i, h_i)\&p_i(x_i, g_i,\ y_i, q_{i2}, z_i, r_i, h_i). \qquad (2.18)$$

Наприклад, для речення природної мови *«Свіжі та соковиті яблука регулярно подають до обіднього столу»* буде побудована логіко-лінгвістична модель типу 7, що матиме вигляд:

$L^{S_1} = p_1(x_1, g_1,\ y_1, q_{11}, z_1, r_1, h_1)\&p_1(x_1, g_1,\ y_1, q_{12}, z_1, r_1, h_1) =$
$p_1(0,0,\ y_1, q_{11}, z_1, r_1, h_1)\&p_1(0,0,\ y_1, q_{12}, z_1, r_1, h_1),$

L^{S_1} = *подають (0, 0, яблука, свіжі, столу, обіднього, регулярно)&*
подають (0, 0, яблука, соковиті, столу, обіднього, регулярно).

Для речення російської мови *«Свежые и сочние яблоки регулярно подают к обеденному столу»* буде побудована логіко-лінгвістична модель типу 7, що матиме вигляд:

$L^{S_1} = p_1(x_1, g_1,\ y_1, q_{11}, z_1, r_1, h_1)\&p_1(x_1, g_1,\ y_1, q_{12}, z_1, r_1, h_1) =$

$p_1(0,0,\ y_1, q_{11}, z_1, r_1, h_1)\&p_1(0,0,\ y_1, q_{12}, z_1, r_1, h_1),$

$L^{S_1} =$ *подают (0, 0, яблоки, свежые, столу, обеденому, регулярно)&*

подают (0, 0, яблоки, сочние, столу, обеденому, регулярно).

Для речення англійської мови *«New modern disciplines are regularly included in curricula»* буде побудована логіко-лінгвістична модель типу 7, що матиме вигляд:

$L^{S_1} = p_{11}\&p_{12}(x_1, g_1,\ y_1, q_{11}, z_1, r_1, h_1)\&p_{11}\&p_{12}(x_1, g_1,\ y_1, q_{12}, z_1, r_1, h_1) =$

$p_{11}\&p_{12}(0,0,\ y_1, q_{11}, z_1, 0, h_1)\&p_{11}\&p_{12}(0,0,\ y_1, q_{12}, z_1, 0, h_1),$

$L^{S_1} =$ *are&included (0, 0, disciplines, new, curricula, 0, regularly)&*

are&included (0, 0, disciplines, modern, curricula, 0, regularly).

– Тип 8. Описує просте речення з однорідними додатками, внаслідок чого кількість частин, що мають закінчений зміст, дорівнює потужності множини предметів відношення:

$$L^S = \overset{\wedge}{z \in Z_p^S}(x, g, y, q, h) L_p^S(x, g, y, q, z, r, h). \qquad (2.19)$$

Логіко-лінгвістична модель (2.19) конкретного речення S_i з двома однорідними додатками має вигляд:

$$L^{S_i} = p_i(x_i, g_i,\ y_i, q_i, z_{i1}, r_i, h_i)\&p_i(x_i, g_i,\ y_i, q_i, z_{i2}, r_i, h_i). \qquad (2.20)$$

Наприклад, для речення природної мови *«Свіжі яблука регулярно подають до обідніх столу та стільців»* буде побудована логіко-лінгвістична модель типу 8, що матиме вигляд:

$L^{S_1} = p_1(x_1, g_1,\ y_1, q_1, z_{11}, r_1, h_1)\&p_1(x_1, g_1,\ y_1, q_1, z_{12}, r_1, h_1) =$

$p_1(0,0,\ y_1, q_1, z_{11}, r_1, h_1)\&p_1(0,0,\ y_1, q_1, z_{12}, r_1, h_1),$

$L^{S_1} =$ *подають (0, 0, яблука, свіжі, столу, обідніх, регулярно)&*

подають (0, 0, яблука, свіжі, стільців, обідніх, регулярно).

Для речення російської мови *«Свежые яблоки регулярно подают к обеденному столу и стульям»* буде побудована логіко-лінгвістична модель типу 8, що матиме вигляд:

$L^{S_1} = p_1(x_1, g_1,\ y_1, q_1, z_{11}, r_1, h_1)\&p_1(x_1, g_1,\ y_1, q_1, z_{12}, r_1, h_1) =$
$p_1(0,0,\ y_1, q_1, z_{11}, r_1, h_1)\&p_1(0,0,\ y_1, q_1, z_{12}, r_1, h_1),$
$L^{S_1} =$ *подают (0, 0, яблоки, свежые, столу, обеденому, регулярно)&*
подают (0, 0, яблоки, свежые, стульям, обеденому, регулярно).

Для речення англійської мови *«New disciplines are regularly included in curricula and educational programs»* буде побудована логіко-лінгвістична модель типу 8, що матиме вигляд:

$L^{S_1} = p_{11}\&p_{12}(x_1, g_1,\ y_1, q_1, z_{11}, r_1, h_1)\&p_{11}\&p_{12}(x_1, g_1,\ y_1, q_1, z_{12}, r_1, h_1) =$
$p_{11}\&p_{12}(0,0,\ y_1, q_1, z_{11}, 0, h_1)\&p_{11}\&p_{12}(0,0,\ y_1, q_1, z_{12}, r_1, h_1),$
$L^{S_1} =$ *are&included (0, 0, disciplines, new, curricula, 0, regularly)&*
are&included (0, 0, disciplines, new, programs, educational, regularly).

2.2. Алгоритм побудови логіко-лінгвістичної моделі представлення знань

В математиці абсолютно точно встановлено, що зміст висловлювання – це предикат, що виражається цим висловлюванням, аргументами такого предикату є змінні, які присутні у даному висловлюванні.

Отже, текст на природній мові можна ототожнити з математичною формулою, записаною за певними правилами, а змістом речення природної мови є деякий предикат. Тоді в математиці, до якої належить формула, повинні бути присутні такі елементи, як змінні, операції над змінними, операції над операціями. Послідовність операцій над змінними та операціями буде записана у вигляді формули.

Структура речення виражається структурою формули предиката. Зміст речення виражається функцією, що реалізується за допомогою предикату. Два тексти різної структури зазвичай мають різну структуру у формулах предикатів, однак, якщо ці предикати виражають тотожно рівні функції, то і тексти виражають однаковий зміст. Тотожність функцій двох різних предикатів означає, що за будь-якої ситуації ці два предикати мають рівні значення. Речення складаються із слів, а слова з літер, тому

з’являється можливість оперувати даними з різних лінгвістичних рівнів, використовуючи один математичний апарат. За допомогою алгебри кінцевих предикатів довільного порядку можна побудувати моделі словотворення і словозміни будь-якої мови, що представляють собою синтаксичні моделі речень природної мови. Засобом формалізації текстової інформації, тобто математичною формулою за допомогою якої можна представити речення природної мови і вилучити з неї зміст, являється логіко-лінгвістична модель, структура формули якої базується на синтаксичній структурі речення. Саме на твердженні «синтаксис речення – це структура формули предикату, а функція, що реалізується предикатом – зміст цього речення» ґрунтується ідея методу автоматизованого формування логіко-лінгвістичної моделі текстової інформації.

Метод автоматизованого формування логіко-лінгвістичних моделей полягає в автоматизації процесу перетворення синтаксичних конструкцій (речень) природної мови в формули логіки предикатів першого порядку. Метод дозволяє вилучити з вхідної текстової інформації знання, формалізовані у відповідності з певними структурними правилами, і які комп’ютер зможе автономно використовувати при вирішенні задач за наперед вибраними алгоритмами, такими, як логічний вивід.

Метод автоматизованого формування логіко-лінгвістичних моделей включає в себе декілька етапів, кожен з яких представляє собою складний механізм роботи формальної системи, а її елементи відіграють важливу роль для вилучення знань із тестової інформації. В основу методу покладено відповідність між формулами логіки предикатів та концептами, що належать реальному світу. Ототожнивши вирази логіки предикатів зі словами природної мови згідно до синтаксичних правил природньої мови, можна засобами мови предикатів відображати зміст тексту. Механізмом для здійснення процесу перетворення речення в формули логіки предикатів служить автоматизований синтаксичний аналізатор, на основі результатів якого формується логіко-лінгвістична модель.

Проведення паралелі між структурою речення та структурою логічного висловлювання, визначення правил формування логіко лінгвістичну модель на основі синтаксичної структури речення, відображення в наборі аксіом, що складають логіко-

лінгвістичну модель, всіх зв’язків між словоформами, дозволяє надалі аналізувати логіко лінгвістичну модель з метою відшукання протиріч у різноманітних текстах. На рисунку 2.1 зображені етапи формування логіко-лінгвістичних моделей текстової інформації(рис. 2.1).

Першим етапом методу є розбиття вхідної текстової інформації на словоформи і ототожнення їх з елементами формальної системи, що дозволить розглядати речення як систему, взаємозв’язаними елементами якої є слова та словосполучення, які фактично представляють собою набір символів, розділених пробілами та знаками пунктуації. Розбиття тексту на словоформи та ідентифікація його з формальною системи надасть можливість розглядати речення як систему взаємозв’язаних елементів, правила взаємодії яких служать основою формування логіко-лінгвістичної моделі, а отже визначають зміст введеної текстової інформації.

Для того, щоб прослідкувати зв’язки між елементами формальної системи, необхідно володіти інформацією про властивості цих елементів, що і забезпечує етап концептуалізації. На даному кроці здійснюється морфологічний аналіз слів речення, тобто визначається, до якої частини мови належить те чи інше слово і якими граматичними значеннями воно володіє. Морфологічний аналіз являється підготовчим етапом перед роботою механізму синтаксичного аналізатору – засобу визначення структури речення. Для визначення морфологічної та синтаксичної будови речення використовується база даних, що містить граматичний та морфологічний словники флективної мови, представлені у вигляді реляційних таблиць, та базу знань, яка інтерпретується як набір аксіом формальної системи і містить правила визначення синтаксичних ролей слів, правила пунктуації та правила формування логіко-лінгвістичних моделей.

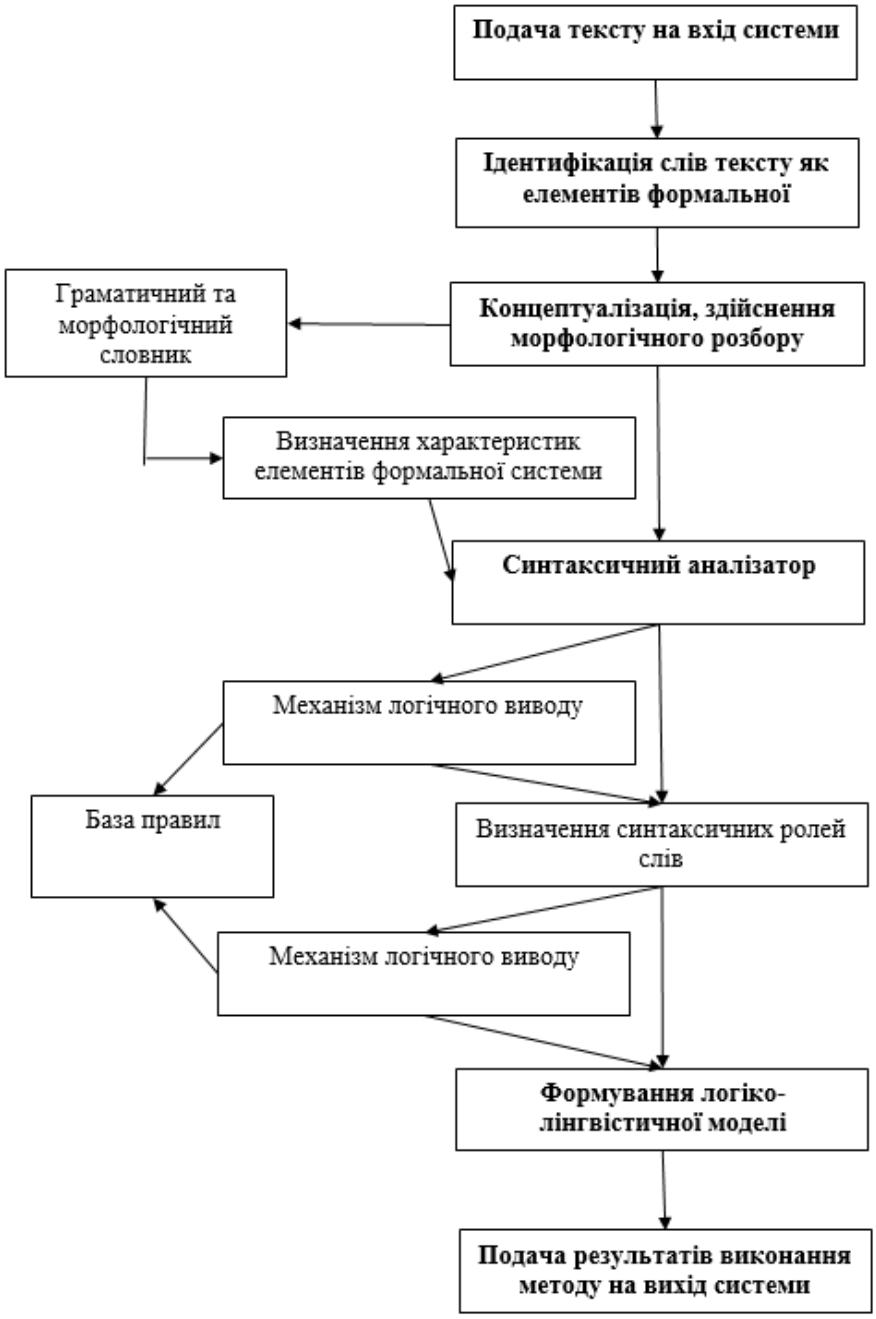

Рис. 2.1. Етапи методу формування логіко-лінгвістичних моделей

Логіко-лінгвістична модель являється засобом відображення відповідності між поняттями, що описуються термінами природної мови, та предикатними формулами. Ця відповідність базується на правилах формування логіко-лінгвістичних моделей згідно тих синтаксичних ролей та синтаксичних правил флективної мови, які були визначені на попередніх етапах методу.

За допомогою механізму логічного виводу бази знань формальної системи в логіко-лінгвістичну модель підставляються слова вхідної текстової інформації

відповідно до того, яку синтаксичну роль виконує дане слово в реченні (дія, об'єкт, суб'єкт, обставина, характеристика). Таким чином шляхом аналізу слів речення, знаків пунктуації, що до нього входять, та визначення його синтаксичної структури, визначається зміст речення, тобто те, про що у ньому йдеться, про які дії, суб'єкти та об'єкти, засобом вираження чого є логіко-лінгвістична модель, здатна відображати зміст тексту.

Фактична реалізація методу автоматизованого формування логіко-лінгвістичних моделей текстової інформації розуміє під собою сортування масиву слів конкретного речення у відповідності з чітко сформованими правилами. Основною ідеєю цього перетворення є визначення відношень між підметом (суб'єктом) та присудком (предикатом), а також їх загальне відношення до того, що вони виражають в дійсності, та формують найголовнішу граматичну властивість речення – його суть – предикативність.

2.3. Правила для автоматизованого формування логіко-лінгвістичних моделей

Ставиться у відповідність кожному слову речення S_i ($i = \overline{1,r}$, де r– кількість слів у реченні) набір характеристик:

$$Z_i(S_i) = \{cm_i, g_i, n_i, k2_i, t_i, h_i, l_i, ch_i\},$$

де $cm_i = \overline{1,11}$ – граматична характеристика, що позначає частину мови, кожному цифровому значенню якої відповідає іменник – 1, прикметник – 2, числівник – 3, займенник – 4, дієслово – 5, дієприкметник – 6, дієприслівник – 7, прислівник – 8, прийменник – 9, сполучник – 10 або частка – 11 відповідно;

$g_i = \overline{1,7}$ – морфологічна ознака, що відповідає за відмінок;

$n_i = \overline{1,2}$ – граматичний параметр, що означає число;

$k2_i = \overline{1,4}$ – граматичний параметр, що означає рід;

$t_i = \overline{0,3}$ – граматичний параметр, що означає час;

$h_i = \overline{1,3}$ – граматичний параметр, що означає спосіб;

$l_i = \overline{1,3}$ – граматичний параметр, що означає особу;

$ch_i = \overline{1,5}$ – параметр, що означає синтаксичну роль (підмет, присудок, додаток, означення, обставина).

Тобто у формалізованому вигляді характеристики кожного слова можна представити як одновимірний масив. Таким чином, кожне слово речення буде характеризуватися набором цифр, наприклад, характеристики $Z_i(S_i) = \{1,1,1,1,0,0,0,0\}$ означають, що слово S_i– іменник у називному відмінку, чоловічого роду однини. Деякі з правил формування словосполучень розглянуто нижче.

Тобто внаслідок здійснення морфологічного розбору на етапі концептуалізації створюється двовимірний масив (рис. 2.2), в якому рядки відповідають за номер елемента в системі, а стовпчики – за характеристики цих елементів.

$$Z = \begin{vmatrix} cm_{11} & g_{12} & n_{13} & k2_{14} & t_{15} & h_{16} & l_{17} \\ cm_{21} & g_{22} & n_{23} & k2_{24} & t_{25} & h_{26} & l_{27} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ cm_{i1} & g_{i2} & n_{i3} & k2_{i4} & t_{i5} & h_{i6} & l_{i7} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ cm_{n1} & g_{n2} & n_{n3} & k2_{n4} & t_{n5} & h_{n6} & l_{n7} \end{vmatrix}$$

Рис. 2.2. Масив характеристик елементів системи

Правило I трактується наступним чином: якщо частина мови для першого слова прикметник, а для другого – іменник, відмінки, число та рід обох слів співпадають, то слова утворюють словосполучення.

Тобто, якщо у реченні зустрінуться два слова, набір характеристик кожного з яких відповідає одновимірному масиву $Z_i(S_i) = \{2,2,1,2,0,0,0,0\}$ та $Z_j(S_j) = \{1,2,1,2,0,0,0,0\}$ відповідно, то такі два слова є словосполученням. Прикладом для такого набору значень граматичних характеристик є словосполучення «гармонійної

мелодії» – прикметник та іменник знаходяться в родовому відмінку однини, жіночого роду.

Наприклад, словосполучення «*гармонійна мелодія*»:

$$if((cm(S_i)=2)and(cm(S_{i+1})=1))and(g(S_i)=g(S_{i+1}))$$
$$and(n(S_i)=n(S_{i+1}))and(k2(S_i)=k2(S_{i+1}))$$
$$then(S_j=S_i\cup S_{i+1})$$

Якщо S$[i]$ – прикметник а $S[i+1]$, де $(k=1,n-i)$ – іменник та в S$[i]$ і $S[i+1]$ відмінок, число та рід (якщо однина, тобто $n(S_i)$=1) співпадають то $(S[i]+S[i+k])$ словосполучення і завершити пошук, якщо або ознака $S[i+k]$ "не кома" і $S[i+k+1]$ – не сполучник або $S[i+k+1]$ – дієслово

Правило працює наступним чином: береться перше слово якщо воно прикметник то перевірка наступного слова, якщо воно іменник – проводиться порівняння (відмінок, число та рід, якщо це однина), то словосполучення і завершуємо пошук, якщо або після іменника НЕ стоїть кома і наступне після іменника слово – не сполучник, або наступне після іменника слово – дієслово. У всіх інших випадках продовжується пошук до кінця речення.

Наприклад, "Червоне сонце гріє рудого товстого юного вкритого ковдрою кота на просторому ґанку."

Словосполучення: червоне сонце, рудого кота, товстого кота, юного кота, вкритого кота(правило 4 дієприкметник + іменник), просторому ґанку. Іншими словами перше правило працює при наявності «Прикметник» - «Іменник», якщо $S[i]$ – «прикметник», а $S[i+k]$, де $k-(i+1,n)$ – «іменник» , при цьому(відмінок, рід та число) співпадають то $S[i]$ та $S[i+k]$ – словосполучення.

Але якщо $S[i+k]$ – «дієслово» то по $S[i]$ завершити пошук або якщо $S[i+k]$ – «сполучник» та $S[i+k+1]$ – «іменник»(відмінок та число) співпадають із $S[i]$ то $S[i]$ та $S[i+k]$ – словосполучення. В інакшому випадку слід завершити пошук по $S[i]$. Правило працює добре, крім, умови, якщо сполучник і далі іменник.

Правило II працює із «Порядковий числівник»-«Іменник»: якщо $S[i]$ – «порядковий числівник», а $S[i+k]$, де $k-(i+1,n)$ – «іменник», при цьому

(відмінок, рід та число) співпадають то $S[i]$ та $S[i+\ k]$ – словосполучення. Але якщо $S[i+\ k]$ – «дієслово» то по $S[i]$ необхідно завершити пошук. Або якщо $S[i+\ k]$ – «сполучник» та $S[i+\ k+1]$ – «іменник»(відмінок та число) співпадають із $S[i]$ то { $S[i]$ та $S[i+\ k]$} – словосполучення. В іншому випадку необхідно завершити пошук по $S[i]$.

Правило III працює із «Займенник»-«Іменник»: якщо $S[i]$ – «займенник», а $S[i+\ k]$, де $k-(i+1,n)$ – «іменник», при цьому(відмінок, рід та число) співпадають то $S[i]$ та $S[i+\ k]$ – словосполучення. Але якщо $S[i+\ k]$ – «дієслово» то по $S[i]$ завершити пошук. Або якщо $S[i+\ k]$ – «сполучник» та $S[i+\ k+1]$ – «іменник» (відмінок та число) співпадають із $S[i]$ то { $S[i]$ та $S[i+\ k]$} – словосполучення. В іншому випадку слід завершити пошук по $S[i]$.

Правило IV працює із «Дієприкметник»-«Іменник» – якщо $S[i]$ – «Дієприкметник», а $S[i+\ k]$, де $k-(i+1,n)$ – «іменник», при цьому(відмінок, рід та число) співпадають то $S[i]$ та $S[i+\ k]$ – словосполучення. Але якщо $S[i+\ k]$ – «дієслово», то по $S[i]$ завершити пошук. Або якщо $S[i+\ k]$ – «сполучник» та $S[i+\ k+1]$ – «іменник»(відмінок та число) співпадають із $S[i]$ то $S[i]$ та $S[i+\ k]$ – словосполучення. За інших умов потрібно завершити пошук по $S[i]$ [8].

Правило V Якщо перше слово – дієслово, а друге – іменник (як правило) у родовому або знахідному (обов'язково не в називному) відмінку, то слова утворюють словосполучення. Наприклад, «*написати підручник*».

Правило VI Якщо перше слово – рикметник, а друге – дієслово в неозначеній формі, то вони утворюють словосполучення. Наприклад, «*запрограмований розпізнавати*».

Маємо таку градацію:

– підмет – суб'єкт x;
– присудок – відношення p;
– додаток – об'єкт yабо предмет z відношення;
– означення – характеристика суб'єкта g, об'єкта q або предмета відношення r;

– обставина – характеристика відношення h.

Група слів, пов'язаних між собою логічними зв'язками, буде позначатися $sp_j, j = \overline{1,m}$, де m – кількість словосполучень у реченні. За правилами української мови словосполучення можуть утворювати між собою наступні члени речення (за їх наявності):

– "означення – підмет" – $sp_j = g \cup x$;

– "присудок – додаток" – $sp_j = p \cup y$;

– "означення – додаток" – $sp_j = q \cup y$;

– "додаток – додаток" – $sp_j = y \cup z$;

– "означення – додаток" – $sp_j = r \cup z$;

– "обставина – присудок" – $sp_j = h \cup p$.

Наприклад, нехай потрібно виявити словосполучення у простому реченні: *"Нові дисципліни регулярно включають до навчальних планів"*. Спочатку відбувається ідентифікація ролей для кожного із слів речення:

– підмет – суб'єкт x – *0*;

– присудок – відношення p– *включають*;

– додаток – об'єкт y– *дисципліни*;

– додаток – предмет z відношення – *планів*;

– означення – характеристика суб'єкта g– *0*;

– означення – характеристика об'єкта q– *нові*;

– означення – характеристика предмета відношення r – *навчальних*;

– обставина – характеристика відношення h – *регулярно*.

Таким чином, у заданому реченні наявні такі словосполучення:

– "присудок – додаток" – $sp_j = p \cup y$ – *включають дисципліни*;

– "означення – додаток" – $sp_j = q \cup y$ – *нові дисципліни*;

– "додаток – додаток" – $sp_j = y \cup z$ – *дисципліни планів*;

– "означення – додаток" – $sp_j = r \cup z$ – *навчальних планів;*

– "обставина – присудок" – $sp_j = h \cup p$ – *включають регулярно*.

Загальна форма моделі для простого, не ускладненого однорідними членами речення:

$$L_\mu(S) = p(x, g,\ y, q, z, r, h),$$

де p – присудок, $ch = 2$

x – підмет, $ch = 1$

g – означення, пов'язане з підметом x $ch = 4$

y – додаток, пов'язаний з присудком $ch = 3$

q – означення, пов'язане з додатком y $ch = 4$

z – додаток, пов'язаний з додатком y $ch = 3$

r – означення, пов'язане з додатком y $ch = 4$

h – обставина, пов'язана з присудом p $ch = 5$

Правило – якщо $ch = 2$ і $S[i]$ – "*дієслово*», $i = \overline{1, n}$, де n – кількість слів у простому реченні, то $p = S[i]$. Якщо $ch = 1$ і $S[i]$ – "*іменник*» і відмінок називний, то $x = S[i]$. Якщо $ch = 4$ або $ch = 5$ і $S[i]$ утворює словосполучення з x, то $g = S[i]$. Якщо $ch = 3$і $S[i]$ – "*іменник*» НЕ в називному відмінку**,** і утворює словосполучення з p, то $y = S[i]$. Якщо $ch = 4$ або $ch = 5$ і $S[i]$ утворює словосполучення з y, то $q = S[i]$. Якщо $ch = 3$і $S[i]$ – "*іменник*» НЕ в називному відмінку, і утворює словосполучення з y, то $z = S[i]$. Якщо $ch = 4$ або $ch = 5$ і $S[i]$ утворює словосполучення з z, то $r = S[i]$. Якщо $ch = 4$ або $ch = 5$ і $S[i]$ утворює словосполучення з p, то $h = S[i]$.

Для простого речення природної мови *«Маленька пташка транспортує підсохлого горіха»* логіко-лінгвістична модель буде мати вигляд:

$L(S) = L^S = p_1(x_1, g_1,\ y_1, q_1, 0, 0, 0),$

$L(S) = L^S =$ *транспортує(пташка, маленька, горіха, підсохлого, 0, 0, 0).*

Для простого речення природної мови, ускладненого однорідними присудками *«Прудкий юнак локалізував та нормував небезпечну зону»*:

$L(S) = L^S = p_1(x_1, g_1,\ y_1, q_1, z_1, r_1, h_1)\&p_2(x_1, g_1,\ y_1, q_1, z_1, r_1, h_1) =$

$= p_1(x_1, g_1,\ y_1, 0, z_1, r_1, 0)\&p_2(x_1, g_1,\ y_1, 0, z_1, r_1, 0),$

$L(S) = L^S$ =*локалізував (юнак, прудкий, зону, небезпечну, 0, 0, 0)* & *нормував (юнак, прудкий, зону, небезпечну, 0, 0, 0).*

Для простого речення природної мови, ускладненого однорідними означеннями для підмета *«Прекрасний, сонячний вечір гармонізує яскравим променем.»*:

$L(S) = L^S = p_1(x_1, g_1,\ y_1, q_1, z_1, r_1, h_1)\&p_1(x_1, g_2,\ y_1, q_1, z_1, r_1, h_1) =$

$= p_1(x_1, g_1,\ y_1, 0, z_1, r_1, 0)\&p_1(x_1, g_2,\ y_1, 0, z_1, r_1, 0),$

$L(S) = L^S$ = *гармонізує (вечір, прекрасний, променем, яскравим, 0, 0, 0)* & *гармонізує (вечір, сонячний, променем, яскравим, 0, 0, 0).*

Для простого речення природної мови, ускладненого однорідними означеннями та додатками *«Рясний дощ підганяє працьовитих та втомлених робітників і помічників»*:

$L(S) = L^S = p_1(x_1, g_1,\ y_1, q_1, z_1, r_1, h_1)\&p_1(x_1, g_1,\ y_2, q_1, z_1, r_1, h_1)\&$

$p_1(x_1, g_1,\ y_1, q_2, z_1, r_1, h_1)\&p_1(x_1, g_1,\ y_2, q_2, z_1, r_1, h_1) =$

$= p_1(x_1, g_1,\ y_1, q_1, z_1, r_1, 0)\&p_1(x_1, g_1,\ y_2, q_1, z_1, r_1, 0)\&$

$p_1(x_1, g_1,\ y_1, q_2, z_1, r_1, 0)\&p_1(x_1, g_1,\ y_2, q_2, z_1, r_1, 0),$

$L(S) = L^S$ = *підганяє (дощ, рясний, робітників, працьовитих, 0, 0, 0)* & *підганяє (дощ, рясний, робітників, втомлених, 0, 0, 0)* & *підганяє (дощ, рясний, помічників, працьовитих, 0, 0, 0)* & *підганяє (дощ, рясний, помічників, втомлених, 0, 0, 0).*

Для простого речення природної мови, ускладненого однорідними додатками *«Прекрасні музи надихають митців на нові вершини та звершення»* типова форма логіко-лінгвістичної моделі буде мати такий вигляд:

$L(S) = L^S = p_1(x_1, g_1,\ y_1, q_1, z_1, r_1, h_1)\&p_1(x_1, g_1,\ y_1, q_1, z_2, r_1, h_1) =$

$= p_1(x_1, g_1,\ y_1, 0, z_1, r_1, 0)\&p_1(x_1, g_1,\ y_1, 0, z_2, 0,0),$

$L(S) = L^S$ =*надихають (музи, прекрасні, митців, 0, вершини, нові, 0)* & *надихають (музи, прекрасні, митців, 0, звершення, нові, 0).*

2.4. Висновки до розділу

Розглянуто чотири логіко-лінгвістичні моделі представлення знань, які дозволяють перетворити речення природної мови на сукупність неподільних частин, які в свою чергу містять зміст. Описано всі деталі кожної з представлених моделей – значення, можливий вміст, структура. Логіко-лінгвістичні моделі слугують основою порівняльного аналізу електронних текстових документів за змістом.

Розглянуто сім типів побудови логіко-лінгвістичних моделей для простих речень природної мови: з однією частиною, з однорідними присудками, з однорідними обставинами, з однорідними підметами, з однорідними означеннями, з однорідними додатками(кількість частин рівна потужності множини об’єктів), з однорідними означеннями, з однорідними додатками(кількість частин рівна потужності множини предметів відношення).

Для вилучення змісту з текстової інформації природньої мови, її слід формалізувати, тобто записати у вигляді формул, щоб інформацією та знаннями в подальшому міг оперувати комп’ютер. Структура формули логіко-лінгвістичної модель базується на синтаксичній структурі речення, що в свою чергу вирішує питання з формалізацією. Результатом автоматизованого формування логіко-лінгвістичної моделі текстової інформації є предикат, який описує синтаксис речення, і функція, що реалізується предикатом.

Для візуалізації виконання методу автоматизованого формування логіко-лінгвістичних моделей подано зображення виконання етапів методу.

Процес автоматичного формування логіко-лінгвістичних моделей складається з наступних етапів:

- розбиття вхідного тексту на словоформи і ототожнення;
- концептуалізація(морфологічний аналіз слів речення);
- синтаксичний аналіз(визначення структур речення);
- підставляння слів вхідної текстової інформації в модель за синтаксичними ознаками

Основною ідеєю перетворення з речення в модель є визначення відношень між підметом (суб’єктом) та присудком (предикатом), а також їх загальне відношення до того, що вони виражають в дійсності, та формують найголовнішу граматичну властивість речення – його суть – предикативність.

Результатом морфологічного аналізу слова є масив його характеристик, який в свою чергу в майбутньому, під час концептуалізації, буде частиною масиву масивів характеристик слів, що в свою чергу формує двовимірний масив для зберігання предикатів речення.

Розглянуто чотири правила формування словосполучень типу «Прикметник + Іменник», «Порядковий числівник + Іменник», «Займенник + Іменник», «Дієприкметник + Іменник», умови їхнього формування, винятки.

Подана загальна форма моделі для простого, не ускладненого однорідними членами речення та наведені приклади реалізації формування логіко-лінгвістичних моделей із текстової інформації українською мовою.

Сукупність даних правил складатиме основу програмного модуля автоматизованої побудови логіко-лінгвістичних моделей.

РОЗДІЛ 3
АВТОМАТИЗОВАНА ПОБУДОВА ЛОГІКО-ЛІНГВІСТИЧНИХ МОДЕЛЕЙ

3.1. Структура технології автоматизованої побудови логіко-лінгвістичних моделей

Для програмної реалізації технології автоматизованої побудови логіко-лінгвістичних моделей використовується середовище розробки програмного забезпечення *Visual Studio 2017*. Проєкт будується за допомогою засобів *.NET Windows Forms* із застосуванням мови програмування *C#*. На рис. 3.1 зображено схема алгоритму роботи програмного модуля.

Для того щоб створити проєкт *Windows Forms* необхідно в початковому вікні *Visual Studio* натиснути на меню «*File*» після чого відкриється контекстне меню в якому необхідно обрати підменю «*New*», а далі «*Project…*». У вікні, що відкрилося, під назво «*New Project*» слід відкрити вкладку «*Installed*», у ній відкрити вкладку «*Visual C#*», яка запропонує різного роду шаблони для програм різної області застосування. Серед запропонованих варіантів потрібно обрати «*Windows Forms App(.NET Framework*)», заповнити поля «*Name*» і «*Solution name*» та натиснути кнопку «*OK*». Після даних маніпуляцій відкриється вікно проєкту з автоматично згенерованим кодом для формування вікна додатку і саме вікно «*Form 1*». Для того щоб додати елементи, необхідні для програмного модуля, слід відкрити вкладку «*Toolbox*», у якій доступні всі наявні елементи для додавання до вікна продукту. Для того щоб додати елемент слід просто перетягнути його на вікно «*Form 1*», в результаті чого, код, який відповідає за формування елемента, автоматично сформується і долучиться до вже автоматично згенерованого коду вікна «*Form1*».

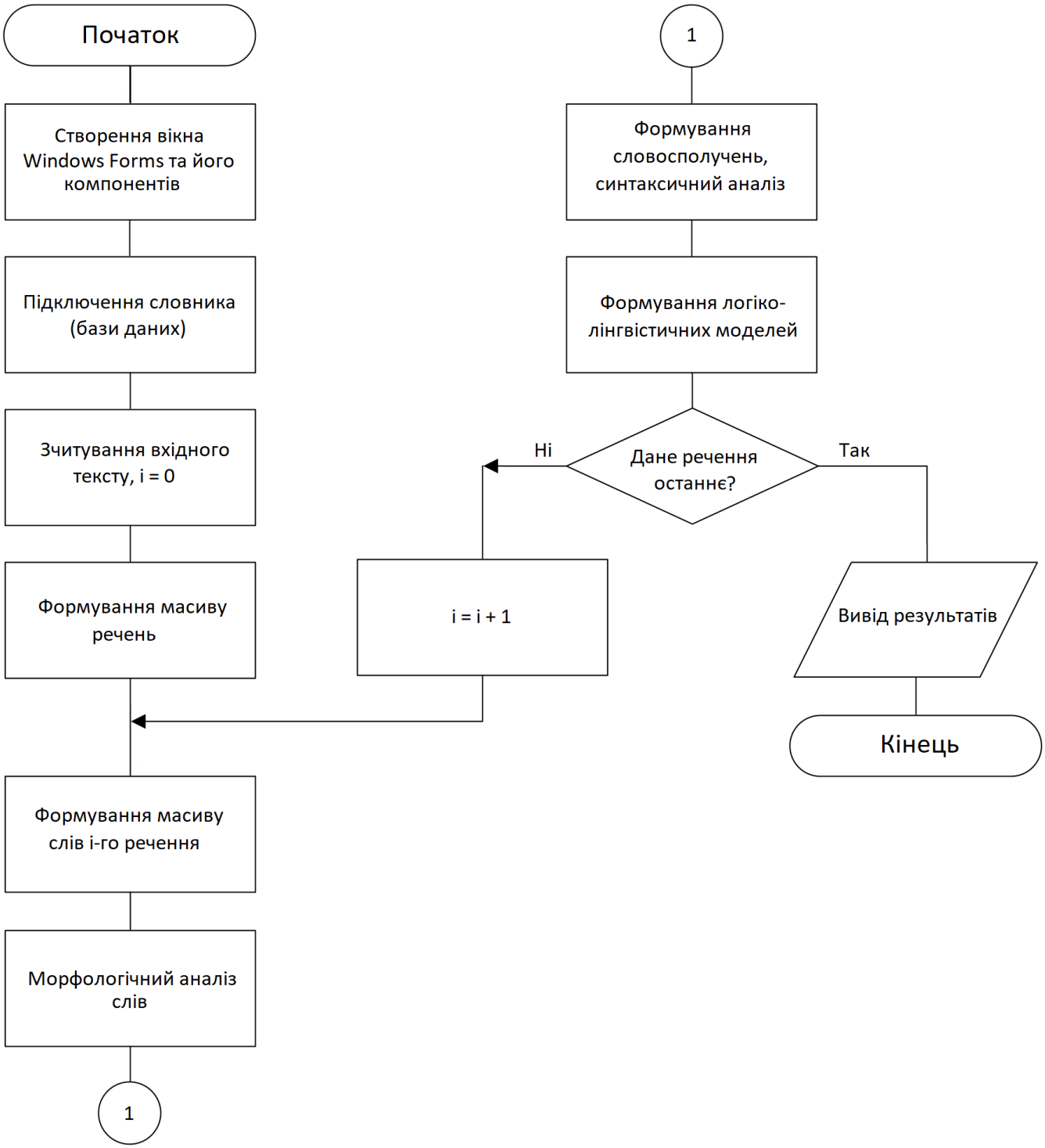

Рис. 3.1. Схема алгоритму технології автоматизованої побудови логіко-лінгвістичних моделей

Так для створення програмної реалізації автоматизованої побудови логіко-лінгвістичних моделей необхідні наступні елементи: два «*FlowLayoutPanel*», чотири «*TextBox*», чотири «*Label*», чотири «*Button*» та «*TableLayoutPanel*».

«*TableLayoutPanel*» слугує формою з кордонами для елементів вікна, вона допомагає пропорційно розміщувати елементи в ній. Необхідні параметри для даного елемента: «*ColumCount*» рівна двом (поділені на 70 і 30 відсотків), «*RowCount*» дорівнює п’ять(поділені на 4.29, 47.93, 4.29, 43.49 відсотків та один рядок висотою 33 пікселі), параметр розміщення відносно внутрішньої частини вікна «*Dock*» слід обрати «*Fill*» зі списку запропонованих.

Один із чотирьох «*TextBox*» слугуватиме для відображення і редагування вхідного тексту природньою мовою і розміщується в другому рядку та першій колонці елементу «*TableLayoutPanel*». Другий «*TextBox*» відображатиме видобуті словосполученя після екстракції і міститься в другому рядку і другому стовпці елементу «*TableLayoutPanel*». Третій «*TextBox*» здійснюватиме відображення сформованих логіко-лінгвістичних моделей і розташований в третьому рядку і першому стовпці «*TableLayoutPanel*». Останній «*TextBox*» слугуватиме для відображення типів формування логіко-лінгвістичних моделей відносно простих речень природньої мови з різними синтаксичними особливостями. Параметри для обох чотирбох(«*Dock*» рівна «*Fill*», «*Multilane*» відповідає «*True*»), окрім одного – «*ReadOnly*» для першого «*TextBox*» даний параметр відповідає значенню «*False*», а для другого, третього і четвертого «*True*».

Один із «*Label*» розміщується над першим «*TextBox*» і має параметр «*Text*» рівний «Текстовий редактор:». Другий «*Label*» розміщується над другим «*TextBox*» і містить параметр «*Text*» дорівнює «Словосполучення:». Третій «*Label*» розташований над третім «*TextBox*» і містить параметр «*Text*» дорівнює «Логіко-лінгвістична модель». Четвертий «*Label*» міститься над четвертим «*TextBox*», параметр «*Text*» якого дорівнює «Тип моделі». Всі «*Label*» вирівнюються по центру.

Елементи «*FlowLayoutPanel*» містяться в п’ятому рядку, першому та другому стовпці елемента «*TableLayoutPanel*» відповідно. Вони слугують для автоматизації розміщення у ній елементів, в даному випадку двох «*Button*», з того чи іншого боку. Параметр «*FlowDirection*» у першому «*FlowLayoutPanel*» рівний «*RigtToLeft*», а в другому «*TopDown*».

Два елементи «Button», що слугуватимуть клавішами у додатку, розміщуються у першому «*FlowLayoutPanel*» і мають параметр «*Text*» рівний «Відкрити» та «Зберегти» відповідно. Двоє інших «*Button*» розташовуватимуться у другому «*FlowLayoutPanel*» і мають параметр «*Text*» рівний «Сформувати» та «Зберегти результат».

У результаті додавання елементів сформовано вікно програмної реалізації технології автоматизованої побудови логіко-лінгвістичних моделей, зображене на рис. 3.2.

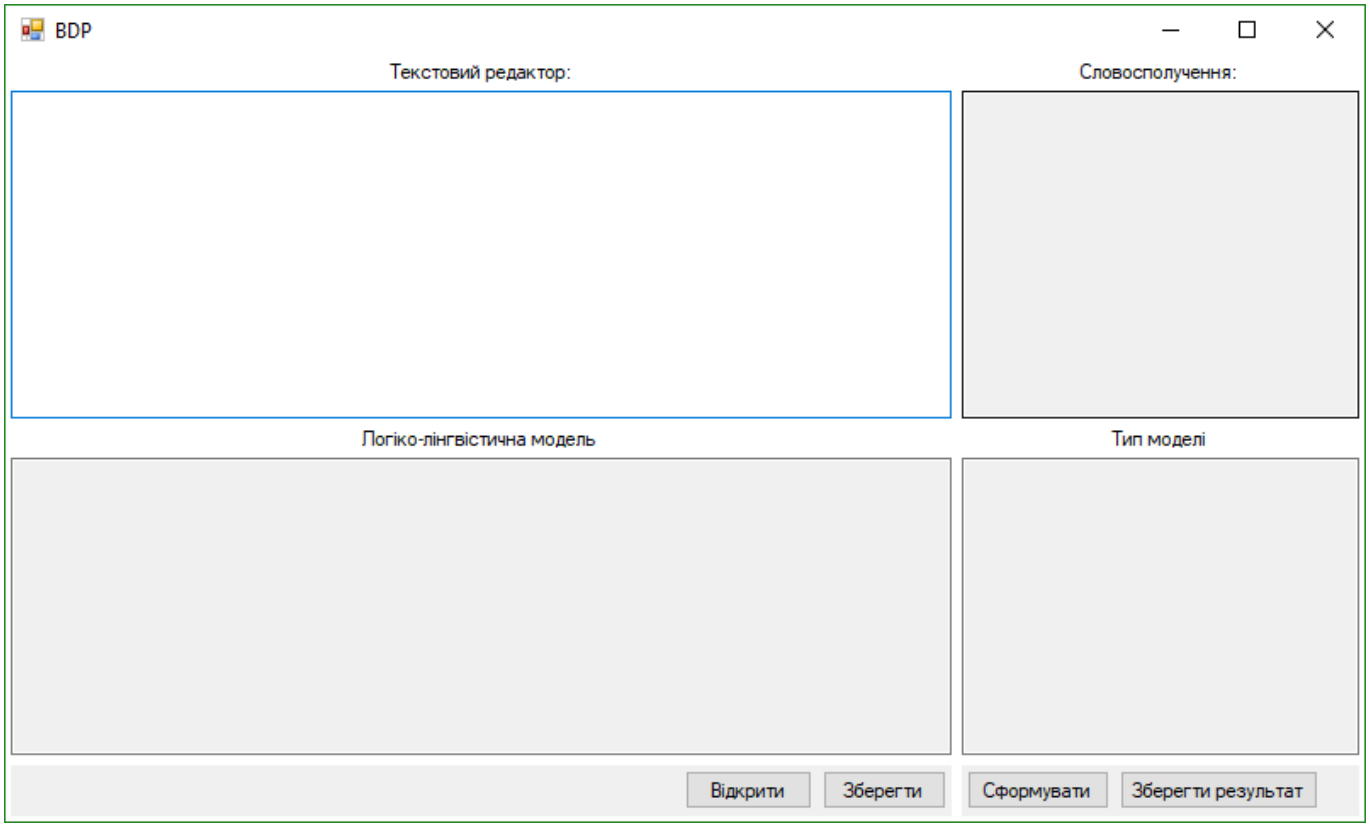

Рис. 3.2. Результат додавання компонентів вікна *Windows Forms*

Одним із найважливіших елементів програмної реалізації технології автоматизованої побудови логіко-лінгвістичних моделей являється словник, що представлений базою даних *MS Access*, який містить у собі слова, відсортовані за частинами мови, родами, відмінками і т.д. Частина електронного словника зображено на рис. 3.3.

E	F	G	H	I	J	K	L	M	
Field4	Field5	Field6	Field7	Field8	Field9	Field10	Field11	Field12	Field13
Аа"хен	Аа"хена	Аа"хену, Аа"хенові	Аа"хен	Аа"хеном	на/в/по Аа"хені, Аа"хену	Аа"хене	<>	<>	<>
абажу"р	абажу"ра	абажу"ру, абажу"рові	абажу"р	абажу"ром	на/в/по абажу"рі, абажу"ру	абажу"ре*	абажу"ри	абажу"рів	абажу"
абажу"рчик	абажу"рчика	абажу"рчику, абажу"рчикові	абажу"рчик	абажу"рчиком	на/в/по абажу"рчику	абажу"рчику*	абажу"рчики	абажу"рчиків	абажу"
аба"з	аба"за	аба"зу, аба"зові	аба"з	аба"зом	на/в/по аба"зі, аба"зу	аба"зе*	аба"зи	аба"зів	аба"за
абази"н	абази"на	абази"нові, абази"ну	абази"на	абази"ном	на/в/по абази"нові, абази"ні, абази"ну	абази"не	абази"ни	абази"нів	абази"
абази"нець	абази"нця	абази"нцеві, абази"нцю	абази"нця	абази"нцем	на/в/по абази"нцеві, абази"нцю, абази"нці	абази"нцю	абази"нці	абази"нців	абази"
абази"нка	абази"нки	абази"нці	абази"нку	абази"нкою	на/в/по абази"нці	абази"нко	абази"нки	абази"нок	абази"
аба"зія	аба"зії	аба"зії	аба"зію	аба"зією	на/в/по аба"зії	аба"зіє*	<>	<>	<>
Аба"й	Аба"я	Аба"єві, Аба"ю	Аба"я	Аба"єм	на/в/по Аба"єві, Аба"ї, Аба"ю	Аба"ю	<>	<>	<>
Аба"й	Аба"я	Аба"ю, Аба"єві	Аба"й	Аба"єм	на/в/по Аба"ї, Аба"ю, Аба"єві	Аба"ю	<>	<>	<>
аба"к	аба"ка	аба"ку, аба"кові	аба"к	аба"ком	на/в/по аба"ку	аба"ку*	аба"ки	аба"ків	аба"ка
аба"ка	аба"ки	аба"ці	аба"ку	аба"кою	на/в/по аба"ці	аба"ко*	аба"ки	аба"к	аба"ка
Абака"н	Абака"на	Абака"ну, Абака"нові	Абака"н	Абака"ном	на/в/по Абака"ні, Абака"ну	Абака"не	<>	<>	<>
Абака"н	Абака"ну	Абака"ну, Абака"нові	Абака"н	Абака"ном	на/в/по Абака"ні, Абака"ну	Абака"не	<>	<>	<>
Абаку"м	Абаку"ма	Абаку"мові, Абаку"му	Абаку"ма	Абаку"мом	на/в/по Абаку"мові, Абаку"мі, Абаку"му	Абаку"ме	Абаку"ми	Абаку"мів	Абаку"
Абаку"мівна	Абаку"мівни	Абаку"мівні	Абаку"мівну	Абаку"мівною	на/в/по Абаку"мівні	Абаку"мівно	Абаку"мівни	Абаку"мівен	Абаку"
Абаку"мович	Абаку"мовича	Абаку"мовичу, Абаку"мовичеві	Абаку"мовича	Абаку"мовичем	на/в/по Абаку"мовичу, Абаку"мовичі, Абаку"мовичеві	Абаку"мовичу	Абаку"мовичі	Абаку"мовичів	Абаку"
абандо"н	абандо"ну	абандо"ну, абандо"нові	абандо"н	абандо"ном	на/в/по абандо"ні, абандо"ну	абандо"не*	абандо"ни	абандо"нів	абандо
аба"т	аба"та	аба"тові, аба"ту	аба"та	аба"том	на/в/по аба"тові, аба"ті, аба"ту	аба"те	аба"ти	аба"тів	аба"та
абати"са	абати"си	абати"сі	абати"су	абати"сою	на/в/по абати"сі	абати"со	абати"си	абати"с	абати"

Рис. 3.3. Частина електронного словника

Для підключення бази даних до програми використовують рядки коду, які зображені на рис. 3.4.

```
public partial class Form1 : Form
{
    //зберігання розташування бази dicua.mdb та провайдера у формі рядка
    public static string connectString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=dicua.mdb;";
    //об'єкт який використовується для підключення
    private OleDbConnection myConnection;
    private string name,
                   text,
                   word;
    private string[] sentences,
                     words;

    public Form1()
    {
        InitializeComponent();
        //встановлення зв'язку з базоюданих
        myConnection = new OleDbConnection(connectString);
        myConnection.Open();
    }
```

Рис. 3.4. Програмне підключення до бази даних *MS Access*

Для автоматичного відключення бази даних під час закривання програмного модуля використовується метод поданий на рис. 3.5.

```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    //роз'єднання зв'язу з базою даних під час завершення роботи
    myConnection.Close();
}
```

Рис. 3.5. Програмне від’єднання від бази даних *MS Access*

Для зчитування вхідного тексту із файлу формату «*.txt*» і відображення його на екрані вікна програмного модуля з можливістю редагування вмісту необхідно виконати декілька кроків. Для початку слід додати до вікна додатку два невидимі компоненти, які позначатимуть процес появи діалогового вікна для відкриття текстових файлів та їхнього зберігання – «*openFileDialog*» та «*saveFileDialog*». Далі необхідно поєднати процес відкриття файлів та процес збереження файлу із відповідними клавішами «Відкрити» та «Зберегти». Для цього після подвійного натискання на елемент «*Button*», якому відповідатиме відкриття, автоматично відкриється вкладка із кодом програми у якому буде згенеровано шаблон методу для функціонування клавіші. Ті ж маніпуляції слід також зробити із клавішою, яка відповідатиме за зберігання змін у текстовому документі. Код методів для відкривання і збереження вмісту файлу зображено на рис. 3.6.

```
//метод для відкриття текстового документу
private void button3_Click(object sender, EventArgs e)
{
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        name = openFileDialog1.FileName;
        textBox1.Clear();
        textBox1.Text = File.ReadAllText(name);
    }
}
//метод для збереження змін текстового документу
private void button2_Click(object sender, EventArgs e)
{
    if (saveFileDialog1.ShowDialog() == DialogResult.OK)
    {
        name = saveFileDialog1.FileName;
        File.WriteAllText(name, textBox1.Text);
    }
}
```

Рис. 3.6. Методи для відкриття і зберігання файлу після натисканні клавіші

Формування масиву *S* із речення полягає у створенні та заповненні двовимірної матриці десяткових чисел, кожен рядок якої відповідає кожному слову в реченні, таким чином кількість рядків залежить від кількості слів речення, а кожному стовбцю відповідає характеристика, яка належить слову. Іншими словами кажучи, другий рядок матриці кодування слів відповідає другому слову в реченні, а сукупність чисел у даному рядку характеризує його за морфологічними ознаками та особливостями.

Згідно з алгоритмом функціонування, для повноцінної роботи програмного модуля необхідно мати наступні відомості про слово: частину мови, рід, число, відмінок та наявність коми після слова, член у реченні, синтаксичні дані членів речення. Отже, для кодування вхідних речень достатньо використати матрицю, яка складається із *n* рядків, де *n* – кількість слів у реченні, та семи стовпців. Тому матриця кодування вхідного речення матиме вигляд, як показано на рис. 3.7.

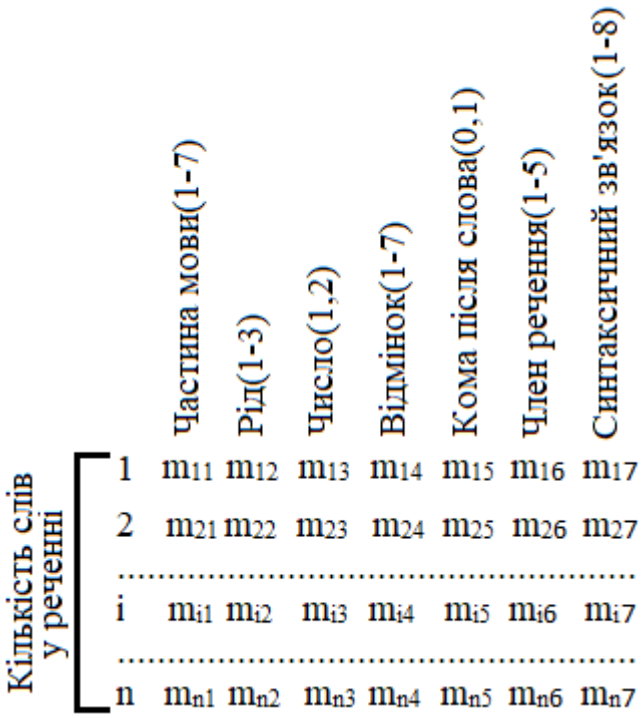

Рис. 3.7. Матриця кодування речення

Для заповнення матриці використовують базу даних, словник термінів української мови, що містить різні форми слів, які застосовуються в реченні, згруповані за певними ознаками. Щоб визначити частину мови, рід число та відмінок, береться слово з речення і порівнюється по черзі з різними таблицями бази даних, кожна з яких відповідає граматичним характеристикам тієї чи іншої частини мови. Коли до вхідного слова знаходиться відповідник із словника термінів, то дані про його розташування у таблиці дають висновок про належність до частини мови і морфологічні ознаки.

Дані дії кодуються наступним чином. Спочатку формується команда(*SQL* запит) до бази даних про надання певних стовпців певної таблиці. Тоді виконується

даний запит у базі, яку було підключено. Результат запиту поміщується до «Читача Баз Даних», об'єкту типу «*OleDbDataReader*», який пізніше викликається для порядкового читання даних таблиці. Кожен елемент рядка порівнюється із вхідним словом, якщо вони однакові, то дані заносяться в таблицю кодування і пошук продовжується для наступного слова речення, якщо ні, то зчитується наступний рядок і відбувається аналогічне порівняння. Якщо співпадінь не знайшлося, запускається читання іншої таблиці, яка відповідає іншій частині мови. У ній відбувається порівняння в такий же спосіб. Фрагмент коду, що відображає частину даних маніпуляцій, зображено на рис. 3.8.

```
        sqlcommandDie = "SELECT Field4, Field5, Field6, Field7, Field8, Field9, Field10, " +
    "Field11, Field12, Field13, Field14, Field15, Field16, Field17, Field18, Field19," +
    "Field20, Field21, Field22, Field23, Field24, Field25, Field27, Field28, Field29 FROM die2";
bool find = false;

//перевірка чи іменник
OleDbCommand extractCommand = new OleDbCommand(sqlcommandIm, myConnection);
OleDbDataReader reader = extractCommand.ExecuteReader();
if (!find)
{
    while (reader.Read())
    {
        string compare;
        int k = 0;
        bool endImVidm = false;
        while (k < 14)
        {
            compare = reader[k].ToString();
            compare = compare.ToLower();
            compare = compare.Replace("\"", "");
            //перевірка на місцевий відмінок
            if (k == 5 || k == 12)
            {
```

Рис. 3.8. Програмна реалізація визначення морфологічних ознак слова

Після заповнення матриці кодування речення, порівнявши слова із таблицями частин мови, які необхідні для виконання правил формування словосполучень, проводиться перевірка відповідності слова одному з правил формування словосполучень. Схема алгоритму виконання формування словосполучень за першим правилом зображена на рис. 3.9.

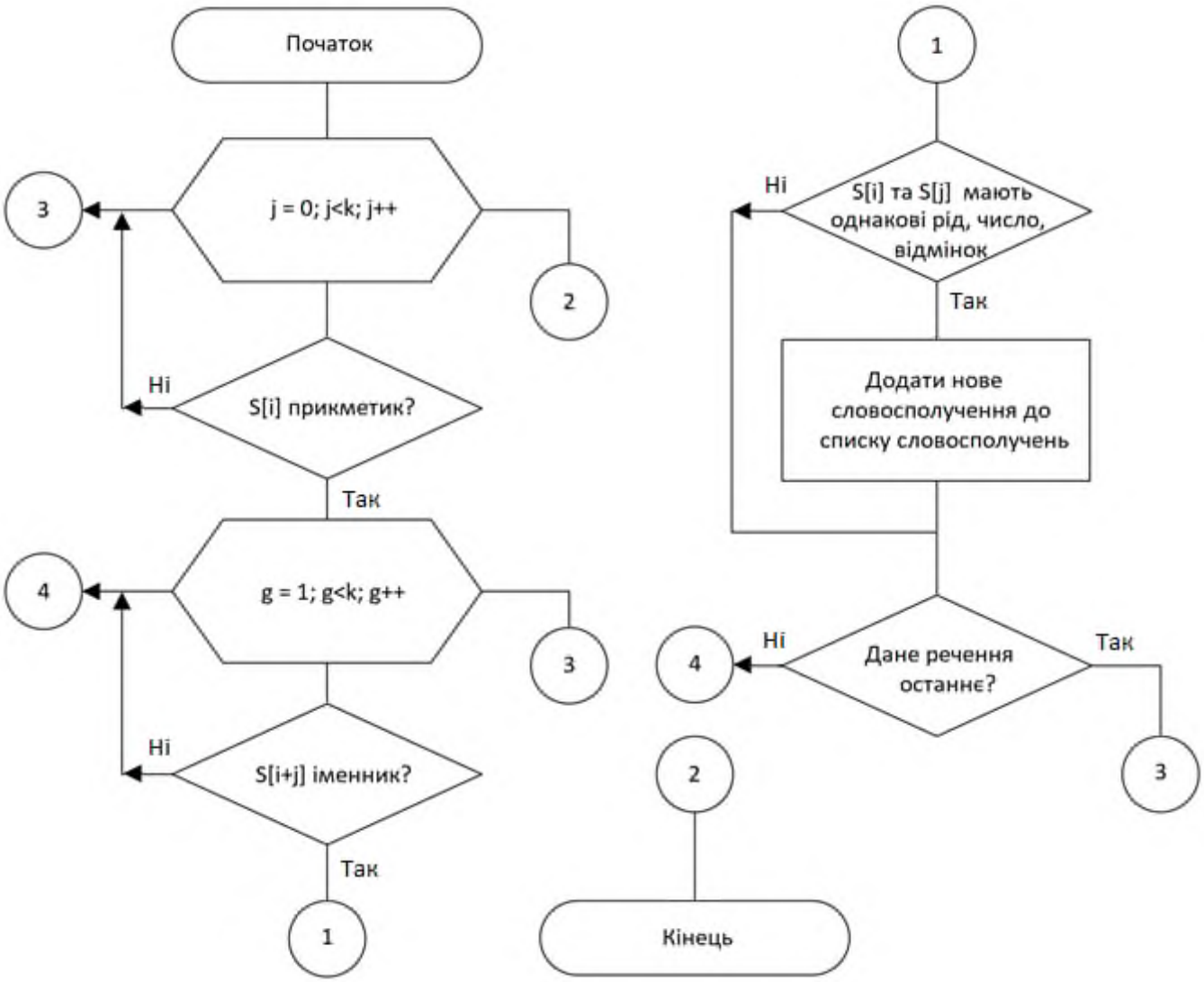

Рис. 3.9. Схема алгоритму формування словосполучень «Прикметник + Іменник»

Завдяки даним із матриці можна легко програмно реалізувати перше правило. Фрагмент коду, що реалізує видобування словосполучення типу «Прикметник + Іменник» зображено на рис. 3.10.

```
//перевірка за першим правилом
for (int i = 0; i < words.Length; i++)
{
    if (masWords[i][0] == 2 || masWords[i][0] == 3 || masWords[i][0] == 4)//якщо прикметник
    {
        for (int j = i + 1; j < words.Length; j++)
        {
            if (masWords[j][0] == 1)//якщо іменник
            {
                if (masWords[j][2] == masWords[i][2] && masWords[j][3] == masWords[i][3])//однакові рід, число, відмінок
                {
                    if (masWords[i][5] == 4 && masWords[j][5] == 1)
                        masWords[i][6] = 5;
                    else
                        masWords[i][6] = 6;
                    textBox2.Text += words[i] + " " + words[j] + "; ";//додати словосполучення
                    if (j + 1 < words.Length)//перевірка чи не має іменників після даного
                    {
                        if (masWords[j][4] == 1 || masWords[j + 1][0] != 6)
                        {
                            continue;
                        }
                        else
```

Рис. 3.10. Програмна реалізація формування словосполучень «Прикметник + Іменник»

Після формування словосполучень в реченні природньої мови, за описаними в другому розділі правилами, здійснюється синтаксичний аналіз речення на основі морфологічного аналізу і сукупності словосполучень речення. Результатом синтаксичного аналізу є елементи предикату, що становить собою логіко-лінгвістичну модель речення. Фрагмент коду який програмно реалізує синтаксис на основі здобутих попередньо даних подано на рис. 3.11.

```
//визначення синтаксису
if (masWords[i][0] == 1 && masWords[i][3] == 1)
{
    masWords[i][5] = 1;
    masWords[i][6] = 1;
}
else if (masWords[i][0] == 1 && masWords[i][3] != 1)
    masWords[i][5] = 3;
else if (masWords[i][0] == 6)
{
    masWords[i][5] = 2;
    masWords[i][6] = 2;
}
else if (masWords[i][0] >= 2 && masWords[i][0] <= 4)
    masWords[i][5] = 4;
else
    masWords[i][5] = 0;
```

Рис. 3.11. Визначення членів речення

Після синтаксичного аналізу відбувається формування логіко-лінгвістичної моделі речення, відповідно до одного з восьми типів формування логіко-лінгвістичної

моделі, враховуючи будову простого речення природньої мови. Код програмної реалізації визначення типу логіко-лінгвістичної моделі простого речення зображено на рис. 3.12.

```
int[] masL = new int[7];
int type = 0;
for(int i = 0; i < words.Length; i++)
{
    if ((masWords[i][6] - 1) >= 0)
    {
        masL[masWords[i][6] - 1]++;
        if (masL[masWords[i][6] - 1] >= 2)
            type = masWords[i][6];
    }
}
if (type == 0)//тип 1
{
```

Рис. 3.12. Визначення типу логіко-лінгвістичної моделі простого речення

Після визначення типу логіко-лінгвістичної моделі простого речення природньої мови, відбувається формування і відображення формули. Фрагмент коду, який відповідає формуванню і відображенню логіко-лінгвістичної моделі першого типу зображено на рис. 3.13.

```
String p = "0", x = "0", y = "0", z = "0", g = "0", q = "0", r = "0", h = "0";
textBox4.Text += "Тип 1";
textBox4.Text += Environment.NewLine;
textBox3.Text += "L = ";
for(int k = 0; k < words.Length; k++)
{
    if (masWords[k][6] == 1)
        x = words[k];
    else if (masWords[k][6] == 2)
        p = words[k];
    else if (masWords[k][6] == 3)
        y = words[k];
    else if (masWords[k][6] == 4)
        z = words[k];
    else if (masWords[k][6] == 5)
        g = words[k];
    else if (masWords[k][6] == 6)
        q = words[k];
    else if (masWords[k][6] == 7)
        r = words[k];
    else if (masWords[k][6] == 8)
        h = words[k];
}
textBox3.Text += p + '(' + x + ',' + g + ',' + y + ',' + q + ',' + z + ',' + r + ',' + h + ')' + Environment.NewLine;
```

Рис. 3.13. Формування логіко-лінгвістичної моделі

Після відображення результату формування логіко-лінгвістичної моделі йде перехід до повторних маніпуляцій з наступним реченням, за умови, що воно існує. В іншому ж випадку, на екрані відображатиметься вікно програмної реалізації технології автоматизованої побудови логіко-лінгвістичних моделей з результатами її роботи. Дану інформацію можна зберегти у *txt* файл, натиснувши кнопку «Зберегти результат».

На рис. 3.14 зображено структуру програмного модуля технології автоматизованої побудови логіко-лінгвістичних моделей.

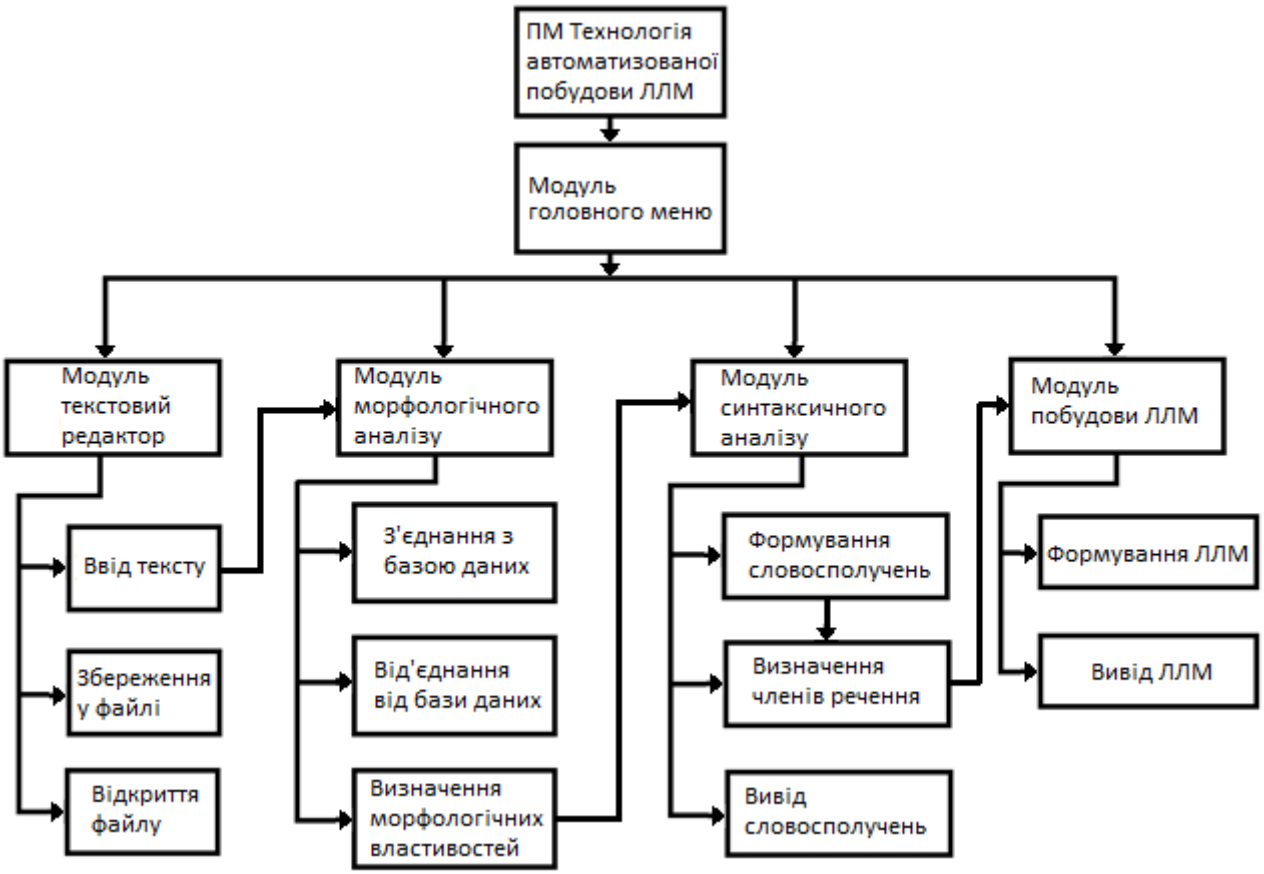

Рис. 3.14. Структура технології автоматизованої побудови логіко-лінгвістичних моделей

3.2. Результати функціонування технології автоматизованої побудови логіко-лінгвістичних моделей

Програмна реалізація технології автоматизованої побудови логіко-лінгвістичних моделей має три варіанти отримання вхідного потоку тексту для здійснення екстракції. Перший варіант найпростіший, він полягає у вводі даних в області під надписом «Текстовий редактор: ». Даний спосіб найшвидший в плані кількості кроків для його реалізації і підходить для тих випадків, коли відбувається ознайомлення з програмою чи є наявний текстовий потік у буфері даних, який можна просто вставити. Другий варіант дає змогу відкрити будь який файл *txt*-формату, який наявний на комп'ютері. Для його виконання слід натиснути клавішу «Відкрити» після чого буде доступне діалогове вікно вибору файлу для відкриття. Процес відкриття текстового файлу зображено на рис. 3.15 та рис. 3.16.

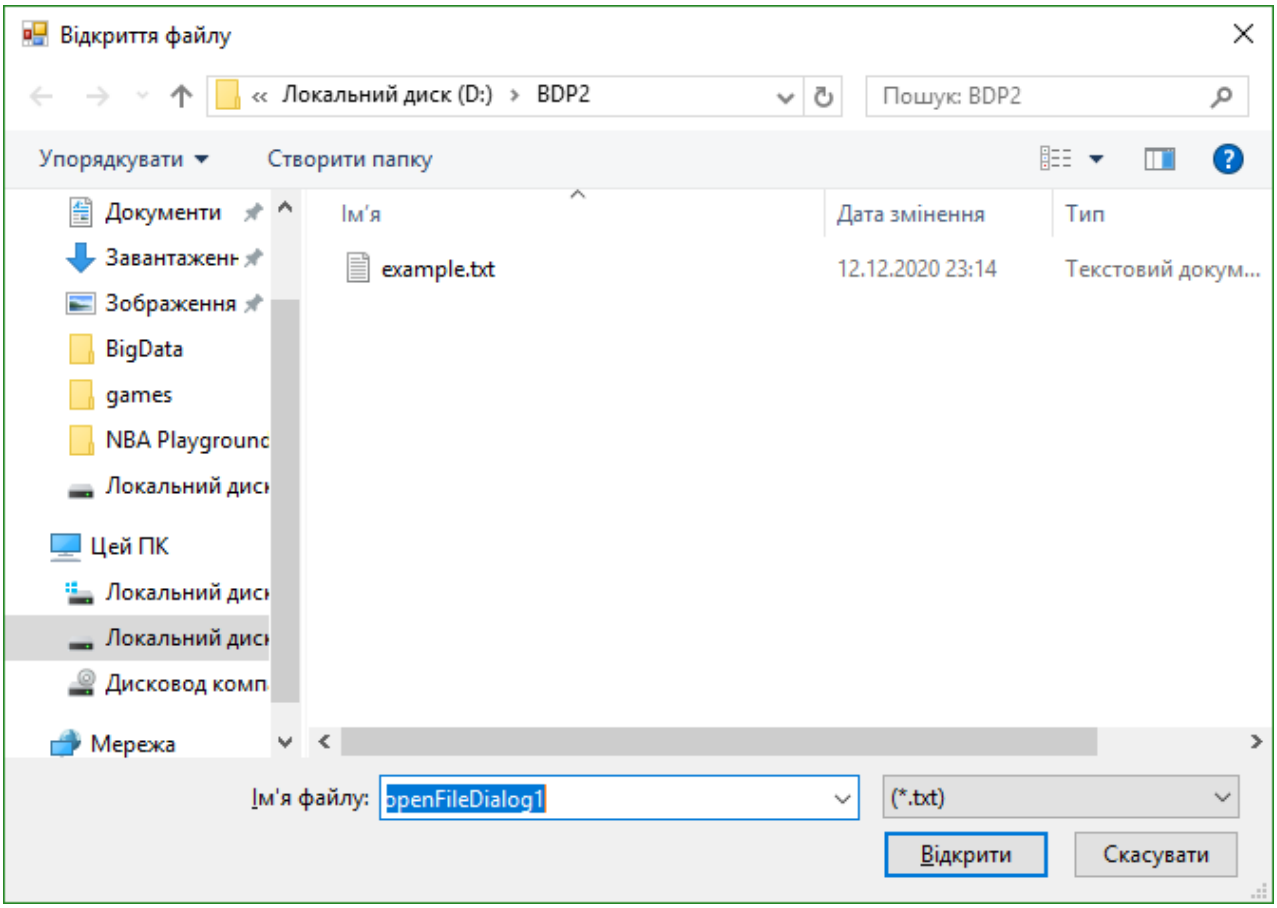

Рис. 3.15. Вікно, котре з'являється після натискання кнопки «Відкрити»

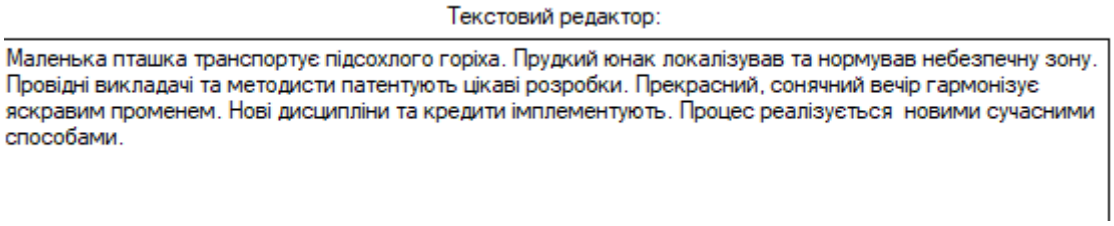

Рис. 3.16. Результат відкриття текстового файлу

Третій варіант для отримання вхідного потоку тексту – комбінований. Він полягає у використанні другого варіанту(відкриття файлу) з можливістю вручну вносити зміни як в першому варіанті. Ще однією особливістю даного програмного модуля являється можливість зберегти зміни у файлі, який був відкритий.

Після натискання клавіші «Сформувати» відбувається основний функціонал даного застосунку – автоматичне формування логіко-лінгвістичних моделей з вхідного потоку текстової інформації(на основі морфологічного аналізу, формування словосполучень та синтаксичного аналізу) з подальшим їхнім виводом на екран в області під надписом «Логіко-лінгвістична модель». Також буде відображено перелік словосполучень, сформованих під час моделювання, під надписом «Словосполучення». Більше того, навпроти кожної сформованої логіко-лінгвістичної моделі, буде відображатися її тип, а в полі під надписом «Тип моделі». Результат натискання клавіші «Сформувати», при наявному вхідному потоці текстової інформації природною мовою, зображено на рис. 3.17.

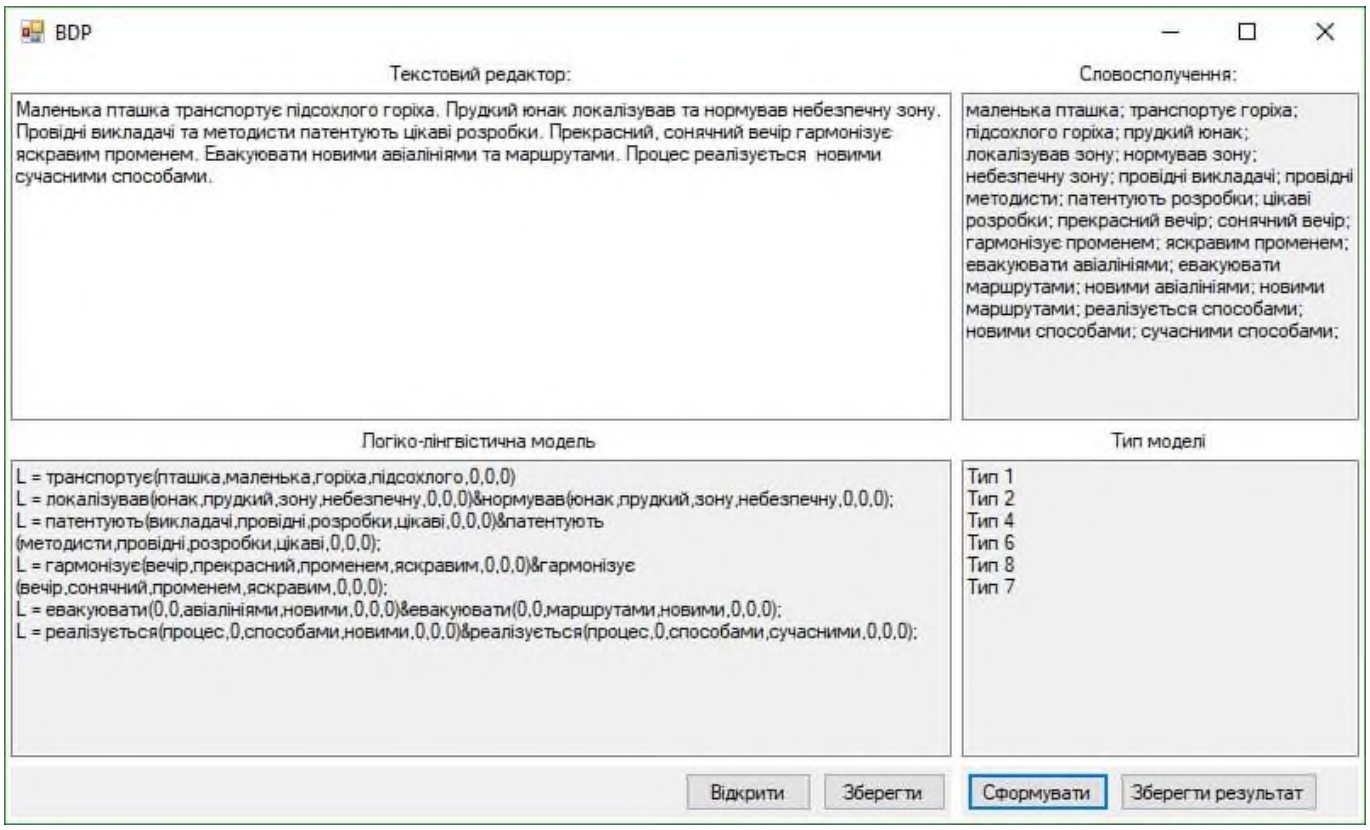

Рис. 3.17. Результат автоматизованого формування логіко-лінгвістичних моделей простих речень

Результат формування логіко-лінгвістичних моделей не піддається ручному коригуванню і його можна зберегти в окремому текстовому файлі, для цього необхідно натиснути клавішу «Зберегти результат», яка розташована з правого боку від клавіші «Сформувати» та під текстовим полем для виводу типу моделі. Вікно збереження вихідного потоку текстової інформації у файлі зображено на рис. 3.18.

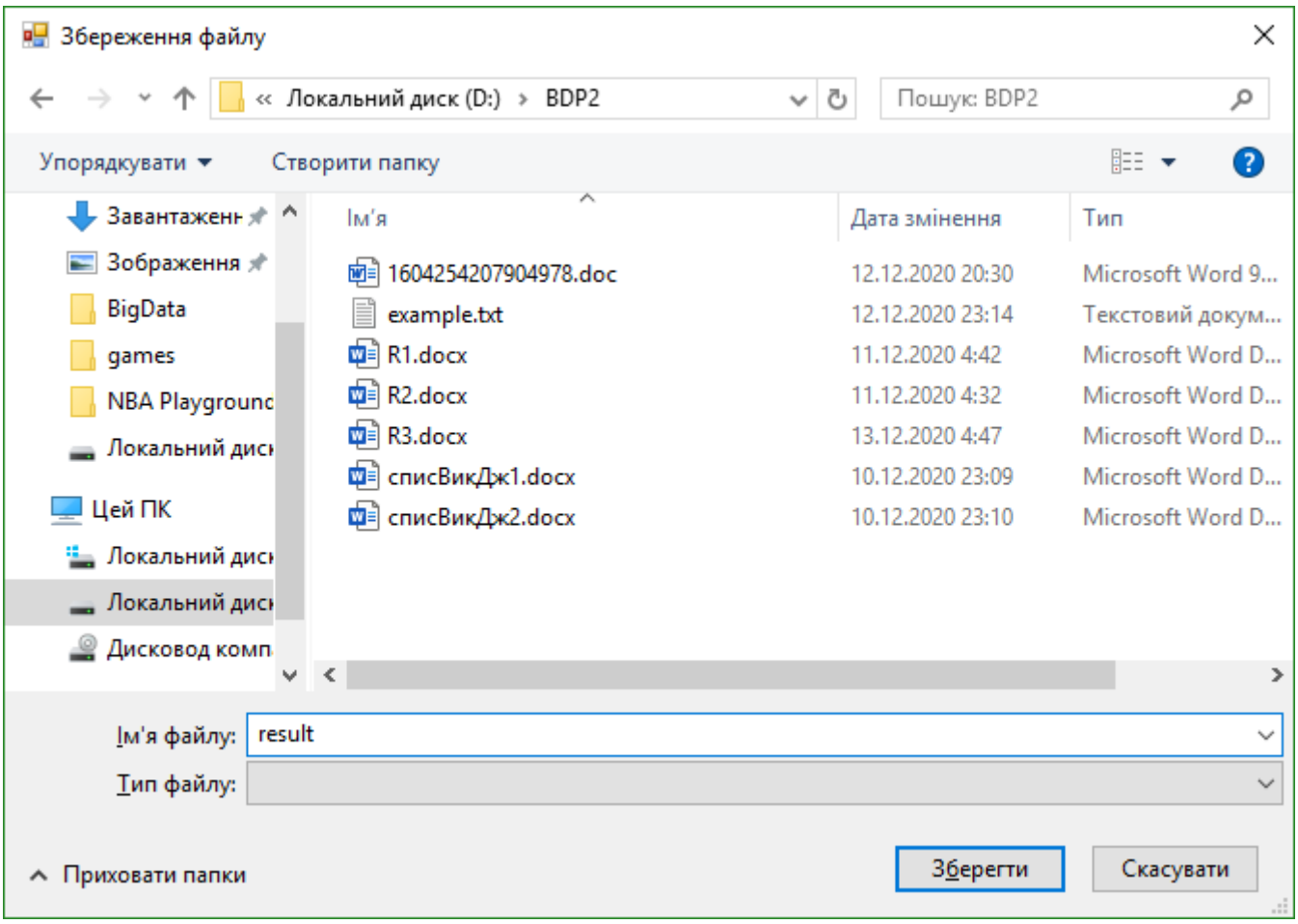

Рис. 3.18. Вікно зберігання результату формування логіко-лінгвістичних моделей

Вміст текстового документа після збереження в ньому результатів формування словосполучень подано на рис. 3.19.

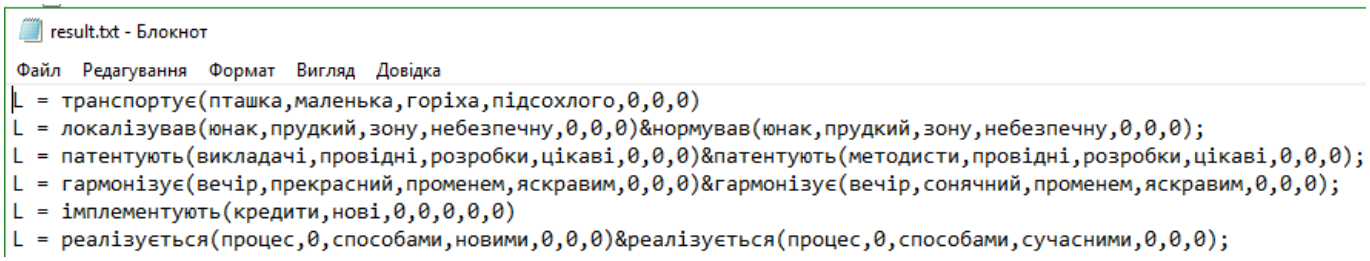

Рис. 3.19. Вміст текстового файлу після збереження результату

Якщо поле під надписом «Текстовий редактор» пусте і була натиснута клавіша «Сформувати», то у полі під надписом «Словосполучення» з’явиться надпис про неможливість екстракції. Результат даних маніпуляцій подано на рис. 3.20.

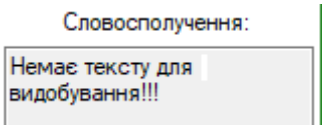

Рис. 3.20. Повідомлення про відсутність вхідного потоку текстової інформації

Лістинг коду програмного модуля генерації змістовних одиниць текстової інформації подано в Додатку А.

Порівняння еталонних показників формування логіко-лінгвістичних моделей речень із результатами наданими програмним модулем подано у Таблиці 3.1.

Таблиця 3.1

Порівняння результатів з еталоном

Речення	Еталон	Результат формування
Маленька пташка транспортує підсохлого горіха.	*L* = транспортує (пташка, маленька, горіха, підсохлого, 0, 0, 0)	*L* = транспортує (пташка, маленька, горіха, підсохлого, 0, 0, 0)
Прудкий юнак локалізував та нормував небезпечну зону.	*L* = локалізував(юнак, прудкий, зону, небезпечну, 0, 0, 0) & нормував(юнак, прудкий, зону, небезпечну, 0, 0, 0)	*L* = локалізував(юнак, прудкий, зону, небезпечну, 0, 0, 0) & нормував(юнак, прудкий, зону, небезпечну, 0, 0, 0)
Провідні викладачі та методисти патентують цікаві розробки.	*L* = патентують (викладачі, провідні, розробки, цікаві, 0, 0, 0) & патентують (методисти, провідні, розробки, цікаві, 0, 0, 0)	*L* = патентують (викладачі, провідні, розробки, цікаві, 0, 0, 0) & патентують (методисти, провідні, розробки, цікаві, 0, 0, 0)

Продовження таблиці 3.1

Прекрасний, сонячний вечір гармонізує яскравим променем.	*L* = гармонізує (вечір, прекрасний, променем, яскравим, 0, 0, 0) & гармонізує (вечір, сонячний, променем, яскрав-им, 0, 0, 0)	*L* = гармонізує (вечір, прекрасний, променем, яскравим, 0, 0, 0) & гармонізує (вечір, сонячний, променем, яскрав-им, 0, 0, 0)
Нові методи та функції імплементують.	*L* = імплементують(0, 0, методи, нові, 0, 0, 0) & імплементують (0, 0, функції, нові, 0, 0, 0)	*L* = імплементують (нові, методи, 0, 0, 0, 0, 0) & імплементують (функції, нові, 0, 0, 0, 0, 0)
Процес реалізується новими сучасними способами.	*L* = реалізується (процес, 0, способами, новими, 0, 0, 0) & реалізується (процес, 0, способами, сучасними, 0, 0, 0);	*L* = реалізується (процес, 0, способами, новими, 0, 0, 0) & реалізується (процес, 0, способами, сучасними, 0, 0, 0);

На рис. 3.21 зображено текстовий файл із реченнями для формування логіко-лінгвістичних моделей для подальшого визначення кількісних показників.

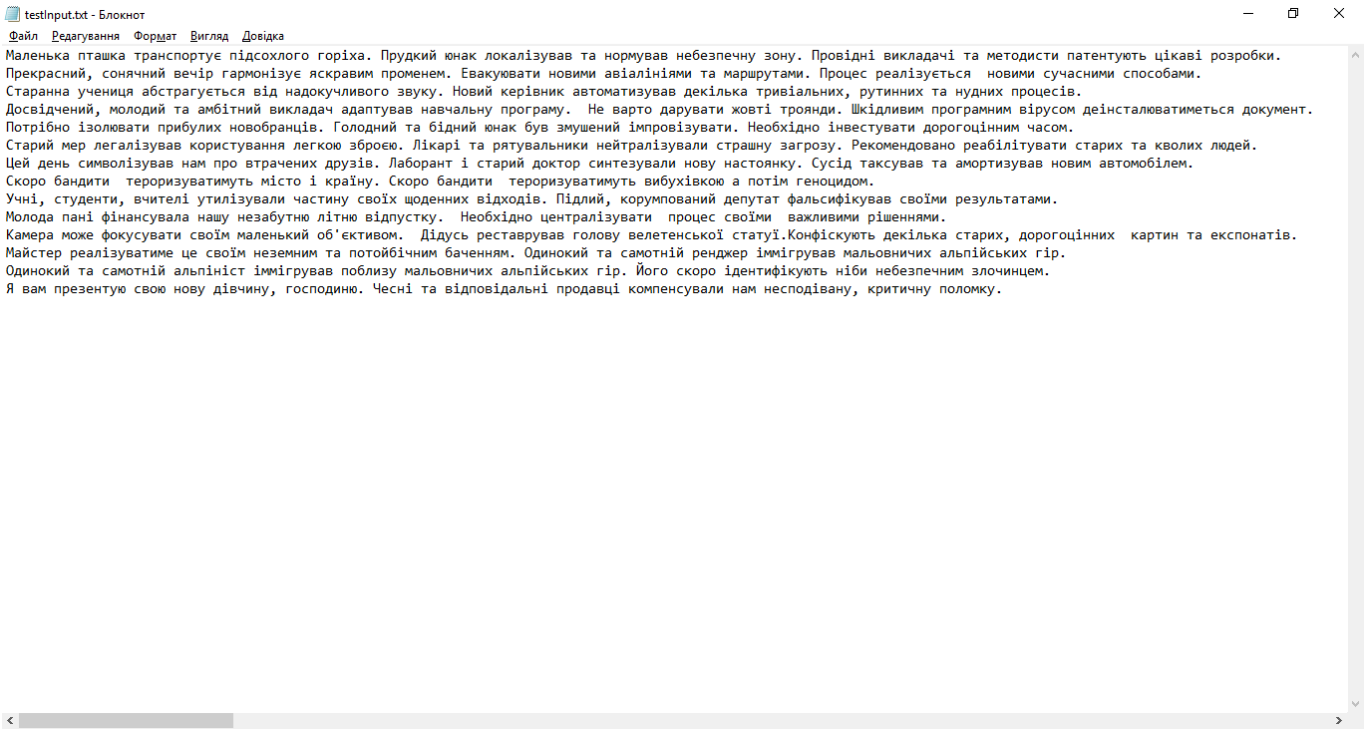

Рис. 3.21. Вхідні дані для тестування

Результат формування логіко-лінгвістичних моделей подано на рис. 3.22.

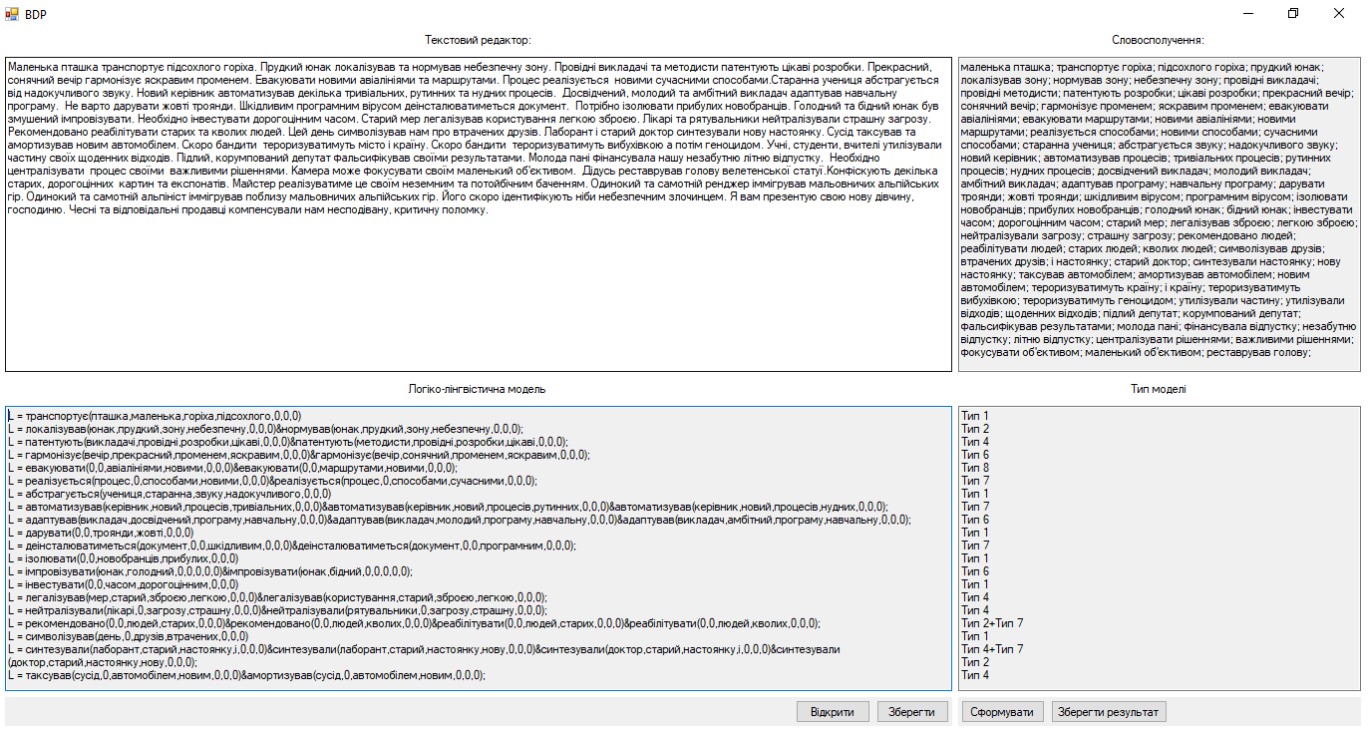

Рис. 3.22. Результат формування логіко-лінгвістичних моделей

На рис. 3.23 зображено вміст текстового документа вміст якого містить результат формування логіко-лінгвістичних моделей.

Рис. 3.23. Вміст текстового документа вміст якого містить результат формування логіко-лінгвістичних моделей

Беручи до уваги дані подані вище, можна зробити висновок, що похибка дорівнює 16%, а відсоток коректності побудови – 86%.

Попри високі результати порівняння з еталонними значеннями дана технологія стикається з деякими недоліками, а саме тривалістю і точністю морфологічного аналізу. Оскільки морфологічний аналіз потребує бази даних всіх частин мови зі всіма формами слів, що в свою чергу збільшує тривалість виконання і будь який недолік пов’язаний з цілісністю бази даних негативно впливатиме на подальші формування словосполучень і синтаксичний аналіз. Також є висока ймовірність повтору різних форм слів, що спричинить похибки в майбутньому.

3.3. Висновки до розділу

Використовуючи засоби *Windows Forms .Net* у середовищі програмної розробки *Visual Studio 2017* було програмно реалізовано технологію автоматизованої побудови логіко-лінгвістичних моделей. Для повного функціонування було використано наступні бібліотеки: для створення вікна додатку і його компонентів – «*System.Windows.Forms*», для роботи з текстовою інформацією – «*System.Text*», для роботи з базою даних(підключення, зчитування даних) – «*System.OleDb*», для роботи з файлами(відкриття текстового документу, збереження нового текстового документу, зчитування з текстового документу).

Програмна реалізація технології автоматизованої побудови логіко-лінгвістичних моделей являє собою вікно на якому наявні: чотири клавіші(для відкриття текстового файлу, для зберігання змін у файлі, для формування логіко-лінгвістичних моделей і для збереження результату), чотири текстові поля(одне для вхідного потоку текстової інформації з можливістю вносити корективи, а три інші для відображення результату, а саме екстракції словосполучень, побудови формул та типів даних формул) і чотири надписи. Програмний модуль може видобувати словосполучення типу «Прикметник + Іменник», «Порядковий числівник + Іменник», «Займенник + Іменник», «Дієприкметник + Іменник», «Дієслово + Іменник», «Прикметник + Дієслово», будувати логіко-лінгвістичні моделі шістьох типів. Аналіз тексту здійснюється зі швидкістю нуль цілих п’ятдесят п’ять сотих слова за секунду.

Технологія автоматизованої побудови логіко-лінгвістичних моделей має наступні недоліки: тривалість морфологічного аналізу, залежність від цілісності і повноти баз даних.

ВИСНОВКИ

Описано наявні на сьогоднішній день системи автоматичного синтаксичного аналізу текстової інформації природньої мови, основні етапи обробки інформації та механізми вилучення знань. Визначено, що вилучення знань найкраще при комбінації морфологічного, синтаксично-семантичного та частотного аналізів. Такий спосіб дає змогу досягнути поставленої цілі враховуючи всі аспекти, необхідні для правильного функціонування.

Розглянуто поняття знання, їхні властивості, моделі їх представлення в електронному вигляді та маніпуляції, котрі можна з ними виконувати. Формування нових знань із вилучених раніше, тобто знання вищого рівня абстракції, можливі лише в системах штучного інтелекту, оскільки вони побудовані на основі моделі людського мислення. Описано шість правил формування словосполучень на основі даних, які є результатом морфологічного аналізу, за якими відбувається екстракція словосполучень в програмному модулі.

Детально розглянуто логіко-лінгвістичну модель представлення знань, котра подає речення природної мови у зрозумілій для комп'ютерних систем формі, у вигляді формули – предикату. Описано вісім типів побудови логіко-лінгвістичних моделей лише для простих речень природньої мови. Розроблено алгоритм побудови логіко-лінгвістичних моделей представлення знань.

Описано правила для визначення елементів предикату – синтаксичних одиниць речення. Для здійснення синтаксичного аналізу спершу слід здійснити морфологічний аналіз, а також екстракцію словосполучень. Відсутність двох останніх унеможливлює визначення членів речення і зв'язків між ними.

Було розроблено програмний модуль котрий реалізує технологію автоматизованої побудови логіко-лінгвістичних моделей, що працює з текстами українською мовою. Під час розробки використовувалися методи роботи з файлами, вікнами та базою даних за допомогою об'єктно орієнтованого програмування. Програмний модуль був розроблений на базі програмного середовища «*Visual Studio 2017*» та комплексом програмних рішень *«.NET Framework»*.

Програмний модуль здатен формувати логіко-лінгвістичні моделі для простих речень української мови, а також видобувати словосполучення типу «Прикметник + Іменник», «Займенник + Іменник», «Дієприкметник + Іменник» і «Порядковий числівник + Іменник», «Прикметник + Дієслово» та «Дієслово + Іменник». Вхідний текст, який можна отримувати шляхом вводу вручну, відкриття текстового файлу чи редагування відкритого тексту, проходить наступні етапи обробки: поділ на речення, поділ речення на слова, кодування слів речення за їхніми морфологічними властивостями та іншими особливостями, перевірка за правилами формування словосполучень, здійснення синтаксичного аналізу, побудова логіко-лінгвістичних моделей, вивід результатів. Після виконання всіх етапів користувачеві відображаються сформовані логіко-лінгвістичні моделі на основі речень, їх типи та словосполучення, які були видобуті з текстової інформації.

Поділ на речення відбувається за наявністю символу крапки. Кодування відбувається за допомогою наявної бази даних, словника термінів української мови, в якій містяться різні форми слів різних частин мови. Порівнюючи з наявними в базі словами визначаються їхні характеристики, що в свою чергу позначаються різними цифрами в матриці кодування слів у реченні. Матриця кодування допомагає із легкістю визначити чи виконується те чи інше правило формування словосполучень що в свою чергу забезпечує визначення синтаксису в реченні і в подальшому забезпечує правильність побудови предикатів – логіко-лінгвістичних моделей.

Було проведено перевірку на швидкість та якість виконання програмного модуля. Як результат даних тестів, було визначено швидкість обробки слів, що дорівнює нуль цілих п’ятдесят п’ять сотих слова за секунду. Похибка становить 16% і спричинена неповнотою бази даних, що в свою чергу унеможливлює морфологічний аналіз, котрий потрібен для формування словосполучень і разом з ними для визначення членів речення і зв речення і зв’язків між ними – синтаксичного аналізу. Також не слід забувати про велику кількість слів у електронному словнику термінів української мови та повторюваність за структурою різних форм слів, що в свою чергу є ще однією причиною тривалості обробки інформації та ймовірних похибок. Для запобігання похибок пов’язаних з повторюваністю різних форм слів слід додати

семантичний аналіз на основі системи штучного інтелекту. Проблема з повнотою бази даних різних форм слів української мови вирішується заміною на нову повноціннішу.

Дана дипломна робота відіграє важливу роль у розвитку аналізу текстової інформації українською мовою, адже такого роду розробки майже не проводяться. На запит в пошуковій системі *Google*: «Синтаксичний аналіз речень українською мовою онлайн», система пошуку видає лише посилання на статті пов’язані із описом процесу знаходження членів речення та на аналізатори, котрі працюють з російською мовою. Дана технологія може слугувати як приклад для дослідження студентів вищих навчальних закладів спеціальностей пов’язаних із інформаційними технологіями в плані не тільки текстового аналізу, а й структури виконання такого роду застосунків. Також даний програмний модуль можна удосконалити, додавши новий функціонал обробки текстової інформації або ж додати як компонент до вже існуючої системи.

СПИСОК БІБЛІОГРАФІЧНИЙХ ПОСИЛАНЬ ВИЕОРИСТАНИХ ДЖЕРЕЛ

1. Палагин А.В., Денисенко Е.Л., Белицкий Р.И., Сигалов В.И. Микропроцессорные системы обработки информации. – Киев: Наук. Думка, 1993. – 352 с.
2. Палагин А.В. К решению основной задачи эмуляции / А.В. Палагин. – УсиМ, 1980. – №3. – С. 24–28.
3. Широков В.А. Елементи лексикографії / В.А. Широков. – К.: Довіра, 2005. – 304 с.
4. Клещёв А.С. Отношения между онтологиями предметных областей. Ч.1. / А. Клещёв, И. Артемьева. – Информационный анализ, Выпуск 1, 2002. – С. 4–9.
5. Марка Д. А. Методология структурного анализа и проектирования / Д.А. Марка, К. МакГоуэн. – М.: "МетаТехнология", 1993. – 239 с.
6. Довгий С.О., Величко В. Ю., Глоба Л. С., Стрижак О. Є., Андрущенко Т. І., Гальченко С. А, Гончар А. В., Гуляєв К. Д., Кудляк В.М., Ляшук К.В., Палагін О. В., Петренко М. Г., Попова М. А., Сидоренко В. І., Слюсаренко О. О., Стус Д.В., Терновой М. Ю. Комп'ютерні онтології та їх використання у навчальному процесі. Теорія і практика [Електронний ресурс]. – 2018. - режим доступу: *https://lib.iitta.gov.ua/10124/1/9.%20%D0%9C%D0%BE%D0%BD%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D1%96%D1%8F%20%D0%A1%D1%82%D1%80%D0%B8%D0%B6%D0%B0%D0%BA_%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%20%D0%BE%D0%BD%D1%82%D0%BE%D0%BB%D0%BE%D0%B3_%D0%B2%D0%B8%D0%B4%D0%B0%D0%BD%D0%BE.pdf* (дата звернення 25.11.2020).
7. О.Бунин Как решить 90% задач *NLP*: пошаговое руководство по обработке естественного языка [Електронний ресурс]. – 2018. - режим доступу: *https://habr.com/ru/company/oleg-bunin/blog/352614/* (дата звернення 26.11.2020).
8. П.Браславский, И.Колычев, Автоматическое реферирование веб-документов с учетом запроса. Грант ООО «Яндекс», 2016. -173 с.

9. Палагин А.В. К решению основной задачи эмуляции / А.В. Палагин. – УсиМ, 1980. – №3. – С. 24–28.
10. Палагин А.В. Организация и функции "языковой" картины мира в смысловой интерпретации ЕЯ-сообщений / А.В. Палагин. – *InformationTheoriesandApplication*, 2000. – Vol. 7, №4. – С. 155–163.
11. Широков В.А. Елементи лексикографії / В.А. Широков. – К.: Довіра, 2005. – 304 с.
12. Клещёв А.С. Отношения между онтологиями предметных областей. Ч.1. / А. Клещёв, И. Артемьева. – Информационный анализ, Выпуск 1, 2002. – С. 4–9.
13. Марка Д. А. Методология структурного анализа и проектирования / Д.А. Марка, К. МакГоуэн. – М.: "МетаТехнология", 1993. – 239 с.
14. Андон Ф.И. Логические модели интеллектуальных информационных систем / Ф.И. Андон, Л.Е. Яшунин, В.И. Резниченко – К.: Наук. Думка, 1999. – 397 с.
15. Палагин А.В. К вопросу проектирования онтолого-управляемой ИС обработки Налимов В.В. Спонтанность сознания: вероятностная теория смыслов и смысловая архитектоника личности / В.В. Налимов – М.: Изд-во «Прометей» МГПИ им. Ленина, 1989. – 288 с.
16. Палагин А.В. Организация и функции "языковой" картины мира в смысловой интерпретации ЕЯ-сообщений / А.В. Палагин. – InformationTheoriesandApplication, 2000. – Vol. 7, №4. – С. 155–163.
17. Палагин А.В. Системная интеграция средств компьютерной техники /А.В. Палагин, Ю.С. Яковлев. – Винница: УНІВЕРСУМ, 2005. – 680 с.
18. Капитонова Ю.В. Парадигмы и идеи академика В.М. Глушкова / Ю.В. Капитонова, А.А. Летичевский. – Киев, Наукова думка, 2003. – 456 с.
19. Башмаков А.И. Интеллектуальные информационные технологии: Учеб. пособие / А.И. Башмаков, И.А. Башмаков. – М.: Изд.-во МГТУ им. Н.Э. Баумана, 2005. – 304 с.
20. Рыков В.В. Управление знаниями / В.В. Рыков – Режим доступа: *http://rykkypc2.narod.ru/part2.doc*. – Дата доступа: 30.11.2020.

21. Павлов В.Т., Руденко К.Ф., Семенов И.С. Логические методы и формы научного познания. – К.: Вища шк., 1984. – 208 с.
22. Вавіленкова А.І. Аналіз і синтез логіко-лінгвістичних моделей речень природної мови: монографія. – К.: ТОВ “СІК ГРУП УКРАЇНА”, 2017. – 152 с.
23. Вавіленкова А.І. Теоретичні основи аналізу електронних текстів: монографія. – К.: ТОВ “СІК ГРУП УКРАЇНА”, 2016. – 192 с.
24. Бойченко С.В., Іванченко О.В. Положення про дипломні роботи (проекти) випускників Національного авіаційного університету. – К.: НАУ, 2017. – 63 с.
25. ДСТУ 3008-95 «Документація. Звіти у сфері науки і техніки. Структура і правила оформлення».
26. ГОСТ 2.301-68. Единая система конструкторской документации. Форматы. – Введ. 2002–01–01. – М. : Вид.-во стандартов, 2006. – 27 с.

Додаток А

Лістинг коду програмного модуля

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;
using System.IO;

namespace BDPr{
   public partial class Form1 : Form{
      //зберігання розташування бази dicua.mdb та провайдера у формі рядка
      public static string connectString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=dicua.mdb;";
      //об'єкт який використовується для підключення
      private OleDbConnection myConnection;
      private string name,   text,word;
```

```
private string[] sentences,
                words;
public Form1()
{
   InitializeComponent();
   //встановлення зв'язку з базоюданих
   myConnection = new OleDbConnection(connectString);
   myConnection.Open();
}

private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
   //роз'єднання зв'язу з базою даних під час завершення роботи
   myConnection.Close();
}

//метод для видобування словосполучень
private void button1_Click(object sender, EventArgs e)
{
   if (textBox1.Text != "")
   {
      textBox2.Clear();
```

```
//поділ вхідного тексту на окремі речення
sentences = textBox1.Text.Split(new char[] { '.' });
//поділ речення на окремі слова та формування матриці кодування слів
for (int a = 0; a < sentences.Length; a++)
{
    if (sentences[a] != "")
    {
        words = sentences[a].Split(new char[] { ' ' });
        int[][] masWords = new int[words.Length][];//матриця кодування слів
                                                   //формування матриці кодування слів
        for (int i = 0; i < words.Length; i++)
        {
            word = words[i].ToLower();
            masWords[i] = new int[7];
            words[i] = word;
            if (word != "")
            {
                if (words[i] == "на" || words[i] == "в" || words[i] == "по" || words[i]
== "у")
                {
                    for (int m = 0; m < 5; m++)
                        masWords[i][m] = 0;
                    if ((i + 1) < words.Length)
```

```
                words[i + 1] = words[i] + ' ' + words[i + 1];
            continue;
        }
        //перевірка на наявність коми після слова
        if (word[word.Length - 1] == ',')
        {
            words[i] = words[i].Trim(new char[] { ',' });
            masWords[i][4] = 1;
        }
        //визначення частини мови
        string sqlcommandIm = "SELECT Field4, Field5, Field6, Field7, Field8, Field9, Field10, " +
            "Field11, Field12, Field13, Field14, Field15, Field16, Field17 FROM imen",  //команда видобування даних з таблиці іменників
              sqlcommandPryk = "SELECT Field4, Field5, Field6, Field7, Field8, Field9, Field10, " +
            "Field11, Field12, Field13, Field14, Field15, Field16, Field17, Field18, Field19," +
            "Field20, Field21, Field22, Field23, Field24, Field25, Field26, Field27 FROM prik",  //команда видобування даних з таблиці прикметників
              sqlcommandPorCh = "SELECT Field4, Field5, Field6, Field7, Field8, Field9, Field10, " +
            "Field11, Field12, Field13, Field14, Field15, Field16, Field17, Field18, Field19," +
```

"Field20, Field21, Field22, Field23, Field24, Field25, Field26, Field27 FROM chispr", //команда видобування даних з таблиці порядкових числівників

sqlcommandDiePr = "SELECT Field4, Field5, Field6, Field7, Field8, Field9, Field10, " +

"Field11, Field12, Field13, Field14, Field15, Field16, Field17, Field18, Field19," +

"Field20, Field21, Field22, Field23, Field24, Field25, Field26, Field27 FROM diepr", //команда видобування даних з таблиці дієприкметників

sqlcommandZaim = "SELECT Field4, Field5, Field6, Field7, Field8, Field9 FROM zajm", //команда видобування даних з таблиці займенників

sqlcommandDie = "SELECT Field4, Field5, Field6, Field7, Field8, Field9, Field10, " +

"Field11, Field12, Field13, Field14, Field15, Field16, Field17, Field18, Field19," +

"Field20, Field21, Field22, Field23, Field24, Field25, Field27, Field28, Field29 FROM die2"; //команда видобування даних з таблиці дієслів

bool find = false;

//перевірка чи іменник

OleDbCommand extractCommand = new OleDbCommand(sqlcommandIm, myConnection);

OleDbDataReader reader = extractCommand.ExecuteReader();

if (!find)

{

while (reader.Read())

```
{
    string compare;
    int k = 0;
    bool endImVidm = false;
    while (k < 14)
    {
        compare = reader[k].ToString();
        compare = compare.ToLower();
        compare = compare.Replace("\"", "");
        //перевірка на місцевий відмінок
        if (k == 5 || k == 12)
        {
            string varMisc;
            compare = compare.Replace(",", "");
            compare = compare.Replace('/', ' ');
            string[] comp = compare.Split(new char[] { ' ' });
            if (comp.Length == 4)
            {
                for (int c = 0; c < 3; c++)
                {
                    varMisc = "";
                    varMisc = comp[c] + ' ' + comp[3];
```

```
            if (varMisc == words[i])
            {
                masWords[i][0] = 1;
                masWords[i][3] = 6;
                if (k == 5)
                    masWords[i][2] = 1;//однина
                else
                    masWords[i][2] = 2;//множина
                endImVidm = true;
                break;
            }
        }
    }
    else if (comp.Length == 5)
    {
        for (int c = 0; c < 3; c++)
        {
            for (int d = 3; d < 5; d++)
            {
                varMisc = "";
                varMisc = comp[c] + ' ' + comp[d];
                if (varMisc == words[i])
```

```
                {
                    masWords[i][0] = 1;
                    masWords[i][3] = 6;
                    if (k == 5)
                        masWords[i][2] = 1;//однина
                    else
                        masWords[i][2] = 2;//множина
                    endImVidm = true;
                    break;
                }
            }
        }
    }
}
//відмінки однини(крім місцевого)
else if ((k < 5 || k == 6) && words[i] == compare)
{
    masWords[i][0] = 1;
    masWords[i][3] = k + 1;
    masWords[i][2] = 1;
    endImVidm = true;
}
```

```
            //відмінки множини(крім місцевого)
            else if ((k < 12 || k == 13) && words[i] == compare)
            {
                masWords[i][0] = 1;
                masWords[i][3] = 7 - (13 - k);
                masWords[i][2] = 2;
                endImVidm = true;
            }
            if (endImVidm)
            {
                find = true;
                break;
            }
            k++;
        }
        if (endImVidm)
        {
            find = true;
            break;
        }
    }
}
```

```
//перевірка чи прикметник
extractCommand = new OleDbCommand(sqlcommandPryk, myConnection);
reader = extractCommand.ExecuteReader();
if (!find)
{
   while (reader.Read())
   {
      string compare;
      int k = 0;
      bool endPrVidm = false;
      while (k < 24)
      {
         compare = reader[k].ToString();
         compare = compare.ToLower();
         compare = compare.Replace("\"", "");
         //перевірка на місцевий відмінок
         if (k == 5 || k == 11 || k == 17 || k == 23)
         {
            string varMisc;
            compare = compare.Replace(",", "");
            compare = compare.Replace('/', ' ');
```

```
string[] comp = compare.Split(new char[] { ' ' });
if (comp.Length == 4)
{
   for (int c = 0; c < 3; c++)
   {
      varMisc = "";
      varMisc = comp[c] + ' ' + comp[3];
      if (varMisc == words[i])
      {
         masWords[i][0] = 2;
         masWords[i][3] = 6;
         if (k == 5)
            masWords[i][2] = 1;//однина
         else
            masWords[i][2] = 2;//множина
         endPrVidm = true;
         break;
      }
   }
}
else if (comp.Length == 5)
{
```

```
for (int c = 0; c < 3; c++)
{
    for (int d = 3; d < 5; d++)
    {
        varMisc = "";
        varMisc = comp[c] + ' ' + comp[d];
        if (varMisc == words[i])
        {
            masWords[i][0] = 2;
            masWords[i][3] = 6;
            if (k == 5)
                masWords[i][2] = 1;//однина
            else
                masWords[i][2] = 2;//множина
            endPrVidm = true;
            break;
        }
    }
    if (endPrVidm)
        break;
}
}
```

```
}
//перевірка на знахідний
if (k == 3 || k == 9 || k == 15 || k == 21)
{
   compare = compare.Replace(",", "");
   string[] comp = compare.Split(new char[] { ' ' });
   for (int c = 0; c < comp.Length; c++)
   {
      if (comp[c] == words[i])
      {
         masWords[i][0] = 2;
         masWords[i][3] = 4;
         if (k == 4)
         {
            masWords[i][1] = 1;//чоловічий рід
            masWords[i][2] = 1;//однина
         }
         else if (k == 9)
         {
            masWords[i][1] = 2;//жіночий рід
            masWords[i][2] = 1;//однина
         }
```

```
            else if (k == 15)
            {
               masWords[i][1] = 3;//середній рід
               masWords[i][2] = 1;//однина
            }
            else if (k == 21)
            {
               masWords[i][1] = 0;//не має роду
               masWords[i][2] = 2;//множина
            }
            endPrVidm = true;
            break;
         }
      }
      if (endPrVidm)
      {
         find = true;
         break;
      }

   }
   //перевірка на інші відмінки
```

```
//прикметники чоловічого роду однини
else if ((k < 3 || k == 4) && words[i] == compare)
{
   masWords[i][0] = 2;
   masWords[i][3] = k + 1;
   masWords[i][1] = 1;
   masWords[i][2] = 1;
   endPrVidm = true;
}
//прикметники жіночого роду однини
else if ((k < 9 || k == 10) && words[i] == compare)
{
   masWords[i][0] = 2;
   masWords[i][3] = 6 - (11 - k);
   masWords[i][1] = 2;
   masWords[i][2] = 1;
   endPrVidm = true;
}
//прикметники середнього роду однини
else if ((k < 15 || k == 16) && words[i] == compare)
{
   masWords[i][0] = 2;
```

```
            masWords[i][3] = 6 - (17 - k);
            masWords[i][1] = 3;
            masWords[i][2] = 1;
            endPrVidm = true;
        }
        //прикметники множини
        else if ((k < 21 || k == 22) && words[i] == compare)
        {
            masWords[i][0] = 2;
            masWords[i][3] = 6 - (23 - k);
            masWords[i][1] = 0;
            masWords[i][2] = 2;
            endPrVidm = true;
        }
        if (endPrVidm)
        {
            find = true;
            break;
        }
        k++;
    }
    if (endPrVidm)
```

```
            {
                find = true;
                break;
            }
        }
    }

    //перевірка чи порядковий числівник
    extractCommand = new OleDbCommand(sqlcommandPorCh, myConnection);
    reader = extractCommand.ExecuteReader();
    if (!find)
    {
        while (reader.Read())
        {
            string compare;
            int k = 0;
            bool endPrChVidm = false;
            while (k < 24)
            {
                compare = reader[k].ToString();
                compare = compare.ToLower();
                compare = compare.Replace("\"", "");
```

```
//перевірка на місцевий відмінок
if (k == 5 || k == 11 || k == 17 || k == 23)
{
    string varMisc;
    compare = compare.Replace(",", "");
    compare = compare.Replace('/', ' ');
    string[] comp = compare.Split(new char[] { ' ' });
    if (comp.Length == 4)
    {
        for (int c = 0; c < 3; c++)
        {
            varMisc = "";
            varMisc = comp[c] + ' ' + comp[3];
            if (varMisc == words[i])
            {
                masWords[i][0] = 3;
                masWords[i][3] = 6;
                if (k == 5)
                    masWords[i][2] = 1;//однина
                else
                    masWords[i][2] = 2;//множина
                endPrChVidm = true;
```

```
                break;
            }
        }
    }
    else if (comp.Length == 5)
    {
        for (int c = 0; c < 3; c++)
        {
            for (int d = 3; d < 5; d++)
            {
                varMisc = "";
                varMisc = comp[c] + ' ' + comp[d];
                if (varMisc == words[i])
                {
                    masWords[i][0] = 3;
                    masWords[i][3] = 6;
                    if (k == 5)
                        masWords[i][2] = 1;//однина
                    else
                        masWords[i][2] = 2;//множина
                    endPrChVidm = true;
                    break;
```

```
                }
            }
            if (endPrChVidm)
                break;
        }
    }
}
//перевірка на знахідний
if (k == 3 || k == 9 || k == 15 || k == 21)
{
    compare = compare.Replace(",", "");
    string[] comp = compare.Split(new char[] { ' ' });
    for (int c = 0; c < comp.Length; c++)
    {
        if (comp[c] == words[i])
        {
            masWords[i][0] = 3;
            masWords[i][3] = 4;
            if (k == 4)
            {
                masWords[i][1] = 1;//чоловічий рід
                masWords[i][2] = 1;//однина
```

```
                }
                else if (k == 9)
                {
                    masWords[i][1] = 2;//жіночий рід
                    masWords[i][2] = 1;//однина
                }
                else if (k == 15)
                {
                    masWords[i][1] = 3;//середній рід
                    masWords[i][2] = 1;//однина
                }
                else if (k == 21)
                {
                    masWords[i][1] = 0;//не має роду
                    masWords[i][2] = 2;//множина
                }
                endPrChVidm = true;
                break;
            }
        }

    }
```

```
//перевірка на інші відмінки
//порядкові числівники чоловічого роду однини
else if ((k < 3 || k == 4) && words[i] == compare)
{
   masWords[i][0] = 3;
   masWords[i][3] = k + 1;
   masWords[i][1] = 1;
   masWords[i][2] = 1;
   endPrChVidm = true;
}
//порядкові числівники жіночого роду однини
else if ((k < 9 || k == 10) && words[i] == compare)
{
   masWords[i][0] = 3;
   masWords[i][3] = 6 - (11 - k);
   masWords[i][1] = 2;
   masWords[i][2] = 1;
   endPrChVidm = true;
}
//порядкові числівники середнього роду однини
else if ((k < 15 || k == 16) && words[i] == compare)
{
```

```
            masWords[i][0] = 3;
            masWords[i][3] = 6 - (17 - k);
            masWords[i][1] = 3;
            masWords[i][2] = 1;
            endPrChVidm = true;
        }
        //порядкові числівники множини
        else if ((k < 21 || k == 22) && words[i] == compare)
        {
            masWords[i][0] = 3;
            masWords[i][3] = 6 - (23 - k);
            masWords[i][1] = 0;
            masWords[i][2] = 2;
            endPrChVidm = true;
        }
        if (endPrChVidm)
            break;
        k++;
    }
    if (endPrChVidm)
    {
        find = true;
```

```
                break;
            }
        }
    }
    //перевірка чи дієприкметник
    extractCommand = new OleDbCommand(sqlcommandDiePr, myConnection);
    reader = extractCommand.ExecuteReader();
    if (!find)
    {
        while (reader.Read())
        {
            string compare;
            int k = 0;
            bool endPrVidm = false;
            while (k < 24)
            {
                compare = reader[k].ToString();
                compare = compare.ToLower();
                compare = compare.Replace("\"", "");
                //перевірка на місцевий відмінок
                if (k == 5 || k == 11 || k == 17 || k == 23)
                {
```

```
string varMisc;
compare = compare.Replace(",", "");
compare = compare.Replace('/', ' ');
string[] comp = compare.Split(new char[] { ' ' });
if (comp.Length == 4)
{
    for (int c = 0; c < 3; c++)
    {
        varMisc = "";
        varMisc = comp[c] + ' ' + comp[3];
        if (varMisc == words[i])
        {
            masWords[i][0] = 4;
            masWords[i][3] = 6;
            if (k == 5)
                masWords[i][2] = 1;//однина
            else
                masWords[i][2] = 2;//множина
            endPrVidm = true;
            break;
        }
    }
```

```
}
else if (comp.Length == 5)
{
    for (int c = 0; c < 3; c++)
    {
        for (int d = 3; d < 5; d++)
        {
            varMisc = "";
            varMisc = comp[c] + ' ' + comp[d];
            if (varMisc == words[i])
            {
                masWords[i][0] = 4;
                masWords[i][3] = 6;
                if (k == 5)
                    masWords[i][2] = 1;//однина
                else
                    masWords[i][2] = 2;//множина
                endPrVidm = true;
                break;
            }
        }
        if (endPrVidm)
```

```
                break;
            }
        }
    }
    //перевірка на знахідний
    if (k == 3 || k == 9 || k == 15 || k == 21)
    {
        compare = compare.Replace(",", "");
        string[] comp = compare.Split(new char[] { ' ' });
        for (int c = 0; c < comp.Length; c++)
        {
            if (comp[c] == words[i])
            {
                masWords[i][0] = 4;
                masWords[i][3] = 4;
                if (k == 4)
                {
                    masWords[i][1] = 1;//чоловічий рід
                    masWords[i][2] = 1;//однина
                }
                else if (k == 9)
                {
```

```
                masWords[i][1] = 2;//жіночий рід
                masWords[i][2] = 1;//однина
            }
            else if (k == 15)
            {
                masWords[i][1] = 3;//середній рід
                masWords[i][2] = 1;//однина
            }
            else if (k == 21)
            {
                masWords[i][1] = 0;//не має роду
                masWords[i][2] = 2;//множина
            }
            endPrVidm = true;
            break;
        }
    }
}
//перевірка на інші відмінки
//дієприкметники чоловічого роду однини
else if ((k < 3 || k == 4) && words[i] == compare)
{
```

```
    masWords[i][0] = 4;
    masWords[i][3] = k + 1;
    masWords[i][1] = 1;
    masWords[i][2] = 1;
    endPrVidm = true;
}
//дієприкметники жіночого роду однини
else if ((k < 9 || k == 10) && words[i] == compare)
{
    masWords[i][0] = 4;
    masWords[i][3] = 6 - (11 - k);
    masWords[i][1] = 2;
    masWords[i][2] = 1;
    endPrVidm = true;
}
//дієприкметники середнього роду однини
else if ((k < 15 || k == 16) && words[i] == compare)
{
    masWords[i][0] = 4;
    masWords[i][3] = 6 - (17 - k);
    masWords[i][1] = 3;
    masWords[i][2] = 1;
```

```
            endPrVidm = true;
        }
        //дієприкметники множини
        else if ((k < 21 || k == 22) && words[i] == compare)
        {
            masWords[i][0] = 4;
            masWords[i][3] = 6 - (23 - k);
            masWords[i][1] = 0;
            masWords[i][2] = 2;
            endPrVidm = true;
        }
        if (endPrVidm)
            break;
        k++;
    }
    if (endPrVidm)
    {
        find = true;
        break;
    }
  }
}
```

```
//перевірка чи займенник
extractCommand = new OleDbCommand(sqlcommandZaim, myConnection);
reader = extractCommand.ExecuteReader();
if (!find)
{
   while (reader.Read())
   {
      string compare;
      int k = 0;
      bool endImVidm = false;
      while (k < 6)
      {
         compare = reader[k].ToString();
         compare = compare.ToLower();
         compare = compare.Replace("\"", "");
         if (k == 1 || k == 3 || k == 4)
         {
            compare = compare.Replace(",", "");
            string[] comp = compare.Split(new char[] { ' ' });
            for (int c = 0; c < comp.Length; c++)
            {
               if (comp[c] == words[i])
```

```
        {
            masWords[i][0] = 5;
            masWords[i][3] = k + 1;
            endImVidm = true;
            break;
        }
    }
}
//перевірка на місцевий відмінок
else if (k == 5)
{
    string varMisc;
    compare = compare.Replace(",", "");
    compare = compare.Replace(";", "");
    compare = compare.Replace('/', ' ');
    string[] comp = compare.Split(new char[] { ' ' });
    if (comp.Length == 2)
    {
        for (int c = 0; c < comp.Length; c++)
        {
            if (comp[c] == words[i])
            {
```

```
            masWords[i][0] = 5;
            masWords[i][3] = 6;
            endImVidm = true;
            break;
        }
    }
}
else if (comp.Length == 4)
{
    for (int c = 0; c < 3; c++)
    {
        varMisc = "";
        varMisc = comp[c] + ' ' + comp[3];
        if (varMisc == words[i])
        {
            masWords[i][0] = 5;
            masWords[i][3] = 6;
            endImVidm = true;
            break;
        }
    }
}
```

```
else if (comp.Length == 5)
{
    for (int c = 0; c < 3; c++)
    {
        for (int d = 3; d < 5; d++)
        {
            varMisc = "";
            varMisc = comp[c] + ' ' + comp[d];
            if (varMisc == words[i])
            {
                masWords[i][0] = 5;
                masWords[i][3] = 6;
                endImVidm = true;
                break;
            }
        }
        if (endImVidm)
            break;
    }
}
else if (comp.Length == 7)
{
```

```
        for (int c = 0; c < 3; c++)
        {
            for (int d = 3; d < 7; d++)
            {
                varMisc = "";
                varMisc = comp[c] + ' ' + comp[d];
                if (varMisc == words[i])
                {
                    masWords[i][0] = 5;
                    masWords[i][3] = 6;
                    endImVidm = true;
                    break;
                }
            }
            if (endImVidm)
                break;
        }
    }
}
//інші відмінки
else if ((k == 0 || k == 2) && words[i] == compare)
{
```

```
                masWords[i][0] = 5;
                masWords[i][3] = k + 1;
                endImVidm = true;
            }
            if (endImVidm)
                break;
            k++;
        }
        if (endImVidm)
        {
            find = true;
            break;
        }
    }
}
//перевірка чи дієслово
extractCommand = new OleDbCommand(sqlcommandDie, myConnection);
reader = extractCommand.ExecuteReader();
bool end = false;
if (!find)
{
    while (reader.Read())
```

```
    {
        string compare;
        for (int w = 0; w < 25; w++)
        {
            compare = reader[w].ToString();
            compare = compare.ToLower();
            compare = compare.Replace("\"", "");
            if (compare == words[i])
            {
                masWords[i][0] = 6;
                end = true;
                break;
            }
        }
        if (end)
        {
            find = true;
            break;
        }
    }
}
```

```
        //визначення синтаксису
        if (masWords[i][0] == 1 && masWords[i][3] == 1)
        {
            masWords[i][5] = 1;
            masWords[i][6] = 1;
        }
        else if (masWords[i][0] == 1 && masWords[i][3] != 1)
            masWords[i][5] = 3;
        else if (masWords[i][0] == 6)
        {
            masWords[i][5] = 2;
            masWords[i][6] = 2;
        }
        else if (masWords[i][0] >= 2 && masWords[i][0] <= 4)
            masWords[i][5] = 4;
        else
            masWords[i][5] = 0;
    }
}
//перевірка за першим правилом
for (int i = 0; i < words.Length; i++)
{
```

```
                if (masWords[i][0] == 2 || masWords[i][0] == 3 || masWords[i][0] ==
4)//якщо прикметник
                {
                    for (int j = i + 1; j < words.Length; j++)
                    {
                        if (masWords[j][0] == 1)//якщо іменник
                        {
                            if (masWords[j][2] == masWords[i][2] && masWords[j][3] ==
masWords[i][3])//однакові рід, число, відмінок
                            {
                                if (masWords[i][5] == 4 && masWords[j][5] == 1)
                                    masWords[i][6] = 5;
                                else
                                    masWords[i][6] = 6;
                                textBox2.Text += words[i] + " " + words[j] + "; ";//додати
словосполучення
                                if (j + 1 < words.Length)//перевірка чи не має іменників після
даного
                                {
                                    if (masWords[j][4] == 1 || masWords[j + 1][0] != 6)
                                    {
                                        continue;
                                    }
```

```
                else
                    break;
                }
                else
                    break;
            }
            if (masWords[j][3] != 1 && masWords[i][6] != 5) //іменник не в називному відмінку
            {
                masWords[i][6] = 6;
                //словосполучення ознака + додаток(q-6)
                textBox2.Text += words[i] + " " + words[j] + "; ";//додати словосполучення//словосполучення присудок + додаток
            }
        }
    }
}
else if (masWords[i][0] == 6 )//якщо дієслово
{
    for (int j = i + 1; j < words.Length; j++)
    {
        if (masWords[j][0] == 1 && masWords[j][3] != 1) //іменник не в називному відмінку
```

```
                {
                    //словосполучення присудок + додаток(у-3)
                    masWords[j][6] = 3;
                    textBox2.Text += words[i] + " " + words[j] + "; ";//додати
словосполучення
                }
            }
        }
    }
    //формування логіко-лінгвістичної моделі речення
    int[] masL = new int[7];
    int type = 0, part = 0, aa = 1, bb = 0;
    for(int s = 0; s < words.Length; s++)
    {
        if ((masWords[s][6] - 1) >= 0)
            masL[masWords[s][6] - 1]++;

    }
    for (int s = 0; s < masL.Length; s++)
    {
        if (masL[s] >= 2)
            part++;
    }
```

```
if (part == 0)
   part = 1;
int[] masL2 = new int[masL.Length];
for(int v = 0; v<masL.Length;v++)
{
   if (masL[v] != 0)
      masL2[v] = masL[v];
   else
      masL2[v] = 1;
}
String[] x = new String[masL2[0]];
String[] p = new String[masL2[1]];
String[] y = new String[masL2[2]];
String[] g = new String[masL2[4]];
String[] q = new String[masL2[5]];
int l = 0;
x[0] = "0";
y[0] = "0";
p[0] = "0";
g[0] = "0";
q[0] = "0";
for (int f = 0; f < words.Length; f++)
```

```
{
        if (masWords[f][6] == 1)
        {
            x[l] = words[f];
            l++;
        }
}
l = 0;
for (int f = 0; f < words.Length; f++)
{
    if (masWords[f][6] == 2)
    {
        p[l] = words[f];
        l++;
    }
}
l = 0;
for (int f = 0; f < words.Length; f++)
{
    if (masWords[f][6] == 3)
    {
        y[l] = words[f];
```

```
        l++;
    }
}
l = 0;
for (int f = 0; f < words.Length; f++)
{
    if (masWords[f][6] == 5)
    {
        g[l] = words[f];
        l++;
    }
}
l = 0;
for (int f = 0; f < words.Length; f++)
{
    if (masWords[f][6] == 6)
    {
        q[l] = words[f];
        l++;
    }
}
textBox3.Text += "L = ";
```

```
for (int t = 0; t < part; t++)
{
    bb = 0;
    for (int i = 0; i < masL.Length; i++)
    {
        if (masL[i] >= 2)
        {
            type = i + 1;
            ++bb;
        }
      if (aa == bb)
        break;
    }
    aa++;
    if (type == 0)//mun 1
    {
      if (part > 1&& t!=0)
        textBox4.Text += "+";
      textBox4.Text += "Tun 1";
      if(t+1 == part)
        textBox4.Text += Environment.NewLine;
      if (part == 0)
```

```
                textBox3.Text += p[0] + '(' + x[0] + ',' + g[0] + ',' + y[0] + ',' +
q[0] + ',' + 0 + ',' + 0 + ',' + 0 + ')' + Environment.NewLine;
            else
                textBox3.Text += p[0] + '(' + x[0] + ',' + g[0] + ',' + y[0] + ',' +
q[0] + ',' + 0 + ',' + 0 + ',' + 0 + ')' + Environment.NewLine;
        }
        else if (type == 1)//mun 4
        {
            if (part > 1  && t != 0)
                textBox4.Text += "+";
            textBox4.Text += "Tun 4";
            if (t + 1 == part)
                textBox4.Text += Environment.NewLine;
        }
        else if (type == 2)//mun 2
        {
            if (part > 1&& t != 0)
                textBox4.Text += "+";
            textBox4.Text += "Tun 2";
            if (t + 1 == part)
                textBox4.Text += Environment.NewLine;
        }
        else if (type == 3)//mun 8
```

```
{
    if (part > 1 && t != 0)
        textBox4.Text += "+";
    textBox4.Text += "Tun 8";
    if (t + 1 == part)
        textBox4.Text += Environment.NewLine;
}
else if (type == 4)//mun 5
{
    if (part > 1 && t != 0)
        textBox4.Text += "+";
    textBox4.Text += "Tun 5";
    if (t + 1 == part)
        textBox4.Text += Environment.NewLine;
}
else if (type == 5)//mun 6
{
    if (part > 1 && t != 0)
        textBox4.Text += "+";
    textBox4.Text += "Tun 6";
    if (t + 1 == part)
        textBox4.Text += Environment.NewLine;
```

```
    }
    else if (type == 6)//mun 7
    {
        if (part > 1 && t != 0)
            textBox4.Text += "+";
        textBox4.Text += "Tun 7";
        if (t + 1 == part)
            textBox4.Text += Environment.NewLine;
    }
    else if (type == 7)//mun 3
    {
        textBox4.Text += "Tun 3\n";
    }
}
int ix = 0, ip = 0, iy = 0, iq = 0, ig = 0;
if (type != 0)
{
    do
    {
        iy = 0;
        while (true)
        {
```

```
ip = 0;
while (true)
{
    iq = 0;
    while (true)
    {
        ig = 0;
        while (true)
        {
            textBox3.Text += p[ip] + '(' + x[ix] + ',' + g[ig] + ',' + y[iy]
+ ',' + q[iq] + ',' + 0 + ',' + 0 + ',' + 0 + ")";
            if (ix == x.Length - 1 && ip == p.Length - 1 && iy ==
y.Length - 1 && ig == g.Length - 1 && iq == q.Length - 1)
            {
                textBox3.Text += ";" + Environment.NewLine;
                break;
            }
            else
                textBox3.Text += '&';
            if (ig < g.Length && ig + 1 != g.Length)
                ig++;
            else
                break;
```

```
            }
            if (iq < q.Length && iq + 1 != q.Length)
                iq++;
            else
                break;
        }
        if (ip < p.Length && ip + 1 != p.Length)
            ip++;
        else
            break;
    }
    if (iy < y.Length && iy + 1 != y.Length)
        iy++;
    else
        break;
}
if (ix < x.Length && ix + 1 != x.Length)
    ix++;
else
    break;
}
while (true);
```

```
            }
        }
    }
}
else
    textBox2.Text = "Не має тексту для видобування!!!";
}
//метод для відкриття текстового документу
private void button3_Click(object sender, EventArgs e)
{
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        name = openFileDialog1.FileName;
        textBox1.Clear();
        textBox1.Text = File.ReadAllText(name);
    }
}
//метод для збереження змін текстового документу
private void button2_Click(object sender, EventArgs e)
{
    if (saveFileDialog1.ShowDialog() == DialogResult.OK)
    {
```

```
            name = saveFileDialog1.FileName;
            File.WriteAllText(name, textBox1.Text);
        }
    }
    //метод для збереження словосполучень у txt файлі
    private void button4_Click(object sender, EventArgs e)
    {
        if (saveFileDialog1.ShowDialog() == DialogResult.OK)
        {
            name = saveFileDialog1.FileName;
            File.WriteAllText(name, textBox3.Text);
        }
    }
  }
}
```