КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

Кваліфікаційна робота

на здобуття ступеня магістра

за спеціальністю 121 Інженерія програмного забезпечення

на тему:

Інтелектуальна консультаційна система по організації освітнього процесу в університеті для студентів

Виконала студентка 2-го курсу магістратури

Хміль Наталія Володимирівна _____

Науковий керівник:

доцент, кандидат фіз.-мат. наук

Шкільняк Оксана Степанівна _____

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Роботу розглянуто й допущено до захисту на засіданні кафедри інтелектуальних програмних систем,

протокол № 12 від 12 травня 2021р.

Завідувач кафедри

проф. Провотар О. І. _____

Київ – 2021

РЕФЕРАТ

Об'єктом роботи є дослідження методів обробки природної мови та систем питань-відповідей. Предметом роботи є системи питань-відповідей, що базуються на пошуку інформації.

Метою роботи є розробка інтелектуальної консультаційної системи по організації навчального процесу, здатної знаходити відповідь на питання, сформульоване природною мовою, а також розробка методу обробки запитань та відповідей для реалізації даної системи.

Інструменти розроблення: мова програмування Python 3, середовище розробки PyCharm, бібліотеки для обробки природної мови gensim, transformers, NLP UK.

Результати роботи: в роботі було запропоновано метод обробки питань та кандидатів на відповідь та оцінки відповідності між ними, розроблено систему відповідей на питання та реалізовано запропоновані підходи в межах даної системи.

Обсяг роботи 55 сторінок, 8 ілюстрацій, 1 таблиця, 31 джерело посилань.

Ключові слова: АЛГОРИТМИ ПОШУКУ ІНФОРМАЦІЇ, ОБРОБКА ПИТАНЬ-ВІДПОВІДЕЙ, ОБРОБКА ПРИРОДНОЇ МОВИ, СЕМАНТИЧНІ ДЕРЕВА ЗАЛЕЖНОСТЕЙ, СИСТЕМИ ОБРОБКИ ПРИРОДНОЇ МОВИ, СИСТЕМИ ПИТАНЬ-ВІДПОВІДЕЙ, СИСТЕМИ ПОШУКУ ІНФОРМАЦІЇ.

ЗМІСТ

ВСТУП

Мова є основним інструментом в житті людини. Вона дозволяє висловлювати ідеї, інформацію, емоції та думки, а також переконувати чи запитувати інформацію.

Інтерес до природної мови з точки зору комп'ютерних наук та інформатики з'явився від самого початку розвитку цих галузей, особливо це актуально для сфери розробки інтелектуальних систем та штучного інтелекту. Згідно з одним із перших тестів для визначення того, чи може машина вважатися розумною (тестом Тьюрінга), вона для цього повинна мати здатність до комунікації близькою до людської. Тобто система повинна бути здатною розуміти та генерувати природну мову.

Розрізняють дві основні концепції: комп'ютерна лінгвістика та обробка природної мови. Комп'ютерна лінгвістика стосується дослідження лінгвістичної теорії за допомогою комп'ютерів та обчислювальних методів, тоді як сфера обробки природної мови передбачає розробку додатків чи систем для обробки тексту та вирішення певних задач з практичної точки зору. Іншими словами, комп'ютерна лінгвістика – це наука, а обробка природної мови є множиною усіх її технологічних наслідків.

На сьогоднішній день дедалі збільшується популярність алгоритмів машинного навчання для обробки тексту. Проте, не дивлячись на їх високу ефективність для багатьох задач та підзадач обробки природної мови, таких, як розмітка частин мови, машинний переклад, категоризація тексту тощо, глибоке розуміння лінгвістичної системи конкретної мови, що обробляється, є необхідною умовою для розробки якісної системи. Окрім того, використання алгоритмів машинного навчання вимагає величезної кількості даних, необхідних для навчання, що ускладнює їх застосування у практичних умовах через високі витрати на розробку анотованих ресурсів.

Система відповідей на питання – це класичний застосунок систем обробки природної мови, що має очевидне практичне значення та представляє особливі інтелектуальні виклики.

Для того, щоб відповісти на звичайне запитання, сформульоване у вільній формі природною мовою потрібно зрозуміти, який тип запитань ставиться та мати уявлення про процес побудови питань в межах тієї чи іншої мови. Обмеження включають семантичні та синтаксичні знання та обрамлення такого роду питань із можливими правильними відповідями.

Питання задаються та отримують відповіді щодня. Технологія відповідей на запитання має на меті забезпечити те ж саме за допомогою ком'ютера. Вона йде далі, ніж більш звичний пошук за ключовими словами, як у пошукових системах, намагаючись розпізнати, що висловлює запитання, і відповісти фактичною відповіддю.

На сьогоднішній день велика кількість задач обробки природної мови вирішені теоретично, проте не реалізовані в практичних системах для переважної більшості природних мов. Серед основних причин можна виділити нестачу розроблених лінгвістичних ресурсів, анотованих даних та якісних моделей для більшості мов, що, очевидно, пов'язано зі світовою тенденцією того, що більшість зусиль на поповнення бази цих ресурсів направлені на невелику множину найбільш поширених мов, лідером серед яких є англійська мова.

Знання правил норм та положень щодо організації освітнього процесу є важливим аспектом для кожного його учасника, в тому числі для студентів. У зв'язку з цим є актуальною розробка консультаційної системи, здатної надати якомога більш точну та конкретну відповідь на запитання сформульоване природною мовою, керуючись текстом положення про організацію освітнього процесу.

Така система дозволяє швидко отримувати необхідну інформацію замість довгого пошуку по об'ємному тексту документу вручну, що може бути корисно для вирішення, наприклад, спірних питань чи ситуацій, або ж просто отримання студентами інформації консультаційного характеру з приводу того чи іншого питання, що стосується навчального процесу.

Розробка такої системи вимагає залучення методів та інструментів обробки природної мови для аналізу тексту документу, а також питання. Також, у зв'язку з тим, що більшість наявних методів та алгоритмів орієнтовані на англійську мову, є необхідною адаптація існуючих чи розробка нових підходів, застосовних до української мови.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ ОБРОБКИ ПРИРОДНОЇ МОВИ

Обробка природної мови включає ряд різноманітних технік та методів. Серед них – розпізнавання, обробка або генерація таких мовних одиниць, як фонеми, склади, слова чи речення.

Аналіз природної мови може виконуватися на різних лінгвістичних рівнях, включаючи морфологічний, синтаксичний та семантичний аналіз. Ці типи аналізу природної мови, а також їх основні поняття та підходи розглядаються в даному розділі.

1.1. Морфологічний аналіз.

Морфологія – це наука, що вивчає слова та словоформи, які складаються з морфем. Згідно з [1], морфема – це найменша лінгвістична одиниця, що може мати певне значення. Кожне слово складається з однієї або декількох морфем. В морфології розрізняються два типи морфем: лексичні морфеми, які позначають загальні об'єкти, та граматичні морфеми, які пов'язані зі словами, що відіграють граматичну роль в реченні чи фразі (наприклад прийменники, займенники тощо).

Як описано в [1, 2], розрізняють наступні операції утворення морфологічних комбінацій, що використовуються для формування слів із морфем та автоматичного розпізнавання словоформ:

— *словозміна* – це процес зміни форми слова, при якому основа слова комбінується з граматичною морфемою для узгодження різних синтаксичних умов, таких як число, час і рід,

— *деривація* – це процес зміни форми слова, при якому слово об'єднується з афіксом для отримання слова, що належить до іншої семантичної категорії,

— *словоскладення* – це процес словотворення шляхом об'єднання однієї або більше основ слова в нову словоформу,

— *контамінація* – це процес словотворення шляхом поєднання початку одного слова з закінченням іншого.

1.1.1. Стемінг. Стемінг – це найпростіша техніка в морфологічній обробці. Вона дозволяє звести слово до його кореневої форми (основи). Метою стемінгу є визначення наближеного значення слова. Згідно з [1–5] це дозволяє виділити концептуально схожі, проте морфологічно різні слова.

Існують різні підходи до процесу стемінгу. В цьому розділі розглядається основна ідея деяких з них, що описуються в [1, 2, 6].

Стохастичний підхід шукає межі між морфемами на основі розподілу фонем, що спостерігаються в корпусі. Основним критерієм є кількість літер, які ймовірно відповідають поточній послідовності в корпусі. [1]

Кластеризація на основі N-грам – це підхід, що базується на кластеризації морфологічно подібних слів. Основна ідея даного підходу полягає в отриманні n-грам кожного слова в корпусі та обчисленні їх схожості з іншими словами, використовуючи наступний коефіцієнт [1]:

$$similarity = \frac{2Z}{X + Y}$$

де:

— Z – кількість унікальних біграм, спільних для обох слів;

— X – кількість унікальних біграм у першому слові;

— Y – кількість унікальних біграм у другому слові.

Також для виконання стемінгу можуть бути використані **регулярні вирази** шляхом визначення шаблонів для перетворення слів або розпізнавання різних форм слів під час пошуку у корпусі.

Згідно з [6–10], регулярні вирази можна визначити наступним чином:

Означення 1.1. Нехай Σ заданий алфавіт, тоді:

— $\varnothing$, ϵ – регулярні вирази;

— Якщо $a \in \Sigma$ – літера алфавіту, тоді a – регулярний вираз;

— Якщо r_1 та r_2 – регулярні вирази, тоді $(r_1 + r_2)$ та $(r_1 \cdot r_2)$ також регулярні вирази;

— Якщо r – регулярний вираз, то $(r)^*$ – теж регулярний вираз.

Означення 1.2. Нехай r – регулярний вираз, тоді його позначення $[[r]]$ – це множина рядків, що визначаються наступним чином:

— $[[\varnothing]] = \{\}$ – це порожня множина;

— $[[\epsilon]] = \{\epsilon\}$ – одинична множина, що містить порожній рядок;

— Якщо $a \in \Sigma$ – літера алфавіту, то $[[a]] = \{a\}$ – одинична множина, що містить лише a;

— Якщо r_1 та r_2 – регулярні вирази з позначеннями $[[r_1]]$ та $[[r_2]]$ відповідно, то $[[(r_1 + r_2)]] = [[r_1]] \cup [[r_2]]$ та $[[(r_1 \cdot r_2)]] = [[r_1]] \cdot [[r_2]]$;

— Якщо r – регулярний вираз з позначенням $[[r]]$, то $[[(r)^*]] = [[r]]^*$.

Регулярні вирази дозволяють формально визначити складні мови. Клас регулярних мов – це клас мов, які можна визначити за допомогою регулярних виразів.

1.1.2. Лематизація. Як описано в [3–5], лематизація – це процес, тісно пов'язаний зі стемінгом. Він також спрямований на визначення різних форм слів зі схожим значенням. Проте на відміну від стемінгу, цей процес визначає базову, або канонічну форму (лему) заданого слова. Лематизація в основному базується на пошуку за словником.

1.1.3. Розмітка частин мови. Згідно з [1, 2, 8, 11, 12], розмітка частин мови – це процес автоматичного присвоєння міток частин мови до слів у заданому корпусі. Процес розмітки включає в себе два етапи: визначення потенційних міток для заданих слів та зняття неоднозначності. Перший етап базується на пошуку в словнику, де зберігаються слова з можливими мітками. Другий етап полягає у виборі найбільш відповідної мітки з отриманих кандидатів.

Як зазначається в [1, 8, 11], одним з основних підходів до розмітки частин мови є статистичний підхід. Цей підхід базується на використанні n-грам та обчисленні

частот категорій для кожного слова в анотованому корпусі. Під час розмітки обирається категорія з найвищою ймовірністю для заданого слова. Іншим поширеним підходом до розмітки частин мови є підхід, що базується на граматиці з обмеженнями. Кожне слово анотується морфологічними та синтаксичними ознаками, які використовуються для визначення правил граматики.

1.2. Синтаксичний аналіз.

Синтаксичний аналіз – це процес, який досліджує структуру речення. Метою синтаксичного аналізу є побудова формального опису закономірностей, базуючись на структурі та організації речень, а також відношеннях залежності між послідовностями слів в межах речення. Згідно з [1], синтаксичний аналіз сфокусований на лінгвістичній формі речення, не беручи до уваги його значення. Синтаксис базується на наступних трьох принципах: валентність, узгодження та порядок слів.

1.2.1. Представлення синтаксису. Як описано в [1, 2, 8, 11], основним підходом до опису синтаксису є синтаксичне дерево.

Синтаксичне дерево – це ациклічний неорієнтований граф, у якому вершини представляють морфологічні або синтаксичні категорії. У синтаксичних деревах слова відносять до їх морфологічних категорій, а словосполучення та фрази – до синтаксичних категорій. Слова речення, яке аналізується, є листовими вершинами, а речення – кореневою вершиною дерева. Це дозволяє визначити синтаксичну структуру речення.

Процес автоматичного синтаксичного аналізу називається синтаксичним розбором. Згідно з [11], це процес розпізнавання речення шляхом присвоєння йому синтаксичної структури. Існують різноманітні підходи до синтаксичного розбору. Основні з них, наведені в [1, 2, 6, 8, 11], розглянуто в цьому розділі.

1.2.2. Рекурсивні мережі переходів. Згідно з [1], рекурсивні мережі переходів базуються на розміченому графі, в якому мітки належать до наступних категорій: лексичні, синтаксичні та понятійні. Вони складаються із станів та пе-

реходів, де стани містять наступні елементи:

— *поточна вершина* – містить інформацію про місце обробки,

— *залишок речення* – неопрацьована частина речення,

— *утримувані вершини* – вершини в поточній мережі, що не перетинаються,

— *синтаксичний розбір* – опрацьована частина вхідного речення.

Коли синтаксичний аналізатор знаходиться у певному стані, є три можливі варіанти дій відповідно до типу поточного стану:

— якщо мітка відповідає категорії фраз (підмережа), поточна вершина поміщається в стек і створюється новий компонент для нової категорії;

— якщо мітка відповідає лексичній категорії, перевіряється ідентичність поточного слова, і слово разом з його категорією додається до поточного компонента;

— якщо опрацювання компонента завершено, із стека береться утримувана вершина та поточний компонент інтегрується у компонент вищого рівня.

1.2.3. Контекстно-вільний синтаксичний розбір. Задача синтаксичного розбору може бути вирішена за допомогою контекстно-вільних граматик.

Згідно з [7, 13], контекстно-вільна граматика визначається наступним чином:

Означення 1.3. Контекстно-вільна граматика – це четвірка (N, Σ, R, S), де

— N – це скінченна множина нетермінальних символів;

— Σ – це скінченний алфавіт термінальних символів;

— R – це множина правил виводу форми $A \to \beta$, де $A \in N$ та $\beta \in (\Sigma \cup N)$;

— S – це початковий символ.

В правилах виведення $A \to \beta$, A – це ліва частина, яка може містити лише нетермінальні символи, а β – це права частина, яка може містити послідовність термінальних та нетермінальних символів. Виведення – це послідовність кроків від початкового символу S до кінцевого рядка w, що є результатом виведення. Рядок w належить до контекстно-вільної мови, якщо існує виведення із S, що призводить до w. Виведення представляються деревами із правилами виведення, що застосовуються зверху вниз. [2, 6–8, 14, 15]

Відповідно до [1, 2, 6, 8], синтаксичний розбір – це процес визначення чи можна отримати рядок із заданої контекстно-вільної граматики та яким чином він може бути отриманий. Синтаксичний розбір на основі контекстно-вільних граматик включає наступні кроки:

1. ініціалізація стека спеціальним нетермінальним символом;
2. пошук правила, в якого ліва частина дорівнює символу, розташованому на вершині стеку та заміна цього символу правою частиною цього правила;
3. пошук правила, в якому ліва частина містить передтермінальний символ та права частина містить наступне слово речення, якщо у вершині стека є передтермінальний символ. Якщо таке правило існує, передтермінальний символ видаляється зі стеку, в протилежному випадку алгоритм повідомляє про невдачу;
4. якщо стек порожній і більше немає слів для аналізу, алгоритм успішно закінчує свою роботу, інакше попередні два кроки повторюються.

1.3. Семантичний аналіз.

Відповідно до [2, 6, 8, 11, 16–19] семантичний аналіз – це процес перетворення природної мови в представлення значення. Існує безліч способів представлення значення, серед яких різні типи логік, які розглядаються в цьому розділі.

1.3.1. Пропозиційна логіка. Як описано в [8, 16], у пропозиційній логіці реченням присвоюється значення істинності (істинна чи хиба). Пропозиційні змінні відповідають простим речення чи твердженням, які є істинними або хибними та позначаються грецькими символами. Для представлення більш складних речень використовуються спеціальні оператори, а саме: заперечення, кон'юнкція, диз'юнкція, імплікація та еквівалентність. Пропозиційна логіка дозволяє висловлювати часткові знання, заперечення тощо, проте вона не дозволяє опрацьовувати властивості окремих сутностей або відношення між ними.

1.3.2. Логіка першого порядку. Логіка першого порядку дозволяє будувати представлення на основі взаємозв'язків між сутностями. Як описано в [8, 16, 19], цей тип логік складається з наступних елементів:

— *терми* – можуть бути константами, змінними або функціями,

— *предикати* – використовуються для опису термів або відношень між ними,

— *квантори* – використовуються для вираження фактів, що застосовуються до об'єктів, представлених у формі термів.

Існує два типи змінних: зв'язані змінні, які знаходяться в полі дії квантора і вільні змінні, які не залежать від квантора. Формула, яка не містить вільних змінних – це речення.

Переклад речень на природній мові в еквівалент логіки першого порядку є необхідним кроком для отримання семантичного представлення цих речень, але недостатнім. Логічне представлення повинно бути пов'язане з контекстом, щоб виражати значення. Обчислення значення істинності речення зводиться до підтвердження чи спростування того, що речення відповідає реальності, яку воно описує.

Якщо змінна S використовується для позначення ситуації, описаної реченням, речення P є істинним у ситуації S, якщо $[p]^v = 1$, де $[p]^v$ – значення істинності речення P. І навпаки, речення P є хибним у ситуації S, якщо $[p]^v = 0$. Складність полягає в тому, що речення в природній мові складаються з набору елементів, таких як дієслова, сутності, доповнення об'єкта тощо. Для обчислення значення композиції є необхідними наступні кроки [16]:

1. Інтерпретація логічних символів першого порядку: на відміну від пропозиційної логіки, не все в логіці першого порядку є пропозиційною змінною. Обчислення істинності речення повинне починатися з обробки основних символів, кожен з яких має тлумачення. Мова логіки першого порядку складається із словника V, який є набором констант, функцій та предикатів цієї мови.
2. Визначення домену, який є представленням сутностей та відношень між ними в заданій ситуації S.

3. Функція інтерпретації – роль цієї функції полягає у встановленні зв'язку між логічними представленнями та розширеннями, які відповідають цим представленням у заданій ситуації.
4. Модель – це поєднання домену та функції інтерпретації. Формально це можна представити як $M_n = (U_n, F_n)$, де M – модель, U – множина сутностей, а F – функція інтерпретації. Специфікатори дозволяють розрізняти ситуації: $M_1 = (U_1, F_1)$, $M_2 = (U_2, F_2)$ тощо.
5. Оцінка значень істинності формул: необхідно задати правила (алгоритми), що дозволяють визначити, чи є формула істинною в заданій ситуації.

1.3.3. Лямбда-числення. У галузі формальної семантики, як наведено в [8, 11, 16, 20], лямбда-числення є розширенням логіки Основна одиниця лямбда-числення – це вираз, який, згідно з [16], може бути визначений рекурсивно в наступним чином:

$$< \textit{вираз} >:=< \textit{змінна} > \mid < \textit{функція} > \mid < \textit{застосування} >$$

$$< \textit{функція} >:= \lambda < \textit{змінна} > . < \textit{вираз} >$$

$$< \textit{застосування} >:=< \textit{вираз} >< \textit{вираз} >$$

Існує декілька правил, які дозволяють перетворювати або скорочувати лямбда-вирази:

— *α-перетворення*, яке полягає в тому, що наступні вирази вважаються ідентичними:

$$(\lambda x.x) \equiv (\lambda y.y) \equiv (\lambda z.z)$$

$$(\lambda x \lambda y.F(x)(y)) \equiv (\lambda z \lambda x.F(z)(x)) \equiv (\lambda w \lambda z.F(w)(z))$$

— *β-перетворення*, яке базується на тому, що всі входження x в формулі $\lambda x.P(x)@(a)$ заміняються на a, даючи $P(a)$.

В термінах лямбда-числення семантичні відношення можна описати виразами, а семантичний аналіз можна отримати, застосовуючи α- та β-перетворення до цих виразів.

Відповідно до принципу композиційності, значення висловлювання залежить від значення його компонентів, а також його синтаксичної структури. Деякі семантичні підходи, базуються на безпосередній відповідності між синтаксисом та семантикою, пов'язуючи семантичне правило з кожним синтаксичним правилом. Розглянемо семантичне представлення основних синтаксичних категорій.

Власні назви є типом . В деяких випадках власні назви вважаються наборами властивостей, тобто множиною множин, та належать до типу: $(e \to t) \to t$.

Загальні іменники вважаються властивостями чи множинами та належать до типу $e \to t$. Як правило, лямбда-вирази, що відповідають іменникам, як наприклад "*ліс*", мають форму $\lambda x.forest(x)$ та представляють властивість x, таку як наприклад x є "*лісом*".

Визначники застосовуються до загальних іменників та передають структуру вищого рівня групи слів, що представляють чи описують сутності. Обробка визначників здійснюється у поєднанні з універсальними та екзистенціальними кванторами.

Прикметники мають природу атрибутів, та відіграють роль предикатів або функторів, як приймають іменні елементи в якості аргумента. Іменні елементи належать до типу (e, t) та визначають тип прикметників як $((e, t) \to (e, t))$.

Дієслова відіграють роль логічного предикату і можуть приймати нуль або декілька аргументів в залежності від того, перехідні вони чи ні. Випадок неперехідних дієслів найпростіший: вони належать типу $(e \to t)$ і задаються виразами форми $\lambda x.V(x)$. В свою чергу, перехідні дієслова представляються виразами форми $\lambda X \lambda z.(X@\lambda x.verb(z, x))$.

РОЗДІЛ 2

СИСТЕМИ ПОШУКУ ІНФОРМАЦІЇ

Розрізняють два основних типи інформаційних систем: інформаційні системи та інформаційно-пошукові системи або просто пошукові системи.

У свою чергу, пошукові системи мають мету пошуку документів, які є релевантними до запиту, заданого користувачем. Тому їх можна вважати частковим випадком інформаційних систем. Їх специфіка стосовно інформаційних систем полягає у відсутності обмежень щодо запитів, збору даних та результатів роботи системи. У випадку текстового пошуку користувачі можуть формувати запит для системи, використовуючи рядок слів, повне речення, або комбінацію ключових слів, які можуть містити мета-символи на основі елементарного синтаксису, прийнятого пошуковою системою. Теж саме стосується документів, що утворюють пошуковий простір, а також вихідних даних системи, що складається з підмножини документів у колекції.

Іншою характеристикою пошукових систем є тип колекції документів, який вони включають; вони можуть бути обмеженими, як у випадку із системою пошуку документальної інформації, або повністю відкритими, як у випадку з Інтернетом.

Пошуковій системі включають дві основні функції. Індексація - це крок, який виконується в автономному режимі і складається з побудови спрощеного подання колекції, щоб зменшити простір, необхідний для її зберігання, і пришвидшити доступ до інформації. Індексація здійснюється як на основі вмісту кожного документа, так і за допомогою його взаємозв'язків, використовуючи гіперпосилання на інші документи.

2.1. Підходи до пошуку інформації.

Усі тексти, незалежно від типу, утворені послідовністю слів, вирівняних відповідно до морфологічних, синтаксичних, семантичних обмежень тощо. Тому природно починати з розуміння вкладу, який роблять лексичні знання у інформацію, яку передає текст і, отже, до можливих спрощень, які можна здійснити на цьому рівні.

У галузі пошуку інформації зазвичай здійснюється пошук текстів, які є найбільш релевантними до запиту, сформульованому одним або кількома ключовими словам. Така задача зводиться до пошуку текстів, де ці слова відіграють вагому роль. Простого входження слова в текст недостатньо, щоб зробити висновок, що це слово має значиму роль у цьому тексті.

Розглянемо детальніше основні підходи до пошуку інформації, описані в [16, 21].

2.1.1. Сканування повного файлу. Найпростішим методом встановлення зв'язку між запитом та документом є пошук документів, що містять це слово, у всій базі доступних документів. Основним недоліком цього методу є те, що він забезпечує дуже поверхневий аналіз відношення запит-документ. Наприклад, він не розрізняє слова, які відіграють важливу роль у розрізненні двох документів та слів, роль яких обмежена. Окрім того, пошук у всіх документах значно обмежує швидкість обробки.

2.1.2. Інвертовані списки. Відповідно до цього методу кожен документ може бути представлений низкою ключових слів. Для прискорення процесу пошуку створюється інвертований файл. Це забезпечує перелік ключових слів з усіх документів, упорядкованих за алфавітом або відповідно до їх загальних частот, і кожного, релевантного документу, в яких вони з'являються. Як правило, граматичні слова вилучаються із цього списку, щоб зменшити його розмір та зробити пошук більш ефективним.

Також для підвищення ефективності цього підходу використовують рішення,

що базується на подвійному проході. На другому рівні всі слова, що починаються з перших двох літер, зберігаються в одному місці, тоді як перший рівень містить гіперпосилання на другий рівень. Перевагами цього підходу є його легка реалізація, швидкий пошук та можливість легкої обробки синонімів. До недоліків можна віднести розмір інвертованого файлу, який іноді може бути досить великим, та труднощі, пов'язані з оновленням змін до документів, особливо у випадку динамічного середовища.

2.1.3. Файли підписів. Цей підхід полягає у скороченні документа до піддокументу, що містить лише інформацію, яка є значимою для пошуку. Документ, який є результатом процесу фільтрації, зберігається в окремому файлі, який називається файлом підпису. Такий файл є значно меншим, ніж оригінальний документ, а тому на його пошук необхідно менше часу.

Як і інші підходи до пошуку інформації, даний підхід починається зі спрощення документа. Граматичні слова виключаються, а всі нестандартні слова зводяться до своїх лем. Після цього виконується сегментація файлів. Для цієї операції існують різні методи, включаючи цифрові методи та та підходи, що базуються на N-грамах. Ідея цієї сегментації полягає у визначенні релевантних слів за сегментами. В деяких інших методах обробка файлів підписів здійснюється на двох рівнях. На першому рівні файли представлені у вигляді дерев. На другому рівні інформація у файлах поділяється відповідно до їх частоти.

Основною проблемою цього методу є пошук ідеального компромісу між деталями та швидкістю пошуку: чим більший розмір файлу підпису, тим точніше він представляє оригінальний документ, проте при цьому збільшується і час, необхідний для пошуку.

2.1.4. Підходи на основі кластеризації. Підходи, що базуються на кластеризації, є однією з найпоширеніших форм алгоритмів навчання. Кластеризація не вимагає втручання людини-експерта, щоб спочатку розмітити дані для алгоритму навчання.

Основна ідея кластеризації в контексті пошуку інформації полягає в тому, що

подібні документи об'єднуються у групу одного типу. Перевагою такого підходу є те, що зазвичай, документи, що утворюють кластер, мають тенденцію відповідати однаковим запитами або задовольняти однаковим потребам з точки зору інформації. Кластер можна використовувати як для документів, так і для слів.

2.1.5. Підходи на основі подібності. Ці підходи використовують методи, засновані на графах. Основним принципом цієї техніки є обчислення подібності (відстані) між двома документами. Перший крок полягає у виборі функції подібності, що дає змогу виміряти відстань між документами, представленими у формі матриці.

Щоб визначити наскільки схожі два документи в корпусі, спочатку виконуються дві операції попередньої обробки: фільтрування стоп-слів та лематизація. Після чого створюється матриця термінів-документів, де частота слова в документі ми розглядається як його вага.

Обчислення схожості двох документів виконується за допомогою наступної рівності:

$$similarity(D_i, D_j) = \sum_{k=1}^{n} W_{ik} \cdot W_{jk},$$

де W_{ik}, W_{jk} – вага k-го терміну в документах D_i та D_j, відповідно.

Після обчислення матриці подібності вибирається поріг схожості, тобто значення, вище якого повинно бути значення метрики схожості, щоб два документи вважалися подібними.

Останній крок такого типу підходів полягає у застосуванні певного алгоритму кластеризації. Ці алгоритми вимагають значення подібності між термінами кластера.

2.1.6. Ітераційні підходи. Ці підходи базуються безпосередньо на описі тексту на основі хеш-ключа, без обчислення схожості між документами. Алгоритмічна складність цього підходу дуже хороша. Проте така ефективність досягається разом з деякими обмеженнями. Наприклад, такими як чутливість до

порядку, в якому аналізувались тексти, або неможливість прогнозувати рівень помилок в описі документа.

Загальна схема цих підходів включає два етапи:

1. визначення початкового розподілу;
2. повторення ітерації та додавання документів до кластерів, до тих пір, поки не буде більше якісних асоціацій.

Окрім того, слід зазначити, що можуть використовуватися і гібридні методи. Ітераційний метод може бути використаний для розділення документів на кластери, а потім, наприклад, метод, заснований на графах, може бути використаний для розділення кожного з отриманих кластерів.

2.2. Моделі пошуку інформації.

Модель пошуку інформації передбачає та пояснює, яку інформацію буде надано користувачу відповідно до його запиту. Правильність передбачень моделі можна перевірити в контрольованому експерименті.

Модель служить проектом, який використовується для реалізації фактичної системи пошуку інформації. Розглянемо детальніше деякі основні моделі пошуку інформації, описані в [11, 16, 22].

2.2.1. Булева модель. Згідно з [22], булева модель – це перша модель пошуку інформації. В цій моделі термін запиту розглядається як однозначне визначення множини документів. Наприклад, термін запиту "*освіта*" просто визначає множину усіх документів, які індексуються терміном "*освіта*". Використовуючи оператори математичної логіки (а саме логічні "*І*", "*АБО*" та "*НЕ*"), терміни запитів та відповідні їм множини документів можна об'єднати, щоб сформувати нові множини документів. Поєднання термінів за допомогою оператора "*І*" визначає множину документів, яка менша або рівна множинам документів будь-якого з окремих термінів. Наприклад, запит "*хімічний І фізичний*" дає множину документів, які індексуються як терміном "*хімічний*", так і терміном "*фізичний*", тобто отримуємо перетин множин. Поєднання термінів за допомогою оператора "*АБО*"

визначає множину документів, яка більша або дорівнює множинам документів будь-якого з окремих термінів, тобто маємо об'єднання множин.

Перевагою булевої моделі є її прозорість. Одразу зрозуміло, чому отримано той чи інший документ, за заданим запитом. Якщо отримана множина документів занадто мала або занадто велика, чітко видно, які оператори результують відповідно в більшу чи меншу множину. Головним недоліком даної моделі є те, що вона не визначає рейтингу отриманих документів. Іншими словами, модель або знаходить документ або ні, що може призводити до того, що система прийматиме некоректні рішення.

2.2.2. Регіональні моделі. Згідно з [22], регіональні моделі – це продовження булевої моделі, які розглядають довільні частини текстових даних, що називаються сегментами, ступенями або регіонами.

Регіональні моделі моделюють колекцію документів як лінійний рядок слів. Будь-яка послідовність слів називається регіоном. Регіони визначаються початковою та кінцевою позиціями. Деякі регіони можуть бути заздалегідь визначені оскільки вони представляють логічний елемент у тексті, наприклад, рядок. Регіональні системи не обмежуються отриманням документів. Залежно від системи, можуть з'являтися запити на пошук цілих п'єс, новел чи романів.

Якщо булева модель оперує множинами документів, де документ представлений іменним ідентифікатором, то регіональна в свою чергу діє на множинах регіонів, де регіон представлений двома порядковими ідентифікаторами: початковою та кінцевою позицією у колекції документів. Булеві оператори "*і*", "*або*" та "*не*" можуть бути визначені на множинах регіонів прямолінійно як перетин множин, об'єднання множин і доповнення множин. Проте у регіональних моделях в свою чергу використовуються принаймні ще два оператори: "*містить*" і "*міститься в*". Системи, що підтримують регіональні моделі, можуть обробляти більш складні запити.

У деяких регіональних моделях визначаються також оператор "*слідує за*", необхідний для визначення початкових та кінцевих тегів. Це дозволяє більше дета-

лізувати запит.

2.2.3. Векторна модель. Як описано в [16, 22], у векторній моделі пошуку інформації документи та запити представлені у вигляді векторів ознак, що задають терміни (слова), які зустрічаються у корпусі. Значення кожної ознаки називають вагою терміна і, як правило, воно обраховується за допомогою функції частоти терміну в документі, разом із іншими факторами.

Згідно з [16], вектор для документа визначається наступним чином.

Означення 2.1. Вектор для документа d_i представляється як

$$\overrightarrow{d_i} = (w_{1,i}, w_{2,i}, w_{3,i}, \dots w_{n,i}),$$

де d_i позначає конкретний документ, а вектор містить ознаку ваги для кожного із n термінів, що зустрічаються в колекції в цілому;

Таким чином $w_{j,i},$ де $i \in \overline{1,n}, j \in \overline{1,m}$, задає вагу, яку термін j має в документі, де m – загальна кількість документів. Аналогічним чином можна представити запит.

В даній моделі доцільно розглядати ознаки, що використовуються для представлення документів та запитів як виміри в багато-вимірному просторі, де ваги ознак служать для розміщення документів в цьому просторі. Коли запит користувача визначається як вектор, він позначає точку в просторі. В такому випадку документи, які знаходяться ближче до запиту, можна вважати більш релевантними, ніж документи, що знаходяться далі.

При векторному пошуку інформації зазвичай використовується метрика косинуса, а не фактичний кут. Відстань між двома документами вимірюється косинусом кута між їх векторами. Коли два документи однакові, значення косинуса дорівнює одиниці, а коли вони ортогональні (тобто не мають спільних термінів), значення косинуса дорівнює нулю. Рівняння для косинуса:

$$sim(\overrightarrow{q}, \overrightarrow{d_i})(= \frac{\sum_{i=1}^{n} w_{i,q} \times w_{i,j}}{\sqrt{\sum_{i=1}^{n} w_{i,q}^2} \times \sqrt{\sum_{i=1}^{n} w_{i,j}^2}}$$

.

Деякі досить специфічні, але ефективні алгоритми пошуку добре обгрунтовані у моделі векторного простору, якщо довжини векторів нормалізуються. Розглянемо наприклад алгоритм зворотного зв'язку щодо релевантності. Нехай $\overrightarrow{q_{old}}$ – вихідний запит, $\overrightarrow{q_{new}}$ – переглянутий запит, $\overrightarrow{d}^{(i)}_{rel}, (1 \leqslant i \leqslant r)$ – один із r документів, які визначені як релевантні, і $\overrightarrow{d}^{(i)}_{norel}, (1 \leqslant i \leqslant r)$ – це один із n документів, визначених як нерелевантні, тоді

$$\overrightarrow{q_{new}} = \overrightarrow{q_{old}} + \frac{1}{r}\sum_{i=1}^{r}\overrightarrow{d}^{(i)}_{rel} - \frac{1}{n}\sum_{i=1}^{n}\overrightarrow{d}^{(i)}_{norel}$$

Нормалізовані вектори документів та запитів можна розглядати як точки в гіперсфері. У даному рівнянні перша сума обчислює центроїд точок відомих релевантних документів. У центроїді кут з відомими релевантними документами зведений до мінімуму. Друга сума обчислює центроїд точок відомих нерелевантних документів. Переміщення запиту в бік центроїда відомих релевантних документів та віддалення від центроїда відомих нерелевантних документів гарантовано покращує ефективність пошуку.

Дана модель документів та запитів в представленні векторів забезпечує всі основні елементи для системи пошуку. Система пошуку документів може просто прийняти запит користувача, створити для нього векторне представлення, порівняти його з векторами, що представляють усі відомі документи та відсортувати результати. В результаті отримується перелік документів, впорядкованих за схожістю із запитом.

Основним недоліком векторної моделі є те, що вона жодним чином не визначає, якими повинні бути значення векторних компонентів. Процес присвоєння відповідних значень векторним компонентам називається зважування термінів. Для цього використовуютья ваги $tf.idf$: комбінація частоти терміна tf, яка є кількістю входжень терміна в документ, та idf, оберненої частоти документа, яка є значенням, оберненим до частоти документа df, тобто кількості документів, що містять термін. Багато сучасних алгоритмів зважування є версіями сімейства алгоритмів зважування $tf.idf$. Ваги $tf.idf$ визначаються як:

$$d_k = q_k = tf(k,d) \cdot \log \frac{N}{df(k)},$$

де $tf(k,d)$ – кількість входжень терміну k в документ d, $df(k)$ – кількість документів, що містять k, а N – загальна кількість документів в колекції.

Іншою проблемою векторної просторової моделі є її реалізація. Для обчислення міри косинуса потрібні значення всіх векторних компонентів, але вони не доступні в інвертованому файлі. На практиці слід використовувати нормовані значення та векторний алгоритм добутку. Або нормовані ваги потрібно зберігати в інвертованому файлі, чи зберігати нормування окремо. І те, і інше є проблематичним у разі поступового оновлення індексу. Додавання одного нового документа змінює частоту термінів, що трапляються в документі, що змінює довжини векторів кожного документа, які містять один або кілька цих термінів.

2.2.4. Модель ймовірнісного пошуку. Модель ймовірнісного пошуку базується на принципі, що документи впорядковуються за $P(R|D)$, тобто за ймовірністю релевантності R, враховуючи опис вмісту документа D. В даній моделі D – це вектор бінарних компонентів, де кожен компонент, як правило, представляє термін. У моделі ймовірнісного пошуку імовірність $P(R|D)$ інтерпретується наступним чином: може бути n документів, які представлені одним і тим же вектором бінарних компонент D. Нехай $n = 10$, якщо вісім із них є релевантними, то $P(R|D) = 0,8$. Для реалізації цього на практиці, використовується наступне правило $P(R|D)/P(\overline{R}|D)$, де $\overline{R}$ позначає нерелевантність. Крім того, припускається незалежність між термінами, які вважаються релевантними.

Для здійснення імовірнісного пошуку використовуються наступні три перетворення, що зберігають порядок. По-перше, документи класифікуються за сумами логарифмічних коефіцієнтів, а не за самими коефіцієнтами. По-друге, пріоритети релевантності $P(R)/P(\overline{R})$ ігноруються. По-третє, оцінка порожнього документа, що обчислюється як: $\sum_k \log P(D_k = 0|R)/P(D_k = 0|\overline{R})$, віднімається від оцінок усіх документів. Таким чином, сума по всім термінам, яка може включати мільйони термінів, включає лише ненульові значення для термінів, які є в документі.

Для обчислення оцінки документа використовується наступне рівняння [22]:

$$matching - score(D) = \sum_{k \in matching-terms} \log \frac{P(D_k = 1|R)/P(D_k = 0|\overline{R})}{P(D_k = 0|R)/P(D_k = 1|\overline{R})}$$

На практиці терміни, яких немає у запиті, також ігноруються у даному рівнянні.

2.2.5. Модель мови. Для пошуку інформації також можуть застосовуватися моделі мови. Вони походять від ймовірнісних моделей генерації мови, розроблених для систем автоматичного розпізнавання мови. Системи автоматичного розпізнавання мови поєднують в собі ймовірності двох різних моделей: акустичної та мовної. Акустична модель може повертати наприклад фрази чи словосполучення у порядку зменшення їх ймовірності. Модель мови в свою чергу може, наприклад, визначати, що певне слово з більшою імовірністю слідує за іншим, якщо така їх послідовність частіше зустрічається в текстовому корпусі [2, 8, 11, 16, 22].

Для пошуку інформації для кожного документу будуються моделі мови. Моделі мови використовують ту ж відправну точку, що і ймовірнісні моделі. Тобто, за заданим документом D, який є релевантним, може бути сформульований запит, що містить термін T з імовірністю $P(T|D)$. Ймовірність обчислюється на основі тексту документа. Якщо певний документ містить, наприклад, сто слів, із них певне слово зустрічається в цьому документі двічі, то ймовірність цього слова за даним документом визначається просто як 0.02. Для запитів, що містять декілька слів, припускається, що кожне слово генерується незалежно від іншого, тобто умовні ймовірності термінів $T_1, T_2, \ldots$ просто перемножуються:

$$P(T_1, T_2, \ldots|D) = \prod_i P(T_i|D)$$

Недоліком, цього підходу є те, що імовірність термінів, які не зустрічаються в документі буде дорівнювати нулю. А оскільки умовні імовірності для кожного терміну множаться одна на одну, документи, що не містять хоча б одного терміну із запиту, також матимуть імовірність нуль. Для деяких систем ця проблема не

є критичною. Наприклад, у веб-пошукових системах запити короткі, і доволі рідко може трапитись так, що жодна веб-сторінка не містить усіх термінів запиту. Проте у багатьох інших системах порожні результати зустрічаються набагато частіше. Тому, для усунення цієї проблеми використовується техніка згладжування. Процес згладжування призначає деяку, відмінну від нуля, ймовірність термінам, що не спостерігаються в документі. Розглянемо наступний підхід до згладжування, який приймає лінійну комбінацію $P(T_i|D)$ та фонову модель $P(T_i)$ наступним чином.

$$P(T_1, T_2, \dots T_n|D) = \prod_{i=1}^{n} \lambda P(T_i|D) + (1-\lambda)P(T_i) \tag{2.1}$$

Фонова модель $P(T_i)$ може бути визначена ймовірністю появи терміну у всій колекції, тобто часткою загальної кількості появи терміну у колекції, поділеною на довжину колекції. У даному рівнянні λ – невідомий параметр, який потрібно встановити емпірично. Лінійне інтерполяційне згладжування пояснює факт того, що деякі слова запиту можуть бути взагалі не пов'язані з релевантністю документів. Тому більш доцільна реалізація лінійних інтерполяційних моделей може бути здійснена за допомогою перетворень, що зберігають порядок, подібних до тих, що використовують моделі ймовірнісного пошуку.

Помножимо обидві сторони рівняння 2.1 на $\prod_i (1-\lambda)P(T_i)$ та візьмемо логарифм. Звідси:

$$matching - score(d) = \sum_{k \in matching-terms} \log 1 + \frac{tf(k,d)}{\sum_t tf(t,d)} \cdot \frac{\sum_t cf(t)}{cf(k)} \cdot \frac{\lambda}{1-\lambda},$$

де $P(T_i = t_i) = cf(t_i)/\sum_t cf(t)$, та $cf(t) = \sum_d tf(t,d)$.

Існує багато підходів до згладжування, більшість з них розроблялись для автоматичного розпізнавання мови. Іншим підходом до згладжування, який також часто використовують для пошуку інформації, є згладжування Діріхле, яке визначається як:

$$P(T_1 = t_1, \dots T_n = t_n | D = d) = \prod_{i=1}^{n} \frac{tf(t_i, d) + \mu P(T_i = t_i)}{\sum_t tf(t, d) + \mu},$$

де μ – дійсне число та $\mu \geqslant 0$. Згладжування Діріхле пояснює факт того, що документи занатдо малі, щоб можна було надійно оцінити модель мови. Дане згладжування має відносно великий вплив на документи малого розміру, проте відносно невеликий вплив на більші документи.

Наведені вище рівняння визначають ймовірність запиту за документом, проте система повинна оцінювати документи за запитом. Для цього ці дві ймовірності пов'язують за правилом Баєса наступним чином:

$$P(D|T_1, T_2, \dots T_n) = \frac{P(T_1, T_2, \dots, T_n|D)P(D)}{P(T_1, T_2, \dots, T_n)} \tag{2.2}$$

Ліву частину цього рівняння не можна використовувати безпосередньо, оскільки наведене вище припущення про незалежність передбачає, що терміни є незалежними, за заданим документом. Отже, для обчислення ймовірності документа D за запитом потрібно помножити рівняння 2.1 на $P(D)$ і розділити його на $P(T_1, \dots, T_n)$. Самі ймовірності не представляють інтересу, лише оцінка документа за цими ймовірностями. А оскільки $P(T_1, \dots, T_n)$ не залежить від документа, оцінка документів за допомогою чисельника правої частини рівняння 2.2 буде оцінювати їх ймовірність за заданим запитом.

У веб-пошуку зазвичай використовуються так звані статичні рейтинги. Наприклад, документи з великою кількістю посилань, що вказують на них, швидше за все будуть релевантними.

РОЗДІЛ 3
СИСТЕМИ ПИТАНЬ-ВІДПОВІДЕЙ, ЩО БАЗУЮТЬСЯ НА ПОШУКУ ІНФОРМАЦІЇ

Система відповідей на запитання (англ. Question Answering (QA), QA-системи) - це система пошуку інформації, в якій очікується пряма відповідь на подане запитання, замість набіру посилань, які можуть містити відповіді. Основна ідея QA-систем в галузі обробки природної мови полягає у наданні правильних відповідей на запитання. Для реалізації цього необхідні складні методи обробки природної мови, щоб визначити, чого хоче користувач (тобто аналіз питання), і зробити висновок, що певний рядок у реченні документа або множина рядків з декількох речень є правильною відповіддю на запитання (тобто аналіз речень документа та їх отримання).

На відміну від цього, процедури пошуку інформації є доволі простими та складаються з вилучення певних багатих на вміст слів чи фраз із елементів, що призводить до досягнення більшої однорідності та покриття серед варіантів слів. Ці індекси підготовлюються перед пошуком і часто є недостатніми для задоволення потреб QA-систем.

З іншого боку, використовувати систему обробки природної мови, щоб проаналізувати тисячі чи навіть мільйони речень безпосередньо для виділення відповіді доволі громіздке завдання. Для досягнення мети QA-системи в ефективний спосіб логічною поширеною практикою є використання системи пошуку інформації у зв'язці з QA-системою, щоб зменшити велику колекцію документів до меншої, яка містить лише найбільш релевантні документи. Після чого застосовуються вже методи більш глибокого аналізу природної мови лише до отриманих релевантних документів.

Типова архітектура систем питань-відповідей, що базуються на пошуку інформації [11] наведена на рисунку 3.1.

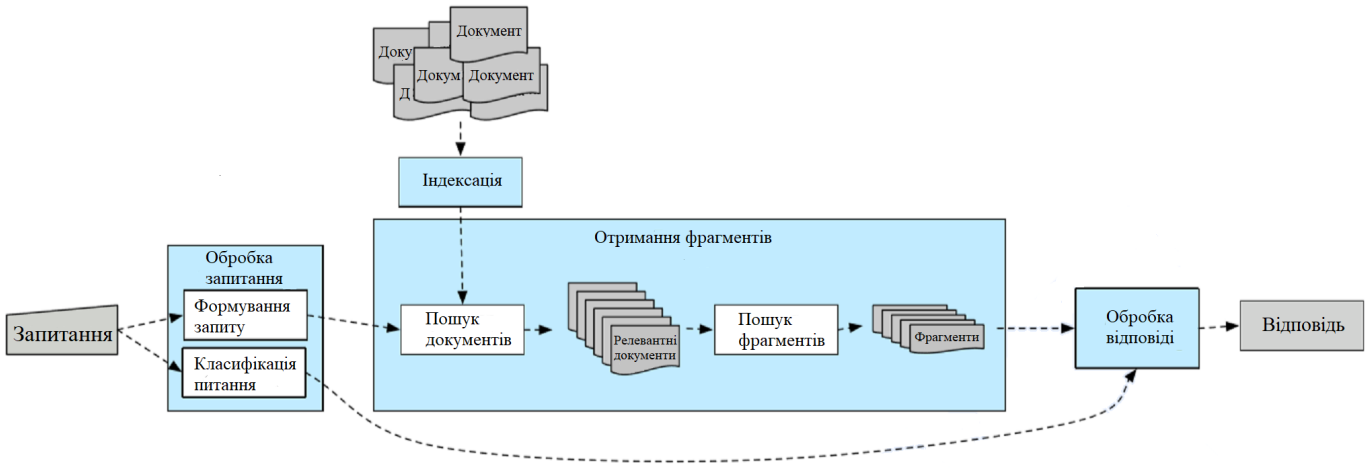

Рис. 3.1. Архітектура систем питань-відповідей, що базуються на пошуку інформації.

3.1. Обробка запитань.

Мета етапу обробки запитань полягає у вилученні двох речей із запитання: запит на основі ключових слів, придатний для того, щоб подати його на вхід до системи пошуку інформації, та тип відповіді, визначення типу сутності, яка могла вважатися коректною відповіддю на запитання.

Розглянемо детальніше процеси формування запиту, класифікації та інтерпретації питань, описаних в [6, 11, 21, 23].

3.1.1. Формування запиту. Процес формулювання запитів дуже схожий на обробку, виконувану для запитів в системах пошуку інформації. Мета полягає в тому, щоб створити із запитання список ключових слів, що формують запит для пошуку інформації.

Який саме запит формувати, залежить від системи питань-відповідей. Якщо пошук відповіді на запитання буде здійснюватись в мережі, можна просто створити ключові слова з кожного слова запитання, дозволивши веб-пошуковій системі автоматично видаляти будь-які стоп-слова. Крім того, ключові слова можуть утворюватися лише з термінів, знайдених в іменних групах у питанні, застосову-

ючи списки стоп-слів для ігнорування функціональних слів та високочастотних, мало змістовних дієслів.

Коли пошук відповіді на запитання здійснюється на менших наборах документів, наприклад, щоб відповісти на запитання по корпоративних інформаційні сторінках, так само використовуються механізми пошуку інформації для пошуку необхідних документів. Проте для цієї меншої множини документів зазвичай необхідно застосовувати розширення запитів.

В мережі відповідь на запитання може з'являтися у багатьох різних формах, і тому, якщо здійснюється пошук по словам із запитання, імовірно, буде знайдено відповідь, написану в тій же формі. На менших наборах, наприклад корпоративних сторінок, навпаки, відповідь може з'явитися лише один раз, а точне формулювання може мати вигляд та форму відмінну від питання. Таким чином, методи розширення запитів можуть додавати терміни до запиту, щоб відповідати певній формі відповіді, в тому вигляді, в якому вона може з'явитись.

Таким чином, можна додати до запиту всі морфологічні варіанти змістовних слів у питанні, а також використати словник синонімів чи інші алгоритми розширення запитів, щоб отримати більший набір ключових слів для запиту. Багато систем використовують семантичну мережу "*WordNet*" в якості словника синонімів, тоді як інші покладаються на словники синонімів спеціального призначення, які спеціально створені вручну для QA-систем. Інший підхід до формулювання запитів полягає у застосуванні набору правил переформулювання запитів. Правила переформулюють питання, щоб воно виглядало як підрядок можливих декларативних відповідей. Можна застосовувати декілька таких правил до запиту та передавати всі отримані запити пошуковій системі.

3.1.2. Класифікація питань. Другим завданням, що підлягає обробці запитань, є класифікація питання за типом очікуваної відповіді. Наприклад, на запитання типу "Хто ...?" очікується відповідь типу "*особа*". Цей процес називається класифікацією запитань або розпізнаванням типів відповідей. Інформація про тип відповіді на запитання дозволяє уникати розгляду кожного речення чи іменних

груп у всій множині документів. Знання типу відповіді також важливе для представлення відповіді. Питання типу "*визначення*", як наприклад "Що таке трикутник?" може використовувати простий шаблон відповіді на зразок "Трикутник - це ...", тоді як відповідь на запитання типу "Хто такий ..?" може використовувати спеціально заданий біографічний шаблон, наприклад, починаючи з громадянства особи, дати її народження та іншої біографічної інформації.

Класифікатори запитань можуть бути побудовані за правилами, розробленими вручну, за допомогою алгоритмів машинного навчання або за допомогою комбінації цих підходів. Однак більшість сучасних класифікаторів питань базуються на техніках машинного навчання. Ці класифікатори навчаються на базах даних питань, які були розмічені вручну типом відповідей. Типові ознаки, що використовуються для класифікації, включають слова у запитаннях, частину мови кожного слова та власні назви у запитаннях.

Також, може використовуватися семантична інформація про слова в питанні.

3.1.3. Інтерпретація питань. Перш ніж дати відповідь на питання за допомогою системи питань-відповідей, воно повинне бути проаналізоване на декількох різних рівнях. По-перше, необхідно визначити складність питання на основі синтаксичних та семантичних знань. По-друге, потрібно визначити контекст питання для кращого розуміння його запиту. По-третє, складно відокремити наміри питання від його формулювання, тому з кожного запитання варто виділити ймовірні наслідки. По-четверте, необхідні механізми, що приймають чи відкидають передбачувані наміри. Всі ці процеси впливають на розуміння питання та на точність відповідей, які повертає система.

Питання які є доволі простими з лінгвістичної точки зору, викликають ряд синтаксичних та семантичних складностей. Зокрема, знання контексту відіграє важливу ролі в процесі формування відповіді на запитання. Розуміння намірів, що ведуть до постановки питання, може допомогти розв'язати лексичні і семантичні неоднозначності та краще визначити відповіді, які потрібно повернути на конкретне питання.

Інтерпретація питання не може бути відокремлена за його контекстом, а також не слід ігнорувати намір того, хто задає питання. При спілкуванні природною мовою наміри не виражаються безпосередньо. Визначення намірів залежить від виведення контексту та від наявних семантичних та прагматичних знань.

Наприклад, якщо контекст питання Q містить кілька посилань на назви дисциплін або відвідування лекцій, неоднозначність намірів користувача може бути вирішена. Однак такі посилання повинні забезпечуватися семантичними знаннями, наприклад, шляхом виведення з цілісної структури концептуальної та реляційної інформації контексту, яка визначає тему навчання замість теми власності.

Інтерпретація питань також спирається на прагматичне знання, яке безпосередньо не випливає з заданого джерела. Для кожної конкретної теми прагматичне знання доповнює семантичне знання і дозволяє робити висновки. Наприклад, для випадку з темою "*навчання*", семантичні поняття, такі як "*дисципліна*", "*студент*" або "*вивчення*", організовані у структури, що визначають їх функції. Такі семантичні структури можуть бути автоматично розпізнані у питаннях або кандидатах для відповіді на них, підкреслюючи роль "*студента*" для "*вивчати*" та нарешті "*опановувати*" програми навчальних "*дисциплін*".

Згідно з [23], інтерпретація складного запитання часто базується на знаннях в галузі, пов'язаних з темою цього запитання. Для кожної підтеми за заданими документами, що відносять до даної підтеми, а також документами, які не мають відношення до заданої підтеми, використовується статистичний підхід, що базується на коефіцієнті схожості, для визначення зваженого списку найбільш релевантних до теми понять, який називають підписом теми. Такі підписи тем можна вдосконалити, шляхом визначення найбільш відповідних відношень між парами понять. Проте ці обидва типи представлення обмежені тим, що вони вимагають ідентифікації документів, які є релевантними до теми, перед визначенням підписів теми.

Розглянемо чотири різні підходи до вирішення задачі пошуку документів, що відповідають кожній підтемі [23]:

Підхід 1: Спочатку усі документи в колекції текстів кластеризують за допо-

могою методу k-найближчих сусідів. Кожен кластер, що містить принаймні одне ключове слово, яке описує підтему, вважається релевантним заданій темі.

Підхід 2: Оскільки певні документи можуть містити елементи дискурсу, що стосуються різних підтем, спочатку здійснюється сегментація всіх документів в колекції текстів на окремі текстові фрагменти. Після чого, ці сегменти дискурсу подаються на вхід алгоритму кластеризації k-найближчих сусідів.

Підхід 3: В цьому підході, релевантні документи визначаються одночасно з отриманням підписів тем. Перш за все, початкове бінарне відношення r_i зв'язується з кожною підтемою S_i. Оскільки початкові відношення належать до певної підтеми, їх можна використовувати для визначення бінарного розподілу колекції документів C на множину релевантних документів R_i (тобто документів, що відповідають відношенню r_i) та множину нерелевантних документів $C - R_i$. Після чого підписи тем обраховуються для множини документів із R_i. Для кожної підтеми S_i документи, які є релевантними відповідному початковому відношенню r_i, додаються до R, якщо відношення r_i задовольняє критерію щільності. Нехай D представляє множину документів, де r_i було розпізнано, тоді критерій щільності визначається як: $|D \bigcap R|/|D \bigcap C| >> |R|/|C|$. Після того як D додано до R_i, для R обчислюється новий підпис теми. Відношення отримані з нового підпису теми можуть далі бути використані для визначення нового розподілу документів, повторюючи обрахунок підпису теми та визначення документів, релевантних кожній підтемі.

Підхід 4: Даний підхід використовує методику, описану в Підході 3, але працює на рівні сегментів дискурсу (або фрагментів тексту), а не на рівні цілих документів.

3.2. Пошук фрагментів.

Запит, створений на етапі обробки запитань, використовується для здійснення запиту в інформаційно-пошуковій системі, або веб-пошуковій системі. Результатом цього етапу пошуку є множина документів.

Хоча множина документів, зазвичай, класифікується за релевантністю, найвищий рейтинг, імовірно, не гарантує відповідь на запитання. Це пов'язано з тим, що документи не є доцільною одиницею для впорядкування відносно цілей QA-системи. Дуже релевантний і великий документ, який чітко не відповідає на питання, не є ідеальним кандидатом для подальшої обробки.

Подальші кроки включають виділення та класифікацію фрагментів, які описані в [11, 21].

3.2.1. Виділення фрагментів. Наступним етапом є вилучення множини уривків із можливими відповідями з отриманої множини документів. Визначення фрагменту в будь-якому випадку залежить від системи, проте типові одиниці включають розділи, абзаци та речення. Наприклад, можна запустити алгоритм сегментації абзаців на всіх повернутих документах і розглядати кожен абзац як фрагмент.

Далі здійснюється пошук потрібного фрагменту. На цьому етапі спочатку відфільтровуються фрагменти у повернутих документах, які не містять потенційних відповідей, а потім решта фрагментів оцінюється відповідно до того, наскільки ймовірно, що вони містять відповідь на запитання. Під час цього процесу, спочатку запускаються алгоритми розпізнавання іменованих сутностей або класифікації типу відповідей на отриманих фрагментах. Тип відповіді, який було визначено з питання, дає інформацію про допустимі типи відповідей, які очікуються у результаті. Отже, можна відфільтрувати фрагменти, які не містять жодної сутності належного типу.

3.2.2. Класифікація фрагментів. На цьому етапі виконується класифікація решти фрагментів. В результаті отримується рейтинг, який базується на відносно невеликому наборі ознак, які можна легко та ефективно виділити з потенційно великої кількості фрагментів з відповідями. До найбільш поширених ознак, що використовуються на даному етапі належать [11]:

— кількість іменованих сутностей потрібного типу в фрагменті,

— кількість ключових слів із питання в фрагменті,

— найдовша точна послідовність ключових слів із питання, яка зустрічається в фрагменті,
— оцінка документу, з якого було виділено фрагмент,
— наближеність ключових слів з оригінального запиту один до одного: для кожного фрагменту визначається найкоротший проміжок, який охоплює ключові слова, що містить даний фрагмент; перевага віддається меншим інтервалам, що містять більше ключових слів,
— перетин N-грам із фрагменту та запитання: обчислюються N-грами в питанні та фрагментами відповідей; перевага надається більшому перетину N-грам.

У випадку пошуку відповіді на запитання в мережі, замість того, щоб виділяти фрагменти з усіх повернутих документів, можна покладатися на веб-пошук, який самостійно робить витяг уривків.

3.3. Обробка відповіді.

Останнім етапом отримання відповіді на запитання є витяг конкретної інформації з фрагменту, щоб мати змогу запропонувати користувачеві відповідь на зразок “Говерла” на запитання “Яка найвища гора в Українських Карпатах?”.

Розрізняють два основних класи алгоритмів, описаних в [6, 11, 21], що використовуються для вирішення задачі виділення відповідей: перший – на основі вилучення шаблону типу відповіді, другий – базується N-грамах.

3.3.1. Алгоритми на основі вилучення шаблону типу відповіді. В методах, що базуються на шаблонах типів відповідей, використовується інформація про очікуваний тип відповіді разом з шаблонами регулярних виразів. Для прикладу, в питаннях з типом відповідей “особа” запускається алгоритм розмітки для типу відповідей або іменованих сутностей на фрагменті чи реченні, яке є кандидатом на відповідь. В якості результату, повертається будь-яка сутність помічена типом “особа”. Проте не всі типи відповідей можуть бути віднесені до певного типу іменованих сутностей. До таких належать наприклад запитання ти-

пу “визначення”. В такому випадку для деяких запитань в якості інструменту для виділення відповіді можуть бути використані регулярні вирази. Окрім того, вони можуть стати в нагоді у випадках, коли фрагмент містить декілька сутностей, що належать до одного і того ж типу.

Ці шаблони специфічні для кожного типу питань і можуть бути побудовані вручну або ж вивчені автоматично. Метою методів вивчення шаблонів є процес вивчення зв’язків чи закономірностей між конкретним типом відповіді та певними аспектами питання. Алгоритм виділення відношень між питаннями та відповідями включає наступні кроки:

1. Для даного відношення між двома термінами алгоритм починає з попередньо визначеного вручну списку коректних пар.
2. Виконуються веб-запити на основі кожної із цих пар та вивчаються перші N отримані документи.
3. Кожен документ розбивається на речення, та зберігаються лише ті речення, що містять обидва терміни.
4. Виділяється шаблон регулярного виразу, що представляє слова та знаки пунктуації, що знаходяться між та навколо цих двох термінів.
5. Зберігаються усі достатньо точні шаблони.

В даному випадку точність вимірюється шляхом здійснення запитів лише за термінами питання, після чого отримані шаблони застосовуються до речень з документу та виділяється відповідь. Оскільки коректна відповідь відома можна підрахувати відсоток випадків, коли шаблон розпізнавав правильну відповідь. Це відсоток використовується для оцінки точності шаблону.

Проте не кожне відношення характеризується однозначними оточуючими словами чи пунктуаційними символами, окрім того в фрагментах відповідей часто зустрічаються декілька сутностей одного і того ж типу. Тож найбільш вдалим методом виділення відповіді є поєднання усіх цих методів разом з іншою інформацією та використання в якості ознак в класифікаторі, який оцінює кандидати на відповіді. Потенційні відповіді виділяють з допомогою іменованих сутностей або шаблонів, або просто перевіряються всі речення отримані після пошуку фра-

гментів. Після чого вони оцінюються використовуючи класифікатор з наступними ознаками:

— співпадіння типу відповіді: істина, якщо кандидат на відповідь містить фразу з потрібним типом відповіді;
— відповідність шаблону: шаблон, що відповідає кандидату на відповідь.
— кількість співпадінь з ключовими словами запитання: кількість ключових слів, що містить кандидат на відповідь;
— відстань до ключового слова: відстань між відповіддю кандидата та ключовими словами запиту (вимірюється середньою кількістю слів або кількістю ключових слів, що зустрічаються в тій самій синтаксичній фразі, що і відповідь кандидата);
— коефіцієнт новизни: істина, якщо в кандидаті на відповідь хоча б одне слово є новим, тобто не міститься в запиті;
— розділові знаки: істинно, якщо після кандидата на відповідь відразу стоїть кома, крапка, лапки, крапка з комою або знак оклику;
— послідовності термінів запитання: довжина найдовшої послідовності термінів із запитання, яка зустрічається у кандидата на відповідь.

3.3.2. Алгоритми на основі N-грам. Даний підхід до вилучення відповідей може бути використаний виключно у веб-пошуку та базується на N-грамах. Його також іноді називають підходом на основі надлишковості.

Цей метод починається з фрагментів, повернутих із веб-пошукової системи та отриманих за допомогою переформульованого запиту. На першому етапі методу вилучається кожна юніграма, біграма та триграма, що зустрічається у фрагменті та обчислюється їх вага. Вага – це функція кількості фрагментів, у яких зустрічається дана N-грама та ваги шаблону переформулювання запиту, який її повернув.

На етапі фільтрування N-грам розраховується оцінка наскільки вони відповідають передбаченому типу відповіді. Ці оцінки обчислюються фільтрами, побудованими для кожного типу відповіді вручну.

На останньому етапі, алгоритм виділення N-грам об'єднує фрагменти N-грам,

що перекриваються, у довші відповіді. Використовується жадібний метод, який починає з кандидата, з найбільшою оцінкою, та зв'язує його з кожним із решти кандидатів. До набору кандидатів додається поєднання з найкращою оцінкою, кандидат з найнижчою оцінкою видаляється. Процес триває, поки не буде побудована одна відповідь.

Для будь-якого з цих методів вилучення відповідей точна відповідь може бути просто представлена користувачеві в тій формі в якій вона є. Проте на практиці користувачів рідко задовольняє єдине число або іменник в якості відповіді, вони хочуть бачити відповідь, супроводжувану достатньою кількістю інформації про фрагменти, щоб обґрунтувати відповідь. Таким чином, користувачеві часто надається цілий фрагмент із підсвіченою або виділеною жирним шрифтом точною відповіддю.

РОЗДІЛ 4
РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ КОНСУЛЬТАЦІЙНОЇ СИСТЕМИ

Розроблювана інтелектуальна консультаційна система повинна надавати відповіді на запитання користувача стосовно організації навчального процесу.

Задача системи полягає в аналізі тексту "Положення про організацію освітнього процесу" [24] та пошуку якомога більш точної та релевантної відповіді на питання, поставлене користувачем. Питання, яке подається на вхід системи формулюється природною мовою.

Оскільки і текст, в якому виконується пошук інформації, і вхідне питання користувача задаються природною мовою, для вирішення даної задачі використовується ряд технік та інструментів для обробки природної мови, а також виконується аналіз тексту та запитання на усіх рівнях, від морфологічного до семантичного.

4.1. Методологія вирішення задачі.

Для вирішення задачі взято за основу методологію систем питань-відповідей, що базуються на пошуку інформації, описану в розділі 3, яка типово включає в себе чотири основні етапи, а саме обробку питання, пошук документу, пошук фрагмента та обробку відповіді. Ці кроки були уточнені, модифіковані та адаптовані для потреби поставленої задачі, а також коректної роботи із врахуванням особливостей та відмінних рис української мови.

4.1.1. Колекція документів, їх аналіз та попередня обробка. В якості джерела інформації для відповідей на питання на питання щодо організації навчального процесу, береться за основу текст "Положення про організацію освітнього процесу" [24].

На першому кроці попередньої обробки та аналізу тексту положення, документ

було переведено у формат текстових даних (англ. plain text) та виконано токенізацію усього тексту на речення та слова. Після чого будувався словник термінів тексту документу, а також частотні списки слів (термінів). Наступним кроком здійснювався процес лематизації (зведення до базової форми) для усіх термінів тексту, та їх входжень в документ.

При виборі підходу до вирішення задачі, було проаналізовано наявні та доступні лінгвістичні ресурси для української мови. В тому числі розглядалися існуючі семантичні мережі, що підтримують українську мову (аналоги мережі "Wordnet"), серед яких "Extended Open Multilingual Wordnet" [25], "ConceptNet.io" [26] та "BabelNet" [27]. Наступним кроком став підрахунок статистики покриття, кожною із цих мереж, термінів та тексту документу. Результати представлені в порівняльній таблиці 4.1, де:

— *покриття термінів* – відсоток термінів (унікальних слів) тексту, що підтримуються даною мережею;

— *покриття тексту* – відсоток тексту, або термінів із урахуванням їх частотності, що покривається даною мережею;

— *обмеження* – наявні обмеження на кількість запитів до мережі.

Таблиця 4.1

Статистика покриття тексту семантичними мережами

	Extended Open Multilingual Wordnet	**ConceptNet**	**BabelNet**
покриття термінів	2.8%	16,5%	31.7%
покриття тексту	6.6%	37.6%	49.5%
обмеження	–	6000 запитів на годину	1000 запитів на добу

Як видно з отриманих результатів, відсотки покриття термінів та тексту даними мережами є досить низькими та, разом з обмеженнями, що накладаються

на кількість допустимих запитів, недостатніми для якісної обробки аналізованого тексту, що робить їх використання не доречними в межах вирішуваної задачі.

На останньому етапі попередньої обробки тексту, що аналізується розроблюваною системою, формувалася колекція документів, де кожен розділ *положення* [24] є окремим документом. Після чого були побудовані семантичні дерева залежностей (англ. dependency tree) для кожного речення в кожному документі, для їх подальшої обробки системою.

4.1.2. Метод обробки питань та кандидатів на відповідь. Запропонований метод обробки питань та кандидатів на відповідь базується на деревах залежностей.

Для кожного речення із колекції документів, дерева залежностей будуються один раз на етапі попередньої обробки. Етап обробки питань та кандидатів на відповідь включає в себе наступні кроки.

На першому кроці відбувається попередня обробка вхідного запитання. Спочатку здійснюється токенізація речення-запитання, та прибираються символи пунктуації, якщо вони є, залишаючи лише послідовність слів (термінів). Далі з речення усуваються запитальні конструкції, такі як: "*як?*", "*яким чином?*", "*за яких умов?*", "*що?*", "*коли?*", "*у якому випадку?*", "*для чого?*" тощо. Після цього виконується лематизація (тобто зведення до базової, інфінітивної форми) усіх слів в реченні та розмітка частин мови. І нарешті, будується дерево залежностей речення-запитання, вершини якого, замість одного терміну, містять трійку (t, l, POS), де t – оригінальний термін із речення у тій формі, в якій він з'являється в реченні, l – визначена лема (базова форма) даного терміну, POS – мітка частини мови даного терміну.

Приклади побудованих дерев залежностей речення-запитання "*За яких умов студентів не включають до рейтингів?*" та для для речення – кандитата на відповідь "*До академічних рейтингів не включають здобувачів освіти, які мають академічну заборгованість на момент укладання рейтингу.*" [24] наведено на рисунках 4.1, 4.2 відповідно.

Рис. 4.1. Дерево залежностей для запитання.

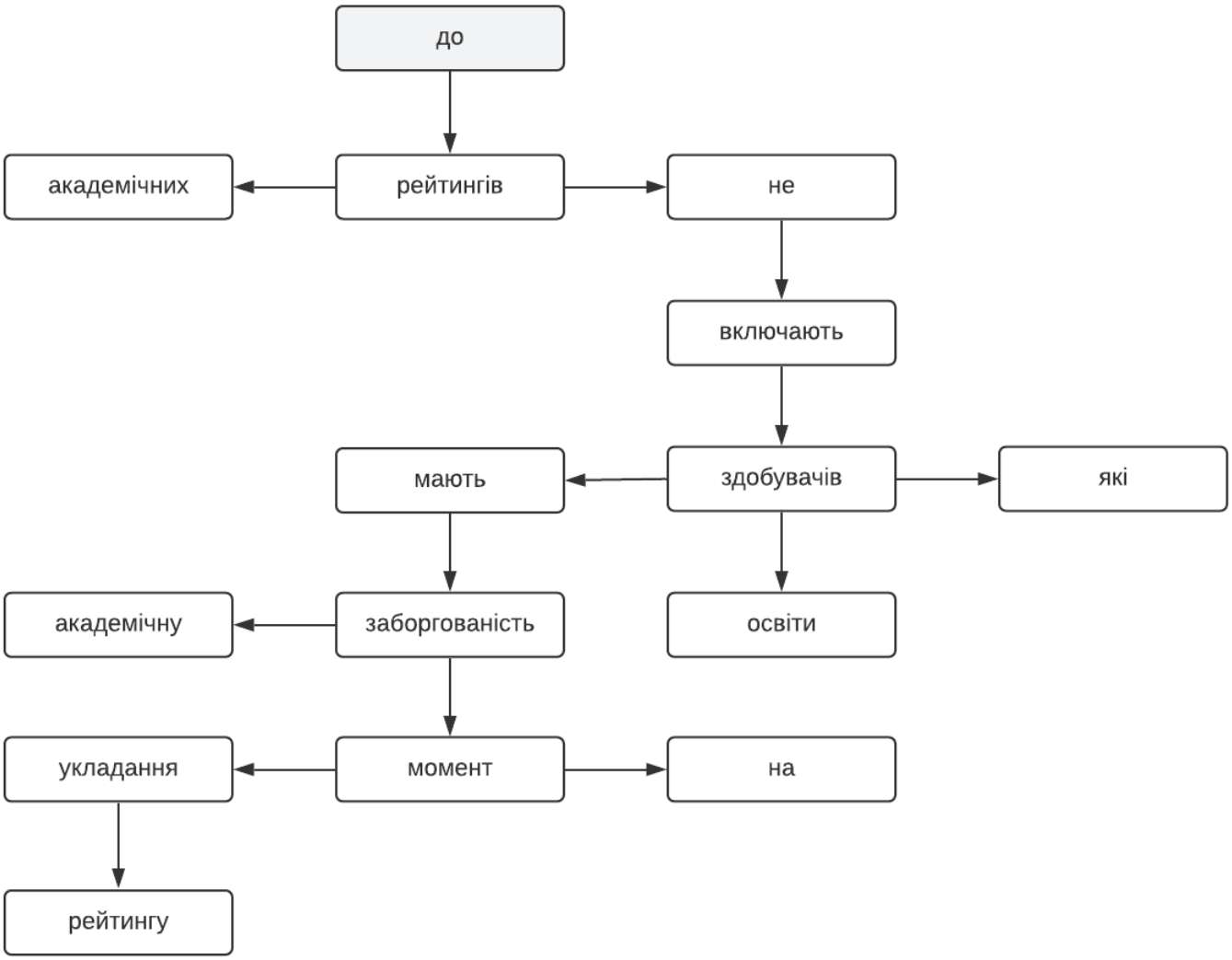

Рис. 4.2. Дерево залежностей для речення-кандидата.

При подальшій обробці побудованих дерев залежностей, алгоритм для підрахунку оцінки того, наскільки дане речення-кандидат підходить для відповіді, працює з лемами, тобто базовими формами слова. Тому дерева 4.1, 4.2, набувають наступного вигляду, наведеного на рисунках 4.3, 4.4.

Рис. 4.3. Дерево залежностей для питання після лематизації.

Оскільки в українській мові допускається відносно вільний порядок слів, але при цьому можна сказати, що конструкція "*не включають*" має таке ж відношення до конструкції "*до рейтингів*" в реченні-запитанні, як і "*до рейтингів*" – до "*не включають*" в реченні-кандидаті, іншими словами відношення між цими конструкціями є рефлексивним. Для коректної обробки таких конструкцій на деревах залежностей вершини зливаються в одну за наступними правилами:

— якщо із вершини, поміченої частиною мови "*прислівник* існує єдиний шлях одиничної довжини у вершину, помічену частиною мови "*іменник*", ці вершини зливаються в одну;

— якщо із вершини, поміченої частиною мови "*іменник*" існує єдиний шлях одиничної довжини у вершину, помічену частиною мови "*прислівник*, ці вершини об'єднуються в одну;

— якщо із вершини, що містить частку "*не*", існує існує єдиний шлях одиничної довжини у вершину, помічену частиною мови "*дієслово*", ці вершини зливаються в одну.

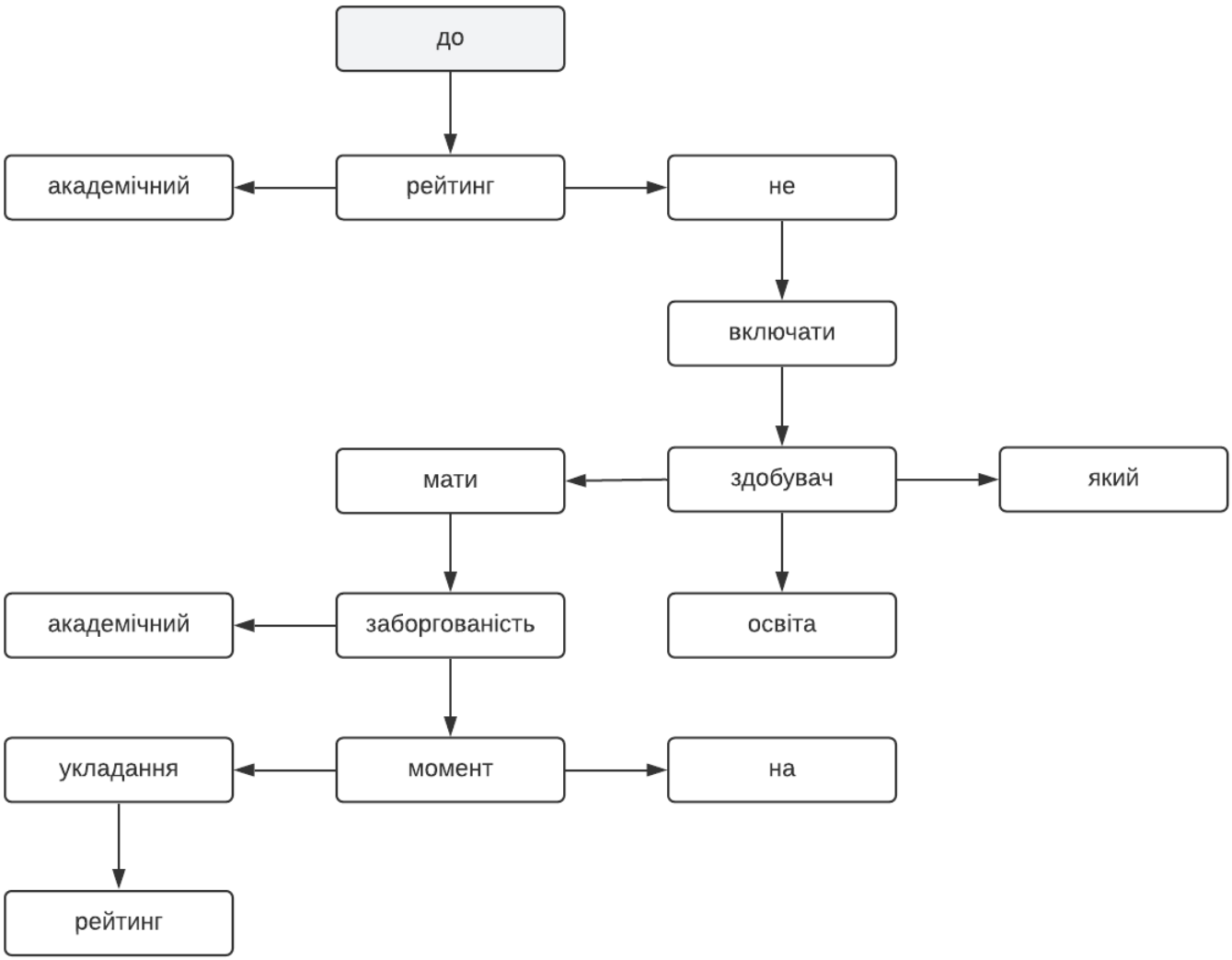

Рис. 4.4. Дерево залежностей для речення-кандидата після лематизації.

Після застосування цих правил та відповідних перетворень в деревах, дерева залежностей 4.3, 4.4, набувають вигляду, як показано на рисунках 4.5, 4.6.

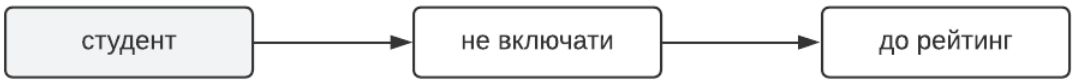

Рис. 4.5. Дерево залежностей для запитання після злиття вузлів.

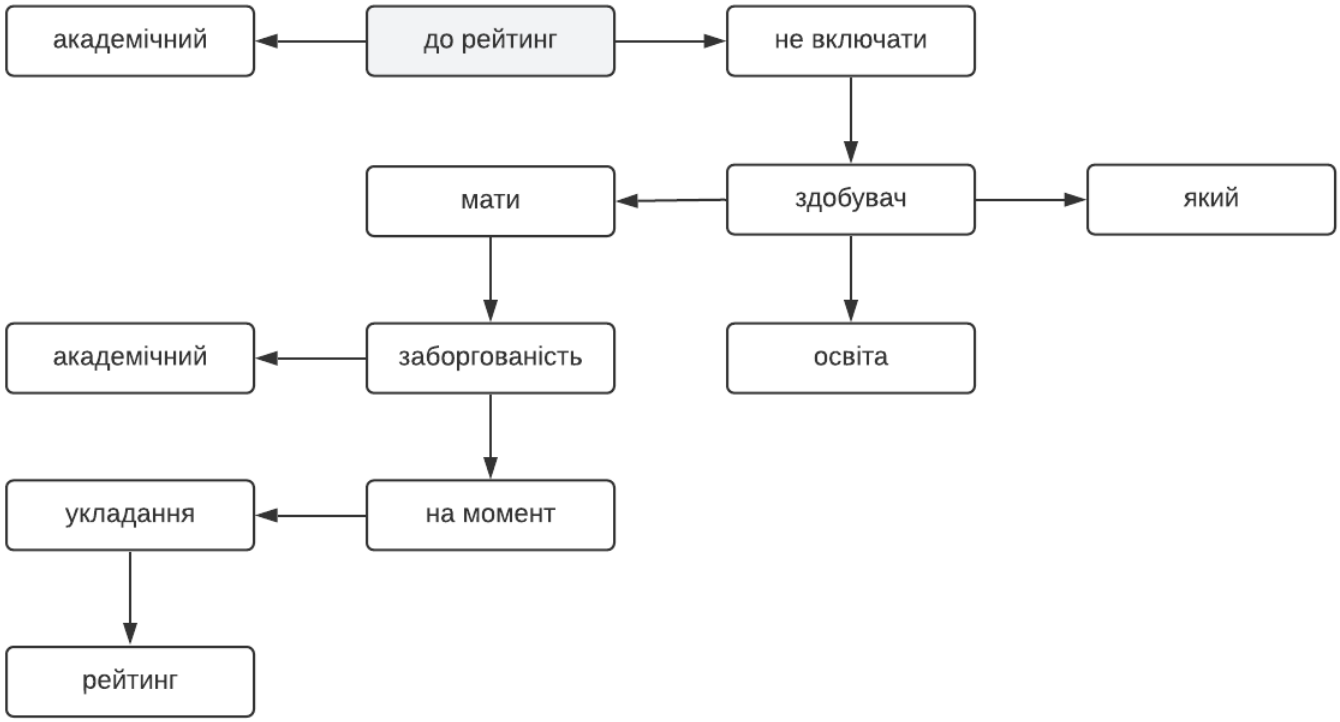

Рис. 4.6. Дерево залежностей для речення-кандидата після злиття вузлів.

На наступному кроці застосовується алгоритм підрахунку оцінки того, наскільки речення-кандидат підходить для відповіді. Даний алгоритм включає наступні три кроки:

1. обертаємо дерево залежностей для речення кандидата, тобто напрям усіх ребер змінюється на протилежний;
2. за допомогою рекурсивного алгоритму 4.1, обраховуються оцінки відповідності s_1 та s_2 між питанням і кандидатом та між питанням і кандидатом за оберненим деревом відповідно;
3. обчислюємо результуючу оцінку як суму s_1 та s_2.

Для відповіді обирається кандидат із найвищим значенням оцінки.

Алгоритм 4.1 Алгоритм для підрахунку оцінки відповідності між деревами залежностей для питання та відповіді

1: **function** GET-SCORE((Q_node, S_node))

2: $score \leftarrow 0$

```
3:  if Q_node.lemma = S_node.lemma then
4:      помітити Q_node та S_node як опрацьовані
5:      score ← score + 1
6:      for all Q_child ∈ нащадки Q_node do
7:          for all S_child ∈ нащадки S_node do
8:              score ← score + GET-SCORE(Q_child, S_child)
9:  else
10:     if S_node не опрацьована then
11:         for all S_child ∈ нащадки S_node do
12:             score ← score + GET-SCORE(Q_node, S_child)
13:     for all Q_child ∈ нащадки Q_node do
14:         if Q_child не опрацьована then
15:             score ← score + GET-SCORE(Q_child, S_node)
16:     return score
```

4.1.3. Метод пошуку фрагментів. В якості одиниць документів розглядаються розділи положення [24]. Пошук здійснюється в два етапи.

На першому кроці документ, тобто розділ, ділиться на менші одиниці – фрагменти. На даному етапі за ці одиниці беруться абзаци. Після чого, за допомогою можливостей інструменту “gensim” [28], виконується пошук по цим фрагментам за оригінальним питанням, з якого усунуто знаки пунктуації та питальна конструкція, як описано в підрозділі 4.1.2. За результатами цього пошуку, виділяється п’ять найбільш відповідних фрагментів, кожен з яких розбивається на менші одиниці – речення. Об’єднання множин, результатів розбиття цих фрагментів, формує множину фрагментів, по яким здійснюється пошук на наступному кроці.

Для другого етапу пошуку, додатково було натреновано модель для української мови, типу питання-відповідь, за допомогою інструменту “tranformers” [29]. На даному етапі для вибору множини кандидатів на відповідь використовуються як можливості інструменту “gensim”, так і підготована модель, та підраховується середнє арифметичне оцінок, отриманих за допомогою цих двох компонентів, для

кожного фрагмента. За результатами цих оцінок формується множина кандидатів на відповідь із усіх фрагментів, значення оцінки яких вище порогового значення.

4.2. Розробка прототипу.

Методи, описані в попередньому підрозділі були реалізовані, як складові частини в розробленому прототипі системи.

4.2.1. Архітектура системи. Архітектура розробленого прототипу системи є модифікацією типової архітектури систем питань-відповідей, що базуються на пошуку інформації, описаних у розділі 3 та наведена на рисунку 4.7.

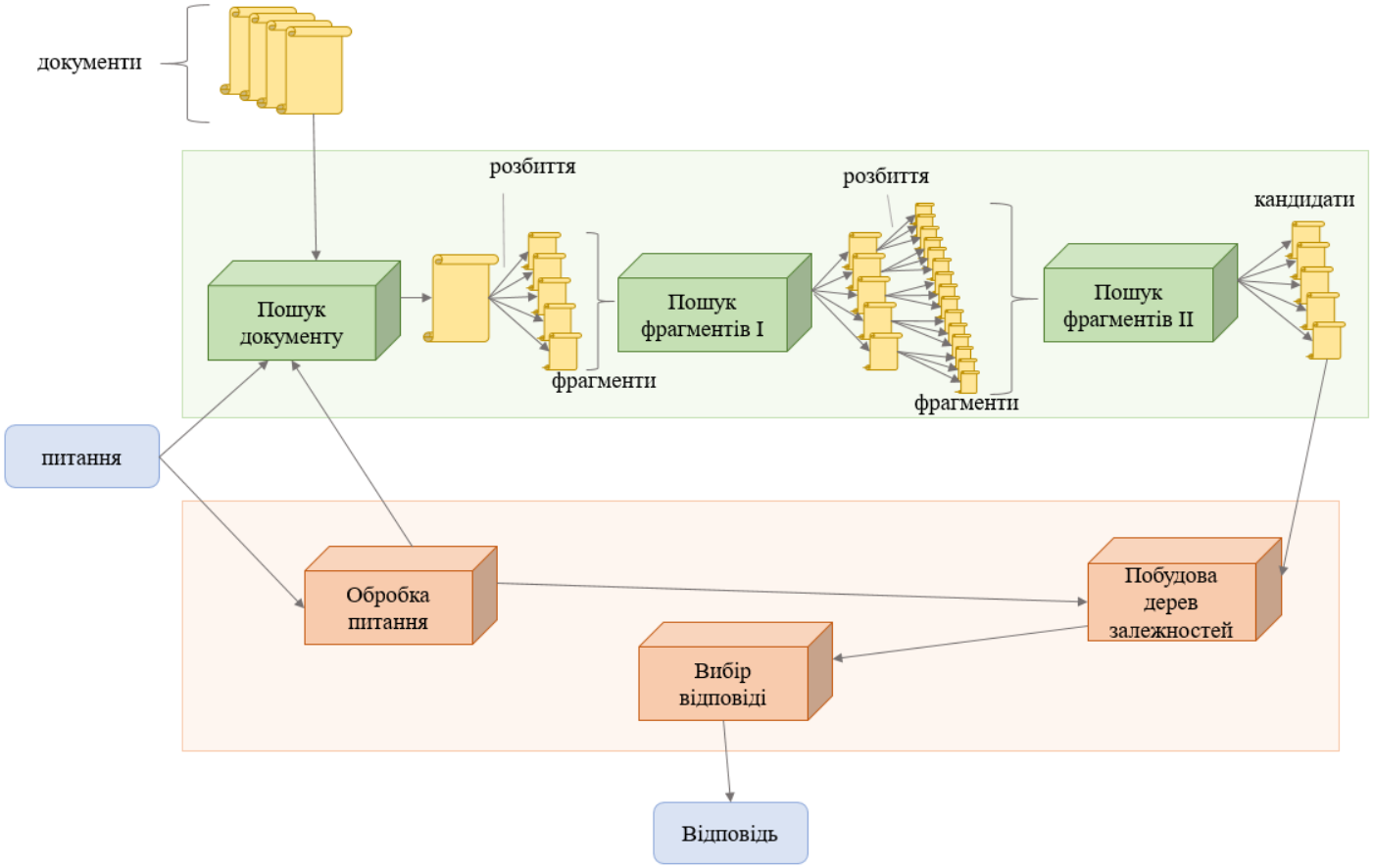

Рис. 4.7. Архітектура системи

На першому кроці здійснюється попередня обробка тексту запитання, після чого воно передається до модулю пошуку за документами. Далі виконується пошук за допомогою інструменту “gensim”, результатом якого є один єдиний документ

(розділ аналізованого тексту), даний крок позначено на рисунку блоком “пошук документу”. Після чого даний документ розбивається на фрагменти, по яким виконується двох-етапний пошук, як описано в підрозділі 4.1.3. Результатом даного кроку є множина речень – кандидатів на відповідь. Ці етапи відображені на схемі блоками “пошук фрагментів I” та “пошук фрагментів II”.

Наступним етапом є обробка питання та речень-кандидатів, побудова дерев залежностей та вибір відповіді за допомогою підходу, описаного в підрозділі 4.1.2.

4.2.2. Використані технології. Прототип системи було реалізовано на мові програмування *Python 3*. В ході розробки було використано наступні інструменти.

Бібліотека для обробки природної мови “gensim” [28] використовувалася на різних етапах для виконання пошуку по документах та фрагментах.

Для тренування моделі та здійснення пошуку на другому етапі пошуку по фрагментах було використано бібліотеку для обробки природної мови “tranformers” [29].

Для побудови семантичних дерев залежностей використовувалася бібліотека “DiaParser” [30]. Також інструмент “NLP UK” [31] було використано для здійснення лематизації та розмітки частин мови.

4.2.3. Напрямки подальшого розвитку. Подальший розвиток системи може бути здійснений шляхом покращення та розширення окремих компонент системи, з метою підвищення точності та якості процесу пошуку інформації, а також розробки додаткових компонент для побудови (генерації) відповіді природним текстом.

Також напрямки подальшої роботи включають розробку та реалізацію методів для здійснення системою висновків на базі тексту знайдених найбільш релевантних відповідей з метою розширення можливостей системи на підтримку класу питань, які передбачають відповіді виду “*так*” і “*ні*” та обгрунтування цих відповідей.

Ще одним шляхом майбутнього покращення системи може бути додавання та обробка нових текстових джерел інформації. Окрім того напрямки можливого розвитку включають розробку мобільного застосунку, веб-застосунку чи інтерфейсу у формі боту для одного чи декількох поширених месенджерів.

ВИСНОВКИ

В роботі було опрацьовано основи обробки природної мови на різних рівнях, включаючи морфологічний, синтаксичний та семантичний аналіз. Більш детально розглянуто такі елементи морфологічного аналізу, як стемінг, лематизація та розмітка частин мови. В межах синтаксичного аналізу розглядалися основи представлення синтаксису, а також підходи засновані на рекурсивних мережах переходів та контекстно-вільний семантичний розбір. Розкрито основні ідеї представлення значення в рамках семантичного аналізу, серед яких такі, що базуються на пропозиційній логіці, логіках першого порядку та лямбда-численні.

Також описано основні принципи систем пошуку інформації та розглянуто основні підходи до пошуку інформації, включаючи підходи, що базуються на: скануванні повного файлу, інвертованих списках, файлах підписів, кластеризації, подібності та ітераційні підходи. Крім того були представлені моделі для пошуку інформації, серед яких булева модель, регіональні моделі, векторна модель, модель ймовірнісного пошуку та моделі мови.

Окрім того було розглянуто основні елементи систем відповідей на питання, а саме принципи обробки запитань, пошуку фрагментів та обробки відповіді. Більш детально описано процеси формування запиту, класифікації та інтерпретації запитань в межах їх обробки. Також розглядалися принципи виділення та класифікації фрагментів на етапі їх пошуку. Детальніше було розглянуто деякі алгоритми обробки відповідей, серед яких алгоритми на основі вилучення шаблону типу відповіді та алгоритми на основі N-грам.

В ході роботи проаналізовано текст “Положення про організацію навчального процесу” для розробки консультаційної системи питань-відповідей на його основі, а також наявні лінгвістичні ресурси для української мови, в тому числі існуючі семантичні мережі “Extended Open Multilingual Wordnet”, “ConceptNet” і “BabelNet”, та підраховано статистику покриття кожною з цих мереж тексту документу з ме-

тою визначення доцільності їх використання в рамках вирішуваної задачі. Окрім того розроблено алгоритм обробки питань та кандидатів на відповідь для оцінки їх відповідності, а також запропоновано метод пошуку фрагментів.

Також було адаптовано типову архітектуру систем відповідей на питання, що базуються на пошуку інформації, до потреб задачі, що вирішувалась. Розроблені алгоритми та архітектура були реалізовані в рамках інтелектуальної консультаційної системи по організації навчального процесу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] M. Z. Kurdi, *Natural Language Processing and Computational Linguistics 1: Speech, Morphology and Syntax*. ISTE Ltd and John Wiley & Sons, Inc., 2016.

[2] R. Hausser, *Foundations of Computational Linguistics*, 3rd ed. Springer, 2014.

[3] D. Chopra, N. Joshi, and I. Mathur, *Maximize your NLP capabilities while creating amazing NLP projects in Python*. Packt Publishing, 2016.

[4] J. Thanaki, *Python Natural Language Processing: Explore NLP with machine learning and deep learning techniques*. Packt Publishing, 2017.

[5] N. Hardeniya, J. Perkins, D. Chopra, N. Joshi, and I. Mathur, *Natural Language Processing: Python and NLTK*. Packt Publishing, 2016.

[6] A. Clark, C. Fox, and S. Lappi, *The Handbook of Computational Linguistics and Natural Language Processing*. Wiley-Blackwel, 2010.

[7] P. Linz, *An Introduction to Formal Languages and Automata*, 6th ed. Jones & Bartlett Learning, LLC., 2017.

[8] J. Eisenstein, *Introduction to Natural Language Processing*. The MIT Press, 2018.

[9] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, 3rd ed. Pearson Education, Inc., 2007.

[10] S. Kandar, *Introduction to Automata Theory, Formal Languages and Computation*. Pearson, 2013.

[11] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition*, 2nd ed. Prentice Hall, 2008.

[12] B. Srinivasa-Desikan, *Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy and Keras*. Packt Publishing, 2018.

[13] A. Meduna and O. Soukup, *Modern Language Models and Computation: Theory with Applications*. Springer, 2017.

[14] A. Meduna, *Formal Languages and Computation: Models and Their Applications*. Taylor & Francis Group, LLC., 2014.

[15] G. E. Revesz, *Introduction to Formal Languages*. Dover Publications Inc., 2012.

[16] M. Z. Kurdi, *Natural Language Processing and Computational Linguistics 2: Semantics, Discourse and Applications*. ISTE Ltd and John Wiley & Sons, Inc., 2018.

[17] V. Brezina, *Statistics in Corpus Linguistics: A Practical Guide*. Cambridge University Press, 2018.

[18] A. Gliozzo and C. Strapparava, *Semantic Domains in Computational Linguistics*. Springer, 2009.

[19] W. Chen and M. Zhang, *Semi-Supervised Dependency Parsing*. Springer, 2015.

[20] B. H. Partee, A. Termeulen, and R. E. Wall, *Mathematical Methods in Linguistics*. Kluwer Academic Publishers, 1990.

[21] G. Cormack, C. Clarke, T. Lynam, and E. Terra, "Question answering by passage selection," in *Advances in Open Domain Question Answering*, ser. Text, Speech and Language Technology, T. Strzalkowski and S. Harabagiu, Eds. Springer, 2008, pp. 283–309.

[22] A. Göker and J. Davies, *Information retrieval: searching in the 21st century*. A John Wiley and Sons, Ltd., Publication, 2009.

[23] S. M. Harabagiu, "Questions and intentions," in *Advances in Open Domain Question Answering*, ser. Text, Speech and Language Technology, T. Strzalkowski and S. Harabagiu, Eds. Springer, 2008, pp. 123–173.

[24] *Положення про організацію освітнього процесу*, Київський національний університет імені Тараса Шевченка. [Online]. Available: http://www.univ.kiev.ua/pdfs/official/Organization-of-the-educational-process.pdf

[25] Extended Open Multilingual Wordnet. [Online]. Available: http://compling.hss.ntu.edu.sg/omw/summx.html

[26] ConceptNet. [Online]. Available: https://conceptnet.io/

[27] BabelNet. [Online]. Available: http://live.babelnet.org/search?&lang=UK

[28] Gensim. [Online]. Available: https://radimrehurek.com/gensim_3.8.3/index.html

[29] transformers. [Online]. Available: https://huggingface.co/transformers/

[30] DiaParser. [Online]. Available: https://pypi.org/project/diaparser/

[31] NLP UK. [Online]. Available: https://github.com/brown-uk/nlp_uk