Міністерство освіти і науки України
Дніпровський національний університет імені Олеся Гончара

Міністерство освіти і науки України
Національна металургійна академія України

Кваліфікаційна наукова
праця на правах рукопису

Ковилін Єгор Романович

УДК 004.822:004.055:519.878(043.5)

ДИСЕРТАЦІЯ

МОДЕЛЬ ГЕНЕРАЦІЇ ВІДПОВІДЕЙ В ПОШУКОВИХ СИСТЕМАХ НА ОСНОВІ НЕСТРУКТУРОВАНОЇ БАЗИ ЗНАНЬ

01.05.02 – математичне моделювання та обчислювальні методи
технічні науки

Подається на здобуття наукового ступеня кандидата наук
Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Ковилін Є.Р.

Науковий керівник Волковський Олег Степанович,
кандидат технічних наук, доцент

Дніпро – 2020

АНОТАЦІЯ

Ковилін Є.Р. Модель генерації відповідей в пошукових системах на основі неструктурованої бази знань. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня кандидата технічних наук за спеціальністю 01.05.02 – математичне моделювання та обчислювальні методи (технічні науки). – Дніпровський національний університет імені Олеся Гончара; Національна металургійна академія України, Дніпро, 2020.

Дисертаційна робота присвячена розробці моделі генерації відповідей на основі неструктурованої бази знань у пошукових системах та програмній реалізації створеної моделі. Об'єктом дослідження є процес автоматичної генерації відповідей з неструктурованої текстової бази знань. Предметом дослідження є модель генерації відповідей з неструктурованої бази знань на основі автоматичного вилучення семантичних характеристик тексту і попередньої класифікації даних.

У роботі проаналізовано існуючі моделі текстів – генеративна граматика, теорія «Сенс ↔ Текст», модель м'якого автоматичного розуміння. Виконано огляд та аналіз існуючих програмних рішень для генерації текстів і вилучення знань на основі нейронних мереж, частотного аналізу та онтологічної розмітки. Проведений аналіз наявної у відкритому доступі науково-технічної літератури і документації показав, що існуючі моделі і програмні рішення для подання слов'яномовних текстів спрямовані саме на опис структурованих знань, і не дозволяють повністю автоматизувати процес опису семантичних властивостей та адаптивного додавання неструктурованих знань до пошукової системи. З огляду на це, визначена доцільність і актуальність розробки моделі генерації відповідей на основі неструктурованої бази знань в пошукових системах.

Обґрунтовано вибір базової моделі для обробки неструктурованих знань – моделі м'якого автоматичного розуміння, до плюсів якої відносяться початкова орієнтованість на синтез, висока роль семантики в моделі, незалежність від синтаксичної структури тексту, вирішення проблеми флексії мови тексту та

орієнтованість на спрощену базу знань. Визначені пріоритетні наукові питання, які поставлені у дисертації, що пов’язані з організацією навчання моделі, подоланням обмежень семантичного словника і гнучким налаштуванням роботи моделі при підключенні нових предметних областей текстових даних.

У роботі поставлена та вирішена актуальна науково-технічна задача обробки семантично-неструктурованих документів для генерації відповідей в пошукових системах. Для цього виконано розробку математичної моделі представлення семантичних властивостей текстів, математичної моделі валідації текстів-кандидатів до включення у неструктуровану базу знань за ступенем їх семантичної зв’язності та моделі генерації текстів у пошукових системах. Отримані нові обґрунтовані результати, які відповідно до поставленої мети дають вирішення актуальної задачі побудови моделі отримання знань із неструктурованих джерел.

В роботі вперше розроблено семантичну модель текстових даних, яка на відміну від існуючих аналогів, дозволяє отримувати кількісні показники семантичних властивостей і сенсові зв’язки між компонентами тексту без необхідності будь-якої попередньої семантичної розмітки, впровадження словників або залучення лінгвістичних знань. Використання такого підходу для роботи із семантичними характеристиками тексту є інноваційним, оскільки:

- сфера застосування латентно-семантичного аналізу в першу чергу стосується задач класифікації документів, тоді як у розробленій моделі, його використання було змінено – модель наближає не документ до терміну, а речення з документу до термінів з документу;
- модель поєднує у собі багато специфічних додаткових етапів, що є нетиповими для підходів побудови семантичної моделі документу. Мова йде про використання алгоритму кластеризації із відповідними методами для визначення необхідних параметрів, алгоритмів синтаксичного, морфологічно і просторового аналізу даних.

Для оцінки отриманих за допомогою семантичної моделі результатів була сформульована та виконана поетапна перевірка семантичних властивостей

розробленої моделі, для визначення залежностей саме від семантичних, а не від частотних характеристик документу. Виконаний ряд порівняльних експериментів довів, що отримана модель є адекватною і може застосовуватися задля безсловникового універсального підходу до кількісного опису семантичних властивостей тексту.

Встановлено, що через значно більшу кількість кластерів-стем в моделях хаотично згенерованого тексту підтверджується головна гіпотеза використання семантичної моделі в задачах автоматичної обробки знань в системах генерації відповідей – терміни не об'єднуються в семантичні мітки, оскільки, не існує ніякого семантичного зв'язку знань в початковому тексті, що наочно показує адекватність розробленої моделі.

Було встановлено, що кількість семантично значимих термінів у кластері-стемі прямо пропорційна до кількості та ваги пов’язаних із ним семантичних контурів речень у побудованих моделях документів, що доводить залежність структури семантичної мережі саме від семантичної складової тексту. Завдяки цьому стає можливим подолати обмеження словникових моделей, яке вимагає використання лінгвістичних знань для створення прикладних програмних моделей автоматичної генерації текстів і автоматичної класифікацій документів.

Вперше створено модель автоматичної класифікації знань за ступенем їх семантичної зв’язності, яка використовує числові дані, отримані із розробленої семантичної моделі тексту, що дозволило збільшити надійність використання моделі генерації відповідей шляхом верифікації первинної інформації.

Проведені експерименти показали коректну класифікацію у 90 відсотках оброблених текстів. При цьому, жоден прогноз щодо незв’язних текстів не був хибним – це стосується як цілком хаотично згенерованих текстів, що вказує на коректну обробку статичних властивостей тексту, так і семантично-фрагментарних текстів, що говорить про коректну обробку семантично незв’язних текстів. Такі результати вказують не тільки на достатню і задовільну точність роботи класифікатора і можливість його подальшого використання, а і на

адекватність семантичної моделі документу в цілому і доцільності її застосування в інтелектуальних системах автоматичної обробки текстів.

Вперше побудовано модель автоматичної генерації відповідей у пошуковій системі із неструктурованої бази текстових знань на основі створених моделей, яка дозволила автоматизувати роботу користувача із пошуковими системами.

Була теоретично визначена та описана база знань системи, яка покладена у основі процесу генерації відповідей, сформульовані теоретичні аспекти побудови корпусів та колекцій текстів, визначені основні властивості, яких необхідно дотримуватися при складанні та обробці корпусу, описаний та обґрунтований розподіл класів текстів, які відображають репрезентативність вибірки. Окрім того, наведено структуру та детальний опис моделі електронної бази даних, яка покладена в основу базу знань моделі.

Вперше розроблено систему оцінок адекватності створеної моделі генерації відповідей на основі неструктурованої бази знань, яка ґрунтується на вже сформованих алгоритмах та даних, і дозволяє з одного боку перевірити тематичну спрямованість відповідей, а з іншого – уникнути ситуації виникнення глибоких семантичних розривів у відповідях. Проведена оцінка отриманої моделі генерації відповідей на основі бального метода. Кількісні результати оцінок, отриманих на обидвох етапах співали і доводять коректність роботи створеної системи в цілому, доцільність її використання як базового інструменту вилучення знань з колекцій текстів та адекватність розроблених моделей.

В роботі отримали подальший розвиток методи організації пошуку інформації: створені моделі дозволяють генерувати релевантні до запиту користувача знання на основі неструктурованої бази знань, чим спрощують роботу користувача із пошуковими системами.

Вдосконалено існуючі семантичні моделі текстових даних для флективно багатих мов із вільним порядком слів: розроблена семантична модель текстових даних дозволяє уникнути процесу ручного опису семантичної структури документа.

На основі створених моделей здійснена програмна реалізація пошукової моделі, яка окрім організації зручного пошукового середовища містить універсальний програмний фреймворк з набором інструментів для проведення автоматичної обробки текстів на семантичному рівні, доступний для будь-якого користувача у вигляді окремої бібліотеки.

Задачі дисертаційної роботи є важливими і нагальними питаннями галузі цифрового моделювання текстів, вирішення яких дозволяє гнучко створювати і обробляти тематичні повнотекстові бази знань без попередньої семантичної розмітки та будувати програмну модель текстових знань формалізованої стильової спрямованості із кількісними характеристиками семантичних властивостей, на основі яких можливо вирішувати інші завдання автоматичної обробки текстів без необхідності залучати будь-які лінгвістичні знання.

Проведені експерименти показали, що використання розробленої моделі дозволяє покращити тематичну якість пошуку текстів-кандидатів для генерації відповідей в 1,7 рази порівняно з прямим вирішенням пошукової задачі. Експертні оцінки, проведені за бальним методом, показали значення у 0,839, що доводить адекватність створеної моделі.

Розроблений програмний додаток впроваджено:

– у міський комунальний заклад культури «Централізована система бібліотек для дітей» м. Дніпро як пошуковий інструмент обробки електронних текстів;

– у ТОВ «Сітал Україна» як засіб автоматичної генерації текстових інструкцій;

– у АТ «ДніпроАзот» як інструмент покращення процесів пошуку в системах електронного документообігу.

Ключові слова: Модель пошукової системи, семантична модель, неструктуровані текстові знання, генерація відповідей, комп’ютерні моделі представлення знань.

ABSTRACT

Kovilin Y.R. Model of answers generating in the search engines based on an unstructured knowledge base. – Qualifying scientific work on the rights of the manuscript.

Thesis for obtaining the candidate degree (Ph.D.) in the specialty 01.05.02 – Mathematical modelling and computational methods (Technical science). – Oles Honchar Dnipro National University; The National Metallurgical Academy of Ukraine, Dnipro, 2020.

The dissertation is devoted to the development of the answers generation model on the basis of unstructured knowledge base in the search engines and the program realization of the created model. The object of research is the process of automatic generation of answers from an unstructured textual knowledge base. The subject of the research is a model of answers generating from an unstructured knowledge base, based on the automatic extraction the semantic characteristics of the text and preliminary data classification.

The existing models of texts – generative grammar, theory "Meaning ↔ Text", model of soft automatic comprehension are analyzed in the dissertation. Review and analysis of existing software solutions for text generation and knowledge extraction based on neural networks, frequency analysis and ontological markup was performed.

The analysis of publicly available scientific and technical literature and documentation showed, that the existing models and software solutions for the presentation of Slavic texts are aimed at describing structured knowledge, and do not fully automate the process of describing semantic properties and adaptive addition of unstructured knowledge to the search engine. Given this, the feasibility and relevance of developing a model for answers generating based on an unstructured knowledge base in search engines were identified.

The choice of the basic model for unstructured knowledge processing is justified – a model of soft automatic comprehension, the advantages of which include the initial focus on synthesis, high role of semantics in the model, independence from syntactic

structure of the text, solving the problem of text language flexion and focus on simplified knowledge base. The priority scientific issues identified in the dissertation are related to the organization of model learning, overcoming the limitations of the semantic vocabulary and flexible configuration of the model when connecting new subject areas of text data.

The actual scientific and technical problem of processing semantically unstructured documents for the answers generating in search engines is set and solved in the work.

To do this, we developed a mathematical model for presenting the semantic properties of texts, a mathematical model for validating candidate texts for inclusion in an unstructured knowledge (by their semantic connectivity) and a model for the texts generating in the search engines. New substantiated results are obtained, which in accordance with the set goal give the solution of the actual problem of a model of obtaining knowledge from unstructured sources building.

For the first time, a semantic model of textual data was developed, which, unlike existing analogues, allows to obtain quantitative indicators of semantic properties and semantic connections between text components without the need for any prior semantic markup, introduction of dictionaries or linguistic knowledge.

Using this approach to work with the semantic characteristics of the text is innovative because:

- the scope of latent-semantic analysis primarily concerns the problems of document classification, while in the developed model, its use has been changed - the model does not bring the document closer to the term, but the model brings the sentence from the documents to the terms from the document;
- the model combines many specific additional stages, which are not typical for approaches to building a semantic model of the document. It is a question of use of algorithm of clustering, with the corresponding methods for definition of necessary parameters, algorithms of syntactic, morphological and spatial analysis of data.

To evaluate the results obtained with the help of the semantic model, a step-by-step verification of the semantic properties of the obtained model was formulated and performed, aimed at determining the dependence of the obtained model on the semantic rather than frequency characteristics of the document.

A number of comparative experiments proved that the obtained model is adequate and can be used for a wordless universal approach to the quantitative description of the semantic properties of the text.

It is established that due to a much larger number of stem clusters in the chaotically generated text models, the main hypothesis of using the model in automatic knowledge processing problems in answer generation systems is confirmed - terms are not combined into semantic labels because there is no semantic connection of knowledge in the initial text that clearly shows the adequacy of the developed model.

It was found that the number of semantically significant terms in the cluster-stem is directly proportional to the number and weight of related semantic contours of sentences in the constructed document models, which proves the dependence of the semantic model structure on the semantic component of the text. This makes it possible to overcome the limitations of vocabulary models, which requires the use of linguistic knowledge to create applied software models for automatic text generation and automatic document classification.

For the first time, a model of automatic classification of knowledge according to the degree of their semantic coherence was created, which uses numerical data obtained from the developed semantic model of the text, which increased the reliability of using the answer generation model by verifying primary information.

The conducted experiments showed the correct classification of 90 percent of the processed texts. At the same time, no prediction for incoherent texts was wrong - this applies to both completely chaotically generated texts, which indicates the correct processing of static properties of the text, and semantically fragmentary texts, which indicates the correct processing of semantically incoherent texts.

Such results indicate not only the sufficient and satisfactory accuracy of the classifier and the possibility of its further use, but also the adequacy of the semantic

model of the document as a whole and the feasibility of its use in intelligent automatic text processing systems.

For the first time, a model of automatic generation of answers in a search engine from an unstructured text knowledge base based on the created models was built, which allowed to automate the user's work with search engines.

The knowledge base of the system, which is the basis of the process of generating answers, was theoretically defined and described, the theoretical aspects of building corpora and collections of texts were formulated, the main properties of the system were followed when compiling and processing the corpus, described and substantiated the distribution of text classes that reflect the representativeness of the sample. In addition, the structure and detailed description of the electronic database model, which is the basis of the model knowledge base, are given.

For the first time a system of assessments of adequacy of the created model of generation of answers on the basis of unstructured knowledge base is developed, which is based on already formed algorithms and data, and allows to check thematic orientation of answers, the received model of generation of answers on the basis of a point method is estimated.

Quantitative results of assessments obtained at both stages sang and prove the correctness of the system as a whole, the feasibility of its use as a basic tool for extracting knowledge from text collections and the adequacy of the developed models.

The methods of information search organization were further developed in the work: the created models allow to generate knowledge relevant to the user's request on the basis of unstructured knowledge base, which simplifies the user's work with search engines.

Existing semantic models of the text data for inflectionally rich languages with free word order have been improved: the developed semantic model of text data avoids the process of manual description of the semantic structure of the document.

Based on the created models, a program implementation of the search engine is carried out, which, in addition to organizing a user-friendly search environment,

contains a universal software framework with a set of tools for automatic semantic processing, available to any user as a separate library.

The tasks of the dissertation are important and urgent issues in the field of digital modeling of texts, the solution of which allows to flexibly create and process thematic full-text knowledge bases without prior semantic markup and build a program model of textual knowledge of formalized stylistic orientation with quantitative characteristics of semantic properties, on the basis of which it is possible to solve other tasks of automatic word processing without the need to involve any linguistic knowledge.

The conducted experiments showed that the use of the developed model allows to improve the thematic quality of the search of candidate texts for generating answers by 1.7 times in comparison with the direct solution of the search problem. Expert assessments conducted by the scoring method showed a value of 0.839, which proves the adequacy of the created model.

The developed software application was implemented:

- in the city municipal cultural institution "Centralized system of libraries for children" in Dnipro as a search tool for electronic text processing;
- in LLC "Sital Ukraine" as a means of automatic generation of text instructions;
- in JSC "DniproAzot" as a tool to improve search processes for electronic document management system.

Keywords: Search engine model, semantic model, unstructured text knowledge, response generation, computer models of knowledge representation.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Наукові праці, в яких опубліковані основні результати дисертації:

1. Волковський О.С. Computer methods for compiling an entry of explanatory combinatorial dictionary belonging to «Meaning↔Text» theory within the task of text automatic generation / О.С. Волковський, Є.Р. Ковилін // Математичне моделювання // науковий журнал – Кам'янське, 2018. – №2 (39). – с. 9–18. Видання включено до НМБ Index Copernicus International.
2. Волковський О.С. Analysis of the modern approaches to the problem of the automatic text generation in the natural language / О.С. Волковський, Є.Р. Ковилін // Системні технології. Регіональний збірник міжвузівських наукових праць // науковий журнал – Дніпро, 2016. – №5 (106). – с. 3–12. Видання включено до НМБ Index Copernicus International.
3. Волковський О.С. Комп'ютерна система автоматичного визначення зв'язності тексту / О.С. Волковський, Є.Р. Ковилін // Системні технології. Регіональний збірник міжвузівських наукових праць. // науковий журнал – Дніпро, 2017. – №1 (112). – с. 11–17. Видання включено до НМБ Index Copernicus International.
4. Волковський О.С. Комп'ютерна система автоматичного аналізу промислових інструкцій. / О.С. Волковський, Є.Р. Ковилін // Системні технології. Регіональний збірник міжвузівських наукових праць. // науковий журнал – Дніпро, 2018. – №3 (116). – с. 28–37. Видання включено до НМБ Index Copernicus International.
5. Волковський О.С. Комп'ютерна система інтелектуального семантичного пошуку з використанням генерації текстів / О.С. Волковський, Є.Р. Ковилін // Вісник Херсонського національного технічного університету // науковий журнал – Херсон, 2018. – №3 (66). – с. 238 –245. Видання включено до НМБ Google Scholar.

6. Волковський О.С. Mathematical model for automatic creation the semantic thesaurus for the scientific text / О.С. Волковський, Є.Р. Ковилін // Системні технології. Регіональний збірник міжвузівських наукових праць. // науковий журнал – Дніпро, 2019. –. №6 (125). – с. 82–88. Видання включено до НМБ Index Copernicus International.
7. Волковський О.С. Модель автоматичної оцінки адекватності комп'ютерних систем «запит-відповідь» з використанням генерації текстів / О.С. Волковський, Є.Р. Ковилін // Системні технології. Регіональний збірник міжвузівських наукових праць. // науковий журнал – Дніпро, 2020 – №4 (129). – с. 50–58. Видання включено до НМБ Index Copernicus International.
8. Volkovsky O.S. Matematical model for constructing the semantic network of a scientific text / O.S. Volkovsky, Y. R. Kovylin. // Modern engineering and innovative technologies // науковий журнал – Карлсруе, Німеччина, 2020 – №11 – с. 128 – 133. Видання включено до НМБ Index Copernicus International.

Публікації апробаційного характеру:

9. O.S. Volkovsky. Computer System of Building of the Semantic Model of the Document / O.S. Volkovsky, Y. R. Kovylin. // 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP): міжнародна науково-практична конференція, 21-25 серпня 2018 р.: тези доп. – Львів, 2018 – с. 322–327. (Конференція включена до НМБ Scopus).
10. Волковський О.С. Система автоматичного аналізу текстів природною мовою / О.С. Волковський, Є.Р. Ковилін // Інформаційні технології в металургії та машинобудуванні: міжнародна науково-технічна конференція, 28–30 березня 2017р.: тези доп. – Дніпро, 2017 – с. 112.
11. Волковський О.С. Автоматична побудова семантичної мережі тексту у системах запит-відповідь / О.С. Волковський, Є.Р. Ковилін // Інформаційні

технології та комп’ютерне моделювання: міжнародна науково-практична конференція, 15 – 20 травня 2017 р.: тези доп. – Івано-Франківськ, 2017 – с. 386 – 389.

12. Волковський О.С. Комп’ютерна система автоматичного аналізу промислових інструкцій.. / О.С. Волковський, Є.Р. Ковилін // Інформаційні технології в металургії та машинобудуванні: міжнародна науково-технічна конференція, 27–29 березня 2018 р.: тези доп. – Дніпро, 2018. – с. 124.
13. Волковський О.С. Комп’ютерна модель семантичної мережі документу в системі запит-відповідь / О.С. Волковський, Є.Р. Ковилін // Проблеми математичного моделювання: всеукраїнська науково-методична конференція, 23–25 травня 2018 р.: тези доп. – Кам’янське, 2018. – с.33–36.
14. Волковський О.С. Комп’ютерна система інтелектуального семантичного пошуку з використанням генерації текстів / О.С. Волковський, Є.Р. Ковилін // Матеріали XIX міжнародної конференції з математичного моделювання, присвяченої 250-річчю з дня народження Жозефа Фур’є, 17–21 вересня 2018 р.: тези доп. – Лазурне, 2018. – с. 49.

ЗМІСТ

ВСТУП

Актуальність роботи. Задача пошуку інформації є однією з ключових в області комп'ютерних наук. Якщо в функціоналі системи присутня необхідність працювати безпосередньо з користувацьким запитом, який часто складається з декількох неформальних критеріїв і вимагає певного семантичного аналізу, то обробка отриманих результатів повністю лягає на плечі користувача. Прикладом такого підходу є модель, яка використовується у багатьох популярних web-пошукових системах: відповіддю на отриманий запит є множина документів, що припускає подальший самостійний аналіз користувачем кожного документа для пошуку відповіді на своє питання. Головним мінусом такого підходу є відсутність глибокого семантичного розуміння вмісту документа моделлю, у результаті чого в отриманому масиві документів міститься велика кількість не пов'язаної із запитом користувача інформації, а також множини повторень однакової інформації, поданої в різних інтерпретаціях, через що процес пошуку значно ускладняється.

Вирішенням цієї задачі є використання процесу автоматичної генерації знань, який дозволить відразу отримати релевантні до запиту користувача знання на основі усіх знань, що містяться у системі, та вдосконалити таким чином процес пошуку даних для людини-оператора. Головною складністю впровадження генерації знань у пошукові моделі є необхідність побудови семантичної моделі документів, доступної для обробки комп'ютером. Окремо слід відзначити ускладнення, пов'язане із властивостями флексії та типології порядку слів мови документа, через що моделі, створені наприклад для англійської мови, не є застосовними для документів на слов'янській мові.

Через це, головним підходом до створення семантичних моделей документів на слов'янських мовах є ручна побудова семантичної структури тексту, яка може бути представлена у вигляді онтологічної розмітки або семантичних словників, і вимагає використання великого обсягу ручної праці для їх складання і пошуку фахівців у прикладній лінгвістиці, що значно ускладнює

використання автоматичної генерації знань у пошукових системах та обмежує застосування моделі для довільної колекції семантично неструктурованих текстів.

Питання розробки схожих моделей представлення текстів для їх подальшої комп'ютерної обробки подані у багатьох роботах, які стосуються галузей штучного інтелекту, математичного моделювання та обробки природної мови. Серед вітчизняних вчених слід відзначити наукові розробки Н.Н. Леонтьевої, М.В. Мозгового, В.А. Тузова, І.О. Мельчука, Ю.Д. Апресяна, спрямовані на математичне моделювання семантичних властивостей тексту на основі онтологій та семантичних словників. Зарубіжні англомовні моделі представлення текстів відштовхуються від жорсткого порядку слів у мові, як наприклад в роботах N. Chomsky. В області вилучення знань із текстів панують онтологічні математичні моделі, описані в роботах B. Yildiz, S. Miksch, R. Navigli, S. Dill, L.K. McDowell, D. Faure.

Проведений аналіз наявної у відкритому доступі науково-технічної літератури і документації показав, що існуючі моделі представлення слов'яномовних текстів спрямовані саме на опис структурованих знань, і не дозволяють повністю автоматизувати процес опису семантичних властивостей та адаптивного додавання неструктурованих знань до пошукової системи. Усі вони мають на увазі залучення значної кількості лінгвістичних знань і використання попередньої семантичної розмітки, створеної лінгвістом-експертом вручну. З іншого боку, зарубіжні моделі орієнтовані в першу чергу на обробку англомовних текстів і не можуть бути застосовані для представлення слов'яномовних текстів.

Таким чином, недосконаліть процесу пошуку інформації і відсутність цілком автоматичних моделей представлення слов'яномовних текстів дозволяє зробити висновок про те, що на сьогоднішній день задача розробки моделі генерації відповідей на основі неструктурованої бази знань в пошукових системах є актуальним завданням.

Зв'язок роботи з науковими програмами. Робота виконана відповідно до закону України «Про пріоритетні напрями розвитку науки і техніки» (Відомості Верховної Ради України (ВВР), 2001, № 48, ст.253), і стосується напряму

«інформаційні та комунікаційні технології» (стаття 3). Обраний напрямок досліджень пов’язаний із виконанням дослідних робіт кафедри комп’ютерних наук та інформаційних технологій Дніпровського національного університету імені Олеся Гончара «Методи та інформаційні технології цифрової обробки багатоканальних даних» (реєстраційний номер 0116U001297).

Мета та задачі дослідження. Метою дисертації є розробка моделі обробки семантично-неструктурованих документів для генерації відповідей в пошукових системах.

Для досягнення мети були поставлені та вирішені наступні задачі дисертаційного дослідження:

1) визначити концепції моделі вилучення інформації та архітектури системи, для чого провести аналіз і порівняльну характеристику фундаментальних моделей уявлення текстових знань, з метою обґрунтованого вибору оптимальної для вирішуваної задачі;
2) розробити математичну модель представлення семантичних властивостей текстів, спроможну працювати із достатньо формалізованим типом тексту і автоматично формувати програмну семантичну модель як окремого документа, так і всього корпусу знань в цілому;
3) забезпечити розроблену модель можливістю використовувати семантично нерозмічений заздалегідь корпус текстів. Модель повинна функціонувати без використання лінгвістичних знань про мову або семантичні властивості текстів;
4) розробити математичну модель валідації текстів-кандидатів до включення у неструктуровану базу знань за ступенем їх семантичної зв’язності, яка захистить неструктуровану базу знань від наповнення помилковими даними;
5) на основі семантичної моделі та моделі валідації, розробити математичну модель генерації текстів, її алгоритмічне та програмне забезпечення;
6) розробити і застосувати систему оцінок адекватності створеної моделі генерації відповіді на основі неструктурованої бази знань.

Об’єкт дослідження. Процес автоматичної генерації відповіді з неструктурованої текстової бази знань.

Предмет дослідження. Модель генерації відповідей з неструктурованої бази знань на основі автоматичного вилучення семантичних характеристик тексту і попередньої класифікації даних.

Методи дослідження. Для вирішення поставлених задач були використані методи частотного аналізу текстів, факторного латентно-семантичного аналізу, кластерного аналізу, індукційного і дедуктивного аналізу, вимірювання, експерименту, гіпотези, припущення, комп’ютерного моделювання отриманих результатів. Застосовано засоби теорії множин, штучного інтелекту і проекційного методу.

При розробці програмної реалізації побудованих моделей були застосовані: технологія об’єктно-орієнтованого програмування, реляційна модель зберігання даних і засоби CASE-проектування архітектури програмного додатку.

Для дослідження адекватності розроблених моделей використано статистичні методи і метод бальних оцінок.

Наукова новизна отриманих результатів. В роботі **вперше** отримано такі результати:

1) розроблено семантичну модель текстових даних, яка на відміну від існуючих аналогів, дозволяє отримувати кількісні показники семантичних властивостей і сенсові зв’язки між компонентами тексту без необхідності будь-якої попередньої семантичної розмітки, впровадження словників або залучення лінгвістичних знань;
2) створено модель автоматичної класифікації знань за ступенем їх семантичної зв’язності, що використовує числові дані, отримані із розробленої семантичної моделі тексту, яка дозволила збільшити надійність використання моделі генерації відповідей шляхом верифікації первинної інформації;

3) побудовано модель автоматичної генерації відповідей у пошуковій системі із неструктурованої бази текстових знань на основі створених моделей, яка дозволила автоматизувати роботу користувача із пошуковими системами;
4) розроблено систему оцінок адекватності створеної моделі генерації відповіді на основі неструктурованої бази знань;

отримали подальший розвиток:

5) семантичні моделі текстових даних для флективно багатих мов із вільним порядком слів: розроблена семантична модель текстових даних дозволяє уникнути процесу ручного опису семантичної структури документа;
6) методи організації пошуку інформації: створені моделі дозволяють генерувати релевантні до запиту користувача знання на основі неструктурованої бази знань, чим спрощують роботу користувача із пошуковими системами.

Практичне значення одержаних результатів. Розроблено математичну модель генерації відповідей в пошукових системах на основі неструктурованої бази знань та побудовано на її основі комп'ютерну систему, яка окрім організації зручного пошукового середовища, утворює універсальний програмний фреймворк з набором інструментів для проведення автоматичної обробки текстів на семантичному рівні, доступним для будь-якого користувача у вигляді окремої бібліотеки.

Формат реалізації додатку пояснюється актуальністю задачі через відсутність альтернативних API із відкритим доступом. Задачі дисертаційної роботи є важливими і нагальними питаннями галузі цифрового моделювання текстів, вирішення яких дозволяє:

1. гнучко створювати і обробляти тематичні повнотекстові бази знань без попередньої семантичної розмітки;
2. будувати програмну модель текстових знань формалізованої стильової спрямованості із кількісними характеристиками семантичних властивостей, на основі яких можливо вирішувати інші завдання автоматичної обробки текстів без необхідності залучати будь-які лінгвістичні знання.

Особистий внесок здобувача. Основні результати дисертаційної роботи опубліковано в статтях:

[1, 2] – аналіз існуючих розробок і моделей автоматичного уявлення текстових знань та оцінка можливості їх застосування для вирішення задачі побудови моделі генерації відповідей із неструктурованої бази знань;

[3, 4] – розробка моделі автоматичної класифікації формалізованих текстових знань на основі моделі автоматичного уявлення семантичних характеристик тексту із неструктурованої бази текстових знань;

[5] – побудова моделі і комп'ютерної системи інтелектуального семантичного пошуку з використанням генерації текстів;

[6, 8] – побудова моделі автоматичного уявлення семантичних характеристик тексту із неструктурованої бази текстових знань;

[7] – розробка моделі автоматичної оцінки адекватності комп'ютерних систем «запит-відповідь» з використанням генерації текстів та оцінка адекватності створеної пошукової моделі.

Апробація результатів дисертації. Основний зміст та висновки дисертаційної роботи викладені та обговорені на 6 міжнародних та всеукраїнських конференціях: міжнародній науково-технічній конференції «Інформаційні технології в металургії та машинобудуванні-2017» (м. Дніпро), міжнародній науково-практичній конференції «Інформаційні технології та комп'ютерне моделювання-2017» (м. Івано-Франківськ), міжнародній науково-технічній конференції «Інформаційні технології в металургії та машинобудуванні-2018» (м. Дніпро), міжнародній науково-технічній конференції «IEEE Second International Conference on Data Stream Mining-2018» (м. Львів, **НМБ Scopus**), XIX міжнародній науково-технічній конференції з математичного моделювання, присвяченої 250-річчю з дня народження Жозефа Фур'є (м. Херсон), всеукраїнській науково-методичній конференції «Проблеми математичного моделювання-2018» (м. Кам'янське).

Розроблений програмний додаток впроваджено у:

1. Міський комунальний заклад культури «Централізована система бібліотек для дітей» м. Дніпро як пошуковий інструмент обробки електронних текстів.
2. ТОВ «Сітал Україна» як засіб автоматичної генерації текстових інструкцій.
3. АТ «ДніпроАзот» як інструмент покращення процесів пошуку в системах електронного документообігу.

Публікації. . Результати дисертаційної роботи опубліковані в 14 наукових працях, в тому числі 8 статей у журналах, рекомендованих МОН України для публікації результатів дисертацій, та закордонних виданнях:

- «Математичне моделювання» – 1 (НМБ Index Copernicus).
- «Системні технології. Регіональний збірник міжвузівських наукових праць» (НМБ Index Copernicus) – 5.
- «Вісник Херсонського національного технічного університету» (НМБ Google Scholar) – 1.
- «Modern engineering and innovative technologies» (Німеччина, НМБ Index Copernicus) – 1.
- у тезах доповідей та трудах міжнародних та всеукраїнських конференцій – 6.

Обсяг і структура дисертації. Дисертаційна робота викладена на 233 сторінках тексту, складається зі вступу, 3 розділів, загальних висновків, списку використаних джерел із 107 найменувань та 19 додатків. Обсяг основного тексту дисертації складає 146 сторінок тексту. Робота ілюстрована 16 таблицями та 46 рисунками.

РОЗДІЛ 1. МАТЕМАТИЧНІ МОДЕЛІ ТЕКСТОВИХ ЗНАНЬ У ПОШУКОВИХ СИСТЕМАХ

Перш ніж приступити до безпосередньої розробки моделі генерації відповідей із неструктурованих баз знань у пошукових системах, необхідно здійснити вибір базової концепції такої моделі, аналізуючи при цьому існуючі підходи до реалізації процесу розуміння знань. До появи ЕОМ питаннями розуміння текстових знань займалась теоретична лінгвістика, завданням якої є пояснення механізмів розуміння природної мови. Створення ЕОМ стало потужним інструментом у руках математиків і лінгвістів, і вже в 50-х роках відбулася перша публічна демонстрація пристрою машинного перекладу – Джоржтаунський експеримент, що заклав основи для появи окремої гілки теоретичної лінгвістики, – комп'ютерної лінгвістики, предметом якої є реалізація механізмів розуміння природної мови безпосередньо для ЕОМ.

Незважаючи на велику кількість різноманітних розробок, успіхи сучасної комп'ютерної лінгвістики у галузі обробки знань можна охарактеризувати як змінні. Переконатися в цьому можна розглянувши, наприклад, роботу систем машинного перекладу, досить далеку від ідеалу. Це пов'язано з кількома факторами, основним з яких є недостатня формалізація правил природної мови, що представляє собою текстові знання, яка необхідна для побудови надійних алгоритмів мовного аналізу – процес розуміння тексту людиною базується на складних механізмах психофізичної взаємодії, реалізація яких для комп’ютера наразі є недосяжним завданням. У свою чергу, це має на увазі, що механічне нарощування алгоритму є тупиковим шляхом. Виходом з цієї проблеми повинні стати різноманітні статистичні та семантичні алгоритми, що дозволяють гнучко оцінити формальність вхідних даних, по суті, створюючи окрему машинну граматику розуміння знань.

Аналізуючи наукові матеріали і розробки в області обробки текстів можна відзначити два «піки» розвитку методів автоматичної обробки текстових знань, перший з яких припав на 50-60-рр., а другий на 80-90 рр. XX століття [2].

Так, 50-ті і 60-ті роки XX століття ознаменувалися рядом розробок, спрямованих на створення комплексних моделей мови (І.А. Мельчук, Н. Хомський), здатних організувати універсальний проміжний рівень між людською мовою і логікою ЕОМ. Однак, не дивлячись на високий інтерес і великі надії наукового співтовариства, пізніше, в силу недостатньої ефективності створених автоматизованих систем і трудомісткості їх розробки, наступив період зміщення інтересу в область більш простих статистичних алгоритмів вирішення прикладних завдань, пов'язаних з обробкою текстів. Ці підходи найчастіше включали в себе досить слабке лінгвістичне ядро, проте саме вони дозволили створити дійсно функціонуючі системи. Якість результатів роботи цих систем була обмежена можливостями моделей, що лежать в основі їх реалізацій, а факт, що далеко не завжди враховувалася така важлива особливість мовних одиниць, як морфологічна, синтаксична і лексико-семантична неоднозначність (через закладання статистичних, а не семантичних моделей у фундамент системи), приводив до неоднозначних результатів.

До 90-х років XX століття підходи до автоматичної обробки текстових знань, засновані на повністю статистичних моделях, стали пануючими (методи, засновані на N-грамах, методи кластерного аналізу, нейронні мережі), і підходи, що використовують знання про світ відійшли на другий план [3]. Лише у кінці 90-хроків XX століття, у зв'язку з широким розповсюдженням нових поколінь обчислювальної техніки, з'явилася можливість створення високопродуктивних систем автоматичної обробки текстів, заснованих на лінгвістичних підходах: формальних граматиках, семантичних моделях і лінгвістичних алгоритмах. У ці роки активно розвиваються комп'ютерні реалізації моделей, створених ще в 60-ті роки XX століття, проте так і не реалізованих внаслідок слабкого розвитку комп'ютерних технологій. Була розроблена система «ЕТАП», що реалізувала модель «Сенс ↔ Текст», різні варіанти синтаксичних парсерів, заснованих на граматиках породження. Крім того, модель «Сенс ↔ Текст» дала поштовх розвитку самостійних моделей оброки текстових знань (Н.М. Леонтьєва, В.А. Тузов) і комп'ютерних реалізацій цих моделей (ФРАП, семантичний

аналізатор Тузова). Можливість реалізації подібних структурних моделей мови дозволяє вирішувати цілий клас прикладних задач комп'ютерної лінгвістики. З'явилися можливості закладати в основу систем автоматичної обробки текстів не тільки статистичні, а і більш лінгвістичні моделі, що містять в собі як банки синтаксичних структур, так і засоби їх інтелектуальної обробки.

Коли справа доходить до безпосередньої реалізації інтелектуальної моделі автоматичної обробки тексту, розробник стикається з декількома науковими питаннями. З одного боку, алгоритмічні проблеми пов'язані з необхідністю вибору апарату опису та реалізації текстових знань на рівні і в формі, доступній машині. З іншого – лінгвістичні проблеми пов'язані безпосередньо з властивостями мови – її нестабільністю і неоднорідністю правил. Крім того, говорячи про мову як про об'єкт моделювання, постає питання про оцінку отриманих результатів – недостатньо просто отримати текст, необхідний глибокий оціночний аналіз результатів роботи системи. Адже текстові знання це не просто набір слів, пов'язаний граматичними правилами – пріоритетним завданням є отримання саме осмисленого тексту, що, в свою чергу, призводить багатьох розробників до необхідності врахування семантичних зв'язків не тільки між окремими словами, а й між реченнями і навіть між документами. З цієї точки зору, інтелектуальна генерація тексту є найбільш складним завданням, оскільки її реалізація має на увазі найбільш повне вираховування усіх важливих смислових відношень в документі.

Менш очевидною складністю є розрив між лінгвістичними описом мови і його прикладної реалізацією. Лінгвістика орієнтується в першу чергу на опис природи мови як свого об'єкта, маніпулюючи, найчастіше, поняттями з психології, філософії, антропології та подібних, недостатньо формалізованих наук. При реалізації систем автоматичної обробки текстів, розробники, використовуючи інструменти комп'ютерної науки, змушені адаптувати їх для роботи з мовою, вирішуючи проблеми що аж ніяк не відносяться до класичної лінгвістики. Саме це і породило таку гібридну область науки як комп'ютерну лінгвістику, об'єктом якої є вже математичне моделювання текстового знання. Крім усього

вищесказаного, необхідно враховувати, що в поняття генерації тексту можуть входити досить різноманітні системи. Так, наприклад, автоматичне складання деяких шаблонних документів (типовий приклад – автоматична документація програмного коду) не може зрівнятися за складністю і семантичними властивостями зі складанням анотацій і квазірефератів до текстів на природній мові. І хоча обидві системи генерують на виході деякий текст, підходи до їх реалізації мають значущі відмінності. Саме тому, першочерговою задачею що має вирішити розробник, стає саме обґрунтований вибір комп'ютерної моделі текстового знання, що буде покладена у основу усієї системи.

Розробка методів вирішення цієї проблеми пройшла декілька історичних етапів і тісно пов'язана із описаними хвилями розвитку автоматичної обробки текстів та вдосконаленням комп'ютерних технологій. Більшість інновацій, що відбувалися у галузі протягом останніх п'ятдесяти років приносили власні підходи і ідеї щодо обробки текстових знань за допомогою ЕОМ, нові задачі та нові шляхи їх вирішення.

Не дивлячись на те, що серйозною проблемою перших розробок в області комп'ютерної лінгвістики був недостатній розвиток комп'ютерних технологій і вчені того часу були змушені боротися з обмеженими можливостями ЕОМ (а амбітні проекти здавалися неможливими до реалізації) – саме тоді були сформульовані фундаментальні теорії про природну мову, що зробили переворот у лінгвістиці в цілому і комп'ютерної лінгвістиці зокрема, і, отримавши розвиток у сучасності, є пануючими напрямами для створення систем автоматичної обробки текстових знань і цільовими підходами для вирішення нашої задачі.

1.1 Вибір фундаментальної моделі природної мови

Перша модель мови, що може бути покладена у алгоритмічну основу нашої моделі є система граматичного опису, сформульована американським лінгвістом А.Н. Хомським, яка отримала назву генеративна граматика. Парадигми, описані Хомським, заснували окрему велику течію – генеративізм.

Основні постулати генеративної теорії були висунуті Хомським у книзі «Синтаксичні структури» [4] в 1957 р. Генеративна граматика відштовхується від припущення про існування явища мовної компетенції – вродженої здатності людини до засвоєння і розуміння людської мови, незалежно від моделі мови. Слідуючи цьому, генеративна граматика ставить перед собою мету змоделювати цю здатність в рамках породження правильних речень, використовуючи певний кінцевий набір правил, алфавіт і початковий символ речення, з якого, за допомогою правил, можна розгортати нескінченну множину правильних схем побудови речення – безпосередні складові. Набори безпосередніх складових Хомський об'єднує в глибинні структури, які, після застосування трансформаційних правил перероджуються в структури поверхневі – з яких можливе отримання «фонетичного уявлення» – після застосування відповідних фонологічних правил. Крім того, в стандартній теорії присутній семантичний компонент, який виражений в структурах лексичних правил, застосування яких до глибинних структур дасть нам семантичне подання – дерево, що містить семантичні ознаки і мітки категорій [4]. Слід зазначити, що спочатку стандартна теорія Хомського переслідувала мету формалізувати тільки граматичну правильність формування речень, упускаючи при цьому опис їх сенсу.

Альтернативна модель текстових знань була сформульована у 60-70-х роках, групою вчених (І.А. Мельчук, Ю.Д. Апресян, А.К. Жовківський) в СРСР і отримала назву «Сенс ↔ Текст». Найбільш повно ця теорія описана в книзі Мельчука «Досвід теорії лінгвістичних моделей «Сенс ↔ Текст» [5]. В основі моделі лежать механізми розуміння мови як перетворювача між текстом і його сенсом (в обох напрямках), що дає можливість організувати процес мовного синтезу. У моделі реалізовано декілька рівнів обробки тексту, семантичне ж

представлення реалізовано за допомогою семантичного рівня і системи правил переходу до глибинно-синтаксичного рівня, що утворюють семантичний компонент.

У своїй роботі семантичний компонент взаємодіє з тлумачно-комбінаторним словником – особливим словником, в якому містяться статті для кожної семантичної одиниці. За допомогою такого словника відбувається вичленення узагальнених слів. Словникова стаття у тлумачно-комбінаторному словнику складається з 10 зон: морфологічні відомості; стилістична мітка; тлумачення (часто з використанням змінних, що індексують пов'язані поняття); модель управління; обмеження до моделі управління; ілюстрації (мовні приклади) до моделі управління та обмеження; лексичні функції; ілюстрації до лексичних функцій; енциклопедична інформація, необхідна для правильного використання заголовної лексеми; ідіоми [5]. Для кожного слова в тлумачно-комбінаторному словнику наводяться слова, пов'язані з ним за змістом. Це його парадигматичні варіанти – заміни конкретного слова в залежності від застосовуваного контексту, і синтагматичні партнери – ідіоматичні вирази різних смислів при даному слові. Ці засоби називаються лексичними корелятами, а зв'язки слів з корелятами описуються лексичними функціями.

Теорія «Сенс ↔ Текст» з самого початку створювалася як спроба вирішити прикладну проблему автоматичного перекладу – за допомогою цієї теорії можна було побудувати динамічну функціональну модель мови – поточні моделі мови вченого не влаштовували. Модель була реалізована в деяких системах машинного перекладу, розроблених в СРСР – система англо-російського автоматичного перекладу ЕТАП, розроблена групою під керівництвом Ю. Д. Апресяна [7]. Крім того, елементи ідеології теорії «Сенс ↔ Текст» були використані в ряді інших систем машинного перекладу, що створювалися у Всесоюзному центрі перекладів під керівництвом Н. М. Леонтьєвої, Ю. С. Мартем’янова, З. М. Шаляпіна та ін. Всі ці системи відносяться до числа експериментальних, тобто їх промислове використання не є можливим [8].

Модель «Сенс ↔ Текст» отримала потужний поштовх, пов'язаний зі збільшенням числа кінцевих користувачів ЕОМ і новими обчислювальними можливостями комп'ютера. Вельми значущою розробкою стала модель м'якого автоматичного розуміння тексту, розроблена в 1970-х роках під керівництвом проф. Н.М. Леонтієвої, що є однією з послідовників Мельчука [1]. Н.М. Леонтьєва відходить від класичного поняття «автоматична обробка тексту», переходячи до більш вузькоспеціалізованого терміну «автоматичне розуміння тексту», визначаючи його «... через результат, який може або прагне побудувати комп'ютер ... результат повинен бути іншим об'єктом, відмінним від вхідного тексту» . При цьому, для опису процесу розуміння повинен виконатися ланцюжок: Автор → Текст_1 → Комп'ютер → Текст_2 → Адресат, в якому особливе місце відводиться оцінці отриманих результатів адресатом.

«Результат повинен бути іншим об'єктом, відмінним від вхідного тексту (Т2 відрізняється від Т1). Так, якщо ми ввели якийсь текст з клавіатури в ЕОМ, а потім роздрукували його на принтері, підключеному до цієї ЕОМ, ми не вважаємо, що текст був зрозумілий. Але якщо ми ввели текст на одній мові (скажімо, англійською), а ЕОМ після роботи системи видала текст іншою мовою (наприклад, російською), то можна вже говорити про «розуміння». Правда, тут важлива ступінь розуміння: якщо машина видала такий нечленороздільний текст, який називають «абракадаброю», ми скажемо, що вона нічого не зрозуміла» [1].

У моделі м'якого розуміння тексту Леонтьєва поєднує лінгвістичний та інформаційний підходи. З боку лінгвістики особливу роль відіграють поняття семантики тексту і його зв'язність, визначення понадфразних одиниць і зв'язків між ними. Інформаційні підходи представлені методами організації бази даних і методами вилучення знань з неї. Обидва підходи організують інформаційно-лінгвістичну модель, сенс якої зводиться до того, що з одного і того ж тексту можна витягати різну інформацію, що обумовлено в першу чергу інтересами користувачів (або комп'ютерними модулями знань, підключеними до системи). У цьому і полягає «м'яка» сторона моделі – зміна семантичних властивостей тексту в залежності від зовнішніх умов.

Леонтьєва зазначає, що поняття автоматичного розуміння тексту безпосередньо пов'язано з інформаційним аналізом – послідовністю операцій, необхідних для вилучення з текстів релевантної інформації. Це передбачає, що для кожного документа необхідно побудувати деяке інформаційне подання – мінімально стислий текст, з якого можна, за допомогою деяких баз знань, розгортати нові тексти без втрати основного сенсу.

Автоматичне розуміння текстів і «м'яка» модель є прямими спадкоємцями теорії Мельчука «Сенс ↔ Текст». Тому коли постало питання про застосування цієї моделі, основною предметною областю був обраний автоматичний переклад. Проте проблеми, які при цьому виділяє Леонтьєва, є фундаментальними для всіх областей автоматичної обробки текстових знань. У першу чергу це проблема формальної неоднозначності ізольованих речень тексту – а саме зміна сенсу речення при виключенні його з контексту; необхідність організації гнучкого підключення різноманітних баз знань предметних областей в процесі автоматичного розуміння; побудова схеми оцінки тексту як цілого утворення, що має на увазі створення якоїсь загальної характеристичної одиниці. Леонтьєва також наполягає на необхідності створення деякої мови-посередника, здатної налагодити взаємодію між природною і машинною мовами.

Саме методи і підходи до створення моделі мови, що були описані вище, ми будемо розглядати в якості гіпотетичних кандидатів для використання у алгоритмічному ядрі моделі, оскільки вони є фундаментальними для задачі формалізації текстових знань [9]. Робити це найкраще виходячи із завдання, яке ми перед собою ставимо – використання генерації тексту для вирішення пошукової задачі.

Говорячи про теорії Хомського, можна відзначити їх низьку придатність для задачі автоматичної генерації тексту. Для розуміння проблематики цього підходу (в рамках нашого завдання) необхідно вказати на рівні відносин у тексті, що визначає Н.М. Леонтьєва [1] – від синтаксичного, як нижчого, до понадфразного – необхідного для побудови узагальненої семантичної характеристики фрагмента тексту (або ж міждокументної взаємодії). Очевидно, що саме понадфразний

зв'язок є ключовим для задачі генерації тексту, а точніше – автоматичне складання з його допомогою формального узагальнення, організації зв'язків узагальнень між собою, та ін. На такому етапі необхідно явно ототожнювати поняття «Сенс» з поняттям «Узагальнення», тобто по суті, діяти подібно когнітивним здібностям людини, де синтаксичний розбір відходить на другий план, поступаючись місцем семантичному.

Тепер повернемося до підходу Хомського. Узагальнення його застосуванню підводить Мозговий М.М .: «... граматики Хомського призначені, насамперед, для опису структури речення. Не менш важливе питання опису смислів окремих слів залишається за межами їх можливостей» [9]. Дійсно, генеративна граматика Хомського ніколи не виходила за межі рівня синтаксису. Згадаймо – первісною метою є породження граматично правильних речень з деякого алфавіту, використовуючи ланцюжки глибинного рівня. І якщо теоретично ми можемо виводити нескінченно велику кількість таких ланцюжків, що, власне і дозволяє описати цією моделлю абсолютно будь яку мову, то на практиці це не представляється можливим – навіть якщо відкинути таку властивість мови як мінливість, кількість ланцюжків буде хоч і не нескінченна, але, безумовно, величезна. І чим більш виражена флексія в мові – тим складніше буде її описати у доступній для ЕОМ формі.

Флексія мови – ще одна проблема для граматик Хомського. Вона пояснюється існуванням спеціальних закінчень, які морфологічно і лексично змінюють структури слів, найчастіше – в залежності від контексту. Чим флективно багатша мова, тим вона більш складна, і тим більш вільний порядок слів у реченні. Це нас призводить до проблеми мовної залежності граматик Хомського – реалізація їх для англійської мови, мови з відносно бідною флексію і жорстким порядком слів у реченні – не підходить для слов’яномовних систем: «... в граматиках Хомського порядок слів вказується безпосередньо, тому проблема лінеаризації взагалі не виникає. Однак тим самим різко знижується виразна міць моделі. Для англійської мови з її строгим порядком слів у реченні

обмеження методу Хомського не є критичними, однак при роботі зі слов'янською мовою їх вже не можна ігнорувати» [9].

І нарешті, поглянемо власне на застосування генеративних граматик: «... завдання аналізу формальної мови (що виникає, наприклад, при компіляції програм на Паскалі) не є тривіальним: воно було повністю вирішено лише в 60-70-і рр. після появи робіт Н. Хомського ... » [9]. Дійсно, основна область використання граматики породження – парсинг мови програмування, де глибинні структури представлені ланцюжками формальної мови, і граматична правильність стоїть над смисловою семантикою. Однак, до таких завдань як автоматичний переклад, рубрикація, генерація текстів – підхід складно застосувати: «Залишається неясним, як в рамках якої-небудь з гілок генеративної теорії перетворити певну семантичну мережу в послідовність дерев залежності або складових, відповідних окремим реченням» [10].

Модель Мельчука – теорія «Сенс ↔ Текст» відокремлює семантику від синтаксису, наполягаючи при цьому на її науковому описі. З різних причин, (від політичних до економічних) модель не отримала широкого розповсюдження у світі, залишившись в «інформаційному вакуумі» пострадянського простору. Тим не менш, багато вчених характеризують її як ту, що випередила свій час. Все частіше звучать твердження, що граматикам складових тільки належить еволюціонувати до деяких парадигм моделі «Сенс ↔ Текст» [10]. Щоб розібратися в проблематиці теорії, нам необхідно поглянути на її основи через призму завдання генерації тексту, що продемонструє нам основні плюси і мінуси моделі в прикладній реалізації. Крім того, необхідно постійно проводити паралелі з моделлю Хомського, оскільки зараз саме вона є домінуючим напрямком прикладної лінгвістики.

Першим, безумовним полюсом, є орієнтованість на синтез тексту. І.А. Большаков зазначає, що «... наскільки нам відомо, синтез тексту довільно заданої семантичної мережі серйозно продумувався саме в рамках моделі Сенс ↔ Текст ...» [10]. Завдання синтезу тексту, на відміну від завдання його аналізу, вимагає опис глибинних семантичних відносин (понадфразних відносин), які, як

зазначалося вище, в граматиках Хомського просто не враховуються. Лексико-семантичні правила розширених генеративних теорій спрямовані на вирішення граматичних неоднозначностей, не можуть вирішувати задачу синтезу. Особливо відчувається недосконаліть граматики складових при спробі синтезувати мову з вільним порядком слів, де граматика неоднозначна, а в якості опори використовується смислові одиниці. Це приводить нас до другого плюса моделі Сенс ↔ Текст – незалежність від порядку розташування слів у реченні. Дійсно, знаючи семантику речення, нам не обов'язково опускатися до синтаксичного рівня, що дає нам можливість обійти вільний порядок слів: «основна перевага граматик залежності вбачається в тому, що саме зв'язки між словами зберігаються на семантичному рівні, а для граматик складових їх зазвичай доводиться виявляти на семантичному рівні окремим механізмом» [10]. Такий підхід дає можливість реалізувати формальну модель мови для всієї індоєвропейської групи, а не тільки для англійської, де порядок проходження відіграє ключову роль. Ці факти говорять про перевагу граматик залежності перед граматиками складових у задачах синтезу тексту.

Основним мінусом моделі «Сенс ↔ Текст» є тлумачно-комбінаторний словник, пункти якого описані вище. Механізм сем має на увазі, що для кожної семи буде описаний повний набір її властивостей – від лексичних і морфологічних, навіть до таких високих семантичних рівнів як пов'язані ідіоми. Необхідність складання повного словника можлива тільки вручну – ми не зможемо вибрати сценарій поведінки системи побудованої на основі моделі «Сенс ↔ Текст» у разі аналізу невідомого слова, оскільки структура статті є занадто складною: при можливості автоматичного вибору морфологічних ознак з деякою, допустимою помилкою, інші пункти, як, наприклад ідіоми, заповнити автоматично без спеціальних семантичних знань неможливо. Довірити ж складання статей оператору-людині є досить складним масовим завданням. Як зазначає І.А. Большаков: «Технологія складання тлумачно-комбінаторних словників залишилася не розробленою. Їх складання виявляється досі під силу тільки тим, хто освоював модель багато років, а по суті створював і

удосконалював її. Ймовірно, саме відсутність ясної і масової технології розробки словників стала однією з основних історичних причин відставання моделі від західних «конкурентів»... » [10]. Однак, навіть незважаючи на вищевказані мінуси, модель «Сенс ↔ Текст» є більш зручною для розробки системи синтезу тексту, ніж генеративна граматика.

Подальший розвиток модель «Сенс ↔ Текст» отримала в моделі «м'якого розуміння» Леонтьевої, основні ідеї яких були описані вище. Теоретична основа м'якого розуміння не дозволила вирішити основну проблему – необхідність складання тлумачно-комбінаторного словника, однак, великим плюсом стало деяке скасування і конкретизація оригінальної теорії Мельчука. Модель, з одного боку, наполягає на ситуативному сприйнятті сенсу тексту, з іншого – обмежує роботу моделі рамками одного корпусу і однієї предметної області. Це дозволяє значно зменшити масштаби завдання, роблячи можливим його практичне застосування. Ці факти вказують на можливість деякого спрощення моделі з метою зниження кількості помилок у роботі цільової системи.

Підсумувавши, можна стверджувати що модель «м'якого» розуміння Леонтьєвої в комбінації зі статистичними методами аналізу може дати оптимальні результати в задачі генерацій текстів. До плюсів такого підходу можна віднести початкову орієнтованість на синтез, високу роль семантики в моделі, незалежність від синтаксичної структури, вирішення проблеми різноманітності флексії слов’янської мови та орієнтованість на спрощену базу знань (у порівнянні з моделями Мельчука). Як наслідок, пріоритетними науковими проблемами стають питання, пов'язані з організацією навчання системи, подолання обмежень бази знань і гнучке налаштування роботи системи при підключенні нових предметних областей.

1.2 Загальна концепція текстового автомата. Структура моделі генерації відповідей

Після вибору базової алгоритмічної моделі мови постає питання її програмної реалізації. Процес формування відповіді за допомогою генерації текстів є складним інтелектуальним завданням автоматичної обробки текстів, заснованим на виділенні з текстів за допомогою спеціальних інформаційних дескрипторів найістотнішої інформації з подальшим породженням нових текстів, які в деякій мірі корелюють з сенсом вмісту вихідного запиту. Для цього необхідно створити складну багаторівневу систему аналізу даних, що містить у собі велику кількість допоміжних утілит та компонентів штучного інтелекту. Крім реалізації формальної моделі машинного розуміння мови, необхідно наповнити систему знаннями, організувати автоматизований процес навчання і адаптації моделі, оцінку отриманих результатів. Розробка монолітного програмного продукту у цьому випадку може викликати проблеми високої зв’язності компонентів між собою та порушення принципу інверсії залежності [11]. До того ж, необхідно чітко сформулювати критерії до архітектури програмного додатку, що в умовах недостатнього розвитку наукової сфери генерації текстів є досить складним завданням. Дані проблеми приводять нас до концепції лінгвістичного автомата як найбільш оптимальної для нашої задачі. Лінгвістичний автомат «розглядається як збалансований комплекс апаратних, програмних, лінгвістичних, а іноді і лінгводидактичних засобів, що взаємодіють з потужною базою лінгвістичних даних і знань» [12]. У загальному випадку, функціонал автомату прагне до реалізації наступних завдань [12]: мінімізація інформаційних втрат при здійсненні автоматичної обробки текстів; перешкоджання виходу системи з ладу – як з технічної, так і з логічної точки зору, що передбачає створення спеціальних сценаріїв поведінки системи у випадку обробки некоректної, або незнайомої текстової інформації; побудова адаптивної системи, здатної еволюціонувати разом з об'єктом своєї роботи – знаннями; можливість підключення до різних каналів зв'язку, через які буде здійснюватися введення і виведення даних.

Самі ці завдання і стали початковими критеріями для створення базової архітектури моделі генерації відповідей на основі неструктурованої бази знань. На прикладному рівні програмної реалізації автомат являє собою набір окремих програмних інструментів автоматичної обробки текстів, сукупність функціоналу яких може бути використана задля вирішення задачі інтелектуального аналізу текстових даних. Таким чином, автомат є модульною сервісною структурою, і до його складу повинно входити інтегроване середовище для користувача, здатне організувати як налаштування окремих утиліт користувачем, так і автоматичне виконання цільового завдання (у нашому випадку – генерацію тексту) з мінімальною участю користувача.

Програмні модулі, що входять до складу автомату формують рівні процесу обробки тексту за своїм безпосереднім функціоналом. В якості найбільш загальних рівнів, Луканін виділяє лексико-морфологічний (словниковий), синтаксико-семантичний і ситуативно-прагматичний рівні [12]. І якщо повернутися до обраної нами моделі м'якого розуміння Леонтьєвої, стає очевидна паралель між ієрархічним процесом роботи автомата і рівнями розуміння мови, що дозволяє не відходити від основних положень як в концепції текстового автомату, так і в теорії м'якого розуміння при програмній реалізації моделі. Загальна розроблена структура текстового автомата, що реалізує у собі описані вище теоретичні основи обробки текстів зображена на рис. 1.1.

Створена модель має два входи, на які можуть надходити Q – текстовий запит користувача або DCN – текст-кандидат до включення у базу знань системи. Якщо на вхід системи поступив текст DCN, то він проходить через деяку оцінку його адекватності за допомогою семантичного фільтру F, що необхідно для запобігання наповнення бази знань невалідними даними. У випадку, коли фільтр F класифікував документ як адекватний, наступним етапом стає побудова моделі текстового знання за допомогою конструктора моделі C та збереження моделі у базу знань системи, що являє собою множину моделей текстів $\{DC_1 \cdots DC_i\}$.

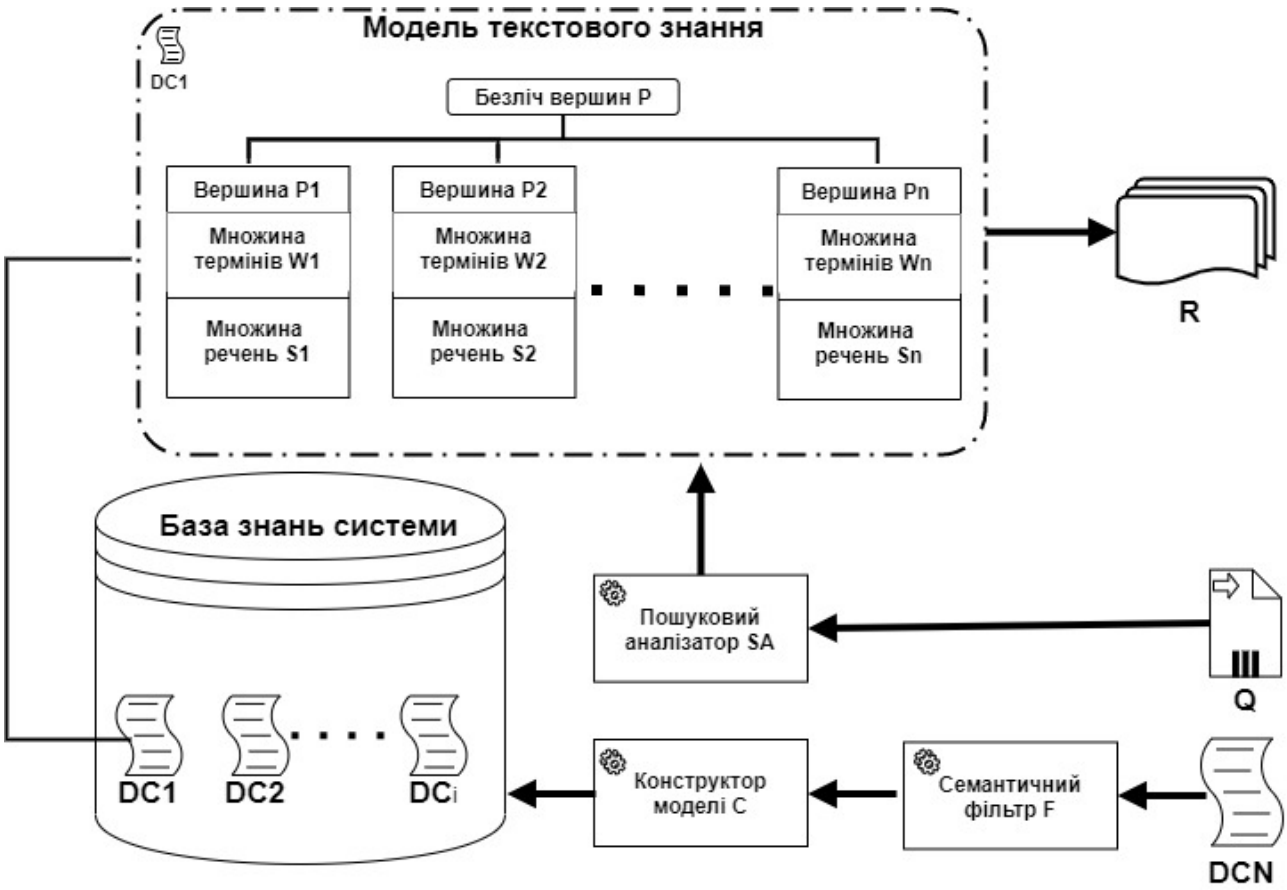

Рис. 1.1. Загальна структура текстового автомату

Приклад моделі текстового знання для одного із документів бази знань DC_1 можна побачити на рис. 1.1. Теоретичною основою для побудови моделі текстового знання була обрана семантична мережа, що складається із множини семантичних блоків P, які представляють собою пов'язану пару множини термінів W і множин речень із тексту S: $P = \{P_1 = \{W_1, S_1\}, P_2 = \{W_2, S_2\} \cdots P_N = \{W_N, S_N\}\}$, що автоматично формуються на етапі роботи конструктора моделі C.

Якщо на вхід системи надходить запит користувача Q, то над кожною моделлю із бази знань системи виконується операція пошуку і формування відповіді R, завдяки роботі пошукового аналізатора SA. Згенерована таким чином відповідь R повертається користувачу з єдиного виходу моделі, і являє собою результат виконання пошукової операції.

Для більш глибоко розуміння етапів роботи створеної моделі, розглянемо більш детально її алгоритм, зображений на рис. 1.2.

Рис. 1.2. Загальний алгоритм роботи системи запит-відповідь на основі генерації текстів

Першим кроком, який виконується у нашій моделі, є отримання даних користувача. Як було вказано вище – дані, які надходять до неї на вхід, можуть

існувати у двох варіантах – неструктурований текст або текстовий запит користувача.

Під цифровим неструктурованим текстом розуміється текст формалізованого жанру на природній людській мові, що не містить в собі будь-яку попередню семантичну розмітку або супутні семантичні словники, доступні для обробки за допомогою комп'ютера. У рамках нашої роботи, такий текст і семантична модель, побудована на його основі, буде служити елементом для наповнення повнотекстової бази знань, на основі якої буде проводитися генерація відповіді. Оскільки, відповідно до мети роботи, складання семантичної моделі є повністю автоматизованим етапом, додавання нового тексту у базу знань моделі не вимагає залучення значних обсягів лінгвістичних правил і описів у вхідному тексті, під якими в рамках нашої роботи, розуміються такі лінгвістичні знання, які вимагають перманентної підтримки і містять в собі семантичні властивості термінів або фрагментів бази знань. Це дозволить нам використовувати в якості вхідних даних будь-який набір текстової інформації формалізованого напряму. Вибір науково-технічного формалізованого цільового жанру предметної області пояснюється шкалою жанрової формальності – науковий стиль знаходиться ближче до ділового, однак, не має фіксованої семантичної структури елементів у своєму складі, що робить його найбільш цікавим об'єктом для дослідження.

Запитом користувача до системи є текстове повідомлення, що містить у собі послідовність ключових слів для пошуку близької за семантичними ознаками відповідної інформації у базі знань системи і формуванні безпосередньої відповіді, завдяки чому і реалізується основна концепція моделі запит-відповідь.

У випадку, коли на вхід системи потрапляє неструктурований текст, над ним здійснюється синтаксичний аналіз – етап, що найчастіше є загальним для більшості систем автоматичної обробки текстів. На цьому етапі відбувається виділення речень, слів, визначення частин мови і частотний аналіз тексту. Над синтаксично розміченим текстом виконується операція створення програмної моделі тексту, в основі якої лежить модель семантичної мережі. Загальне визначення семантичної мережі можна сформулювати наступним чином:

«семантична мережа – це множина понять (слів і словосполучень), пов'язаних між собою. У семантичну мережу включаються слова тексту, що зустрічаються найчастіше, і які несуть основне сенсове навантаження. Для кожного поняття формується набір асоціативних (сенсових) зв'язків, тобто список інших понять, у поєднанні з якими воно зустрічалося в реченнях тексту. При цьому вважається, що чим частіше зустрічаються разом два поняття в реченнях тексту, тим вища ймовірність того, що вони пов'язані за змістом» [13]. У нашій роботи ми дещо змінили концепцію семантичної мережі – у вершинах мережі стоятимуть множини термінів, що формують інформаційні семантичні мітки тексту. Початково, сенс змісту слів в кожній множині нам невідомий. Після утворення кожної інформаційної мітки, ми припускаємо, що поняття сенсу у неї тотожне сукупному семантичному значенню термінів, що входять до її складу. До кожної семантичної мітки приєднується множина відповідних до неї речень, що відповідають загальному сенсу семантичної мітки. Така схема розуміння сенсу тексту відрізняється від механізмів розуміння людини, однак, для інформаційних комп'ютерних моделей це цілком допустимо.

На основі побудованої програмної моделі тексту проводиться визначення доцільності включення вхідного тексту у базу знань. Важливість цього кроку пояснюється необхідністю недопущення виникнення семантичного сміття у базі знань системи через помилки користувача або неоднозначності щодо визначення жанру тексту – документи, що не пройшли валідацію на цьому етапі не потраплять до бази знань системи. У першому варіанті моделі для реалізації цього етапу пропонувалося кластерізувати корпус за синтаксичною близькістю документів між собою. Проте, у ході виконання роботи була висунута і підтверджена більш складна гіпотеза про використання створеної на минулому етапі семантичної моделі, яка може надати нам такі данні, що дозволять проводити класифікацію іншого рівня, що визначає клас документу не тільки за ступенем схожості текстів, а в першу чергу за структурою його семантичної моделі. І якщо у підході із кластерізацією цей крок є лише проміжним і необхідним для фільтрації даних, то пропонований підхід, який буде розглянуто у подальших главах може буде

використаний і як програмний засіб виконання критерію безпечності функціонування текстового автомату.

Якщо ж на вхід моделі надходить запит, над ним виконується операція виділення термінів, на основі яких відбувається пошук по множинам всіх ключових слів у кожній семантичній мітці, у результаті чого будується перелік потенційно задовольняючих запиту текстових інформаційних відповідей – моделі в базі знань шикуються в інформаційний градієнт відповідно до ключових слів у запиті, що є аналогом семантичної сили класів від максимально інформативного документа до документа із мінімальною інформативністю відносно запита користувача. Описаний підхід базується на висунутій гіпотезі, що семантична мітка із найбільшою сумарною вагою зв’язків із множинами речень має найбільшу концентрацію термінів, що відносяться до тематики текстового знання і має найбільшу семантичну силу для подальшого аналізу при генерації відповіді, а терміни, що не входять в предметну область, опиняться в семантичних мітках із меншою сумарною вагою зв’язків із множинами речень. Провівши аналогію з людським мисленням, можна охарактеризувати роботу системи як спробу опису деякого об'єкта його основними характеристиками, не називаючи при цьому сам об'єкт. При цьому основні характеристики представлені множиною термінів у семантичній мітці, що має найбільшу сумарну вагу зв’язків із множинами речень. Окрім того, зміна множин відповідей і інформаційних градієнтів залежна від запиту користувача, що являє собою саме ситуативну обробку тексту із концепції м’якого розуміння. Додавши до цього механізм визначення морфологічних характеристик, ми можемо будувати гіпотезу про семантичну близькість термінів, подолавши таким чином обмеження тлумачно-комбінаторного словника. Більш детально дана модель і гіпотеза описана у другій главі дисертаційної роботи.

Завдання генерації тексту, що полягає у створенні текстової відповіді на запит що надійшов, реалізовано за допомогою семантичного квазіреферування, подібно генерації автореферату в роботі [14], однак, з використанням додаткових засобів Data і Text Mining. У загальному випадку, модель буде спиратися на наближення понять в результаті формування ключових семантичних міток. За

допомогою попередніх операцій створення семантичних міток і побудови програмної моделі документу, вибираються найбільш інформативні частини, описані в отриманих множинах знань, з подальшим включенням їх у результуючу відповідь системи в порядку слідування в інформаційному градієнті. Таким чином, відповіддю на запит користувача є множина фрагментів текстів із бази знань моделі, що є релевантними до тематики термінів вхідного запиту.

Розглянемо відповідність роботи запропонованої моделі до визначених критеріїв текстового автомату.

Критерій мінімізації інформаційних втрат забезпечується використанням у системі бази знань, яка реалізується інструментами баз даних і формуванням семантичної моделі тексту з її подальшим збереженням у реляційному форматі. Таким чином, усі данні що надійшли до моделі і були визначені як задовільні зберігаються всередині моделі і можуть бути отримані за потребою користувача.

Критерій перешкоджання виходу системи з ладу реалізується за допомогою двох основних інструментів. По-перше мова йде про використання інформаційного фільтру, що не дозволить наповнювати базу знань моделі невалідними даними, які несуть у собі інформаційне сміття і можуть призвести, при накопичуванні у великій кількості, до отримання помилкових результатів. По-друге, у фінальну комп'ютерну систему, що реалізує нашу модель, була включена автоматична перевірка результатів її роботи, заснована на виставленні оцінок отриманому тексту по критерію тематичної зв'язності інформаційного градієнта, що визначається на основі відповідності теми запиту і отриманої множини документів-кандидатів.

Критерій адаптивності системи реалізований за допомогою вилучення тлумачно-комбінаторного словника з загальної схеми текстового автомату і можливості роботи моделі без необхідності у глибокій зміні алгоритмічних правил її функціонування або залучення лінгвістичних знань. Це дозволило організувати динамічне наповнювання системи текстами різної тематичної спрямованості через розширювання вбудованої бази знань у процесі роботи.

Критерій можливості підключення до різних каналів зв'язку реалізований завдяки формату передачі даних – utf-8 plain text, що може бути адаптованим до будь яких цифрових каналів обміну інформації. Окрім того, головні компоненти текстового автомату, як показано далі, були реалізовані у вигляді сервісних компонентів з відкритим API, що дозволяє гнучко змінювати оболонку системи по власному бажанню користувача.

Підбивши межу під вищесказаним можна стверджувати, що виходячи з розробленого алгоритму роботи системи основні вимоги до програмного текстового автомата виконані і задовільнять описаним критеріям. Окрім того, повернувшись до обраної моделі природної мови, можна провести паралель між її основними концепціями – рівневою структурою автоматичної обробки текстів і ситуативним розумінням та розробленою моделлю генерації відповідей. Так, система виділяє декілька етапів обробки вхідних текстових даних – синтаксичний, на якому проводиться базовий синтаксичний аналіз неструктурованого тексту (або аналіз вхідного запиту користувача), семантичний – на якому відбувається формування семантичної моделі документа та створення семантичних міток і рівень міждокументної взаємодії, на якому відбувається генерація текстової відповіді на поставлене питання, що співвідноситься з рівнями м’якого розуміння тексту. Можливість динамічно змінювати семантичну цінність одного і того самого тексту під впливом зовнішнього запиту є реалізацією концепції ситуативного розуміння, завдяки чому можна стверджувати про успішне впровадження моделі м’якого розуміння.

1.3. Аналіз існуючих розробок та моделей

Перш ніж створювати програмний текстовий автомат генерації відповідей, був проведений глибокий аналіз існуючих розробок і основних підходів, що можуть бути покладені в основу нашої моделі і дозволять уникнути дублювання вже реалізованого функціоналу. Першим етапом був пошук доступних для використання програмних засобів на основі генерації текстів, що можуть бути застосовані для вирішення нашої задачі. Основними критеріями, за якими здійснювався пошук були можливість працювати з флективно багатими мовами, а саме українською або будь-якою іншою слов'янською мовою, опис та зрозумілість алгоритмів функціонування системи та, власне, існування доступної програмної реалізації. Аналіз розробок за цими критеріями показав, що серед сучасних моделей автоматичної обробки текстів напрямок генерації текстів є не дуже розвинутим, проте була знайдена система, що підпадає під описані критерії, яка є передовим прикладом засобу генерації текстів і представляє собою цілий клас алгоритмів які базуються на механізмах штучного інтелекту, а саме – реалізації автоматичної обробки текстів за допомогою нейронних мереж. Розробники таких систем використовують найчастіше три типу мереж [15]: проста рекурентна мережа для обробки послідовностей, таких як речення; рекурсивна авто-асоціативна пам'ять для обробки лінгвістичних структур, представлених у вигляді дерев та самоорганізована карта Кохонена для категоризації таких уявлень. Для оцінки якості методів нейронних мереж в задачі автоматичної генерації текстів звернемося до роботи [15], де штучна рекурентна мережа використовується для створення описів про товари деякого інтернет магазину. Процес її функціонування полягає у пошуку коментарів реальних покупців під кожним існуючим товаром і створення на їх основі опису товару, за допомогою рекурентної мережі (у автоматичному режимі). Приклад результатів роботи системи наведені в таблиці 1.1:

Таблиця 1.1

Результати роботи рекурентної нейронної мережі

Товар №1 (сматрфон) Якість звуку, зручний інтерфейс, дуже довго тримає заряд. Чуйний екран, гучний дзвінок, великий шрифт, робочий день. Приємно лежить в руках, 2 сім карти виручають. Якість збірки, батарея, зручне меню, стійкий до впливу води. Явно лідирують, соковитий дисплей, якість зв'язку
Товар №2 (ноутбук) Акумулятор, швидкість красива. Дизайн, звук, функціонал, маса різних днів вистачає. Красів, кілька назад, процесор чуйний сенсор. Гарний екран, передача кольору. Дизайн, батарея, не гальмують, практичний.

Як можна побачити, результати виходять досить неоднозначні. З одного боку – безумовними плюсами такого підходу є повна автоматизація процесу генерації тексту та низькі витрати на її налаштування та впровадження. З іншого – очевидні деякі проблеми появи «смислового сміття», на подобі «швидкість красива, чуйний екран, робочий день». Очевидно, що незважаючи на удавану наявність інтелектуальної обробки, система не розуміє сенсу того що генерує, відштовхуючись лише від заздалегідь заданих шаблонів. Окрім того, при розробленому підході знижається адаптивність системи – існування семантичної розмітки у вигляді попередньо заданої категорії товару і обмеження предметної галузі не дозволяють використовувати підхід як універсальний інструмент обробки тексту. Подолати цю проблему можна двома шляхами – це ускладнення мережі, що незмінно приведе нас до втрат в адаптивності, або ж впровадження додаткових технологій подання текстових даних – створення «суміщених» моделей, до яких і належить описана модель текстового автомата.

Наступним етапом аналізу існуючих розробок став пошук і порівняння додатків, що можуть задовільнити критеріям текстових автоматів отримання

знань із текстів. Основними критеріями пошуку моделей для аналізу були можливість вирішення задачі отримання знань із вихідного тексту та робота із слов’янськими мовами. Виявилось, що найбільш передовими розробками здебільш є комерційні продукти, глибокі деталі реалізації яких не розповсюджуються у відкритому доступі. Розглянемо найбільш відомі і цікаві з них.

AlchemyAPI – це програмний продукт, що належить компанії IBM [16]. При роботі використовує машинне навчання (зокрема, глибоке навчання) для виконання обробки природної мови та семантичного аналізу тексту, включаючи аналіз настроїв і комп'ютерний зір (виявлення і розпізнавання осіб). Підходом до обробки бази знань, що покладена у систему є семантичне анотування.

Під час семантичного анотування текст на природній мові супроводжується метаданими, які повинні зробити семантику елементів що містяться в тексті, доступною до розуміння комп’ютером. У цьому процесі, який зазвичай є напівавтоматичним, знання витягуються в тому сенсі, що встановлюється зв'язок між лексичними елементами і, наприклад, поняттями з онтологій. Таким чином отримуємо знання, які розкривають значення сутності в оброблюваному контексті, завдяки чому визначається значення тексту в інформації, що сприймається машиною і з’являється можливість робити логічні висновки у автоматичному режимі.

Програмний продукт NetOwl Extractor виконує витяг об'єктів із неструктурованих текстів з використанням обробки природної мови, машинного навчання і комп'ютерної лінгвістики [17]. Extractor також виконує отримання семантичних відносин і витяг подій, а також геотегінг тексту. Він використовується для різних джерел даних, включаючи як традиційні джерела (наприклад, новини, звіти, веб-сторінки, електронна пошта), так і соціальні мережі (наприклад, Twitter, Facebook, чати, блоги). Він працює на різних платформах аналізу великих даних, включаючи Apache Hadoop і технологію високопродуктивного комп'ютерного кластера LexisNexis. Він був інтегрований з рядом сторонніх аналітичних інструментів, таких як Esri ArcGIS і Google Earth /

Maps. Підходом до обробки бази знань, що покладена у систему є традиційне вилучення інформації.

Традиційне вилучення інформації – це технологія обробки природної мови, яка витягує інформацію з текстів на природній мові та структурує її відповідним чином. Види інформації, які слід виділити, повинні бути вказані в моделі перед початком процесу обробки, ось чому весь процес традиційного добування інформації залежить від розглянутої предметної області. Традиційне вилучення інформації розділяється на наступні п'ять підзадач: розпізнавання іменованих сутностей; вирішення кореференції; побудова елементів шаблону; виявлення зв'язків між сутностями; побудова повного опису події.

Завдання розпізнавання іменованих сутностей полягає в розпізнаванні і категоризації всіх іменованих сутностей, що містяться в тексті (призначення іменованих сутностей зумовлене категорією). Це реалізується шляхом застосування методів, заснованих або на граматиці, або на статистичних моделях. Вирішення кореференції встановлює еквівалентні сутності, які були розпізнані в тексті. Під час побудови елементів шаблону система встановлює описові властивості сутностей. Ці властивості відповідають звичайним якостям, як «червоний» або «великий». Виявлення зв'язків між окремими сутностями встановлює відносини, які існують між елементами шаблону. Ці відносини можуть бути декількох видів, такі як «працює-для» або «розташовано-в», з обмеженням, що як область, так і діапазон відповідають сутності. Повні описи подій, які проводяться в тексті, розпізнаються і структуруються згідно сутностей, розпізнаних на попередніх етапах.

Цікавою розробкою є програмний продукт ontoX [18], який надає функціонал вилучення знань з тексту за допомогою методу виділення інформації на основі онтологій, який є підобластю вилучення інформації, в якій використовується щонайменше одна онтологія для управління процесом вилучення інформації з тексту на природній мові. Дана модель використовує методи традиційного добування інформації для розпізнавання понять, сутностей і

відносин в тексті, які будуть структуровані в онтологію після процесу роботи системи.

Головною особливістю системи ontoX є інтегроване напівавтоматичне середовище управління онтологіями. Автори продукту стверджують «... важливим питанням є підтримка онтології в актуальному стані, тому що домен, який вона представляє, неминуче зміниться. Ми мотивуємо використання онтологій в інформаційних системах добування інформації, особливо для автоматизації процесу генерації правил вилучення, який в даний час є основною перешкодою для масштабованих інформаційних систем» [18].

Як можна помітити, аналіз найбільш передових розробок в області свідчить про декілька фактів: по-перше, очевидна явна нестача розробок, які коректно взаємодіють з українською (або іншою флективно багатою) мовою, що пояснюється багаторічною ізоляцією вітчизняної комп'ютерної лінгвістики у світі. По друге, існуючі системи вилучення знань, орім того що є небезкоштовними комерційними продуктами, орієнтовані в першу чергу на роботу із онотологіями, які вимагають залучення лінгвістичних експертів і виконання великого обсягу роботи по впровадженню описаних структур, що найчастіше є досить складним завданням для рядового розробника моделей та систем генерації знань. Крім того, впровадження описаних підходів різко знижує адаптивність моделі, вимагаючи постійної підтримки розмітки і словників при додаванні нових знань в систему. З цього можна зробити висновок що завдання, поставлені в цій роботі, є досить актуальним напрямком в науковій сфері математичного моделювання.

Заключним кроком аналізу існуючих розробок став огляд робіт, що стосуються теми побудови семантичних мереж. Це зроблено для пошуку алгоритмів, які б дозволили створити програмну модель документу, і з метою визначення актуальності створеного підходу до побудови семантичної моделі документу.

В роботі [19] запропонований підхід до створення семантичної мережі, який дозволяє виявити значення невідомого слова U на основі аналізу околів вершини

графа, що відповідає цьому слову. Спочатку з вихідного графа $G=(V;E)$ витягується его-мережа $G_U=(V_U;E_U)$ – околиця вершини $U\in V$ в графі G. Потім з графа G_U виключається вершина U і проводиться кластеризація даного графа за допомогою будь-якого методу жорсткої кластеризації графа. Кожен отриманий кластер в результаті кластеризації відповідає окремому лексичному значенню заданого слова $U\in V$. Даний підхід широко використовується в теоретико-графових методах виведення значень слова. Двома найбільш популярними алгоритмами є марковський алгоритм кластеризації і алгоритм зіпсованого телефону [19]. Обидва алгоритми приймають на вхід неорієнтовний зважений граф і повертають множину непересічних кластерів. Метод виведення значень слова за допомогою жорсткої кластеризації графа дозволяє виявити значення багатозначних слів, але не дозволяє об'єднати ці значення в сінсети. Незважаючи на цей недолік, даний метод широко використовується для побудови інвентарів смислів, які використовуються для вирішення важливої оберненої задачі – дозволу лексичної багатозначності [19].

В роботі [20] представлений MaxMax – метод нечіткої кластеризації, що дозволяє кожній вершині графа бути елементом одного або декількох кластерів. Слова, які опинилися в складі декількох кластерів, неявним чином отримують мітки різних значень вихідного слова. Незважаючи на те, що метод MaxMax є спеціалізованим методом кластеризації для графів спільної зустрічальності слів, автори не накладають обмежень на його використання для обробки інших типів графів. Метод MaxMax включає два основних етапи: побудова допоміжного орієнтованого графа $G'=(V';E')$ на основі вихідного неорієнтованого зваженого графа $G=(V;E)$ та витяг пересічних кластерів з допоміжного графа G'.

Автори алгоритму MaxMax використовують поняття максимально близької вершини: вершина $v\in V$ є максимально близькою вершиною для вершини $u\in V$, якщо серед всіх вершин, суміжних з u, вага $weight(u;v)$ максимальна. Це дозволяє використовувати два припущення:

– якщо u є максимально близькою вершиною для v, то з цього не випливає, що v є максимально близькою вершиною для u ;

– пара вершин $\{u,v\}$ знаходиться в одному кластері тоді і тільки тоді, коли u є максимально близькою вершиною для v, і v є максимально близькою вершиною для u .

Таким чином, на етапі перетворення графа проводиться побудова такого орієнтованого невиваженого графа, що множина його вершин збігається з множиною вершин вихідного графа, а множина ребер формується шляхом підбору максимально близької вершини для кожної вершини. В отриманому орієнтованому графі виконується витяг кластерів. Для цього спочатку всі вершини позначаються як кореневі. Потім, для кожної вершини $u \in V$, всі вершини, досяжні з неї, позначаються як кореневі. В результаті такої операції виходять кластери, які можна отримати з орієнтованого графа за допомогою запуску алгоритму пошуку в глибину з кожної кореневої вершини.

На відміну від методу виведення значень слова за допомогою жорсткої кластеризації графа, метод MaxMax природним чином виробляє і висновок значень слів, і об'єднання цих слів в семантичні одиниці. Присутність слова в складі кількох різних кластерів інтерпретується як наявність декількох значень даного слова, що відповідають кількості кластерів що його містять. Недоліком цього методу є залежність від підходу до зважування графа, оскільки даний метод спроектований з урахуванням нерівномірного розподілу ваг ребер.

В роботі [21] запропонований метод перколяції клік – метод нечіткої кластеризації невиваженого орієнтованого графа. Цей метод широко використовується в області аналізу соціальних мереж для пошуку пересічних співтовариств (кластерів).

Оскільки множина синонімів, що виражають одне й те саме поняття, утворює пов'язане скупчення вершин в графі синонімів, то даний метод є підходящим для побудови сінсетов в графі. Метод перколяції клік формує кластери шляхом виявлення в графі *k*-клік, відповідним повнозв'язним підграфам,

що містить $k \in R$ вершин. Дві *k*-кліки вважаються суміжними за умови, що у них є *(k – 1)* загальних вершин. Кластер визначається як максимальне об'єднання *k*-клік, які досяжні один від одного через послідовності з суміжних *k*-клік.

Недоліком методу перколяції клік є важкодосяжне на практиці припущення про те, що пов'язані скупчення вершин утворюють повні кліки в графі синонімів. У методі не передбачено засобів для відновлення пропущених ребер у вихідному графі.

В роботі [22] запропонований простий і широко використовуваний підхід вилучення з неструктурованих текстів зв'язків між словами $R \subset V \times V$ на основі лексико-синтаксичних шаблонів, також відомий як шаблони Херста. Даний підхід передбачає вилучення з колекції документів упорядкованих пар слів $(w;h) \in R$ відповідних множин заздалегідь складених шаблонів такого вигляду: «*w* є видом *h*»; «такі *w*, як *h*».

Спочатку, даний метод застосовувався для обробки текстів англійською мовою і заслужив високу популярність. Нажаль, цей метод повертає ненадійні результати, що містять помилки [22]. Тому використання такого підходу в загальному вигляді вимагає виключення рідкісних пар слів. Це робиться за допомогою простого порогового фільтра, який видаляє з набору даних таких пар слів, які зустрілися меншу кількість разів ніж задане значення. Граничне значення залежить від конкретної колекції документів і підбирається індивідуально. Обробка флективних мов за допомогою подібних лексико-синтаксичних шаблонів ускладняється тим, що форми слів змінюються при узгодженні речень.

В роботі [23] запропонований метод побудови семантичних зв'язків шляхом розбору слабоструктурованих матеріалів Вікісловника. Вікісловник – це великий багатомовний семантичний ресурс, доступний в тому числі українською та російською мовою. Побудовою Вікісловника займається велика кількість учасників-редакторів, які широко застосовують матеріали інших словників та інших джерел для оперативного реагування на зміни мови. Вікісловник поповнюється на основі матеріалів існуючих електронних словників, при цьому учасниками ресурсу проводиться очищення і дооформлення даних. Вікісловник і

його електронна версія широко використовується для вирішення завдань, що вимагають відомостей про семантичні відношення.

Важливою проблемою Вікісловника як семантичної мережі є відсутність об'єднання слів в сінсети, тому семантичні зв'язки в цьому ресурсі сформовані тільки між окремими словами. Іншою важливою проблемою всіх ресурсів, створюваних волонтерами в мережі Інтернет з використанням «вікі-підходу» є достовірність відомостей і необхідність забезпечення якості правок, внесених учасниками. Експеримент з побудови семантичної мережі Yet Another RussNet шляхом інтеграції матеріалів тезаурусів із застосуванням спільного редагування виявив дві основні проблеми даного підходу [23]: при недостатньому патрулюванні правок учасники ігнорують результати один одного і створюють поняття-дублікати; без проведення ретельного інструктажу учасники не завжди розрізняють тонкощі побудови зв'язків між словами.

Аналіз описаних підходів вказує на схожу проблему побудови семантичної мережі текстів і програмної реалізації м’якого розуміння , а саме обмеження, що накладаються необхідністю використання словника в якості основної еталонної бази знань. Крім того, очевидний явний брак розробок і застосувань технології семантичної мережі для вилучення знань задля подальшої генерації текстів, оскільки для такої задачі недостатньо сформулювати деяку онтологію знань – необхідно ще й задати точні правила виводів текстових повідомлень із мережі. Щодо питань, які стосуються завдання побудови систем генерації відповідей, можна з певністю сказати що міжнародний досвід використання семантичних мереж вказує на перспективність їх застосування у якості програмного ядра таких додатків, проте називати цю задачу повністю вирішеною заважає відсутність достатньої кількості вітчизняних розробок, працюючих з флективними слов’янськими мовами, що як було описано раніше, має велике значення для роботи моделі.

Висновки до розділу 1

Розглянувши основні підходи до створення моделей в завданні генерації відповідей можна сказати, що синтаксичні методи обробки тексту в комбінації з деякими інтелектуальним компонентом є найбільш простим і адаптивним методом автоматичної обробки тексту. Однак у випадку із завданням генерації тексту подібні методи наштовхуються на ряд обмежень як алгоритмічної реалізації, так і власне лінгвістичних властивостей мови. Виходом з цієї ситуації може стати поєднання можливостей алгоритмів, здатних до статичного аналізу, автоматичного навчання і розвитку, з інструментами альтернативної семантичної теорії моделювання мови – теорією Мельчука – «Сенс ↔ Текст». Її безумовним полюсом, є високий ступінь орієнтованості на синтез тексту [24]. Такий підхід дає можливість реалізувати формальну модель мови для всієї індоєвропейської групи, а не тільки для англійської, де порядок слів у реченні відіграє ключову роль. Головним мінусом теорії «Сенс ↔ Текст» є тлумачно-комбінаторний словник [25], який базується на механізмі опису семантичних характеристик сем. Це має на увазі, що для кожної семи буде описаний повний набір її властивостей (в спеціальній статті) – від лексичних і морфологічних до таких високих семантичних рівнів як пов'язані ідіоми. Необхідність складання повного словника (в оригінальному вигляді) можлива тільки вручну, що робить задачу створення такого словника надважкою. Однак, навіть незважаючи на вищевказані мінуси, теорія «Сенс ↔ Текст є більш зручною для розробки системи синтезу тексту, ніж граматично-орієнтовані моделі. Використовуючи механізми штучного комп'ютерного інтелекту, можливо домогтися високих результатів, як наприклад, в системах М.М. Леонтьєвої, орієнтованих на комп'ютерний переклад. З точки зору генерації тексту, теорія «Сенс ↔ Текст» не розглядалася, хоча, як видно з вищесказаного, її застосування в цій області більш ніж виправдано. Тому першою задачею що була вирішена в рамках цієї роботи стало подолання обмежень тлумачно-комбінаторного словника за допомогою моделі м’якого розуміння Леонтьєвої і розробки автоматичного компоновника та аналізатора спеціальних семантичних структур даних [25].

Сформульовано та розглянуто попередню модель інтелектуальної системи генерації відповідей на основі неструктурованої бази знань. За її основу була взята концепція програмної обробки текстів на природній мові, що була висунута Луканіним – текстовий автомат. Головними плюсами даної концепції є: модульність – автомат представляє собою набір програмних інструментів аналізу та обробки тексту, що дозволяє гнучко досліджувати та будувати загальний алгоритм цільової задачі; рівневість – автомат об'єднує власні механізми за рівнями розуміння тексту (синтаксичним, лексико-морфологічним, семантичним або іншими, сформованими користувачем за необхідністю), що співвідноситься із рівневою моделлю м'якого розуміння Леонтьєвої; орієнтованість на використання бази знань – автомат виконує свою цільову задачу по обробці тексту завдяки роботи із деяким набором знань про мову і текст (у нашому випадку такими даними є корпус текстових документів); безпека – процес розробки автомату вимагає створення системи перевірки отриманих результатів та тестування власних компонентів [26, 27].

У ході досліджень була запропонована схема текстового автомату що складається із двох рівнів – рівня інформаційного представлення, який відповідає за фільтрацію вхідних документів, аналіз запитів користувача і генерацію текстових відповідей , та рівня глибинного синтаксичного аналізу, який відповідає за побудову семантичних моделей документу, зважування та ранжування текстових одиниць, а також оцінку отриманих результатів. Використання концепції семантичної моделі документа дозволило обійти обмеження тлумачно-комбінаторного словника. Таким чином, була висунута гіпотеза про можливість фільтрації даних і оцінки отриманих результатів завдяки автоматичному аналізу структури семантичної моделі, а використання зв'язків множини зважених термінів і запиту користувача не тільки відповідає ситуативному розумінню м'якої моделі а дозволяє уникнути необхідності використання лінгвістичних знань у процесі роботи системи.

Основні положення цього розділу викладені у публікаціях автора [24, 25, 26, 27]

РОЗДІЛ 2. ПОБУДОВА МАТЕМАТИЧНОЇ СЕМАНТИЧНОЇ МОДЕЛІ ДОКУМЕНТА

Одним з основних компонентів розробленої в даній роботі моделі генерації відповідей є спеціально розроблена семантична структура – модель, яка використовується для зберігання і подальшого семантичного аналізу текстових даних, що представляє собою спеціалізовану семантичну мережу.

З точки зору теоретичного визначення семантичної мережі у науковій літературі існує її різне розуміння, проте у загальному плані, поняття «семантична мережа» включає в себе три аспекти [28]: спосіб мислення про знання, в яких є поняття і відносини між ними; схематичне уявлення, що включає деяку комбінацію закладок, стрілок-покажчиків і міток; комп'ютерне уявлення того, що дозволяє активувати і формувати логічний висновок з використанням алгоритмів, які працюють за даними уявленнями.

В рамках дисертаційної роботи створена семантична модель підпадає під всі три описані критерії семантичної мережі: перший критерій використовується для опису семантичних властивостей бази знань системи, яка представлена нерозміченим корпусом текстів, другий критерій використовується для зручного графічного відображення побудованої моделі тексту, а третім критерієм є безпосередньо процес інтелектуальної автоматичної обробки вхідного тексту.

З точки зору програмної реалізації, питання щодо структури семантичної мережі є ще більш розмитим. Так, на зорі інженерії знань і обробки природної мови під семантичною мережею розумівся розмічений орієнтований граф, вершини якого відповідають деяким сутностям (поняттям, подіям, характеристикам або значенням), а дуги висловлюють зв'язок між цими сутностями [29]. Однак, сьогодні подібні визначення відносяться саме до області онтологій, тоді як семантичні мережі можуть бути представлені у вигляді графа деяких залежностей, не обмежуючись тільки понятійним описами. В основі поняття семантичної мережі лежить те, що процес накопичення знань в пам'яті відбувається через систему асоціативних зв'язків між поняттями, тому семантична мережа представляє не тільки систему зберігання інформації, але і структуру

моделі процесів мислення [29]. В ході моделювання даних процесів визначається порядок проведення міркувань, створення понять, хід логічного мислення на основі побудови програмних умовиводів. Що стосується структур даних, які представляють семантичну мережу – в мінімальній комплектації вони містять вузли, які відображають поняття і кордони між поняттями – абстрактні уявлення ідей, думок і одиниць знань і сенсу [29]. Це дозволяє використовувати семантичну мережу як формальну модель для зберігання і аналізу практично будь-яких знань, що задовольняє проблемній галузі нашої задачі, оскільки система використовує набір семантичних текстових міток як поняття для створення сенсового програмного відображення документа, що, на відміну від більш розповсюдженої онтологічної структури семантичної мережі, є новим підходом до опису семантичної структури тексту.

Дослідники виділяють шість різних типів семантичних мереж [29, 30], найбільш близькими з яких до даної дисертаційної роботи є гібридні мережі – семантичні мережі, що можуть створюватися як комбіновані підвиди інших типів семантичних мереж і, можуть як містити у собі усі їх властивості, так і ігнорувати деякі з них. Оскільки найбільш поширеним класом семантичних відносин є бінарні відносини [31], в якості робочого визначення в даній роботі буде використано наступне визначення: семантична мережа – це орієнтований граф, вершини якого – поняття, а дуги – зв'язки між поняттями.

Існують і інші форми подання знань, такі як продукційні правила, фрейми і формальні логічні моделі [32]. Їх розгляд виходить за рамки даної дисертаційної роботи, оскільки великою перевагою семантичних мереж є те, що вони не накладають обмежень на структуру знань або конкретну предметну область до тих пір, поки ці знання можливо уявити у вигляді орієнтованого графа [30]. Тому основна увага в даній роботі буде присвячена семантичним прикладним моделям як способу представлення і зберігання знань про документ, найближчим еталонним аналогом яких є гібридні семантичні мережі.

2.1. Процес автоматичної побудови семантичної моделі документа

Проведений раніше аналіз існуючих моделей в галузі автоматичної генерації текстів показав, що модель м'якого розуміння Леонтьєвої є найоптимальнішою для використання у нашій роботі. Данна модель визначає поняття ситуативного розуміння знання, при якому розуміння кожного значення термінів документа змінюється в залежності від дії зовнішніх чинників. Якщо адаптувати цю модель для нашої мети, таким зовнішнім фактором є запит користувача, що надійшов в систему. Уявити це можливо за допомогою терм-матриці D деякої текстової бази знань (2.1):

$$D = \begin{vmatrix} w_{1,1}(t_1, d_1) & \cdots & w_{1,n}(t_n, d_1) \\ \vdots & \cdots & \vdots \\ w_{m,1}(t_1, d_m) & & w_{m,n}(t_n, d_m) \end{vmatrix} \times \beta,$$
$$w(t,d) = \frac{n_t}{\sum_k n_k}, T \subset \{t_1 \cdots t_n\}, \beta = \begin{cases} 1, t \in T \vee t \in L(V,T) \\ 0, t \notin T \wedge t \notin L(V,T) \end{cases}, \quad (2.1)$$

де n – загальна кількість термінів в базі знань, m – загальна кількість документів в базі знань, w – вага певного терміну щодо документа, отримана за моделлю *TF*, n_t – частота поточного терміну в документі d, $\sum_k n_k$ – сума частот усіх термінів у документі d, T – множина термінів запита користувача, β – ознака наявності терміна в документі, рівна 1, якщо термін належить *T,* або належить підмножені семантично пов'язаних із запитом термінів, отриманих в результаті виконання функції $L(t,T)$ над семантичним словником V.

Дана модель чітко описує процес розуміння тексту за допомогою ЕОМ, вводячи при цьому кілька рівнів обробки тексту – синтаксичний, семантичний і міждокументний. Однак, модель не автоматизує процес створення словника знань, будучи таким чином, лише теоретичною основою для нашої системи, що вимагає значних доопрацювань. Тому, було прийнято рішення першим кроком розробки моделі генерації текстів створити програмний інструмент

автоматичного створення семантичної моделі тексту, структура якої буде схожа з гібридною семантичною мережею.

Головними задачами що були сформульовані до функціоналу майбутньої розробки є:

– відсутність необхідності додавання семантичного словника знань, попередньої семантичної розмітки тексту, або будь-яких інших лінгвістичних знань, що вимагають глобального ручного створення чи постійної підтримки;

– отримана модель має спиратися саме на семантичні властивості тексту і має бути орієнтована на генерацію текстових відповідей та роботу із запитом користувача;

– отримана модель має містити кількісні характеристики семантичних властивостей тексту для подальшого інтелектуального автоматичного аналізу.

У ході роботи із існуючими алгоритмами побудови програмних семантичних моделей тексту стало очевидно, що задля отримання необхідних результатів потрібно зміщати вектор досліджень у сторону статичних технологій кластеризації тексту і пошуку подібних текстових документів, що не отримали широкого застосування у задачі побудови семантичних моделей тексту – а саме, алгоритму латентно-семантичного аналізу [33].

Латентно-семантичний аналіз (також відомий як латентно-семантичне індексування) – це основний метод, який використовується для аналізу відносин між документами і термінами в колекції задля вилучення високорівневих понять і перетворення подання документів відповідно до ідентифікованих відносин [33].

У загальному випадку, латентно-семантичний аналіз переносить документи колекції і терміни в них в прихований (латентний) простір властивостей, в якому розмірності і вимірювання ідеально відповідають високорівневим поняттям або компонентам [33]. Тому, кожен документ представляється у вигляді зваженого поєднання таких компонентів, в той час як кожен термін може в різній мірі бути аналогічним чином пов'язаний з іншими поняттями. Ця схема дуже схожа на метод головних компонент, який використовується для відображення векторного

простору з можливими взаємозв'язками між вимірами, на інший простір, у якого немає таких взаємин [33].

Рис. 2.1. Алгоритм проведення латентно-семантичного аналізу над текстовим документом

Алгоритм роботи латентно-семантичного аналізу, зображений на рис.2.1, виглядає наступним чином: нехай є колекція з n документів і m різних термінів, витягнутих з них. Для застосування моделі латентно-семантичного аналізу, потрібно побудувати $m \times n$ матрицю «термін-документ» X з осередками $x_{i,j}$, що містять вагові коефіцієнти терміну t_i в документі d_j. Стовпці матриці X на практиці відповідають мультімножені слів для документа, при цьому можуть бути використані терміни, «зважені» за якої-небудь схемою: наприклад, це може бути розповсюджена схема TF-IDF [34, 35], або будь-яка інша схема, заснована на частотному аналізі.

Усередині даної матриці для оцінки взаємозв'язку (кореляції) може бути обчислений скалярний добуток між двома рядками (термінами) або двома стовпцями (документами). Повна матриця взаємозв'язків для термінів або документів може бути отримана шляхом обчислення XX^T або X^TX відповідно.

У матриці «термін-документ» застосовується сингулярне розкладання [35], математична методика, яка обчислює розкладання вихідної матриці X на три матриці (2.2).

$$X = \mathrm{U}\Sigma V^T \ , \qquad (2.2)$$

де, матриці U і V є ортогональними матрицями розмірності $m \times r$ і $n \times r$ відповідно, а матриця Σ – це $r \times r$ діагональна матриця, яка містить власні значення. Обґрунтування латентно-семантичного аналізу полягає в тому, що кожне з r власних значень відповідає одному з вищезазначених високорівневих компонентів, що відслідковуються в колекції документів, і позначає, наскільки цей компонент актуальний у всій колекції [35].

Власні значення упорядковуються відповідно до діагоналі матриці Σ в порядку убування, так що ті власні значення, які йдуть першими, пов'язані з найбільш важливими компонентами. Це дозволяє легко відрізати найменш важливі компоненти, просто видаливши відповідні рядки і стовпці в матрицях. Це скорочення потенційно дозволяє видалити «шум» в даних, який може бути складений, наприклад, з термінів або груп, що з'являються тільки в декількох документах і погано пов'язані з іншими. Сам по собі алгоритм латентно-семантичного аналізу не призводить до формування структури даних, що змогла би задовольнити поставленим раніше вимогам, проте він має важливу властивість – робити семантичні виводи про тексти виходячи лише із частотного портрету документа, що дало підставу продовжити дослідження щодо його використання як основи у вирішенні цільової задачі. Для цього був розроблений інтелектуальний алгоритм автоматичної побудови семантичної моделі тексту, етапи роботи якого зображені на рис. 2.2.

Програмною платформою для реалізації розробленої моделі стала мова java та супутні виконавчі технології [36]. Такий вибір зумовлений не тільки можливостями запуску додатків незалежно від платформи, а і великою кількістю як вбудованих інструментів роботи із текстом, так і додаткових безкоштовних програмних бібліотек, які програмно реалізують у собі більшість необхідних інструментів для вирішення нашої задачі.

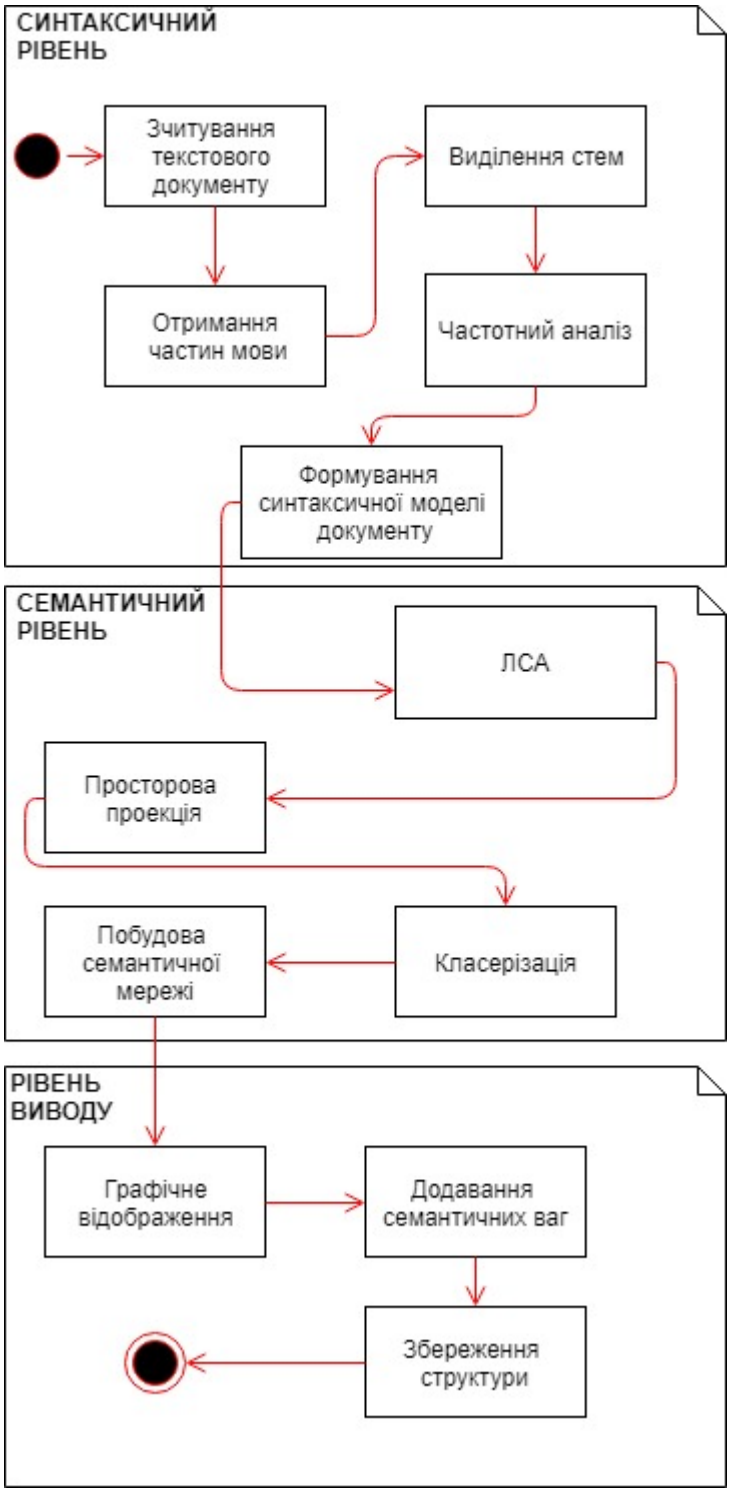

Рис. 2.2. Загальна структура побудови семантичної моделі тексту

Загальний алгоритм роботи моделі має лінійну структуру, яка поділена на три рівня обробки документу: синтаксичний рівень, що об’єднує у собі деякі типові задачі обробки тексту, такі як виділення слів і речень із вхідного тексту, визначення стем, зважування елементів тексту, автоматичне визначення частини

мови, тощо; семантичний рівень, що реалізує у собі компонент латентно-семантичного аналізу і додаткові засоби інтелектуальної обробки даних, завдяки яким відбувається формування фінальної семантичної моделі вхідного тексту; рівень виведення, що відповідає за відображення і збереження отриманої семантичної мережі у базу знань системи.

Розглянемо процес роботи цих рівнів детальніше, і почнемо з синтаксичного рівня роботи моделі (рис. 2.3), що зазвичай є типовим для моделей автоматичної обробки текстів.

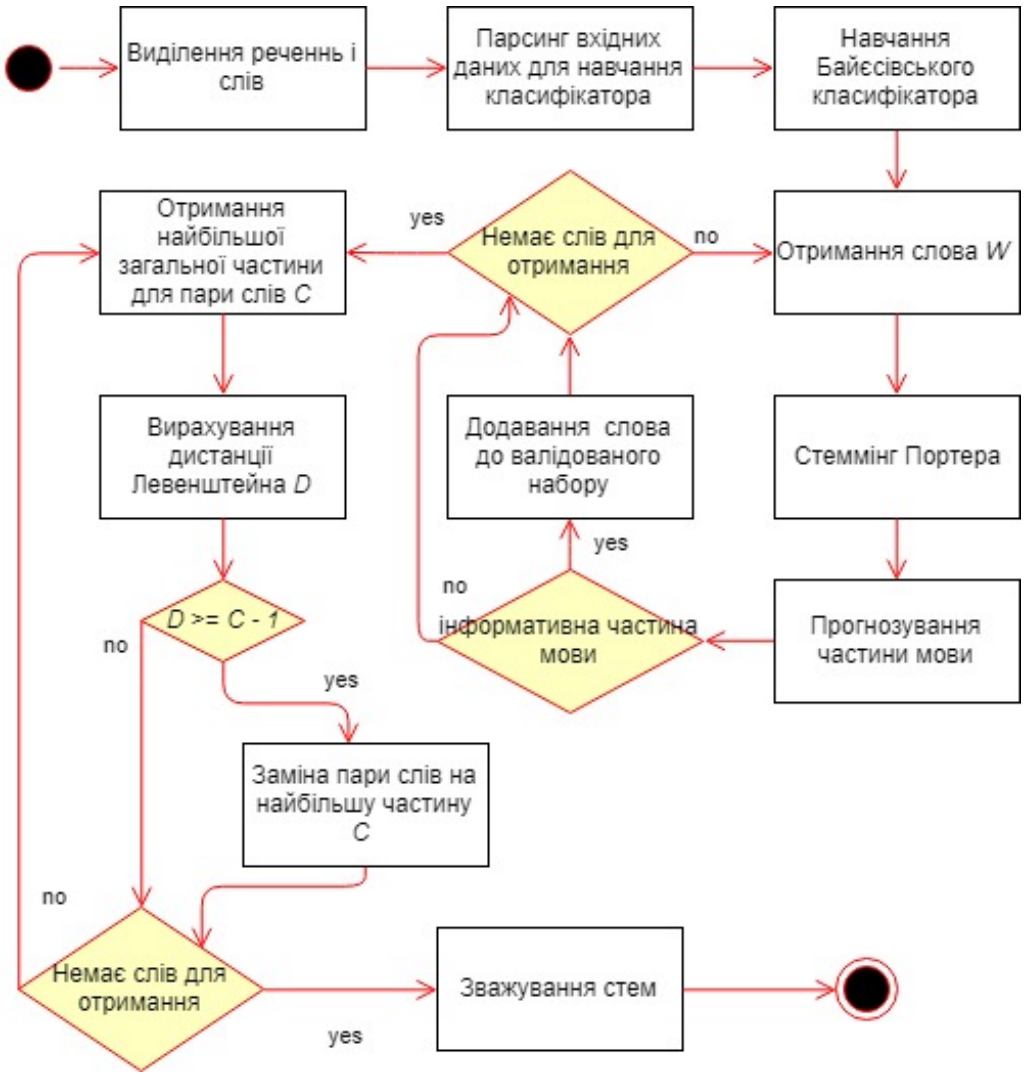

Рис. 2.3. Загальна структура роботи синтаксичного рівня

На вхід рівень приймає текстовій документ у форматі «plain text» (лише текст, без зображень, формул, таблиць, тощо) у якому відсутня будь-яка додаткова

семантична розмітка, після чого першим завданням, що виконується над отриманим текстом, стає синтаксичний аналіз, який ставить перед собою за мету виділення речень та слів з тексту. Ця задача хоча і не є надскладною, проте її точна реалізація дуже важлива задля коректної роботи моделі, оскільки речення та слова (або їх похідні – семи) є мінімальними семантичними одиницями створюваної моделі. Окрім того, при вилученні слів та речень необхідно враховувати багато факторів, що можуть спровокувати помилку (наприклад – робота із скороченнями). Тому для вирішення цієї задачі було застосовано механізми бібліотеки BreakIterator[37] із вбудованого в java пакету засобів роботи із текстом.

Як тільки слова і речення були переведені у програмні сутності, наступною задачею стає очистка тексту від стоп-слів, що співвідносяться із неінформативними частинами мови, до яких в рамках цієї дисертаційної роботи відноситься всі частини мови, шо не є іменником, прикметником, прислівником або дієсловом. Для цього необхідно використовувати механізм автоматичного визначення частини мови і, оскільки використовування словника суперечить головній меті даної роботи, то було прийнято рішення про використання системи оцінки імовірності належності слова до тієї чи іншої частини мови на основі наївного байєсівського класифікатору [38]. Для цього, в системі задається кінцевий набір пар $P < W, POS >$, де W – деяке слово, POS – його частини мови, набір стоп слів S, де $s \in S$ – має частину мови союз, частка, вигук або займенник, та набір значних частин мови POS_I. В цьому випадку, термін T потрапляє в результуючий набір T_R, якщо виконується умова 2.3:

$$T \in T_R \Leftrightarrow T \notin S \wedge B(last2(T), last3(T), P < W, POS >) \in POS_I, \qquad (2.3)$$

де $B(\ldots)$ – функція байєсівського класифікатора, що повертає клас – частину мови, $last2(T)$– функція, яка повертає ознаку двох останніх символів слова, $last3(T)$ – функція, яка повертає ознаку трьох останніх символів слова.

Даний метод заснований на теоремі Байеса, яка полягає у формулі обчислення апостеріорної ймовірності [38]. Такий підхід вже добре

зарекомендував себе при роботі із текстом – прикладами може послужити фільтрація спам-листів, або класифікація смс-повідомлень. Задля його реалізації в першу чергу необхідно мати еталонний набір слів та відносних до них частин мови, що дозволило би навчити систему і отримувати нові припущення при класифікації. Виходом із цієї ситуації став національний корпус російської мови [39], невелика частина якого (1.5 тисячі слів) розповсюджується вільно у електронному форматі XML, доступному для програмного парсингу. В тестовому файлі містяться речення разом із семантичними зв'язками між ними, де кожному слову приписані деякі морфологічні признаки, у тому числі – частина мови. І хоча такий об'єм замалий для навчання семантичного компоненту, для вирішення нашої цільової задачі на російській, і у випадку перекладу – українській мові його достатньо. Використання еталонного набору хоча і має на увазі обробку деякої бази знань, проте не викликає проблем словника, описаних раніше, оскільки навчання моделі буде проходити один раз і не потрібує подальшої підтримки протягом функціонування. Задля програмної обробки була використана бібліотека Daslaboratorium classifier [40], що реалізує у собі функціонал байєсівського класифікатора і має необхідне API для роботи із класами строк. Класами для прогнозування у нашій системі є, власне, частини мови, а критеріями прогнозування є закінчення слів: першим критерієм є дві останні літери слова, а другим – три. Для перевірки отриманого результату вже навчена система була апробована на частині корпусі, що не був використаний при її навчанні, і її точність при виконанні 750 тестів відповідно склала 98.78% для української мови, що цілком достатньо для прийнятної роботи цього етапу.

Наступним важливим етапом роботи системи є стемінг слів – приведення однокореневих слів до деякої загальної форми. Необхідність цього етапу полягає в подальшому підрахунку частот унікальних слів, визначаючи таким чином вагу для процесу подальшого латентно-семантичного аналізу. Під стемінгом розуміють виділення основи слова, яка буде незмінною для ряду форм незалежно від відмінку та наявних суфіксів [41]. При цьому отримана форма слова не обов'язково відповідатиме його кореню [41]. Через існування значних

відмінностей у процесі словотворення навіть споріднених та близьких мов, у залежності від мови тексту, що аналізується, необхідно застосовувати різні правила стемінгу [41]. З цієї причини неможливо розробити універсальний перелік правил, придатних для будь-якої мови. Існує декілька підходів до організації процесу стемінгу, проте вони найчастіше передбачають обробку текстів, наведених на розповсюджених у світі мовах, до яких українська не належить. На даний момент в літературі запропоновано стемери, що спираються на алгоритми з використанням підходу опису правил відсікання слів (стемер Портера [42], Paice/Husk стемер [43]), а також ті, що спираються на лінгвістичні дані та використовують семантичну інформацію слова (Y-стемер [44]), або використовують статистичну інформацію (аналіз K-грам [45], Stemka [46], MyStem [47]). Проте основною проблемою, що виникає при використанні стемерів є обробка слів, які при утворенні різних граматичних форм змінюють не тільки закінчення, а й основу слова. Наприклад, іменник «кішка» в знахідному і родовому відмінку множини має форму «кішок». Через такі мігруючи голосні стемер повинен або ігнорувати подібні форми, приводячи слово «кішки» до «кішок» і втрачаючи частину форм слова.

Виходом із цієї ситуації може стати використання попереднього процесу лематизації, як зроблено в роботі [41]. Лема – первісна, основна форма слова. Для іменників і прикметників, такою є форма однини називного відмінка. Для дієслів – відповідь на питання «що робити?». Відповідно, лематизація – це перетворення слова в словниковий вид або лему. Даний метод використовується в алгоритмах пошукових систем при індексуванні інтернет-сторінок. Наразі, багато лематизаторів представлені в інтернеті у відкритому доступі, деякі – безкоштовні, проте всі вони базуються на певних словниках, що обмежує їх використання разом із стемінгом в рамках цієї роботи. Тому було прийнято рішення про розробку і використання іншого підходу, що базується на знаходженні дистанції Левенштейна.

Дистанція Левенштейна [48] (відстань Левенштейна, редакційна відстань, дистанція редагування) визначається між двома рядками і дорівнює мінімальній

кількості операцій вставки одного символу, видалення одного символу і заміни одного символу на інший, необхідних для перетворення одного рядка в інший. Вперше такий спосіб обчислення дистанції між двома двійковими послідовностями запропонував радянський вчений-математик В. І. Левенштейн. Згодом спосіб поширився на послідовності довільного алфавіту. Відстань Левенштейна активно застосовується для виправлення помилок в пошукових системах, в текстових редакторах, а також в біоінформатиці.

Дистанцію Левенштейна можна сформулювати наступним чином: нехай є пара слів $S_1, S_2 \in T_R$, довжиною M и Nвідповідно. Тоді відстань Левенштейна $d(S_1, S_2)$ можна підрахувати за такою рекурентною формулою (2.4) $d(S_1, S_2) = D(M, N)$,

$$D(M,N) = \begin{cases} 0, i = 0, j = 0 \\ i, j = 0, i > 0 \\ j, i = 0\, j > 0 \\ \min \begin{cases} D(i, j-1) + 1, \\ D(i-1, j) + 1, \\ D(i-1, j) + m(S_1[i], S_2[j]) \end{cases}, i > 0, j > 0 \end{cases}, \qquad (2.4)$$

де $m(a, b)$ – дорівнює нулю якщо $a = b$ і одиниці в іншому випадку, $\min\{a, b, c\}$ повертає найменший з аргументів. Тут крок за *i* символізує видалення з першого рядка, по *j* –вставку в перший рядок, а крок за обома індексами символізує заміну символу або відсутність змін. Таким чином, дистанція Левенштейна визначається як мінімальна кількість односимвольних операцій (а саме вставки, видалення, заміни), необхідних для перетворення однієї послідовності символів в іншу. На основі отриманої дистанції для кожної пари слів S_1, S_2 відбувається заміна одного рядка на інший до тих пір, поки істинно (2.5):

$$S_1 = S_2 \Leftrightarrow d(S_1, S_2) < len_{common}(S_1, S_2), \qquad (2.5)$$

де $len_{common}(S_1, S_2)$ – довжина найбільшої загальної частини пари стем. Відносна похибка цього підходу склала 2% і була вирахувана експериментально за такою формулою (2.6):

$$\delta = \left(1 - \frac{\sum N_S}{\sum N}\right) * 100\%, \qquad (2.6)$$

де $\sum N$ – кількість розглянутих пар стема до стеми-кандидату, $\sum N_s$ – кількість правильно встановлених розглянутих пар стема до стеми-кандидату.

Фінальним етапом роботи рівня синтаксичного аналізу стає зважування стем, у результаті чого кожній стемі відповідає кількість її повторень у тексті і зважування речень, де під вагою речення мається на увазі сумарна вага усіх стем речення. Приклад отриманих результатів, зображений у таблиці 2.1, що є частиною тексту з додатку 3, де наведена його повна програмна модель, отримана на цьому етапі.

Для аналізу був використаний технічний текст, що присвячений темі методології розробки програмного забезпечення. Оброблений текст представляє собою програмну сутність, де кожне слово характеризується наступними основними полями: word – початкове слово, до якого був застосований стемінг Портера, stema – найбільша загальна частина слова, знайдена за створеним підходом на основі дистанції Левенштейна, pos – частина мови, яка була встановлена для вхідного слова (S – іменник, V – дієслово, A – прикметник, ADV – прислівник), weight – вага слова у тексті.

Як можна побачити з описаного прикладу, підхід на основі дистанції Левенштейна відділяє загальні частини на достатньо якісному рівні (методологія, методології: методолог; програмного: програм), дозволяючи оперувати не тільки закінченням, а й суфіксальною частиною для отримання спільної частини слова.

Таблиця 2.1

Приклад фрагменту технічного тексту після етапу синтаксичного аналізу.

Початковий текст	Результат
Методологія розробки програмного забезпечення – сукупність методів, що застосовуються на різних стадіях життєвого циклу програмного забезпечення і мають загальний філософський підхід.	[word = "Методолог", stema = "методолог", pos = S", weight = 28"] [word = "розробки", stema = "розробк", pos = S", weight = 3"] [word = "програмного", stema = "програм", pos = A", weight = 25"] [word = "забезпечення", stema = "забезпечен", pos = S", weight = 4"] [word = "сукупн", stema = "сукупн", pos = S", weight = 1"] [word = "метод", stema = "метод", pos = S", weight = 6"] [word = "застосовуються", stema = "застосов", pos = V", weight = 2"] [word = "циклу", stema = "цикл", pos = S", weight = 1"] [word = "програмного", stema = "програм", pos = A", weight = 25"] [word = "забезпечення", stema = "забезпечен", pos = S", weight = 4"] [word = "мают", stema = "мают", pos = S", weight = 1"] [word = "загальний", stema = "загальн", pos = S", weight = 3"] [word = "лософськ", stema = "лософськ", pos = A", weight = 2"]

Це дає можливість отримувати основи близьких слів і об’єднувати їх із більшою якістю, ніж при використанні алгоритму безсловникового стемінгу за Портером (програмного – програм). У деяких випадках підхід не дозволяє обійти проблему зміни літер при словотворі (програмного, програмування), проте, у рамках нашої задачі це не матиме значного впливу на результат роботи системи, оскільки такі помилки у великій кількості утворять окрему незалежну стему, а у маленькій – просто проігноруються при подальшому аналізі. Слід звернути увагу на те що текст є очищеним від неінформативних частин мови – наведені слова так чи інакше відносяться до однієї з чотирьох частин мови, що є корисними для подальшого семантичного аналізу. Не дивлячись на те, що іноді присутні помилки

визначення точної частини мови ([word = "зазвичай", stema = "зазвича", pos = S", weight = 2"]), для нашої цільової задачі цілком достатньо точності у перевірці належності слова саме до інформативної підмножини частин мови, оскільки саме присутність службових частин мови у тексті впливає на роботу етапу отримання стем.

Найважливішим етапом роботи синтаксичного блоку є зважування стем, що стає можливим через визначення і заміну слів на унікальні загальні частини. Умова використання текстів формалізованого жанру зумовлена саме цим етапом – робота синтаксичного блоку базується на гіпотезі про те, що слова із найбільшою частотою будуть напряму залежати від семантичних характеристик тексту у подальшому. Вже на цьому етапі можливо простежити зв’язок між темою тексту та вагою стем: слова «методологія» ([word = "Методолог", stema = "методолог", pos = S", weight = 28"]) та «програма» ([word = "програмного", stema = "програм", pos = A", weight = 25"]), які найбільш відносяться до змісту викладу у тексті, мають найбільшу вагу у порівнянні з іншими стемами, оскільки мають найбільшу кількість пов’язаних основ слова. Зона розповсюдження синтаксичного рівня обмежується текстом і не виходить на рівень корпусу. Це означає, що синтаксичні данні одного тексту не залежать від іншого тексту у корпусі, що робить синтаксичний аналізатор універсальним інструментом для аналізу текстів. Наприклад, розглянемо результат обробки іншого тексту «Періодизація філософії» (додаток 4), фрагмент якого наведено у таблиці 2.2. Вага і форма однієї і тієї самої стеми у обох текстах відрізняється ([word = "лософ", stema = "лософ", pos = S", weight = 8"] та [word = "лософський", stema = "лософськ", pos = A", weight = 2"]), оскільки вираховувалась окремо для кожного з текстів і залежить як від розміру тексту, так і від слів-пар, з якими іде порівняння. Так, текст з додатку 3 має розмір у три рази більше ніж з додатку 4, проте у першому тексті стем з основою «лософ» лише 2, тоді як у другому тексті – їх вісім, оскільки тексти мають різну тематичну спрямованість.

Таблиця 2.2

Приклад фрагменту гуманітарного тексту після етапу синтаксичного аналізу

Початковий текст	Результат
Середньовічну філософію умовно можна розділити на наступні періоди: 1) введення або патристика (II – VI ст.); 2) аналіз можливостей слова, пов'язаний з християнською ідеєю творення світу по Слову і Його втілення в світі (VII – X ст.); 3) схоластика (XI – XIV ст.).	[word = "Середньов", stema = "середн", pos = S", weight = 5"] [word = "лософ", stema = "лософ", pos = S", weight = 8"] [word = "умовно", stema = "умовн", pos = ADV", weight = 2"] [word = "можна", stema = "можн", pos = S", weight = 4"] [word = "наступн", stema = "наступн", pos = S", weight = 1"] [word = "введення", stema = "введен", pos = S", weight = 1"] [word = "патристика", stema = "патристик", pos = S", weight = 2"] [word = "можливостей", stema = "можлив", pos = S", weight = 2"] [word = "слова", stema = "слов", pos = S", weight = 3"] [word = "язаний", stema = "язан", pos = S", weight = 2"] [word = "християнською", stema = "християн", pos = S", weight = 2"] [word = "творення", stema = "твор", pos = S", weight = 2"] [word = "Слову", stema = "слов", pos = S", weight = 3"] [word = "лення", stema = "лен", pos = S", weight = 1"] [word = "схоластика", stema = "схоласти", pos = S", weight = 2"]

Не дивлячись на те, що вже на етапі синтаксичного аналізу існує залежність між даними і семантичними властивостями тексту, її ще недостатньо для ствердження про існування процесу інтелектуального розуміння семантики тексту у моделі. Для його організації виконується ряд обробок статистичних даних, отриманих на рівні синтаксичного аналізу, сукупність яких утворює семантичний рівень моделі, основні кроки якого зображено на рис. 2.4.

Рис.2.4. Загальна структура роботи семантичного рівня

Розглянемо кроки алгоритму роботи семантичного рівня більш детально. Розмічений текст D, що було отримано на синтаксичному рівні проходить етап частотного аналізу, в результаті якого текстовим даним відповідає матриця $M(D)$, рядки якої відображають речення, стовпці – стеми, а значення формуються як число входжень стеми в речення. Над отриманою таким чином матрицею виконується операція сингулярного розкладання порядку $m \times n$, що була описана в рамках латентно-семантичного аналізу (2.7):

$$M(D) = \begin{vmatrix} w_{1,1}(t_1) & \cdots & w_{1,n}(t_n) \\ \vdots & & \vdots \\ w_{m,1}(t_1) & & w_{m,n}(t_n) \end{vmatrix}, M(D) = U \sum V, \qquad (2.7)$$

де n – кількість термінів, m – кількість речень, $w_{m,n}(t_n)$ – число входжень терміну n в речення m; $\sum$ – матриця розміру $m \times n$ з невід'ємними елементами, у якій елементи, що лежать на головній діагоналі – це сингулярні числа (а всі елементи, що не лежать на головній діагоналі, є нульовими), U (порядку m) и V (порядку n) – це дві унітарні матриці, що складаються з лівих і правих сингулярних векторів відповідно (а V^*–це поєднано-транспонована матриця до V).

Фрагмент отриманих лівої і правої сингулярних матриць для тексту з додатку 5 зображений в таблиці 2.3.

Таблиця 2.3

Фрагмент сингулярних значень для лівої U і правої V матриць

U		V	
0.016999045213105635	-0.06238172491801002	-0.020427669879958622	0.2367204172528457
0.07904916949311086	0.09542183140658986	0.17181578966075053	-0.18727319819904106
0.03916883576202113	0.013348234862403946	-0.4165417511865938	-0.12199691988484404
0.0653159957354656	0.13505812647285426	0.3436851721963842	-0.04386936342578947
0.027716291130482854	-0.02409203793381208	-0.036662026737650316	0.0126333954653221
0.12417631386624298	-0.08160324639714774	-0.2233441091414292	0.15317485106147077
0.13802618197015548	-0.04626863038612519	-0.2788559746834593	-0.2264955875703946
9.633627513608026E-4	-1.3007390995266652	-0.05980273743880148	0.13746809455093353

Оскільки сингулярне розкладання є стійким, стає можливим прибрати значення лівої і правої матриці, які відповідні низьким сингулярним значенням, залишивши тільки два найбільших, що представляють собою координати для відображення на двомірну площину. Приклади конкретних координат що залежать від стеми або речення зображені у таблиці 2.4.

Таблиця 2.4

Приклад залежності координат і значень для сем і речень

Координати	Значення
0.0022231829759011 71; – 0.006914238190982283	Програми
0.0750574537326258; 0.1730125924893532	Алгоритм
0.01983259551958578; 0.04288953463604017	Текст
0.07927810329284618; – 0.042323591356239386	Програм
0.016999045213105635; – 0.06238172491801002	Клас
0.001733942545363416; 0.005480368431279287	Доступ
-0.2788559746834593; – 7.014130929455636E-4	для цього в класах передбачені спеціальні методи функції звані конструкторами
-0.06150091503947243; – 0.1782250160751949	звичайні поля створюються по одній копії для кожного конкретного об'єкта екземпляра класу
0.07539971090049394; – 0.2920219055902991	існує основне правило ніщо в одному класі не може бачити приватних елементів іншого класу

Отримані на попередньому етапі координати проектуються на двомірну площину як масив точок для стем і для речень. Приклад цих операції для тексту з додатку 5 представлений на рис.2.5.

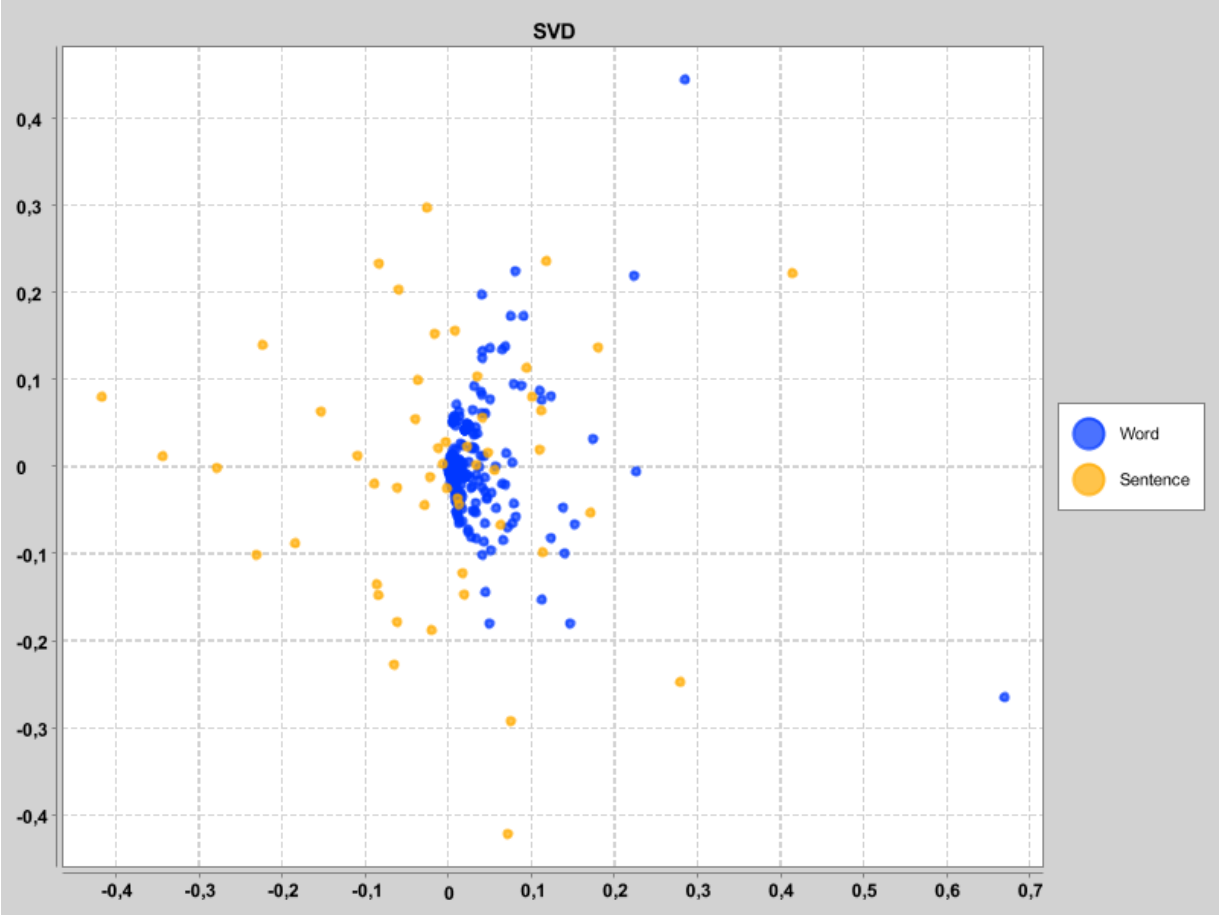

Рис. 2.5. Проекція сингулярних координат сем і речень документу. Сіньми точками позначені стеми, жовтими – речення

Після завершення етапу латентно-семантичного аналізу і проектування даних на площину, наступним кроком стає кластеризація отриманих точок-координат для стем і для речень за алгоритмом k-means [49]. Даний алгоритм розбиває множину елементів векторного простору на заздалегідь відоме число кластерів k. Дія алгоритму така, що він прагне мінімізувати середнє відхилення на точках кожного кластера (2.8).

$$V = \sum_{i=1}^{k} \sum_{x \in S_i} (x - \mu_i)^2 , \qquad (2.8)$$

де k – число кластерів, S_i – множина точок-координат кластеру, $i = 1, 2, \ldots, k$, а μ_i –центр мас точок-координат із кластера S_i. Головними особливостями алгоритму k-means є те, що, по-перше, необхідно заздалегідь знати кількість

кластерів, а по-друге алгоритм дуже чутливий до вибору початкових центрів кластерів – класичний варіант має на увазі випадковий вибір кластерів, що дуже часто є джерелом похибки. Тому в рамках розробки етапу семантичного аналізу були створені методи для встановлення необхідних параметрів кластеризації. По-перше, кількість кластерів для стем і для речень *cl* визначається за формулою (2.9):

$$cl(W, W_U) = \frac{count(W)}{count(W_U)}, \tag{2.9}$$

де $count(W)$ – кількість слів, $count(W_U)$ – кількість стем, а центроїдами кластерів-стем є координати стем з найбільшою кількістю входжень в текст, що визначаються за формулою (2.10):

$$C_{st}(W_U) = \max(W_0 \cdots W_{cl}), \tag{2.10}$$

де W_0 ... W_{cl} – кількості входжень стем із кластеру у текст. По-друге, центроїдами кластерів-речень є координати речень з найбільшим загальною вагою стем, яка визначається по формулі (2.11):

$$C_S(W_S) = \max\left(\sum_{i=0}^{SN} W_i\right), \tag{2.11}$$

де W_i – кількість входжень стеми із речення у текст, *SN* – кількість стем в реченні. В якості міри відстаней між об'єктами кластера було вибрано відстань міських кварталів (2.12):

$$d_1(\vec{p}, \vec{q}) = \left\|\vec{p} - \vec{q}\right\|_1 = \sum_{i=1}^{n} \left|\vec{p}_i - \vec{q}_i\right|, \tag{2.12}$$

де $\vec{p},\vec{q}$ – вектори. Ця відстань є звичайним середнім різниць по координатах. У більшості випадків ця міра відстані приводить до таких же результатів, як і для звичайної відстані Евкліда [50], проте для цього підходу вплив окремих великих різниць (викидів) зменшується – оскільки вони не зводяться в квадрат, що і стало головним аргументом її використання в рамках нашої моделі.

Останнім етапом роботи семантичного рівня є формування семантичних контурів у двомірній площині. Для цього, на основі координат точок кожного кластера-стеми будується контур опуклої фігури за алгоритмом Джарвіса [51], суть якого полягає у наступному: нехай на площині задана кінцева множина точок *A*. Оболонкою цієї множини називається будь-яка замкнута лінія *H* без самоперетинів така, що всі крапки з *A* лежать всередині цієї кривої. Якщо крива *H* є опуклою (наприклад, будь-яка дотична до цієї кривої не перетинає її більше ні в одній точці), то відповідна оболонка також називається опуклою.

Виходячи із вищесказаного, семантичним контуром кластеру називається опукла фігура, що була сформована для множини точок із одного кластеру-стеми, або кластеру-речення. Для програмної реалізації цього алгоритму була використана бібліотека GiftWrapper [52]. Приклад отриманих результатів можна побачити на рис. 2.6 для кластерів-стем і рис. 2.7 – для кластерів-речень, де кожному кольору точки відповідає кластер, отриманий за алгоритмом k-means, а лінії обмежують приблизний контур опуклої фігури, отриманої за алгоритмом Джарвіса.

В таблиці 2.5 наведено приклад складу семантичних контурів стем, які показують основний вихід роботи семантичного рівня. Як можна побачити, у результаті семантичної оброки тексту стеми розділилися на групи, частина яких концентрує у собі найбільшу кількість пов’язаних за семантичною відповідністю стем (кластер №5).

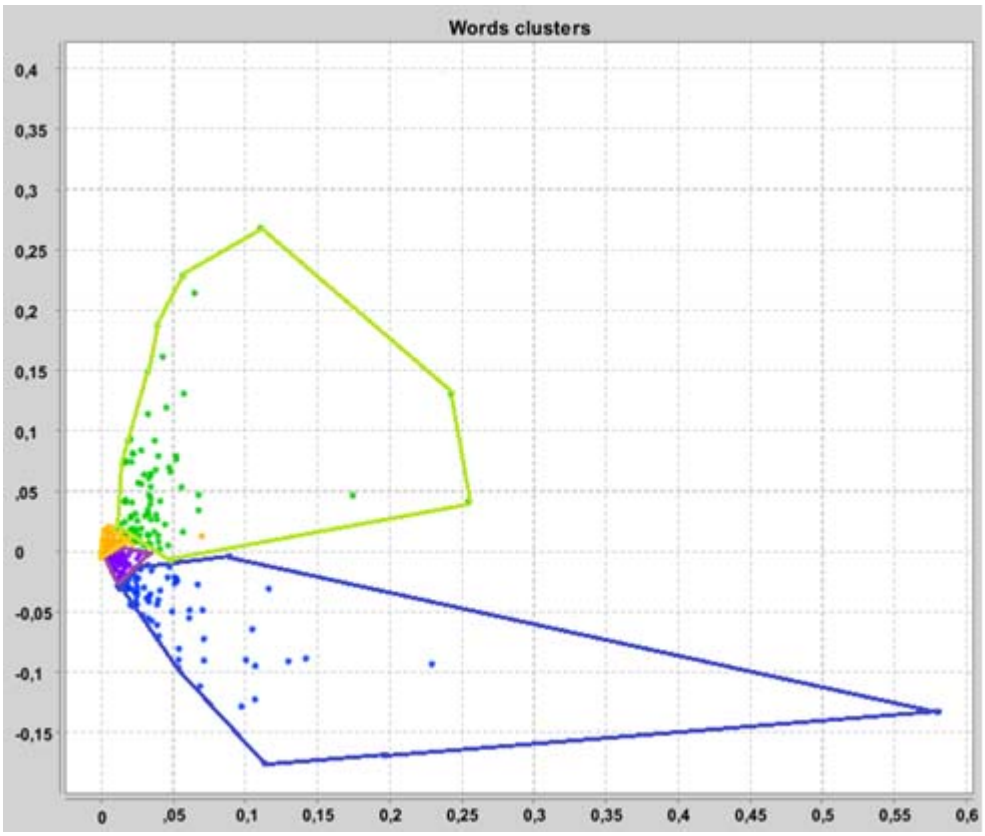

Рис. 2.6. Опуклі контори кластерів-стем

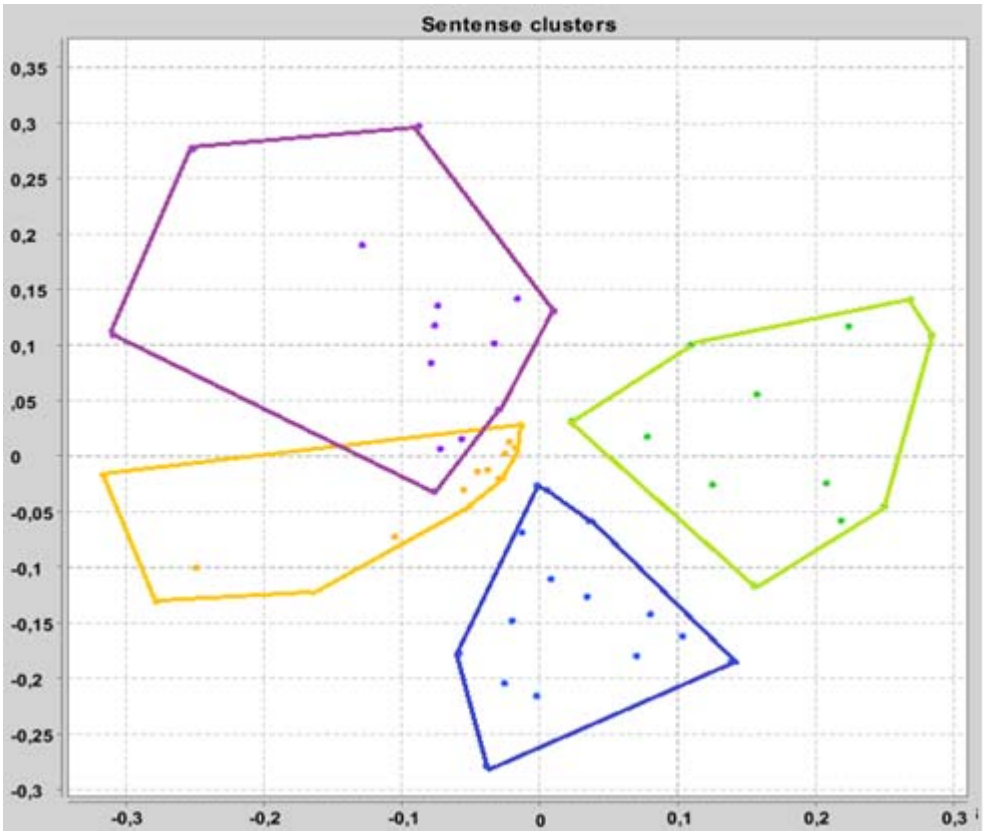

Рис. 2.7. Опуклі контури кластерів-речень

Таблиця 2.5

Приклад семантичних контурів стем (1-5)

№	Контур
0	члено грають прийшла ремінь java конструктор патерн інтелекту python взаємодії місць прихований фіналізатор параметром атрибути object згадують груші член більшості зовнішні речей серед oberon с шаблон матическі тримаючи огляд заду ruby
1	квітн платформ значен батьківськ займаємося виготовленням орга яблук практи новий прізвищ людин істот захищен ование private вирізання сенс зувати масив оформл дан несе одн раз
2	характеризуються прийшла кількостей класифікації цілому ключовим ставши патернів поклику кадрів інтелекту доступ програми можна олжн альн вого ботаніки зоології прототипна зв нуля вити властивостей нескінченності рекомендацій lua self ставлення
3	значен специальн жніх ращаются редст личн доступ мент порядку текст можн нащад ого флаже сподівання вирізання зв'язність осіб осту несе варіант овані передбачається вико прид змовившись формалізованого
4	робіт числ існуюч інтер проце можут конкретн опуска ічески абстрактн рам дан алгоритм функці повн
5	модификатор держ мова визнч protected public нащадок тип структ належ клас друг об’ект орієнтован тренн екземпляр рукт програм констант метод

Мова йде про стеми «модифікатор» «мова» «protected» «public» «нащадок» «клас» «об’єкт» «орієнтован» «екземпляр» «програм» «констант» «метод» та інші, що відносяться до теми тексту «клас» як поняття з програмування. Для подальшого аналізу такий контур має першочергове значення, оскільки він створений навкруги головної і найважчої стеми тексту – «клас», що є кореневим поняттям на основі якого буде проводитися аналіз запиту користувача –

семантичною міткою. При цьому інші стеми, що не мають кореневого поняття у своєму складі, увібрали менш значні з семантичної точки зору поняття і проігноруються при подальшій роботі.

Більшість наведених стем з 5 контуру не мають безпосереднього синтаксичного зв’язку у тексті із кореневим поняттям, проте належать до спільного кластеру, що дає можливість використовувати їх як суміжні до семантичного кореня знання – семантичні одиниці, створюючи таким чином аналог тлумачно-комбінаторного словника, а описаний алгоритм роботи семантичного рівня дозволяє обійти обмеження ручного складання такої бази знань, оскільки вхідні данні не проходили попередньої ручної семантичної розмітки або категоризації.

Безумовно, програма все ще не «розуміє» на своєму рівні що таке програмування, клас або об’єкт, проте вона розпізнає належність понять до деякої абстрактної сутності. Це дає можливість описувати семантичний клас на програмному рівні як сукупність понять, що його визначають: будь-яка стема із наведеного прикладу може нести у собі декілька сенсів (клас – чого саме?; модифікатор – з якої предметної галузі?) проте якщо розглядати їх разом вони однозначно визначають галузь «інформаційні технології», що і реалізує ситуативне розуміння семантичних понять за Леонтьевою, яке було описане раніше.

У таблиці 2.6 наведено приклад кластеру-речення, що утворює власний семантичний контур. Використання кластерів-речень розкривається нижче, і зумовлено необхідністю автоматичного формування зв’язків між наборами стем і речень – не дивлячись на те, що семантичні відповідності стем у тексті встановлені, програмна модель тексту ще не побудована, оскільки по-перше не описані міжфразові зв’язки у тексті, а по друге не виведені кількісні характеристики семантичних властивостей документу, що приводить нас до заключного етапу роботи моделі – рівня виведення, на якому відбувається формування семантичної моделі документа (рис. 2.8).

Таблиця 2.6

Приклад семантичного контуру речення

Контур
Клас – це елемент ПО, що описує абстрактний тип даних і його часткову або повну реалізацію. Поряд з поняттям «об'єкта» клас є ключовим поняттям в ООП (хоча існують і безкласові об'єктно-орієнтовані мови, наприклад, Self, Lua; докладніше дивіться прототипне програмування). Тобто «екземпляр класу» в даному випадку означає не «приклад деякого класу» або «окремо взятий клас», а «об'єкт, типом якого є якийсь клас». Наприклад, по можливості, клас «рядок тексту» буде містити всі основні методи / функції / процедури, призначені для роботи з рядком тексту, такі як пошук в рядку, вирізання частини рядки, тощо. Як і поля, код у вигляді методів / функцій / процедур, що належать класу, може бути віднесений або до самого класу, або до екземплярів класу. Метод, що належить класу і співвіднесений з класом (статичний метод) може бути викликаний сам по собі і має доступ до статичних змінних класу. Метод, співвіднесений з екземпляром класу (звичайний метод), може бути викликаний тільки у самого об'єкта, і має доступ як до статичних полів класу, так і до звичайних полях конкретного об'єкта (при виклику цей об'єкт передається прихованим параметром методу). Правила, що визначають можливість або неможливість безпосередньо змінювати будь-які змінні, називаються правилами завдання областей доступу. Спадкування за типом protected робить все public-члени батьківського класу protected-членами класу-спадкоємця (C ++); public (відкритий член класу) – звернення до члена допускаються з будь-якого коду.

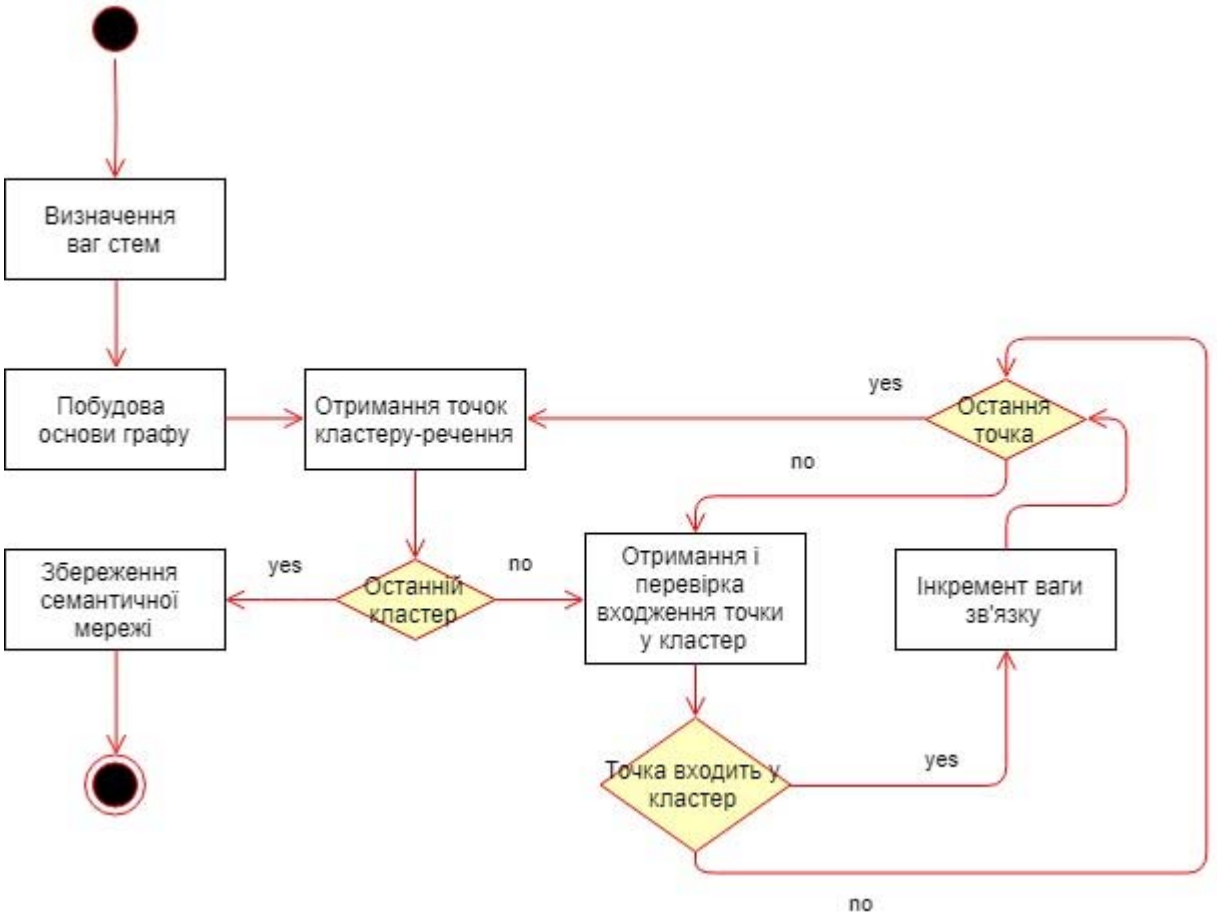

Рис 2.8. Загальна структура роботи рівня виведення

Першим кроком рівня виведення є визначення кількості координат-стем C^S, що входять в кожний кластер-стему, після чого будується семантичний граф зв'язків кластерів-стем в порядку убування C^S, який є основою для побудови семантичної мережі. Для кожної фігури кластерів-стем, отриманої по алгоритму Джарвіса, перевіряється попадання точок, які формують кожен кластер-речення. Якщо такі точки знайдені – кластер речення приєднується в мережі до кластеру-стеми, де вага зв'язку – це кількість точок, що потрапили в контур кластера-стеми. Отримана таким чином семантична структура M зберігається в базу знань і представлена множиною множин термінів T і множиною множин речень S, зв'язок між якими визначається матрицею семантичних ваг W (2.13):

$$T = \left\{ T_1\{t_1 \dots t_{w_{T_1}}\} \cdots T_n\{t_n \dots t_{w_{T_n}}\} \right\},$$
$$S = \{S_1\{s_1 \dots s_{l_1}\}\} \cdots S_1\{s_1 \dots s_{l_n}\}\}, \qquad (2.13)$$
$$W = \begin{vmatrix} \{w_{T1}, w_{S1}\}, & \{w_{T2}, w_{S1}\} & \cdots & \{w_{Tn}, w_{S1}\} \\ \vdots & \vdots & & \vdots \\ \{w_{T1}, w_{Sm}\}, & \{w_{T2}, w_{Sm}\} & \cdots & \{w_{Tn}, w_{Sm}\} \end{vmatrix},$$

де $t_1 \dots t_{w_{T1,}}$ – множина координат стем, що формують кластер-стему, $s_1 \dots s_{S_l}$ – множина координат речень, що формують кластер-речення, $w_{T1,}$ – вага кластера-стеми, S_{l1} – розмір кластера речення, w_{S1} – вага зв'язку між кластером-стемой і кластером-реченням, n – кількість кластерів-стем, $m = n$ – кількість кластерів-речень. Для програмного опису і відображення отриманих проекцій семантичного графу була використана бібліотека XChart [53]. Результат роботи етапу виведення для тексту з додатку 5 зображений на рис. 2.9.

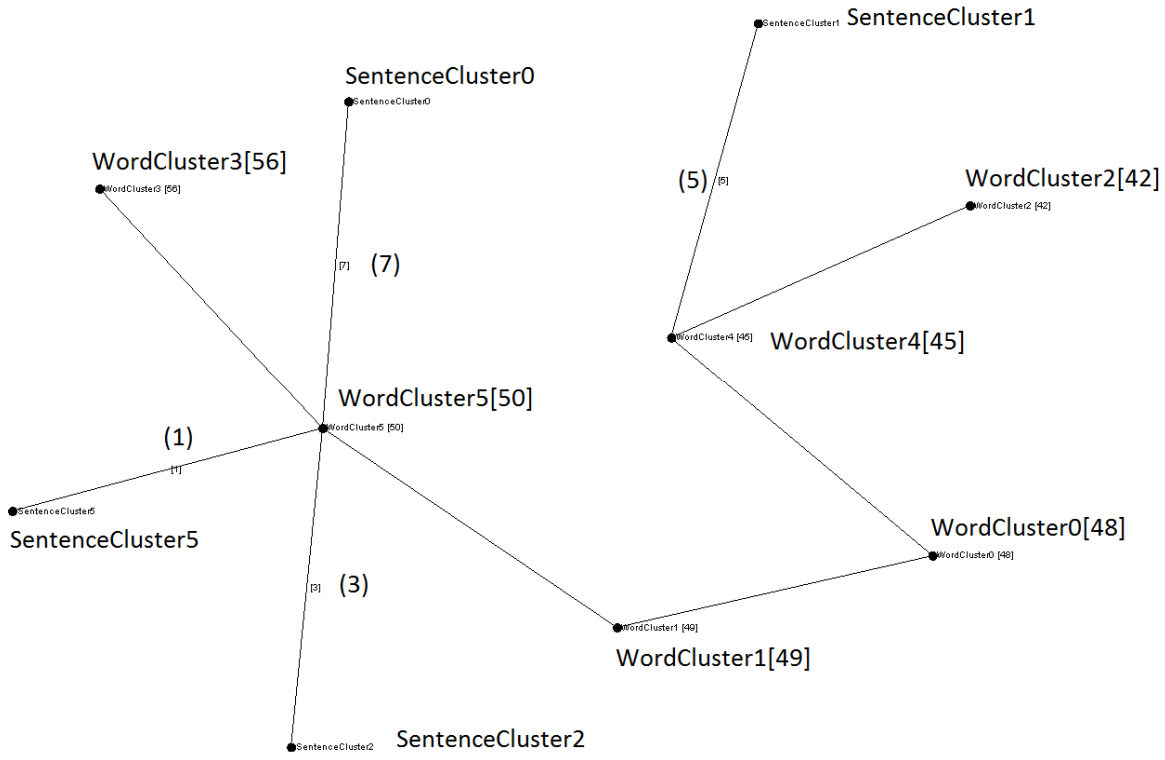

Рис 2.9. Семантична модель документа

Результуюча семантична модель складеться із кластерів-стем (позначено на рисунку як WordCluster) до яких прив'язане значення C^S (зображене на рисунку

у квадратних дужках), та кластерів-речень (позначено на рисунку як SentenceCluster) які пов’язані із кластерами-стемами семантичним відношенням, вага якого позначена у круглих дужках. Отримана семантична модель може бути використана для зняття числових даних, що характеризують семантичні властивості знання, до яких відносяться кількість стем-кластерів та кількість стем-речень, кількість точок у кластерах-стемах, ваги і кількість зв’язків, кількість стем що не увійшли до семантичної мережі, які використовуються для автоматизації процесів оцінки і фільтрації текстів за їх семантичною складовою. Окрім того, створена структура дозволяє встановити прямі відносини між кластерами-стемами, що як було сказано вище є семантичними мітками документа і масивами речень, шо відповідають семантичним міткам і являють собою семантичне подання документу – контури речень, за якими відбуватиметься подальший пошук і автоматична генерація тексту-відповіді.

Головною властивістю створеного підходу є його універсальність при використанні у повнотекстових корпусах що містять у собі тексти різної жанрової спрямованості. Розглянемо її складові детальніше на конкретних прикладах.

Перш за все, кожен новий текст генерує окрему семантичну мережу, що не містить залежностей від інших текстів у корпусі – на рис. 2.10 і рис. 2.11 зображений процес обробки тексту «Методологія програмування» з додатку 3, а у таблиці 2.7 міститься склад семантичного контуру першого кластера – стеми, що являє собою семантичну мітку тексту.

Хоча вхідний текст має схожу тематичну спрямованість (інформаційні технології), проте кластер, який було сформовано навколо найважчих слів має інше наповнення, ніж наведений раніше приклад обробки тексту з додатку 5. Окрім того, структура семантичної мережі на рис 2.11 має меншу кількість вузлів, оскільки об’єм відповідного тексту приблизно у 3 рази менше ніж текст з додатку 5. У мережі також присутній кластер (SentenceCluster1) що не має ніяких семантичних зав’язків. Така велика різниця у структурах текстів пояснюється відсутністю прямих міжтекстових семантичних зв’язків у корпусі та stateless – режимом роботи створеної моделі, де кожен текст і відповідна йому семантична

модель розглядаються як різні одиниці знань навіть у рамках однієї предметної галузі.

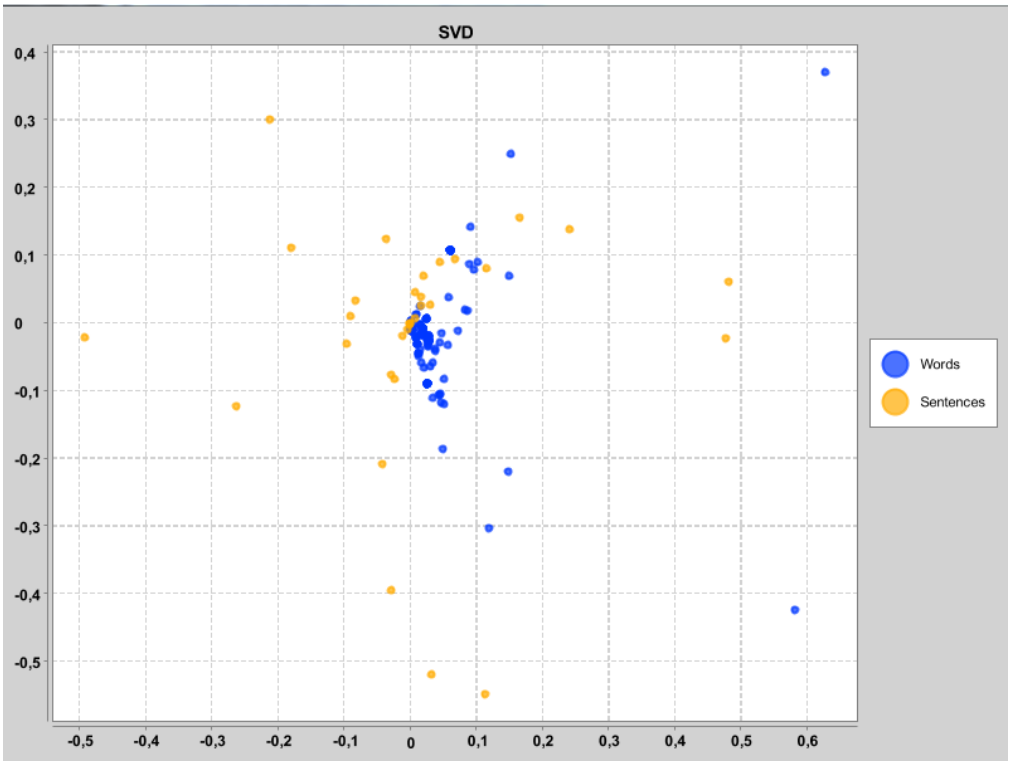

Рис. 2.10. Проекція сингулярного розкладання тексту

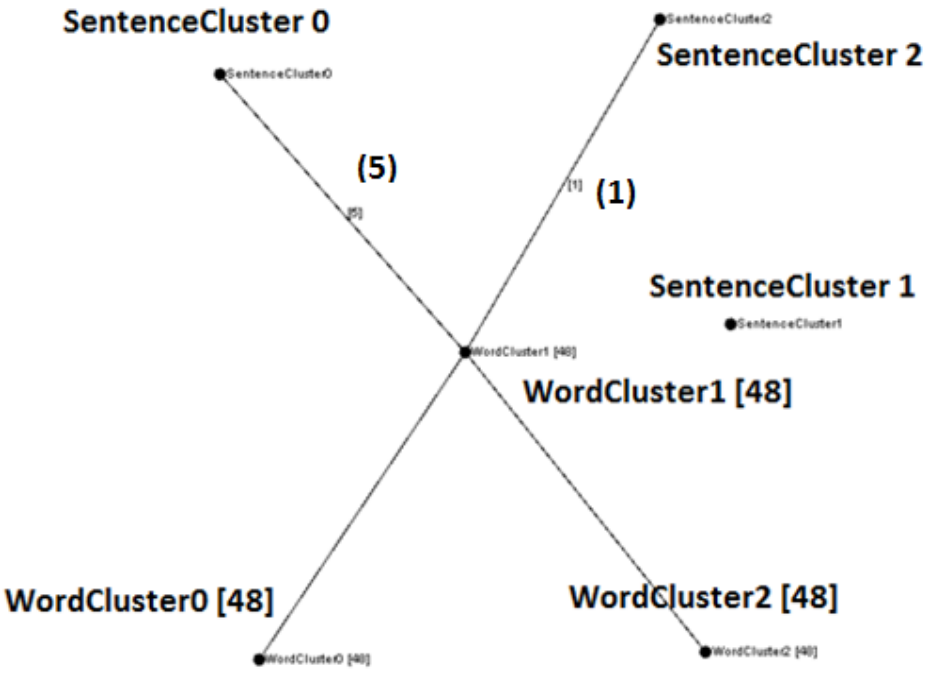

Рис. 2.11. Семантична модель тексту

Таблиця 2.7

Головний кластер-стема тексту WordCluster1

забезпечення ключових метдолог підтримувати модульності концепціями легко ділити мову використовувати пояснити називають парадигмою абстрак характери розробка невластивою циклу рішення фінансові програм часу ресурсів розрахунки більш ефективність життєвого створення системи узгодженим іншого конструювання означає шляху мають зазвичай принцип

Цей факт призводить нас до наступної властивості створеного підходу – можливість оброблювати тексти незалежно від їх предметної галузі. Дійсно, оскільки побудова семантичної моделі працює на рівні тексту а не корпусу, це дає нам можливість використовувати підхід для мультигалузевої бази знань, не регулюючи її тематичне наповнення ніяким чином. На рис. 2.12 і рис 2.13. зображені данні для побудови семантичної мережі для тексту з додатку 4, тематична спрямованість якого стосується теми «філософія», а у таблиці 2.8 – семантична мітка тексту з додатку 4.

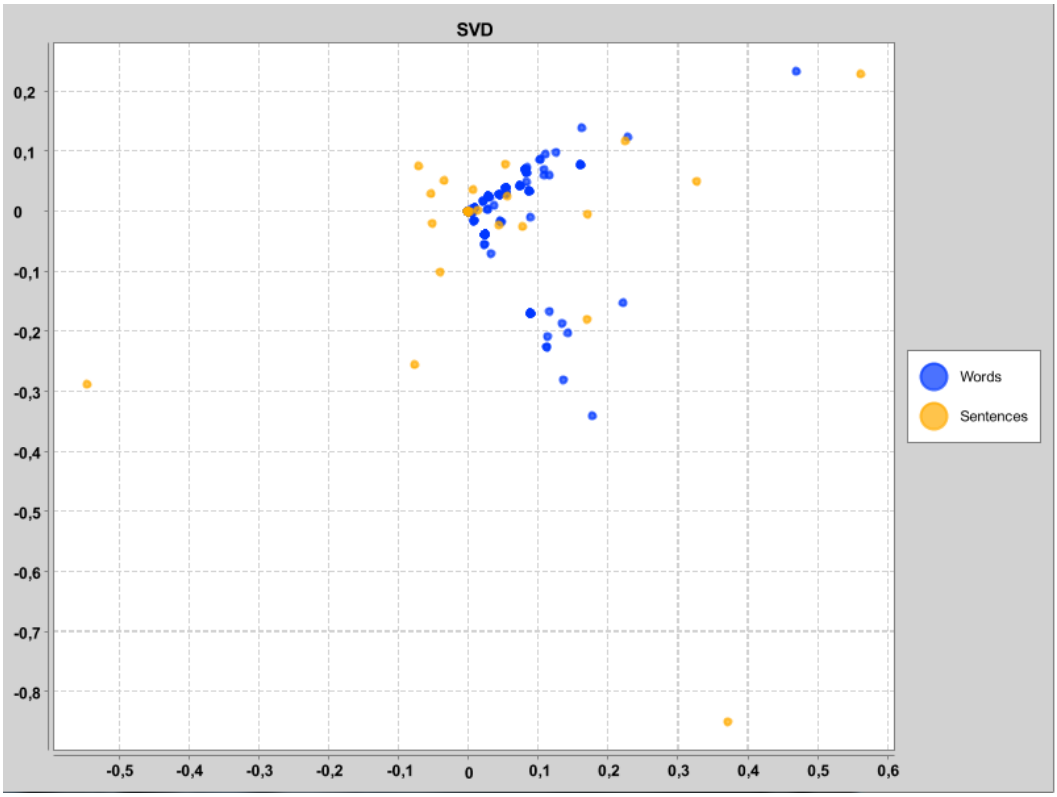

Рис. 2.12. Проекція сингулярного розкладання тексту

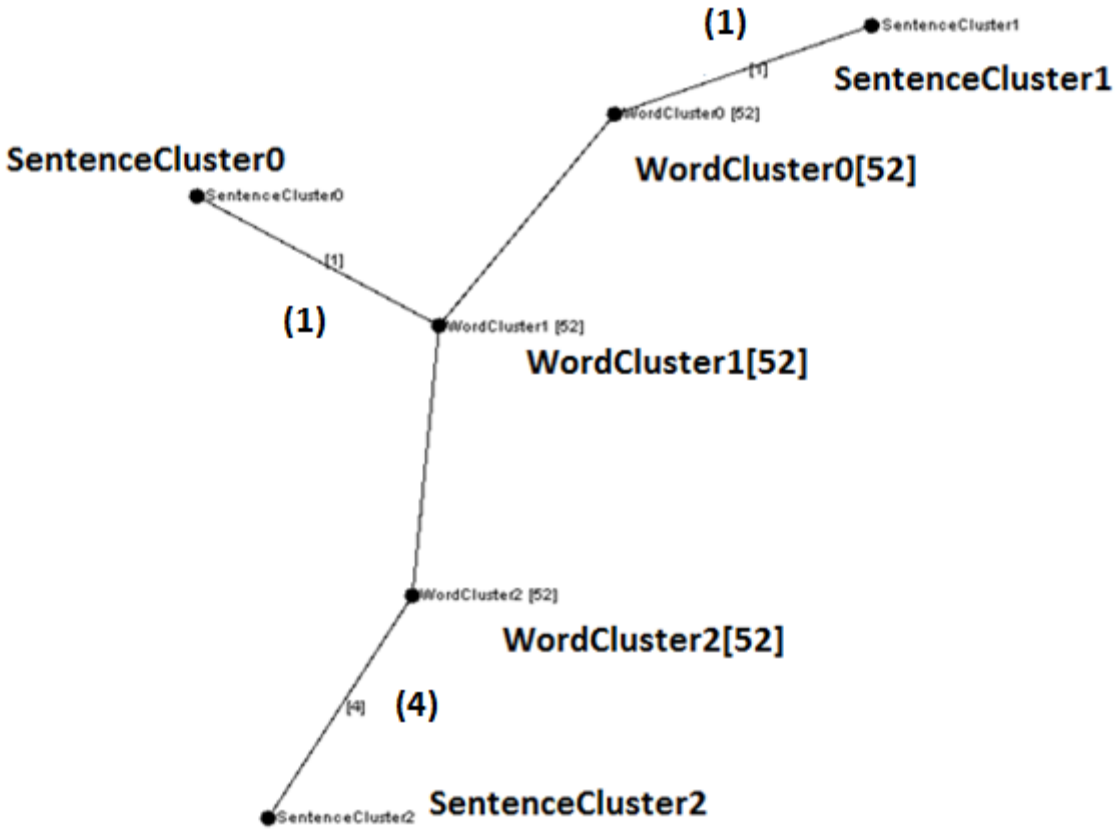

Рис. 2.13. Семантична модель тексту

Таблиця 2.8

Головний кластер-стема тексту

домініканці особняком Аквінський Діонісій Ансельм брабантский інтуїтивному раймонд думка шкіл Абеляр Теодорік філософ греко-візантійського схоластичної максим єдиний Бонавентура францисканці монастирях Дамаскін релігійних комплекс пересічних найбільш пізній розділ бога сповіді Боецій нікейську бернард Ареопагіт Больштедт університетах Клервоский теологією схоластически методом Гроссетест раціонал Кентерберійський світських вільям

Не дивлячись на те, що оброблений текст має гуманітарну спрямованість і, на відміну від проаналізованих раніше текстів відноситься до іншої предметної галузі, системі вдалося побудувати коректну семантичну модель тексту і коректний семантичний контур, що і вказує на властивість незалежності роботи системи від тематики оброблюваного тексту.

Сукупність описаних властивостей дає можливість стверджувати про реалізацію одного із важливіших критеріїв текстового автомата – адаптивності системи. Можливість незалежної обробки документів дозволяє динамічно формувати наповнення бази знань, розширюючи таким чином спектр знань моделі.

При цьому підході повністю виключені будь які допоміжні семантичні словники, на основі яких проводиться глибокий синтаксичний аналіз даних, що значним чином сприяє вирішенню алгоритмічних проблем підходу м’якого розуміння, що були описані раніше, роблячи більш доступним рівень семантики тексту для прикладної програмної обробки.

Хоча вже на цьому етапі роботи системи ми отримали непогані результати, перш ніж продовжити розробку моделі запит-відповідь на основі розробленої моделі, необхідно було впевнитися у її адекватності.

Тому першим кроком, який був виконаний із створеним підходом є розробка і застосування до нього набору тестів, спрямованих на експериментальну перевірку саме семантичних властивостей алгоритму, результати яких мають оцінити доцільність подальшого використання підходу у дисертаційній роботі та його потенціал як універсального інструменту кількісного уявлення семантичних властивостей тексту. Особливу увагу при перевірці адекватності підходу до побудови семантичної моделі текстового знання необхідно звернути на властивість об’єднувати терміни документу у семантичні мітки, і на залежність між кількістю семантичних зв’язків мітки і концентрацією термінів у ній, оскільки саме цей фактор є основним семантичним показником при автоматичній генерації відповіді.

2.2. Перевірка адекватності отриманої семантичної моделі документа

Перевірка адекватності розробленої семантичної моделі тексту і розробленої на її основі прикладної програмної системи викликає ряд складнощів. Справа в тому, що класичні семантичні мережі, які використовуються для онтологізації бази знань тестуються порівнянням із деяким «золотим стандартом», відносно якого і вираховуються критерії успішності побудови семантичної мережі [29]. Проте у нашому випадку такий підхід може використовуватися лише при безпосередньому застосуванні отриманої моделі (у нашому випадку – при генерації текстів), що відсуне питання семантичних властивостей моделі в цілому на другий план. Важливішим же питанням, що постало на цьому етапі роботи є перевірка залежності моделі саме від семантичних характеристик тексту, а не від його статистичного частотного портрету. Тому було прийнято рішення про проведення ряду тестів, спрямованих, в першу чергу, на спробу ввести систему в оману, будуючи моделі семантично невірних документів і порівнювати їх з отриманими результатами для коректних текстів.

Найбільш показовою перевіркою є зіставлення із автоматично згенерованим текстом, а саме – обробка тексту побудованого за допомогою автоматичного SEO-генератора тексту. Генератор тексту – комп'ютерна програма, що створює тексти, коректні з точки зору більшості мовних норм, але, як правило, позбавлені сенсу. [54] Іноді у читача що працює із згенерованим такою програмою текстом може скластися враження, що цей текст є осмисленим і семантично пов’язаним, особливо якщо текст має тематику, з якою читач слабо ознайомлений. Існують різні види генераторів тексту, що розрізняються своїми можливостями (наприклад, деякі з них можуть самостійно формувати нові слова). Генерація тексту шляхом складання з повністю випадкових слів дає сміттєвий результат, безглуздий для розуміння людиною, тому зазвичай застосовується генерація за вручну написаними фразами-шаблонами.

Для перевірки адекватності отриманої моделі система була перевірена на текстах, створених в результаті генерування на основі шаблонів синонімізації. Для

цього був використаний сервіс [55], що дозволяє створювати тексти за обраною тематикою та розміром.

Особливістю згенерованого документу є задовільна для систем статичної обробки тексту частотна картина. Наприклад, розглянемо частотні портрети двох текстів однакового розміру, які наведені в таблиці 2.9.

Таблиця 2.9

Частотні портрети згенерованого тексту і знання

Документ1		**Документ2**	
Слово	*Вага*	*Слово*	*Вага*
Радіовипромінювання	88	астероїдів	92
галактичної	87	існують	23
джерела	83	планетні	22
об'єктів	53	поверхню	18
оптичне	44	земної	18
званих	42	зіткнення	17
зірки	42	залишалися	18

Ґрунтуючись на частотних портретах документів, можна зробити висновок, що обидва тексти мають тематичну спрямованість «космос» і перший текст присвячений темі галактичного радіовипромінювання, а другий – астероїдів, однак це не зовсім так, оскільки перший текст є автоматично згенерованим текстом, а другий – семантично важливим знанням, що описує тему астероїдів. Приклад такого автоматично згенерованого тексту наведений у додатку 6. Не дивлячись на те, що стема «галакт» має вагу 14, що є досить великою для тексту такого об’єму, сам текст має слабкі смислові зв'язки між своїми частинами. З цього витикає гіпотеза, що якщо система орієнтується не тільки на синтаксичний і частотний, а й на семантичний рівень розуміння, то у процесі роботи із згенерованим текстом повинні відбутися зміни кількісних характеристик

семантичних властивостей створеної моделі документа (за умови однакової стилістичної спрямованості і близьких фізичних об’ємів). Задля її перевірки було проведено ряд експериментів із згенерованим текстом і порівняння отриманих результатів із схожими семантично нормальними текстами. Роздивимось деякі приклади отриманих результатів. Результат обробки документу із додатку 6 зображено на рис. 2.14 (проекція кластерів стем та речень) та на рис. 2.15 (результуюча семантична модель). Незважаючи на те, що не тільки обсяг автоматично згенерованого тексту співпадав з прикладом із додатку 3, наведеним раніше, а і його технічна спрямованість, та сама отримана семантична модель має значні яскраво виражені відмінності. Перш за все мова йде про кількість кластерів-стем, кластерів-речень що мають зв’язки із кластерами-стемами та вагу цих зв’язків – кількість «вільних» кластерів стем і їх загальна кількість значно перевищує аналогічні показники у еталонного тексту, а як було сказано раніше, саме ці данні і виражають кількісні характеристики семантичних властивостей тексту.

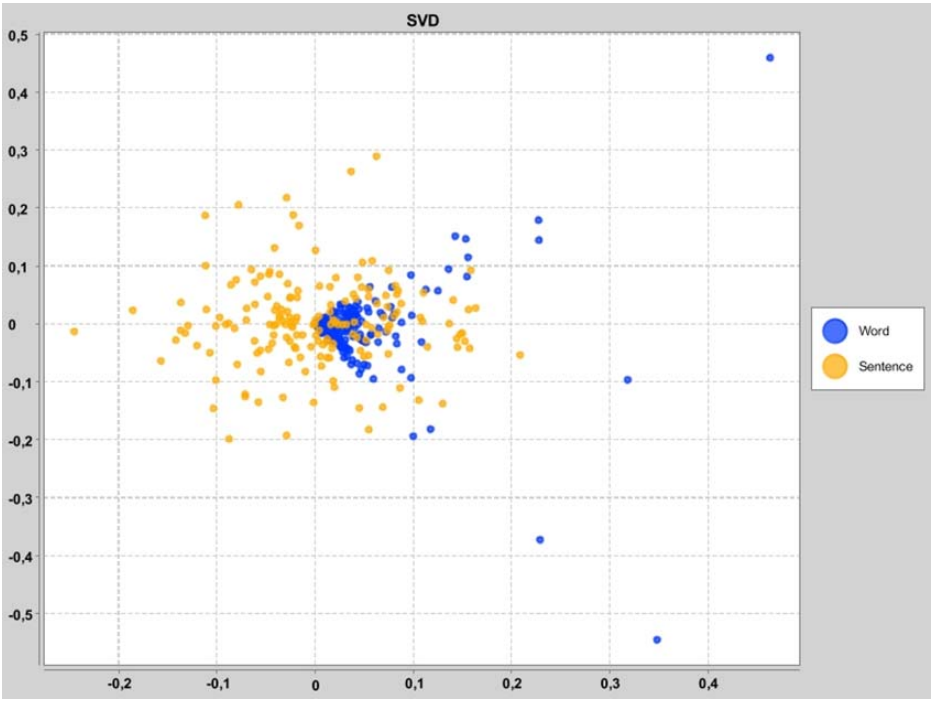

Рис. 2.14. Проекція сингулярного розкладання для тексту зі слабкими семантичними зв’язками: Word – координати-стеми, Sentence – координати-речення

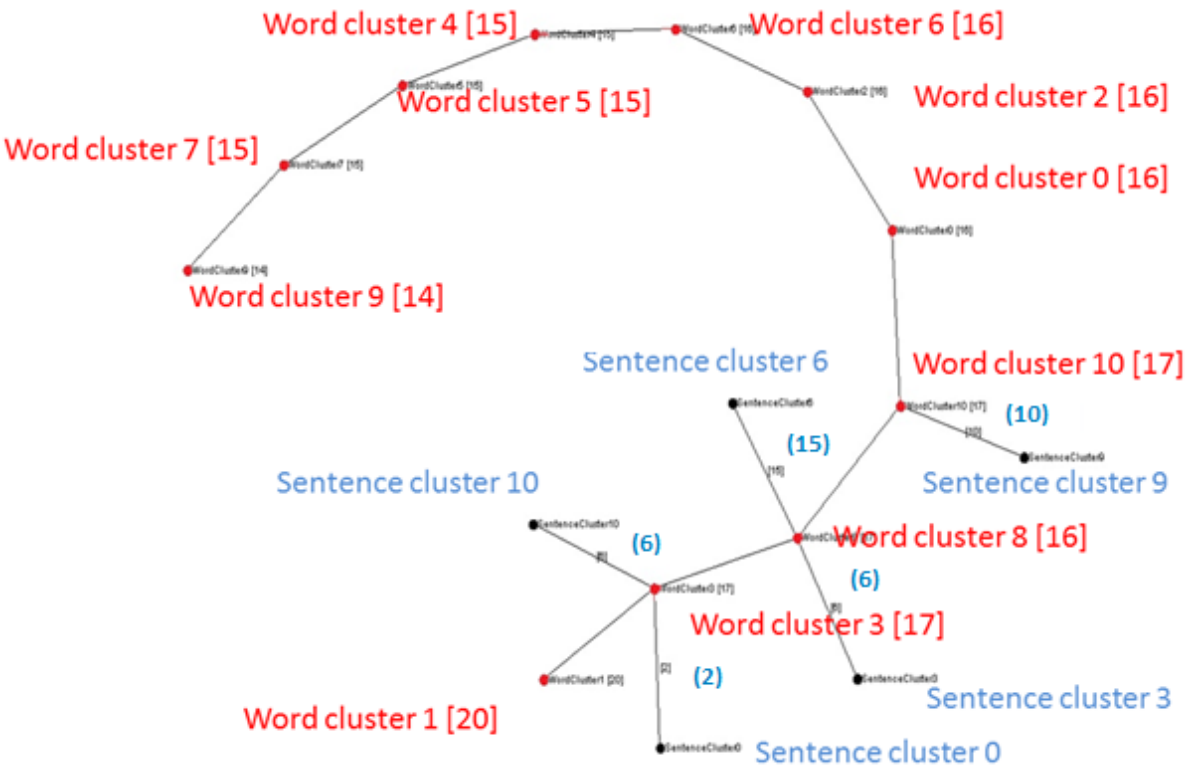

Рис. 2.15. Результат роботи системи для технічного тексту зі слабкими семантичними зв'язками – семантична модель документа: WordCluster відповідають кластерам-стемам, SentenceCluster відповідають кластерам-реченням

Для оцінки роботи моделей на згнерованих текстах було виконано попарне порівняння числових значень семантичних моделей згенерованого тексту і знання щодо значення середнього коефіцієнта семантичної значущості моделі μ (2.14):

$$\mu(D) = \frac{\sum_{i=0}^{N} \frac{f_{\min}(D_i, D_{i+1})}{f_{\max}(D_i, D_{i+1})}}{N},$$

$$f_{\min}(D_1, D_2) = \min\left\{\frac{\sum w_{SD_1}}{N_{D_1}}, \frac{\sum w_{SD_2}}{N_{D_2}}\right\}, \quad (2.14)$$

$$f_{\max}(D_1, D_2) = \max\left\{\frac{\sum w_{SD_1}}{N_{D_1}}, \frac{\sum w_{SD_2}}{N_{D_2}}\right\},$$

де $\sum w_{SD}$ – вага всіх семантичних зв'язків в моделі знання D, N_D – кількість кластерів в моделі знання D, $f_{\min}\ (D_1, D_2), f_{\max}\ (D_1, D_2)$ – функції, які повертають мінімальне і максимальне значення відносин, N – кількість експериментів. Дане значення характеризує розподіл семантичних міток документа щодо семантичних зв'язків, і показує залежність цього значення для пар текстів в корпусі знань $D = \{D_1 \dots D_N\}$. Коефіцієнт семантичної значущості моделі був розрахований для D_g – корпусу автоматично створених документів, D_I – корпусу знань и D_{gI} – корпусу, в якому зіставляється згенерований текст і знання між собою. Дослідження показали, що зміни коефіцієнта семантичної значущості для корпусу згенерованого тексту і корпусу знань при порівнянні текстів однакових розмірів практично не відбувається – $\mu_g(D_g) = 0{,}911;\ \mu_I(D_I) = 0{,}932$ відповідно. Однак, якщо порівнювати між собою знання і згенерований текст, очевидно значне падіння значення коефіцієнта до $\mu_{gI}(D_{gI}) = 0{,}335$. Це відбувається через те, що на відміну від згенерованого тексту, семантичні мітки знання мають значно більшу вагу і відповідно об'єднують більшу кількість семантично пов'язаних термінів, підвищуючи таким чином семантичну значущість моделі. Через значно більшу кількість кластерів-стем в моделях згенерованого тексту, підтверджується головна ідея використання моделі в задачах автоматичної обробки таких знань в системах генерації відповідей – терміни не об'єднуються в семантичні мітки, оскільки не існує ніякого семантичного зв'язку знань в початковому тексті, що наочно показує адекватність розробленої моделі.

Так, на рис. 2.16 зображена семантична модель згенерованого тексту гуманітарної спрямованості із додатку 7, що відповідає темі «fish-текст» і співвідноситься розміром із текстом з додатку 4. Не дивлячись на те, що головна стема «риб» має вагу 10, що навіть більше ніж у стемі «лософ» у еталонному тексті, семантична модель має значно більшу кількість кластерів-стем, оскільки у тексті відсутня будь-яка семантична основа, за даними якої можна було б провести адекватний латентно-семантичний аналіз.

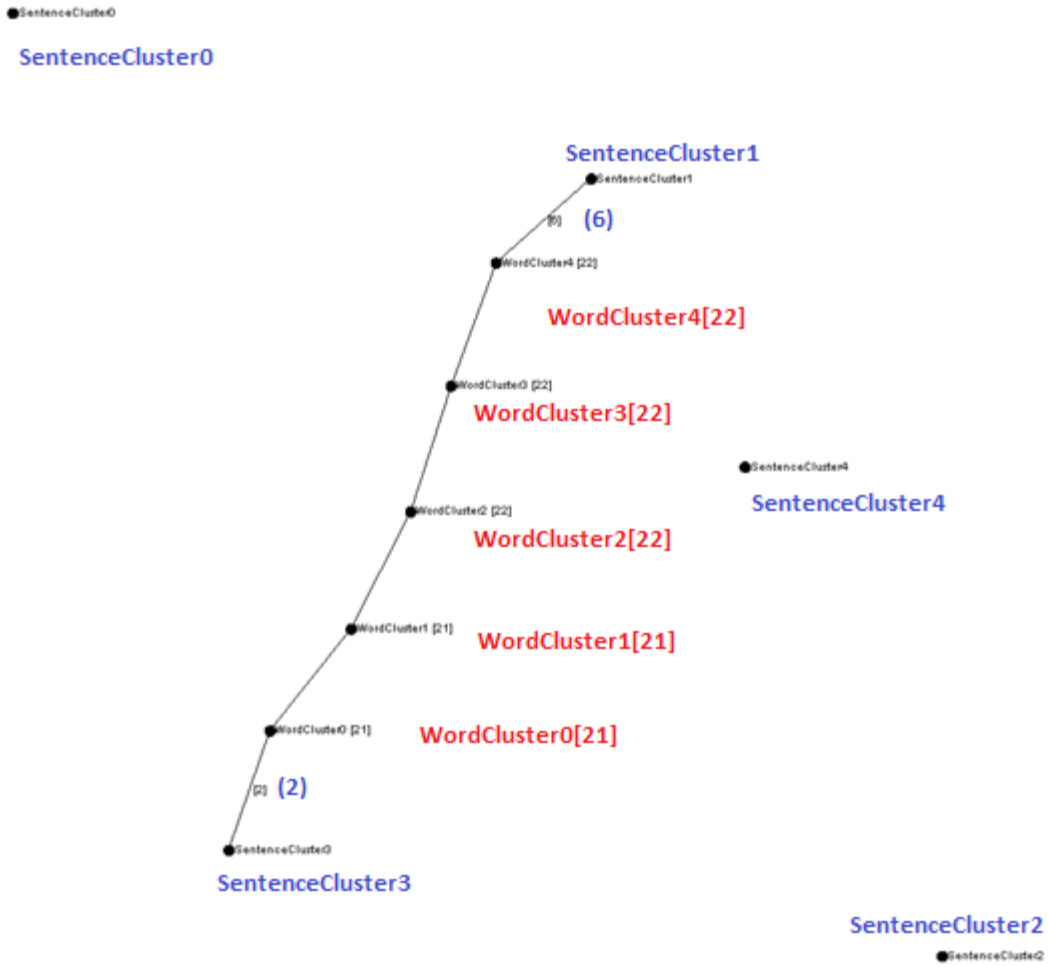

Рис. 2.16. Результат роботи системи для тексту гуманітарної спрямованості зі слабкими семантичними зв’язками

Особливо чітко можна побачити збільшення явища дисперсії у семантичній структурі при збільшенні об’єму згенерованого тексту – прикладом є семантична модель тексту з додатку 8, що зображена на рис. 2.17. Розмір і технічна спрямованість тексту співвідноситься із текстом з додатку 5, проте структура семантичної мережі явно вказує на особливість збільшення кількості вільних кластерів-стем у порівнянні із еталонним текстом.

Підвивши межу під вищесказаним можна стверджувати що результати, отримані у процесі емпіричних дослідів над структурою семантичної мережі дозволяють зробити припущення про залежність семантичних характеристик тексту від кількісних даних із семантичної моделі, що підтверджує її адекватність

на даному етапі перевірки і виключає залежність підходу від частотного портрету сем документу.

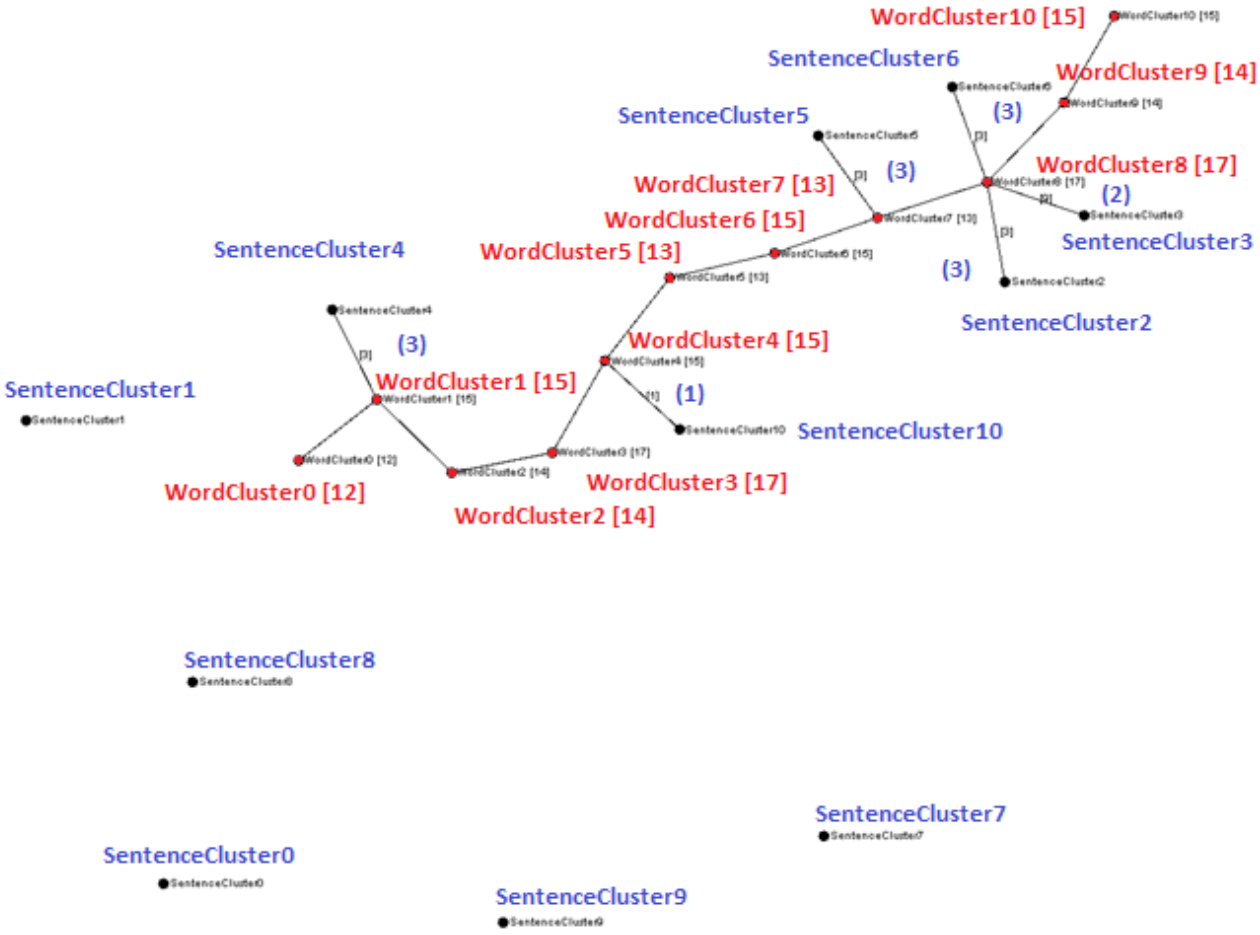

Рис. 2.17. Результат роботи системи для тексту великого об’єму зі слабкими семантичними зв’язками

Окремою і важливою перевіркою адекватності алгоритму побудови семантичної моделі тексту є оцінка сенсової місткості семантичних міток документу. Гіпотеза, сформульована у першому розділі, має на увазі, що утворені семантичні мітки документа мають найбільшу кількість перетинів із семантичними контурами тоді, коли вони містять найбільшу кількість семантично значимих стем у своєму складі. Якщо це так, то отримана модель спроможна автоматично складати семантичні тезауруси понять, встановлюючи відповідності між наборами речень та семантичною міткою документу, реалізуючи при цьому схему розуміння абстрактного поняття завдяки набору значних слів. Розглянемо

перший прилад обробки технічного тексту з додатку 9, тематика якого стосується теми космосу, а саме – космічного реліктового випромінювання.

Таблиця 2.10

Приклади кластерів-стем тексту «Реліктове випромінювання»

WORDCLUSTER 0
стало передбачене вибуху Всесвіту поляризацією повинна електромагнітне який ізотропності Експериментально пов'язаної перестали етапів плазму отримали Робертом час лінії Вільного могли перестали космологічним Великого атоми частинками враховує викликана основного процес мікрохвильове зміщенням через Реліктове результаті викликало рекомбінацію Дікке гравітаційних анізотропії речовиною Коли розсіюватися ефективну причиною Зельдовичем цього взаємодіяли фонове об'єкт супутникових енергією називаються ається дуже зарядах поширення Таким сповільнилися наслідок розширення гелію водню заряджених Сюняева щільністю області рання встановити Сюняева з'являтися Сюняева іонізованої простору нейтральних фонове постійно знову червоним температурою теорії спектром фотонів дипольні гарячу складається найбільш середовищі сповільнилися випромінювання компоненти Видимий складають Землі ультрафіолетовим перестали вимірювання відповідав електронів буде компоненти Сюняева існування володіли космічне Більш можливість Сюняева подальшого сьогоднішній основи флуктуацій високої фонове випромінюються розсіювання
WORDCLUSTER 3
трапилося максимальна оцінки перевірити впливом чином Виявилося менш Згідно захисту вивчаючи Згідно максимум дзвінок Походження точністю Виділяються захисту місце захисту Яковом неоднорідну Згідно вивчаючи сфера захисту знизилася менш явище охолодження охолодження дзвінок іншими підпорядковується іншими амплітуда віці теплового калібрування наявність неоднорідну ознакою протягом підпорядковується Висновок теплового кількість згідно протягом вивчаючи точністю менше кількість Походження вивчаючи декількома захисту протягом ознакою калібрування довгий широко вивчаючи захисту Походження Обробка калібрування вивчаючи декількома іншими може вибиті абсолютно іншими дзвінок охолодження Яковом максимум Обробка Висновок дзвінок телескопом точністю широко Згідно іншими калібрування телескопом вивчаючи підтвердили Виділяються охолодження вивчаючи порядку молекулами Планка Згідно іншими вивчаючи Висновок охолодження точністю перестали захисту сфера охолодження менш дзвінок ознакою Термін

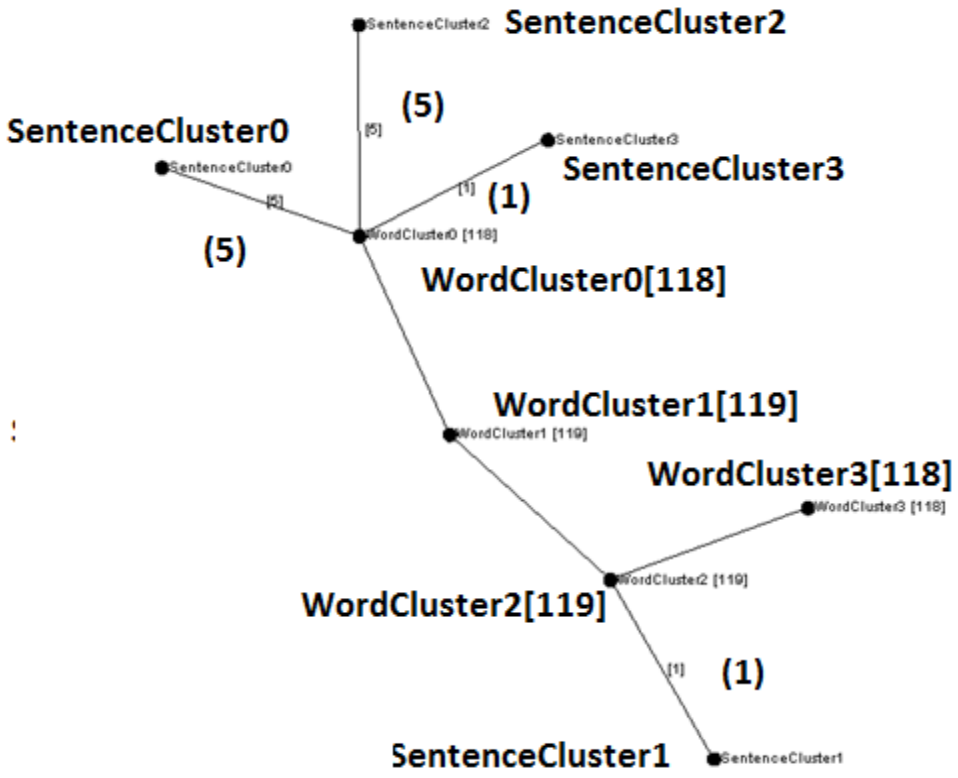

Рис. 2.18. Семантична модель тексту «Реліктове випромінювання»

На рис. 2.18 зображена семантична модель тексту, на основі якої видно, що кластер-стема із номером 0 має найбільшу сумарну вагу зв’язків із семантичними контурами ніж інші, а кластер-стема із номером 3 не має перетинів із семантичними контурами стем. Тоді, відповідно до гіпотези що перевіряється, нульовий кластер має бути найінформативнішим і однозначно вказувати своїм змістом на тематику тексту, а третій кластер – мати у своєму складі неінформативні і семантично неоднозначні стеми. Зміст отриманих кластерів наведено у таблиці 2.10. Дійсно, порівнявши кластери між собою можна побачити значущу різницю – нульовий кластер містить стеми (всесвіт, реліктове гравітаційних, супутникових, випромінювання, космічне та ін.), що не тільки пов’язані із головною тематикою тексту, а і семантично пов’язані між собою галузевою спрямованістю. З іншого боку, кластер що не потрапив у семантичну мережу не представляє ніякої сенсової цінності, на весь кластер присутня лише одна стема що має відношення до теми космосу (телескоп), інші – являють собою семантично узагальнені слова, що не визначають ніякої предметної галузі.

Описаний результат обмежується виключно ступенем формалізованості вхідного тексту, що повинен дозволяти складати базовий частотний портрет документу, і не залежить від предметної галузі тексту що обробляється. На рис. 2.19 зображена семантична модель тексту гуманітарної спрямованості теми «знання і мова», а у таблиці 2.11 – приклади стем із максимальною і мінімальною вагами зв’язку із відповідними семантичними контурами. Наведений текст знаходиться у додатку 10 і має достатню ступень формалізованості, тож описані стеми мають відповідати наведеній гіпотезі навіть за умови повної зміни тематики і спрямованості тексту.

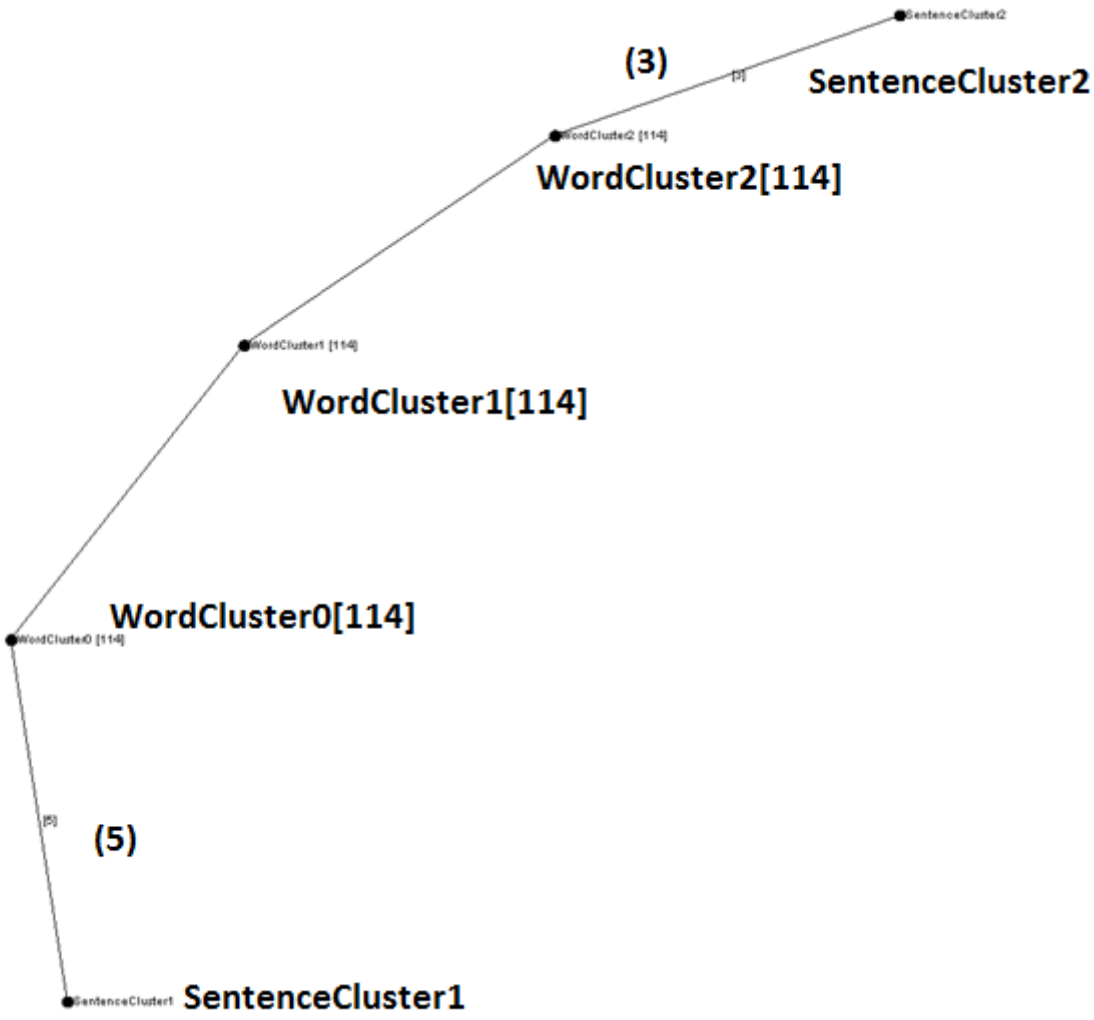

Рис.2.19. Семантична модель тексту «Знання і мова»

Таблиця 2.11

Приклади кластерів-стем тексту «Знання і мова»

WORDCLUSTER 0
навчання ставлення говорити значення Лінгвістичні Останнє фіксується випадку заповітних мозку фіксується людина програмного дуже разі методологічних спрямування є структура Лінгвістичні зовнішньої зовнішньої дослідження наявних Лінгвістичні Особливий приходять технічними розвиток заповітних забезпечення Особливий заповітних напрямки зовнішньої представляє зовнішньої приходять дуже визначення підхід фіксується природного разі Особливий дуже заповітних приходять здатні зовнішньої Особливий заповітних Лінгвістичні випадку Процес усвідомлення вдалося фіксується мовою приходять носіями допомогою результаті недостатньо аналізу звітних Виявлення своєрідного зовнішньої Останнє труднощі конкретних Особливий інтереси вельми вдалося власне дуже зовнішньої людської власне інструменту Лінгвістичні зовнішньої заповітних доступності Лінгвістичні цього заповітних приходять штучного вельми разі створення вдалося більш можливостей сьогодні оволодіння дуже вдосконаленням визначає зовнішній недостатньо фіксується проходять вдалося виступає вдалося Саме зовнішньої
WORDCLUSTER 1
змусив банків фреймів звичайний Реалізація Поява обчислюваності згоди засвоєння єдиного Переконливе обчислюваності Поява процедури суті єдиного Реалізація Конструювання нерозривно означає логічної Тому може фігур єдиного засвоєння є Реалізація Поява фігур Поява потім Переконливе Приступаючи означає потім потім Переконливе обчислюваності більше означає історично потім напрямки є можуть бачиться Поява Реалізація історично фігур Поява Реалізація логічної оди ого може Переконливе є згоди Зауважимо засвоєння фігур єдиного Конструювання є Зауважимо може можуть суті Поява згоди різної Зауважимо Поява є згоди Поява згоди потім обчислюваності фігур бачиться потім більше Переконливе можуть напряму згоди є використовуваних потім бачиться може єдиного Зауважимо може Конструювання обчислюваності потім Конструювання Зауважимо може означає потім фігур засвоєння Переконливе єдиного Переконливе напрямки символів бачиться більше

Наведена семантична модель вказує на те, що кластер із нульовим номером є головною семантичною міткою документа, а його склад (лінгвістичні, мовою, програмного та ін.) на відміну від кластеру із номером 1, має куди більш близьку до теми тексту семантику. До того ж, на прикладі цього тексту добре видно, що зменшення об’єму вхідного тексту негативно впливає на об’єднання даних у

кластери, а саме призводить до зменшення семантичної сили головної мітки документу, що вказує на залежність якості побудови семантичного градієнту від ступеня інформативності тексту що обробляється. Описана поведінка і властивості системи цілком підпадають під модель опису абстрактного поняття – наприклад кластери, наведені в таб. 2.12 однозначно вказують своїм змістом як на семантичну мітку документу, так і на кластер без перетинів із семантичними контурами стем. Це можна зрозуміти без аналізу структури семантичної мережі – очевидно, що перший кластер має більше стем пов'язаних із галуззю інформаційних технологій ніж другий, і вказує своїм змістом на тему «програмування», тоді як другий кластер увібрав у себе узагальнені поняття, за якими неможливо встановити початкову тематику документа. Текст дійсно стосується теми «модульне програмування» (додаток 11), а його семантична модель (рис. 2.20) підтверджує здогадку про те, що кластер із номером 1 містить найбільшу вагу перетинів із семантичними контурами стем.

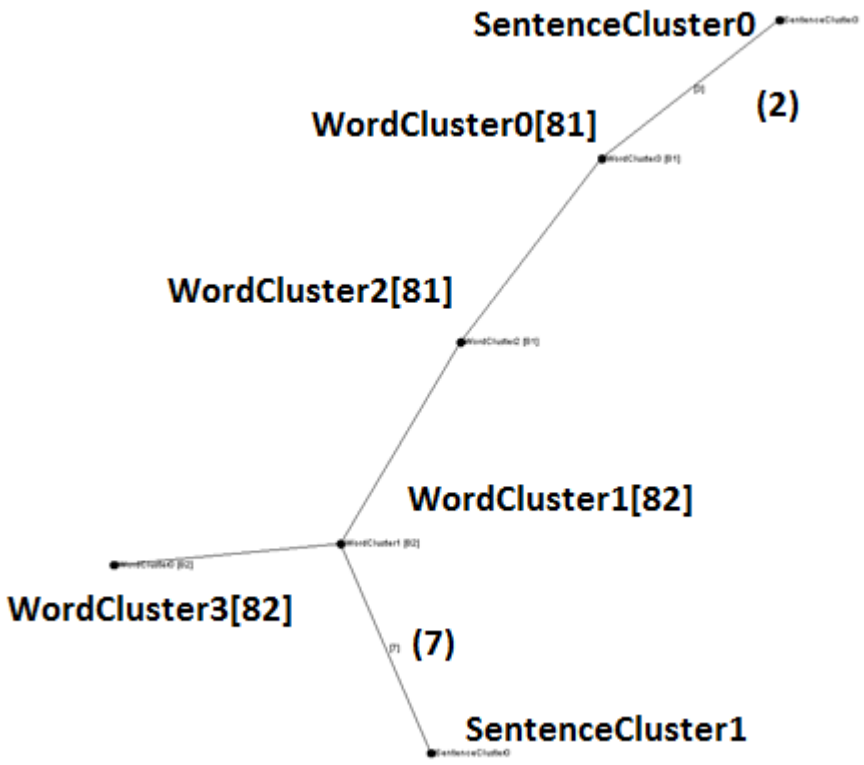

Рис. 2.20. Семантична модель тексту «Модульне програмування»

Таблиця 2.12

Приклади кластерів-стем тексту «Модульне програмування»

WORDCLUSTER 1
готової кількість незалежних чином самого представляють кінцевого інших визнач представлені програмування представлені структура яких сегменті Використання створенні кінцевого окремого Термін кінцевого реалізована конструкції є важлива спростити файлу заміни повинен сукупності невеликих заміни частини можуть Паскаля надають всього дозволяє процедуру виконувані завдання основні Принцип бібліотеки одиниці групами будь-яка компонентів сукупності пакети засобом наприклад правилам Модульне інструменти функцій рівня доступними кожного вихідним кінцевого комплексів сукупності створюваних сукупності сегменті представляють розробки представляють більш можливості організація функціонально представляють Така константи зміни представляють підзадачі сукупності кінцевого
WORDCLUSTER 2
синтаксичні розуміли з'явилася спрощення існували Зручність багатьох високу Вперше Цілком багатьох високу замкнутим Вперше замкнутим реалізацію мінімальних сильно багатьох нову багатьох виділяється замкнутим Вирішити детально Вирішити формально Зручність Вирішити грати детально розбитті високу вийшла включена Цілком передбачають середовищі може розташовуються яку багатьох багатьох грати інформації виділяється приховування мінімальних об'єднання існували високу існували виділяється високу мінімальних багатьох сильно детально висунув виділяється розбитті глобальних Цілком детально Цілком розташовуються існували виділяється яку забезпечує сильно замкнутим грати якусь Оберон високу Зручність замкнутим названа Парнас

Описана оцінка сенсової місткості семантичних міток документу не обмежується кількома тестами, і була проведена для окремої тестової бази знань, оскільки саме вона вказує на залежність структури семантичної мережі і семантичних властивостей документу. Для цього було перевірено 65 семантичних моделей, для яких було розраховано коефіцієнт середньої семантичної ємності ε (2.15) по кожній з тем – економіка, філософія, інформаційні технології і астрономія:

$$\varepsilon = \frac{\sum_{i=0}^{N} \frac{N_H}{N_L}}{N} \qquad (2.15)$$

де N_H – кількість термінів, семантично пов'язаних з тематикою знання в кластері-стемі з найбільшою загальною вагою зв'язків з кластерами-реченнями (сильні кластери), N_H – кількість термінів, семантично пов'язаних з тематикою знання в кластері-стемі з нульовою загальною вагою зв'язків з кластерами-реченнями (слабкі кластери), N – загальна кількість проведених тестів для заданої теми.

Даний коефіцієнт показує, у скільки разів щільність семантично важливих термінів в сильних кластерах перевищує щільність семантично важливих термінів в слабких кластерах.

Отримані результати розрахунку сенсової місткості для кожного тематичного набору текстів зображені на рис. 2.21 – 2.24.

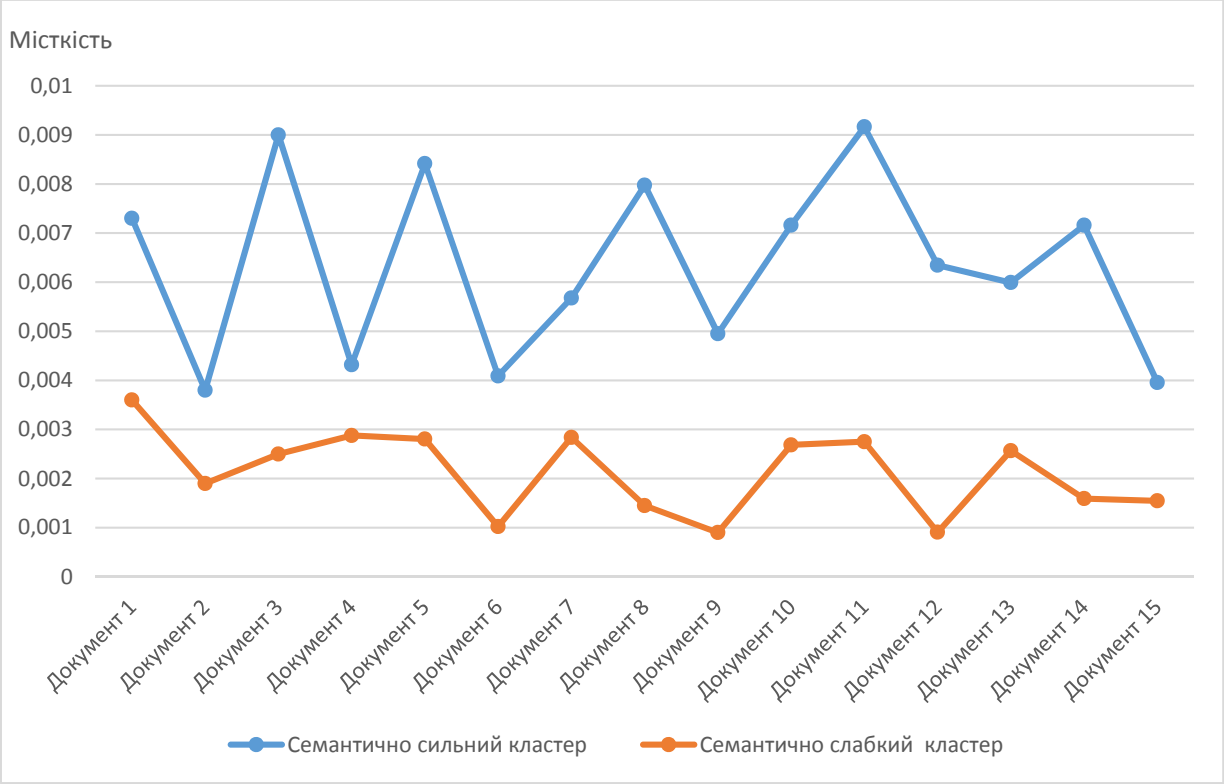

Рис. 2.21. Розподіл значень сенсової місткості для тематики «економіка»

Отримані графіки показують, що у проведених тестах кількість семантично значимих термінів у семантично сильних кластерах значно перевищує кількість термінів у семантично слабких кластерах незалежно від кількості, розміру та

тематичної спрямованості документів – для тематики «економіка» (15 документів) кількість термінів у сильних кластерах у середньому у 2,98 рази більша ніж у слабких, для тематики філософія (21 документ) – у 3,375 рази, для тематики «астрономія» (24 документа) – у 3,473 рази, для тематики «інформаційні технології» (5 документів) – у 4,44 рази.

При проведенні досліджень не враховувалась вага стем або їх будь-яке синтаксичне співвідношення із текстом – пов’язані терміни оцінювалися лише з точки зору належності до тематики документу, тому отримані результати вказують на цілком адекватне формування семантичних міток документу – кількість семантично значимих термінів у кластері-стемі прямо пропорційна до кількості та ваги пов’язаних із ним семантичних контурів речень у побудованих моделях документів, що доводить залежність структури семантичної мережі саме від семантичної складової тексту.

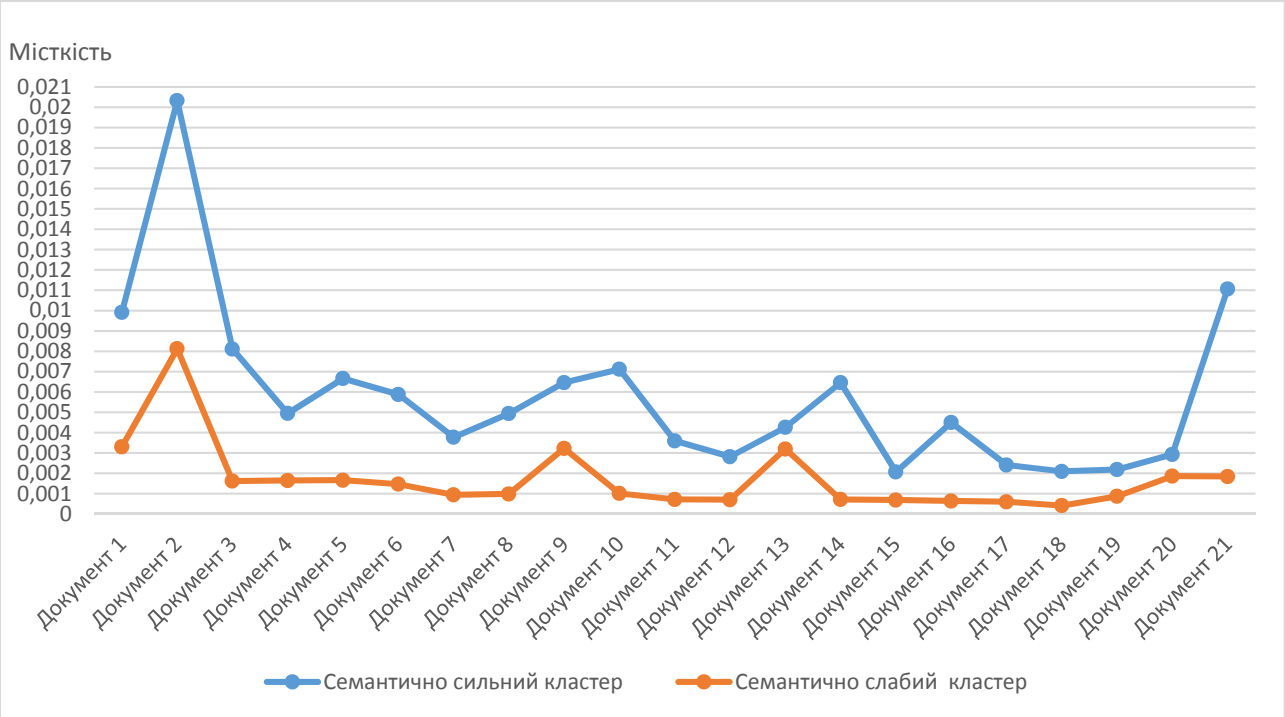

Рис. 2.22. Розподіл значень сенсової місткості для тематики «філософія»

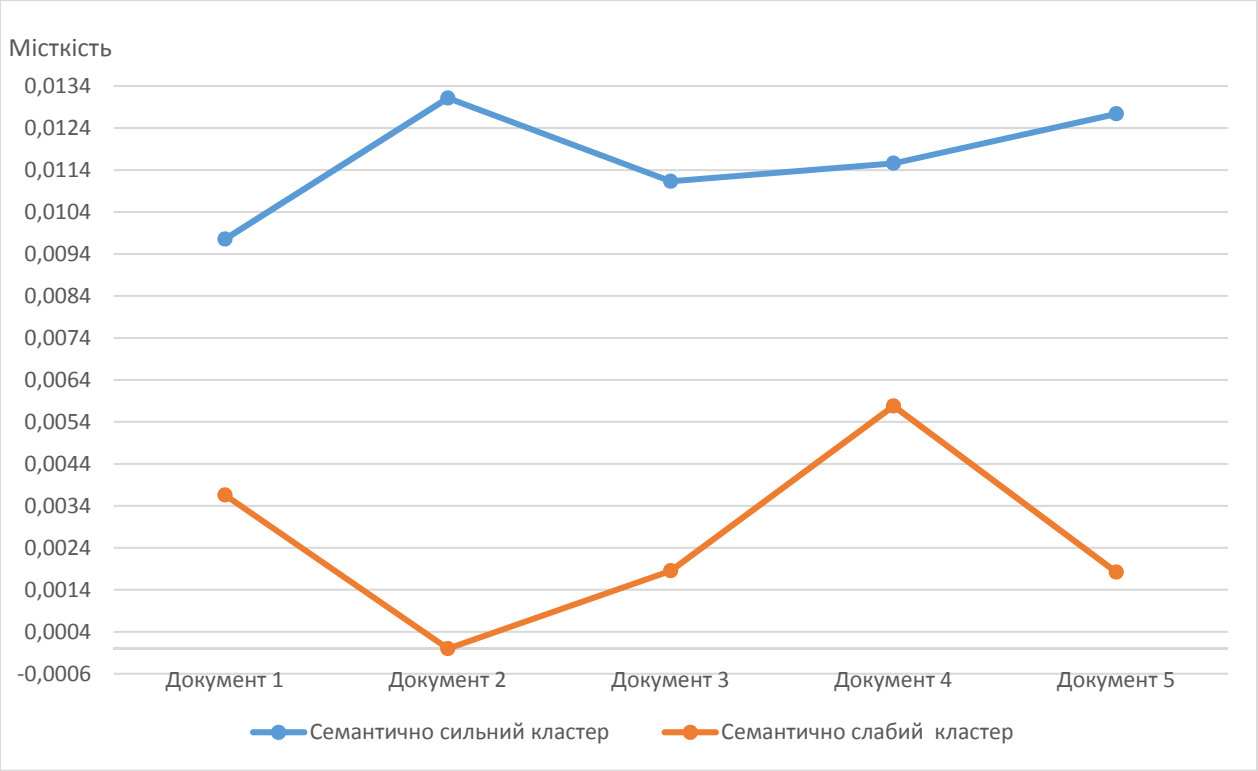

Рис. 2.23. Розподіл значень сенсової місткості для тематики «астрономія»

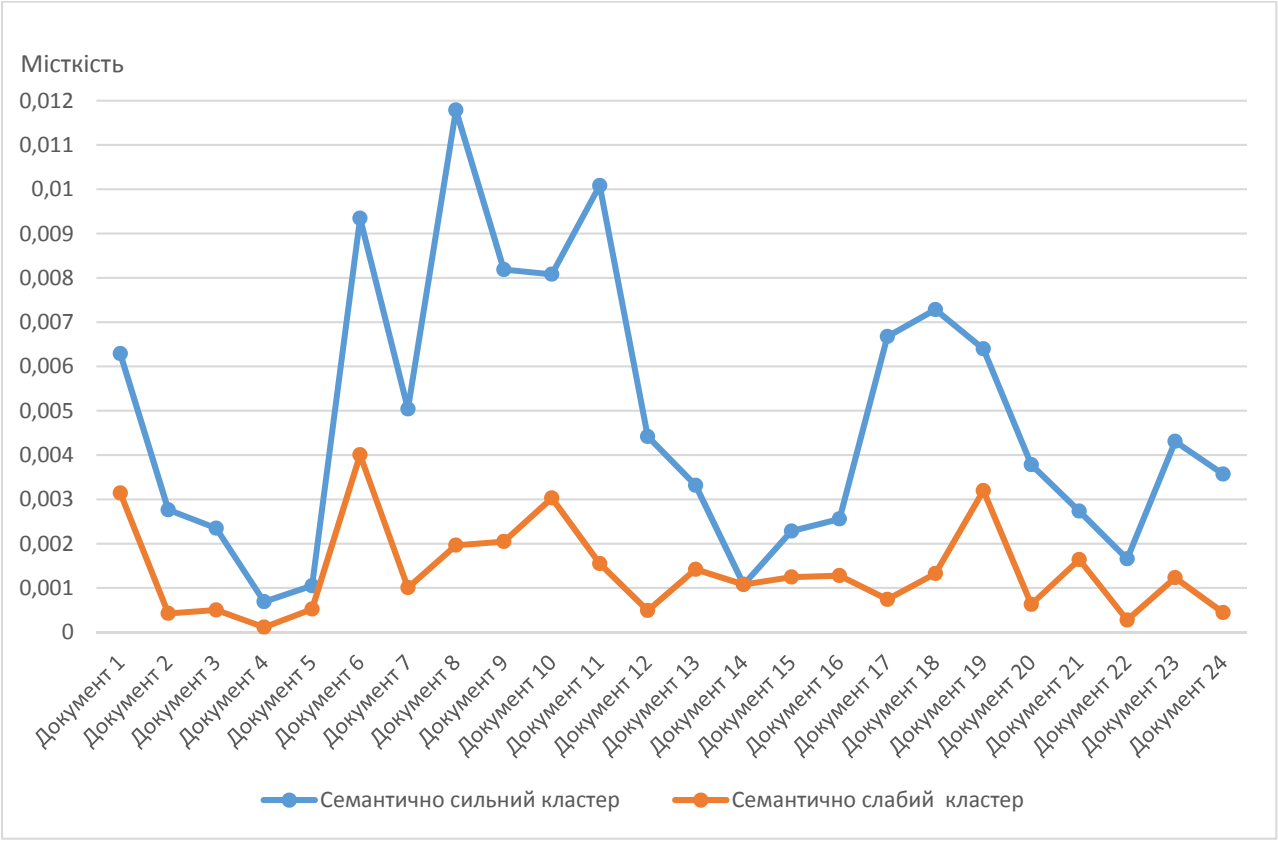

Рис. 2.24. Розподіл значень сенсової місткості для тематики «інформаційні технології»

Висновки до розділу 2

В другому розділі розкрита структура основного компоненту глибинного семантичного рівня – програмної семантичної моделі документа, яка базується на структурі гібридної семантичної мережі. Необхідність і важливість цього етапу виходить насамперед із аналізу існуючих аналогів і алгоритмічних підходів до побудови семантичних мереж документа – усі вони або базуються на словниках, або не пророблені для флективно багатих мовних груп, що суперечить головним цілям цього дослідження. Розроблений підхід базується на алгоритмі латентно-семантичного аналізу що дозволяє знаходити семантичні відповідності на основі вагових характеристик тексту і роботі з проекціями координат для базових текстових одиниць на двомірній площині. Додатково до основного алгоритму, у розділі розглянуті алгоритми автоматичного визначення частин мови для заданих слів та альтернативний розроблений підхід пошуку стем у тексті [56].

Використання такого підходу до роботи із семантичними характеристиками тексту є інноваційним не тільки тому що сфера застосування латентно-семантичного аналізу в першу чергу стосується задач класифікації документів, тоді як у нашій моделі, його використання було змінено, і ми наближаємо не документ до терміну, а речення з документів до термінів документа, а і тому що алгоритм поєднує у собі багато специфічних додаткових етапів, що є нетиповими для підходів побудови семантичної моделі документу. Мова йде про використання алгоритмів кластеризації, із відповідними методами для визначення необхідних для неї параметрів, алгоритмів синтаксичного, морфологічно і просторового аналізу даних [56].

Для оцінки отриманих за допомогою семантичної моделі результатів була сформульована та виконана поетапна перевірка семантичних властивостей розробленої моделі, для визначення залежностей саме від семантичних, а не від частотних характеристик документу.

Виконаний ряд порівняльних тестів довів що отримана модель є адекватною і може застосовуватися задля безсловникового універсального підходу до кількісного опису семантичних властивостей тексту. Встановлено, що

через значно більшу кількість кластерів-стем в моделях хаотично згенерованого тексту підтверджується головна гіпотеза використання семантичної моделі в задачах автоматичної обробки знань в системах генерації відповідей – терміни не об'єднуються в семантичні мітки, оскільки, не існує ніякого семантичного зв'язку знань в початковому тексті, що наочно показує адекватність розробленої моделі.

Було встановлено, що кількість семантично значимих термінів у кластері-стемі прямо пропорційна до кількості та ваги пов’язаних із ним семантичних контурів речень у побудованих моделях документів, що доводить залежність структури семантичної мережі саме від семантичної складової тексту [57]. Завдяки цьому стає можливим подолати обмеження тлумачно-комбінаторного словника, що накладав строгі рамки використання лінгвістичних знань на створення прикладних програмних моделей, і застосувати парадигми м’якого розуміння Леонтьевої для автоматичної генерації текстів, автоматичної класифікацій документів і інших завдань автоматичної обробки текстів [57].

Лінгвістичні знання, що були застосовані у ході роботи є кінцевими і відносно невеликими, до того ж використовуються одноразово і не вимагають подальшої підтримки. Додатковою перевагою створеного підходу є відповідність по-перше – загальний рівневій структурі текстового автомату, що має на увазі використання створеного програмного продукту як окремого інструменту аналізу текстових даних, а по-друге – відповідність конкретній архітектурі текстового автомата, створеної в рамках нашої роботи .

Подальша задача, що виникає перед нами на даному етапі складається у безпосередньому застосуванні отриманої моделі в рамках складної інтелектуальної системи обробки текстів, результати роботи яких і стануть основними для оцінки роботи і адекватності створеної моделі обробки тексту. [58].

Основні положення цього розділу викладені у публікаціях автора [56, 57, 58]

РОЗДІЛ 3. МОДЕЛЬ ГЕНЕРАЦІЇ ВІДПОВІДЕЙ У ПОШУКОВИХ СИСТЕМАХ НА ОСНОВІ НЕСТРУКТУРОВАНОЇ БАЗИ ЗНАНЬ

Семантична модель, що була отримана на попередньому етапі дослідження надає всі необхідні дані для реалізації пошукової моделі, суть якої можна сформулювати наступним чином. Нехай $D = \{d_1, d_2 \ \dots \ d_n\}$ – множина знань, розміром n, що складається з текстів $\{d_1, d_2 \ \dots \ d_n\}$, T – множина термінів, що являє собою запит користувача. Тоді, множина відповідей системи представлена матрицею M (3.1):

$$M = \begin{vmatrix} f_{1,1}(t_1, d_1) & & f_{1,n}(t_n, d_1) \\ \vdots & \cdots & \vdots \\ f_{m,1}(t_1, d_m) & & f_{m,n}(t_n, d_m) \end{vmatrix},$$

$$f(t,d) = \begin{cases} 1, t \in T \wedge t \in V(d) \\ 0, t \notin T \wedge t \notin V(d) \end{cases}, \qquad (3.1)$$

$$T \subset \{t_1 \dots t_n\},$$

де $V(d)$ – множина сильних кластерів-стем для документа d – таких, що мають найбільшу загальну вагу зв’язків із кластерами речень, $f(t, d)$ – повертає 1, якщо термін t належить множинам T та $V(d)$, або 0 у іншому випадку.

Але, перш ніж приступити до безпосередньої імплементації моделі, необхідно визначити вимоги до нашого додатку і його термінологію. У сучасній літературі [59] під системою запит – відповідь часто мають на увазі інформаційну систему, яка є гібридом пошукових, довідкових та інтелектуальних компонентів, що використовує природно-мовний інтерфейс. На вхід такої системи подається запит, сформульований на природній мові, який аналізується з використанням методів програмної обробки текстової інформації, після чого генерується семантично пов’язана відповідь. В якості базового підходу до задачі пошуку відповіді на питання зазвичай застосовується наступна схема: спочатку система тим чи іншим чином (наприклад, пошуком за ключовими словами) відбирає документи, що містять інформацію, пов'язану з поставленим питанням, потім

фільтрує їх, виділяючи окремі текстові фрагменти, які потенційно містять найбільш релевантну відповідь, після чого з відібраних фрагментів модуль генерації синтезує відповідь на питання. Наразі існують дві головні сучасні теоретичні парадигми реалізації описаного підходу [59] – моделі на основі вилучення інформації (IR-based QA) і моделі, заснованої на знаннях (Knowledge-based QA)

Парадигма IR-based QA [59] пропонує 3 послідовні фази до формування відповіді, першою з яких є обробка питання. Мета цієї фази – витяг усієї можливої інформації, що міститься в питанні. З питання витягується тип іменованої сутності, який визначає тип відповіді, що дозволяє сконцентруватися на пошуку конкретної сутності в наборі документів, підготовлених для формування відповіді по знайденому типу. Класифікатор питань може бути побудований на основі рукописних правил (що використовують регулярні вирази), машинному навчанні або комбінувати обидва підходи. Наступним кроком модель вибирає ключові слова, за якими буде здійснено запит для пошуку документів, що потенційно містять відповідь. Отриманий запит може бути сформульований по-іншому, замінюючи ключові слова синонімами або розширюватися словами-параметрами. Далі формується фокус питання, що представляє собою деяку частину цього питання, замінивши яку відповіддю, результат можна представити у вигляді повноцінного речення.

Другою фазою роботи системи є формування релевантних фрагментів. Запит, сформований на попередньому етапі, надходить на вхід моделям вилучення інформації, таким як пошукові системи або набір індексованих документів. Результуючі документи проходять етап вилучення ряду релевантних фрагментів, що потенційно містять відповідь. Одиницею поділу фрагмента може виступати абзац, але допускаються і більш дрібні сегменти – речення. Фрагменти ранжуються за такими властивостями, як кількість іменованих сутностей, число ключових слів, близькість ключових слів, найбільша безперервна послідовність ключових слів, тощо.

Завершальною фазою роботи є безпосередньо формування відповіді. Для цього завдання може бути застосовано два типи алгоритмів витягів відповідей: витяг шаблону за типом відповіді і збір відповіді з N-грам. У першому випадку за типом відповіді витягується іменована сутність, що їй відповідає. Для типів, у яких не існує відповідності, наприклад, «дія» або «опис», формуються вручну регулярні вирази. У другому алгоритмі – призначається вага у N-грамі, що еквівалентна кількості фрагментів, в яких N-грам зустрівся. Далі виконується фільтрація і збір відповідей до тих пір, поки не залишиться єдина відповідь-кандидат.

Альтернативним підходом до створення моделей запит-відповідь є Knowledge-based QA [59]. Питання, що ставляться моделі, відображаються запитом до структурованої бази знань. Подання послідовності тексту в логічну форму здійснюється за допомогою семантико-синтаксичного аналізу. Зазвичай, відображення в логічну форму найчастіше представляється у вигляді числення предикатів або на мові запитів. Структурою зберігання таких даних може виступати як реляційна БД, так і RDF-триплети. У найпростішому випадку, завданням формування відповіді на питання про об'єкт є пошук відсутнього аргументу триплета. Дана парадигма пропонує формувати рукописні правила вилучення відносин що часто зустрічаються в питаннях, які надійшли у систему. Для цього використовують два підходи опрацювання питань класифікатором: заснований на навчанні з учителем і заснований на частковому навчанні. У першому випадку в системі є набір тренувальних даних, де кожен екземпляр представлений у вигляді еталонного питання з певною логічною формою, що є представником деякої підмножини. Даний набір використовується моделлю для відображення нових питань на вже існуючі логічні форми. Процес відображення являє собою зіставлення дерева залежного від розібраного питання і подальше його зведення до логічної форми.

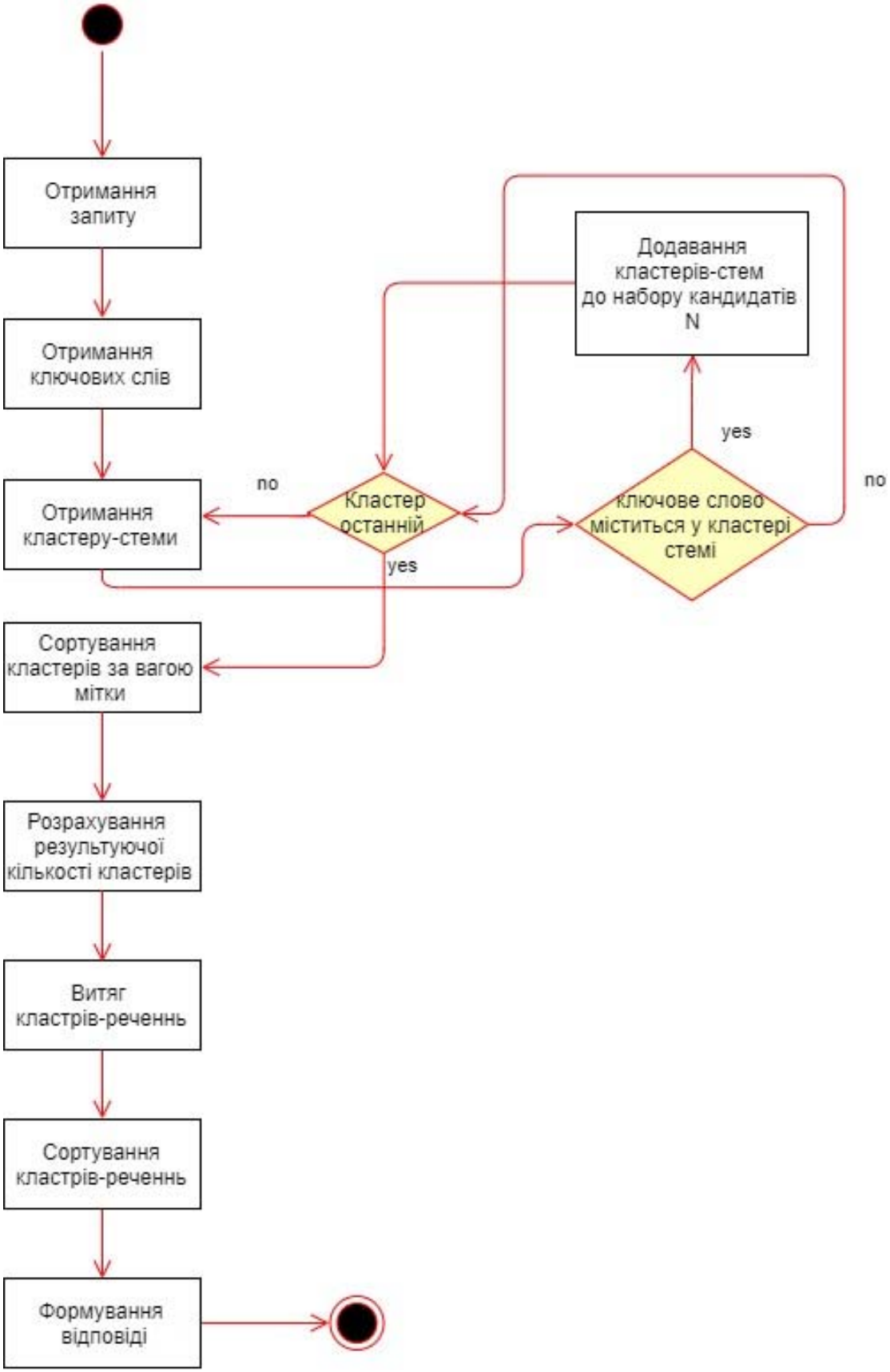

Рис. 3.1. Структура роботи моделі генерації відповідей

У разі класифікатора питань, заснованого на частковому навчанні з учителем, модель намагається подолати обмеження бази знань, що покриває всілякі форми питань. Для цього більшість методів відображення запитань до деяких логічних форм застосовують методи текстових скорочень. Таким чином, виконується зіставлення предиката і його параметрів з унікальним артефактом певного джерела знань.

Загальна структура роботи розробленої моделі генерації відповідей зображена на рис.3.1. Модель відштовхується від парадигми IR-based QA архітектури, і має на увазі етап аналізу і розбору запиту, обчислення релевантних фрагментів з бази знань і безпосереднє формування відповіді. Весь життєвий цикл одного такту роботи системи зав'язаний на роботу з семантичною моделлю документа що була описана раніше, обхід і аналіз якої дозволяє включати в релевантну відповідь семантично пов'язані фрагменти із запитом, не дивлячись на відсутність прямих синтаксичних зв'язків. Розглянемо процес роботи системи докладніше.

Першим кроком алгоритму є отримання і обробка запиту користувача, що є типовою поведінкою моделей запит-відповідь. У нашому випадку, під запитом розуміється набір ключових слів, інформація про які цікавить користувача, розділених між собою знаками пробілу. Наприклад «класи програмування розробка», «космос комети», «телескопи», «теорія економіки». Форма та відмінок слів що складає запит не фіксовані, і може задаватися на бажання користувача. В цілому, модель виходить з того, що семантична складова запиту є цільною і його частини не суперечать одна одній. Така форма запиту є відмінною від звичних морфологічно повних фраз, і була обрана для спрощення перевірки функціонування системи – аналіз повнотекстових запитів користувача є окремим семантико-морфологічним завданням і виходить за рамки цілей цієї роботи.

Для кожного окремого ключового слова із запиту користувача відбувається операція пошуку відповідних кластерів-стем, множини яких формують семантичні мітки документа, за якими і буде генеруватися результуюча відповідь. Це стає можливим, оскільки до системи включена база знань, що представляє

собою корпус документів, тексти з якої мають відповідно побудовану семантичну модель документа.

Для кожного кластера-стеми із семантичних моделей текстів проводиться перевірка входження його елементів в користувальницький запит, для чого використовується розроблений механізм визначення загальної частини на основі відстані Левенштейна, що застосовується для кожного ключового слова і кожної стеми у кластері. Якщо для поточного кластера-стеми таке входження знайдено – то кластер стає кандидатом для включення у результуючу відповідь. Приклад кластеру-стеми тексту на тематику «чорні діри» зображено у таблиці 3.1.

Таблиця 3.1

Приклад фрагменту кластеру-стеми

диску средньомассівних стала аналог важливі витягнутої всередині тертя дозволило Зовнішнє великих завершила стала вік тертя обсерваторія Через що з'єднують виявити з'єднують витягнутої забезпечує даними пророкує будова завершила забезпечує показана средньомассівних вік Альберта об'єкти назад тертя будова завершила рентгенівському речовини Метагалактики средньомассівних Сонця утворення подібних поглинає будова средньомассівних забезпечує Чумацького відстанях забезпечує область забезпечує тертя астрономів білої будова тертя средньомассівних вік Альберта досліджень обсерваторія тертя обсерваторія тертя пророкує Альберта завершила нашої темпом багатьох спостерігача забезпечує становить інших сусідній стала компаньйоном Чорні вік Альберта масивних тертя десятків обсерваторія средньомассівних Виявлення тертя будова існування більш можливо відкрито скупчення визначаються завершила аналог високим тепер джерел

Як можна побачити з наведеного прикладу, у одному кластері містяться стеми, що мають прямий семантичний зв’язок, проте синтаксично вони ніяк не співвідносяться (чорні, обсерваторія, метагалактики, Сонця тощо). Саме це дає можливість системі знаходити семантично близькі речення у тексті, не спираючись при цьому на синтаксичні представлення (якщо ми шукаємо чорні діри – то інформацію про метагалактики також необхідно вважати релевантною,

при умові достатньої сили семантичних зв’язків). Інші «нерелевантні» словоформи (наприклад – розповсюджені дієслова) відсікаються за недостатньою вагою відносно тексту в цілому і конкретного запиту. У цьому і полягає реалізація моделі м’якого розуміння Леонтьєвої, коли ситуація – а власне, запит користувача, змінює сенсові ваги одного і того самого тексту.

Для подальшого аналізу модель повинна ранжувати кластери-кандидати за їх співвідношенням із вхідним запитом, тому для кожного знайденого кластера розраховується ситуативна вага – кількісна міра, що характеризує загальну вагу стем у тексті що збіглися із запитом користувача, після чого усі знайдені на попередньому кроці кластери-стеми сортуються відповідно до розрахованого значення ситуативної ваги.

Для обмеження і зручності аналізу відповідей розмір результуючого тексту, що буде згенеровано для користувача, виходить із кількості кластерів із максимальною ситуативною вагою C_W, що вираховується за формулою (3.1):

$$C_W = \frac{W_C}{S_C}, \tag{3.1}$$

де W_C – загальна кількість стем у базі системи, S_C – загальна кількість речень у базі системи.

Заключним етапом роботи системи є генерація текстової відповіді користувачеві. Для цього вибираються кластери-речення, пов'язані з кожним кластером-стемою загальною кількістю C_W, що стали кандидатами для включення в результуючу відповідь на попередньому етапі. Якщо таких пов'язаних кластерів кілька, то в множину кандидатів потрапляє кластер з максимальною вагою зв'язку. В результаті даної операції формується множина речень-кандидатів для включення у результуючу відповідь, що сортується за номером речення у вихідному документі і загальною вагою стем у реченні. На виході користувач отримує згенеровану текстову відповідь, семантично пов’язану з вхідним запитом.

Відповідно до IR-based QA архітектури, наведений алгоритм роботи системи запит-відповідь можна поділити наступним чином: етап аналізу і розбору запиту співвідноситься із кроком 1, обчислення релевантних фрагментів з бази знань співвідноситься із крокам 2, і безпосереднє формування відповіді – із кроками 3-5. Приклад отриманої відповіді на запит «інтерфейс» наведено у додатку 12.

Отримана відповідь містить інформацію не тільки дані про інтерфейс що цікавить користувача, а й семантично пов’язану інформацію – щодо абстракцій у теорії програмування, об’єктно-орієнтованого підходу до розробки, інформацію про класи, тощо. Згенерований текст був отриманий на основі документів «Клас» і «Інтерфейс». Окрім того, алгоритм є стійким до кількості слів у запиті і їх семантичних зв’язків – додатку 13 наведено приклад запиту що є комбінацією тематично однозначного (космос) і неоднозначного (дослідження) ключових слів.

Результуючий текст був отриманий на основі документів «Досягнення у освоєнні космосу» і «Астрономічна картина миру» та містить інформацію про космічну гонку СРСР та США, висадку людини на місяць, посилання до романів Жуля Верна, та інші релевантні дані, що вказує на коректний процес роботи системи запит-відповідь. Докладніше, відповіді і адекватність роботи системи буде розглянута пізніше, наразі роздивимося деталі реалізації системи, першою з яких є реалізація текстової бази знань.

3.1. Текстова база знань системи і структура бази даних

Головним компонентом кожної моделі генерації текстів є база даних, структура і наповнення якої прямим чином впливає на швидкість і якість одержуваних результатів. Найбільш популярним засобом організації текстової інформації для її подальшого автоматизованого аналізу є створення корпусів текстів. Це поняття прийшло з лінгвістики, в якій визначається наступним чином: «корпус текстів може розглядатися як досить складно організована антологія мовної діяльності, що відображає в собі усе жанрове розмаїття представленого в ньому роду словесності і займає проміжне положення між реальними комунікативними процесами в суспільстві, які він представляє, і формалізованої лінгвістичної теорією, для якої він є джерелом для дослідження» [60]. У класичному прикладному застосуванні, під корпусом текстів розуміють колекцію категорійних текстів, зібраних за певною методикою і представлених в електронному вигляді. Категоризація текстів може проводитися як за інтегральним характеристикам кожного тексту (наприклад належність тексту до певної тематики), так і за специфічними характеристиками окремих термінів (наприклад словоформи, леми, морфеми). Більш того, ця колекція текстів повинна бути організована у вигляді бази даних, щоб мати можливість практичного використання корпусу з метою наукового аналізу [60].

Серед множини визначень корпусу можна виділити наступні основні характеристики: електронний – в сучасному розумінні корпус повинен бути в електронному вигляді; репрезентативний – повинен добре «представляти» об'єкт, який моделює; розмічений – головна відмінність корпусу від колекції текстів; прагматично орієнтований – повинен бути створений під певне завдання.

Однією з найважливіших характеристик корпусу текстів є репрезентативність. Оскільки корпус – це своєрідна модель мови, його репрезентативність визначає достовірність отриманих на його основі даних, тому питання можна розглядати як «проблему адекватного відображення, адаптації або інтеграції великих масивів текстів чи деяких інших фрагментів мовної діяльності в істотно менший за обсягом корпус текстів» [60]. У корпусній лінгвістиці під

репрезентативністю розуміється збалансоване і пропорційне представлення текстів в корпусі.

Ще одним важливим процесом є розмітка корпусу. Під розміткою корпусу розуміється процес, який полягає в приписуванні до текстів і їх компонентів спеціальних тегів: лінгвістичних і зовнішніх (екстралінгвістичних). Виділяють [60] наступні лінгвістичні типи розмітки: морфологічна, семантична, синтаксична, анафорична, просодична, тощо. До деяких корпусів застосовуються подальші структурні рівні аналізу. Зокрема, деякі невеликі корпуси можуть бути повністю синтаксично розмічені. Такі корпуси зазвичай називають глибоко анотованими або синтаксичними, а сама синтаксична структура при цьому є деревом залежностей [60].

Описані вище критерії дозволяють правильно сформулювати визначення бази знань нашої системи. Згідно цілей нашої роботи у тексті не має бути присутня будь-яка семантична або синтаксична розмітка, тому у її основу покладена саме колекція текстів, що буде задовольняти корпусним властивостям електроності, прагматичної орієнтованості та репрезентативності. Розглянемо ці пункти докладніше.

Під електронною властивістю мається на увазі представлення текстів у вигляді plain text, закодованому у універсальному форматі опису символів UTF-8 і доступному для обробки ЕОМ, а їх додаткові дані, що були зняті із колекції під час роботи системи – організовані у вигляді електронної реляційної бази даних (у нашій системі була використана СКБД PostgreSQL 9.5), структура якої зображена на рис. 3.2.

Створена база даних складається із чотирьох таблиць що не мають зв’язків, оскільки контроль і керування посилковою цілісністю перекладено на рівень додатку. Розглянемо призначення і основні компоненти кожної таблиці докладніше.

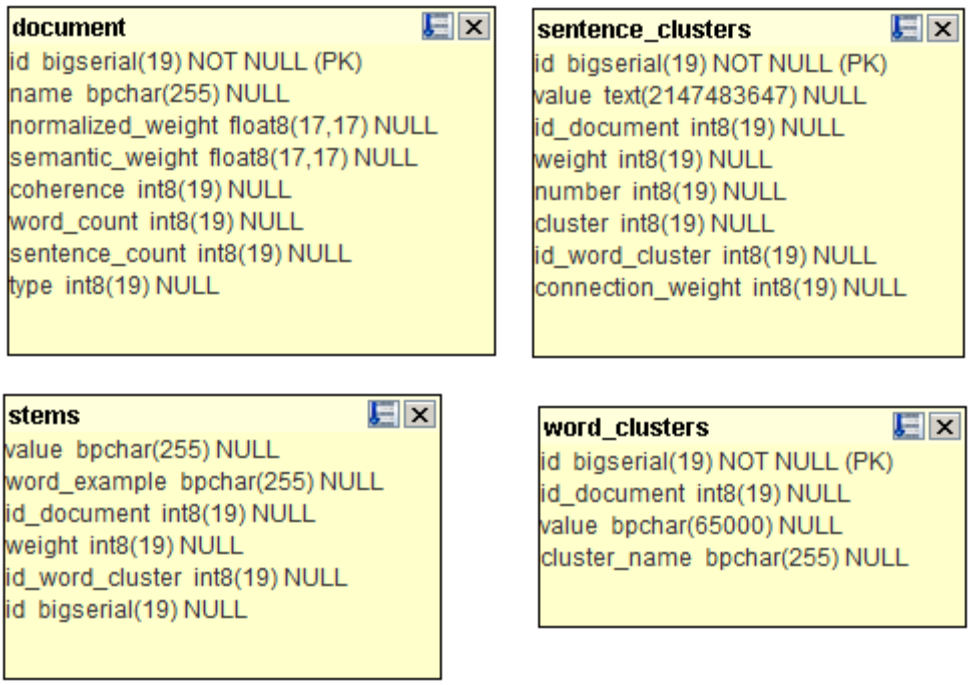

Рис. 3.2. Структура бази даних системи

Таблиця document відповідає за збереження інформації про конкретний текст у корпусі, і є базовою для системи. Атрибут name відповідає за шлях до цифрового файлу з початковим текстом, word_count та sentence_count містить у собі інформацію про кількість слів та речень у тексті відповідно, type відповідає за тип тексту та пов’язаний із репрезентативністю набору текстів, що буде розглянута нижче, а група атрибутів normalazid_weight, coherence, semantic_weight відповідають за процес автоматичної класифікації, що розглянутий у наступному розділі.

Таблиця word_cluster зберігає перелік кластерів-стем тексту: атрибут id_document вказує на зв’язок із сутністю із таблиці document, атрибут value зберігає у собі текстові стеми, що входять до конкретного кластеру, a cluster_name вказує на ім’я кластеру у семантичній моделі.

Таблиця sentence_cluster містить у собі інформацію про перелік кластерів-речень, що зв’язані з конкретним кластером стемою. Кожен запис у таблиці є окремим реченням і складається із наступної інформації: value – текст речення, id_document вказує на зв’язок із сутністю із таблиці document, id_word_cluster

вказує на зв’язок із сутністю із таблиці document, weight і number – на вагу та номер речення у вхідному тексті, cluster – на номер кластеру-речення у побудованій семантичній моделі, а connection_weight – на вагу зв’язку із кластером-стемою.

Таблиця stems зберігає інформацію щодо конкретної стеми у сформованих кластерах-стемах відповідного тексту: атрибут id_document вказує на зв’язок із сутністю із таблиці document, атрибут value зберігає значення текстової стеми, атрибут weight містить вагу стеми відносно вхідного тексту, атрибут word_example містить початкове слово, від якого була знайдена стема, стовбець id_word_cluster містить ідентифікатор кластеру-стеми вхідного тексту, що містить поточну стему.

Під прагматичною орієнтованістю мається на увазі побудова корпусу з точки зору повноти виконання завдання роботи і оцінки отриманих результатів інтелектуальної моделі генерації відповідей у пошукових системах.

Під репрезентативністю розуміється складання корпусу таким чином, щоб максимально задовільнити виконання усіх можливих сторін функціонування системи. Мова йде про використання декількох класів текстів, обробка яких по-перше, повно покриє стилістичні і семантичні особливості мови, а по-друге, стане опорою для проведення повноцінного тестування системи. Для цього була складена колекція із 100 текстів наукової стилістиці, розміри яких розподілені від 6 до 203 Кілобайт чистого тексту, класи яких розподілені у відсотковому співвідношенні у таблиці 3.2, де тип 0 – відповідає текстам із слабкою семантичною зв’язністю, що були створені за допомогою автоматичного генерування або складені із фрагментів різних текстів і використовуються для тестування системи, тип 1 – тексти на економічну тематику, тип 2 – тексти на тематику філософії, тип 3 – тести на тематику космології і астрономії, а тип 4 – тексти на тематику інформаційних технологій і програмування.

Таблиця 3.2

Відсоткове розподілення типів тексту у колекції

Тип	Кількість (%)
0	35
1	15
2	21
3	24
4	5

Таке розподілення переслідує мету створення достатньо репрезентативної вибірки текстів та розробки необхідного набору даних, що дозволить створити гнучку систему оцінок та перевірок адекватності додатку. Велика кількість семантично помилкових текстів створює умови для виявлення випадковостей у створенні відповідей, а велика кількість тематично незалежних кластерів тексту дозволяє змоделювати можливості обробки запиту користувача у полістилистичній колекції текстів.

Таким чином, на основі бази даних корпусу текстів була утворена база знань системи. Тут треба зазначити, що головна відмінність бази даних і бази знань полягає у наступному: «БД являє собою жорстко структуровану модель записів однорідних даних, а БЗ являє собою відкриту модель семантичної мережі, яка може містити різнорідні і різнотипні дані» [61]. Оскільки в структурі розробленої БД містяться, окрім неструктурованих текстів, різноманітна інформація про структуру семантичної моделі, що являє собою правила виведення і генерації нових знань при здійснені пошуку відповіді на запит користувача, стає можливим стверджувати, що описана модель працює саме з базою знань неструктурованих текстів.

3.2. Модель автоматичної класифікації вхідних документів

Питання створення моделі, що базується на деяких знаннях йде пліч о пліч із питанням адаптивності шляху накоплення цих знань. Розробка підходу, заснованого на побудові семантичної моделі документа створює великі можливості у цьому плані, дозволяючи незалежно наповнювати колекцію документів (і базу знань моделі відповідно) текстами формалізованого стилю різної тематичної спрямованості. Модель функціонує таким чином, що у користувача немає необхідності у ручній семантичній розмітці тексту, класифікації або перевірки тексту на відповідність якимось критеріям – достатньо лише завантажити документ у форматі plain text. Проте, такий підхід призводить до можливості наповнення моделі знаннями, що недостатньо якісні для її стабільної роботи, і це може створити надлишкову множину текстів для аналізу. До таких текстів, в першу чергу, відносяться недостатньо формалізовані документи, відповідність викладення інформації в яких не дозволяє отримати адекватний частотний портрет документа. Вирішенням цієї проблеми може стати застосування процесу попередньої оцінки і автоматичної фільтрації документів за їх семантичною зв’язністю.

Проблема класифікації текстів за їх семантичними властивостями не є новою – в результаті рішення задач автоматичної смислової обробки текстової інформації, таких як реферування або машинний переклад текстів з однієї природної мови на іншу, повинен бути отриманий зв'язний текст, який представляє собою послідовність семантично пов'язаних одне з одним речень. Тим не менш, наразі вона піднімає ряд складних і невирішених наукових питань, оскільки необхідно аналізувати не тільки синтаксичні зв'язки всередині речення, але і відношення між ними – міжфразовий зв'язок. Ці зв'язки можуть виражатися різними мовними способами. Згідно викладеної в роботі [61] теорії, документ підпорядковується деяким структурним законам зв'язного тексту, таким як закони повторюваності, скорочення і надмірності сенсу. Повторення елементів сенсу – це необхідна умова існування тесту. При цьому повторення сенсу в зв'язковому тексті можливо лише за умови дії принципу скорочення. Закон скорочення

вимагає передавати певний сенс мінімумом лексичних засобів. Навпаки, закон надмірності дозволяє збільшувати засоби вираження сенсу, що сприяє підвищенню надійності сприйняття тексту. Це вказує на високе різноманіття і неформалізованість семантичних портретів тексту, тому, хоча виконання подібних класифікацій текстів має широкий прикладний характер, проведений аналіз існуючих розробок показав брак досліджень у цій галузі, дозволивши сформулювати лише три суміжних із нашими цілями напрями.

По-перше, одним з найбільш близьких до нашої задачі науковим дослідженням є роботи по визначенню атрибуції текстів, а саме – питання визначення автоматично згенерованого тексту, робота з яким велася в розділі 2. Наприклад, в роботі [63] запропонований алгоритм визначення текстового спаму на основі оцінки різноманітності тематики документа. Метод заснований на аналізі характеристик тексту, що відрізняють пошуковий спам, до яких відносяться велика кількість ключових слів, наявність прихованого тексту, дрібного нечитабельною шрифту, зловживання тегами заголовків, тощо. Так само, дослідження веб-спаму опубліковані зарубіжними авторами C. Castillo, D. Donato і ін. [65]. В основі запропонованого ними методу – аналіз веб-контенту на наявність посилань на певні сторінки і їх розподілу між сторінками.

По друге – визначення текстів, що представляють собою машинний переклад з однієї природної мови на іншу – ще одна задача класифікації зв'язності тексту. Для виявлення таких текстів існує ряд спеціальних алгоритмів, наприклад, BLEU, METEOR, Perplexity, які оцінюють якість машинного перекладу через відповідність характеристик тексту на лексичному рівні деяким еталонним показниками природної мови [65].

Врешті, цікавим є підхід, який базується на автоматичному обчисленні семантичних відношень викладений в роботі [66]. Проводячи виявлення міжфразових зв'язків текстів з суспільно-політичної тематики на порівняно невеликому за обсягом матеріалі, автори спробували знайти залежність частоти виникнення різних типів зв'язків від виду і довжини тексту. Була зроблена спроба виявлення міжфразових зв'язків в суспільно-політичних текстах на основі

автоматичного вирішення анафори за допомогою статистичних алгоритмів обробки даних. Робота алгоритму оцінювалася на випадковій вибірці з 50 текстів, в результаті чого було отримано близько 70% правильно вирішених анафор.

Більш глибоке вивчення інструментів описаних підходів до семантичної класифікації текстів дозволило сформулювати наступні висновки:

– існуючі методи класифікації текстів дозволяють вирішувати завдання виявлення пошукового спаму, а також текстів, які є машинним перекладом документу з однієї природної мови на іншу на надійному рівні, проте спроби створити універсальні інструмент семантичної класифікації тексту наразі показують ненадійні результати;

– через накладені обмеження і особливості розглянутих текстів існуючі методи не можуть бути використані в задачі визначення текстів, створених за допомогою сінонімізаторов або інших засобів автоматичної генерації;

– для досягнення більшої точності у визначенні зв'язності тексту необхідно розглядати всі лінгвістичні рівні тексту, в тому числі синтаксичний;

– якість тексту безпосередньо пов'язана з його тематичними властивостями, тобто пов'язані з ними характеристики тексту повинні бути включені в множину характеристик тексту для аналізу;

– при формуванні набору текстових характеристик, що використовуються для ідентифікації неприродних текстів, слід відштовхуватися від глобальних семантичних властивостей текстів;

– використання лише статичного підходу не дає достатньо точних результатів та зменшує адаптивність використання такого алгоритму.

Виходячи з цих пунктів, було прийнято рішення розробити універсальний підхід до класифікації текстів за семантичною ознакою формалізованості їх стилю, що базується на оброці семантичної моделі документу. Оскільки отримана модель може бути використана для зняття числових даних, що характеризують семантичні властивості документа, такі дані можуть використовуватися для автоматичного визначення зв’язності тексту, у нашому випадку – для навчання деякої нейронної мережі. У процесі досліджень було встановлено, що до таких

характеристик відносяться загальна кількість стем, кількість усіх слів, кількість кластерів-стем, що мають зв'язок із кластерами-реченнями та загальна кількість кластерів-стем.

Алгоритм роботи системи класифікації текстів зображено на рис. 3.3, і першим його кроком є отримання документу для аналізу, що є кандидатом для побудови семантичної моделі і включення у базу знань.

За створеним алгоритмом, кожен з текстів характеризувався двома значеннями – нормалізованим розміром тексту W_N, отриманого за формулою (3.2):

$$W_N = \frac{W_I - W_{\min}}{W_{\max} - W_{\min}}, \tag{3.2}$$

де W_i – загальна кількість слів, W_{min} та W_{max} – найменша та найбільша кількість слів у навчальному корпусі, та нормалізованим семантичним значенням S_N, отриманим за формулою (3.3):

$$S_N = \frac{W_U}{W} * \frac{CW_C}{CW}, \tag{3.3}$$

де W_U – загальна кількість стем, W – загальна кількість слів, CW_C – кількість кластерів – стем, що мають зв'язок із кластерами-реченнями, CW – загальна кількість кластерів – стем. Отримані таким чином данні є критеріями оцінки формализованості тексту і водночас – навчальною вибіркою для нейронної мережі. Приклад отриманих значень наведено в таблиці 3.3.

Необхідно враховувати, що описані семантичні характеристики залежать від розміру тексту, тому зняті данні потребують попередньої нормалізації. В якості даних для навчання використовувалися тексти із описаного раніше корпусу – 50 текстів було використано як набір уроків (40% склали незв’язні тексти, а 60% – зв’язні),

Таблиця 3.3

Приклад отриманих значень для розрахування зв’язності тексту

W_N	S_N	Зв’язність
0.134172558784485	0.0505628511309624	Ні
0.459072768688202	0.0019859226886183	Ні
0.0692661958857845	0.0945273631840796	Так
0.159594719066626	0.0535888275414095	Так
0.0781086889775867	0.122146118721461	Так
0.646300276327909	0.0289232167196729	Так

Для реалізації структури нейронної мережі була використана бібліотека SimpleNeuralNetwork [67], що дозволяє задавати кількість входів у нейрону мережу, має один прихований рівень, кількість нейронів у якому дорівнює кількості уроків, що подано на вхід, та один вихід що може видавати значення 0 або 1 – наші класи тексту відповідно. Для навчання мережі використовується друге правило Хебба (дельта-правило), функцією активації є сигмоїдна функція, а значення зміщення одноразово обираєтеся випадково у проміжку між -0,5 та 0,5. Використана структура нейронної мережі зображена на рис. 3.4. Отримана навчена модель використовується для прогнозування зв’язності вхідних текстів, у випадку, коли користувач бажає доповнити базу знань системи. Якщо відповідно до отриманого прогнозу документ, що завантажується, є достатньо формалізованим, він проходить побудову семантичної моделі та завантажується у базу знань системи, інакше користувач отримує помилку і завантаження не відбувається.

Рис. 3.3. Алгоритм автоматичної класифікації документів

Для тестування отриманої мережі було виконано 50 тестів, данні для яких вибиралися з бази даних колекції текстів – а саме з таблиці document, де стовбець normalazid_weight відповідає за значення W_N, стовбець semantic_weight – відповідає за значення S_N, а стовбець coherence містить у собі оцінку зв'язності – 1 або 0. Важливою особливістю є те, що у тестовому корпусі були присутні як тексти, що згенеровані автоматично, так і тексти, що складалися із зв'язних

фрагментів реальних текстів, які мають семантичний зв’язок між реченнями всередині, і не мають семантичного зв’язку між собою у документі в цілому (приклад такого тексту наведено у додатку 14).

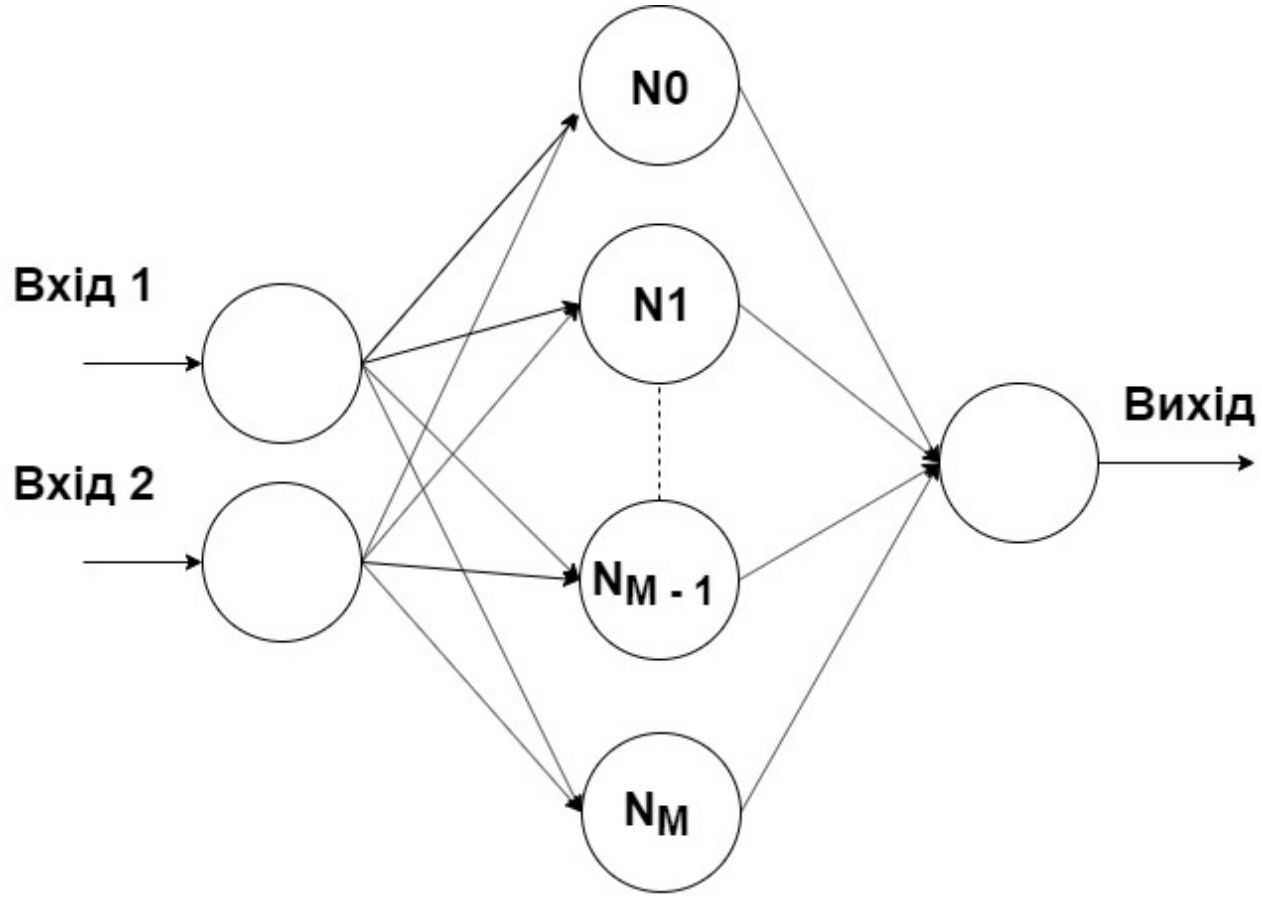

Рис. 3.4. Структура нейронної мережі

Проведені експерименти показали коректну класифікацію що у 90 відсотках оброблених текстів. При цьому, жоден прогноз щодо незв’язних текстів не був хибним – це стосується як цілком хаотично згенерованих текстів, що вказує на коректну обробку статичних властивостей тексту, так і семантично-фрагментарних текстів, що говорить про коректну обробку семантично незв’язних текстів. Такі результати вказують не тільки на достатню і задовільну точність роботи класифікатора і можливість його подальшого використання, а і на адекватність семантичної моделі документу в цілому і доцільності її застосування в інтелектуальних системах автоматичної обробки текстів.

3.3. Загальна програмна архітектура та інтерфейс системи генерації відповідей

В цьому розділі будуть розглянуті основні компоненти програмної архітектури створеної системи, деякі зовнішні бібліотеки, що були застосовані у процесі розробки та особливості функціоналу інтерфейсу додатку.

Оскільки система була розроблена на мові Java, був використаний об'єктно-орієнтовний підхід до розробки програмного забезпечення, а структура розміщення класів розроблялась з урахуванням пакетної можливості об'єднання класів у одну сутність за їх функціональним призначенням. Тому нижче будуть розглянуті саме програмні пакети, чого цілком буде достатньо для розуміння загальної концепції та можливостей отриманого програмного додатку.

Усього система складається з 7 пакетів загального та спеціального призначення:

Пакет semantic.search.application – у основу архітектури додатку був покладний принцип Inversion of Control (IoC) реалізований за допомогою фреймворка Spring Boot 2.0 [68] тому у цьому пакеті міститься один клас, що являє собою головну точку входу програми і має посилання на її графічний інтерфейс. Діаграма класів цього пакету зображена на рис. 3.5.

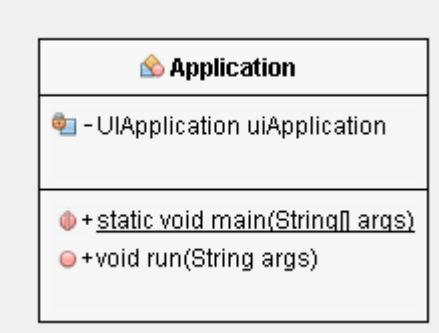

Рис. 3.5. Класи пакету semantic.search.application

Пакет semantic.search.common.util – містить функціонал для виконання алгоритму загортання подарунків Джарвіса і організації процесу стемінгу. Діаграма класів цього пакету зображена на рис. 3.6.

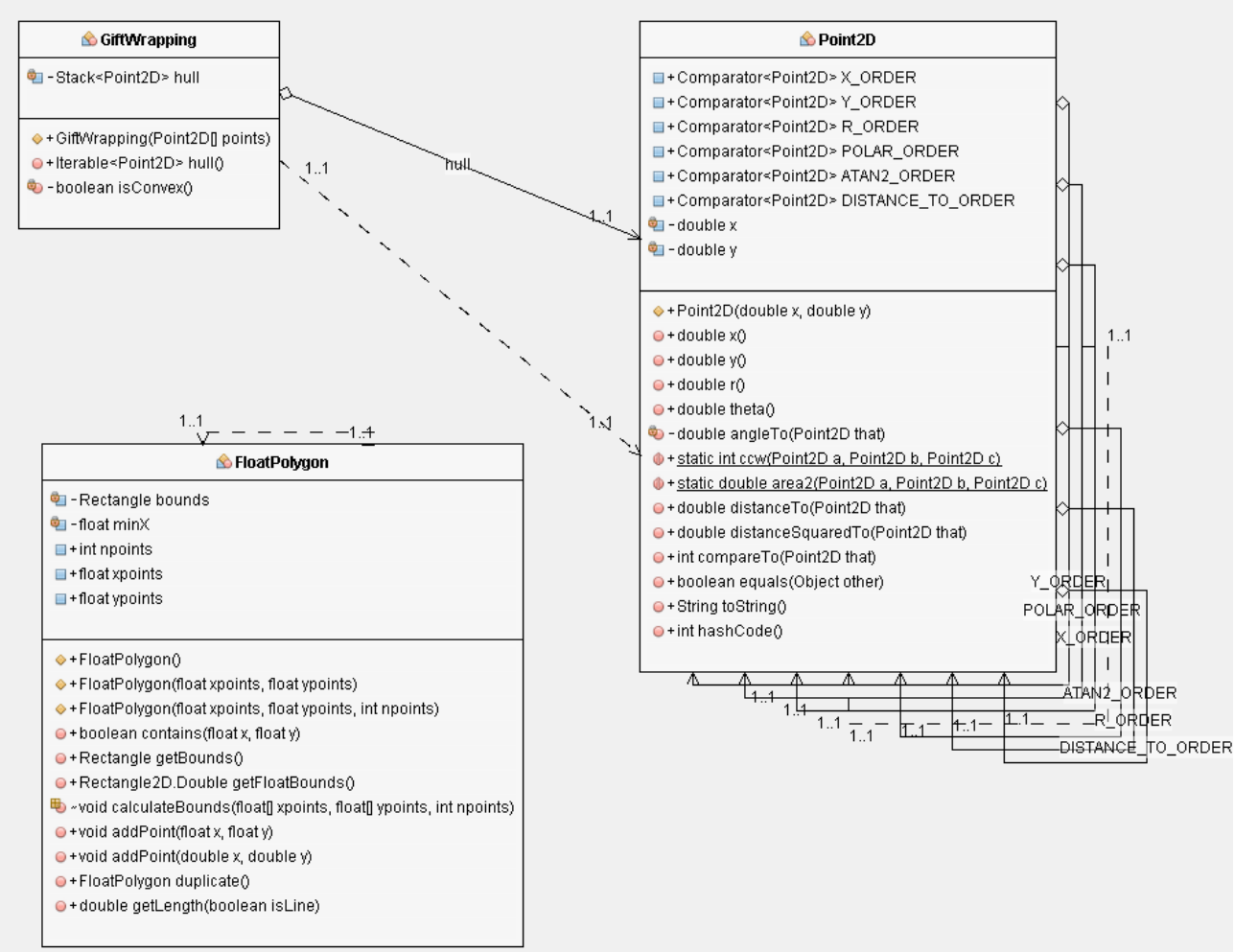

Рис. 3.6. Класи пакету semantic.search.common.util

Пакет semantic.search.model містить у собі класи, що реалізують засоби Java DataBase Connectivity [69] для роботи із базою даних PostgreSQL [70] за допомогою механізму Object-Relational Mapping [71], реалізованого на основі фреймворку Hibernate (JPA version) [72]. При цьому підході кожна таблиця має клас-прототип, для роботи з якими використовується шар data access object, представлений відповідними інтерфейсами. Діаграма класів цього пакету зображена на рис. 3.7.

Пакет semantic.search.ui містить засоби для організації інтерфейсу користувача, функціонал якого написано за допомогою бібліотеки Swing [73]. Діаграма класів цього пакету зображена на рис. 3.8.

Пакет semantic.search.model.vo – містить у собі класи, що використовуються як сутності для передачі та зберігання інформації (Value Object) щодо стеми та

речення, їх координат, ваги, номеру, частини мови та семантичної і пошукової моделей. Діаграма класів цього пакету зображена на рис. 3.9.

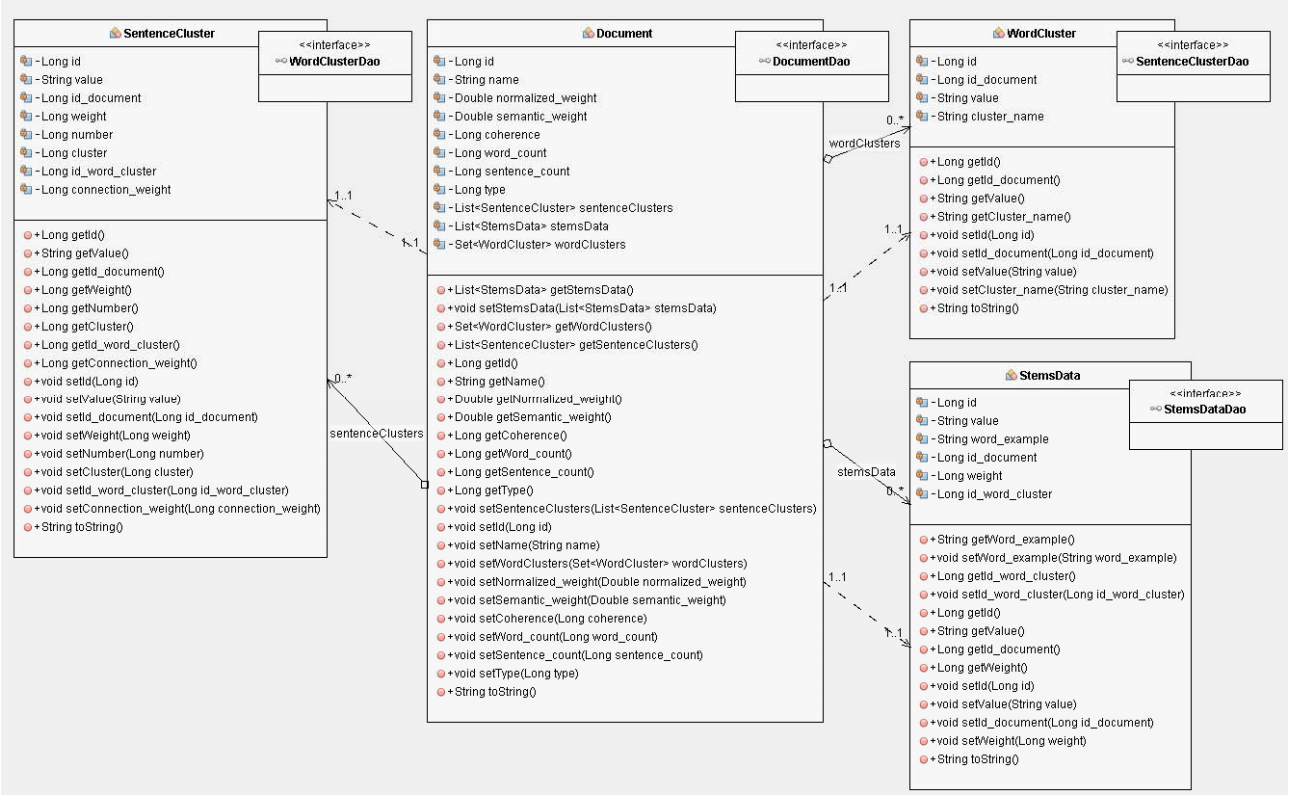

Рис. 3.7. Класи пакету semantic.search.model

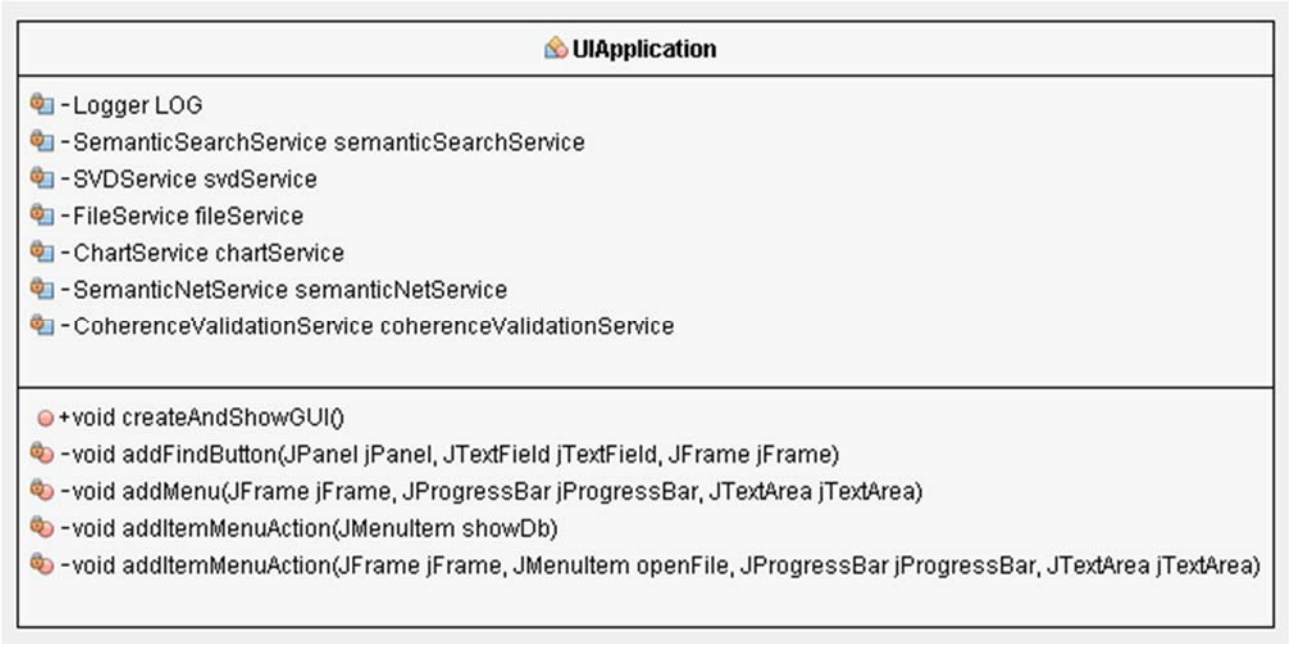

Рис. 3.8. Класи пакету semantic.search.ui

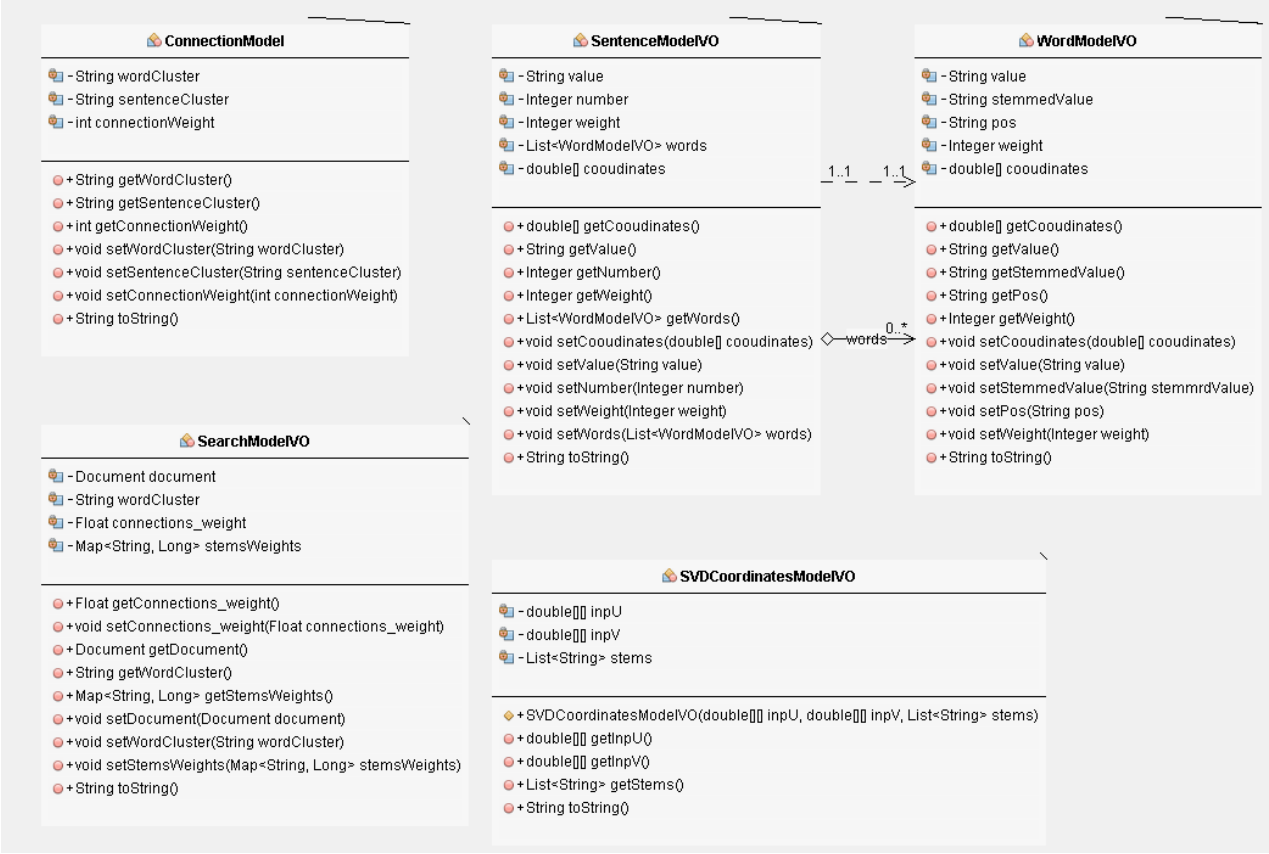

Рис.3.9. Класи пакету semantic.search.model.vo

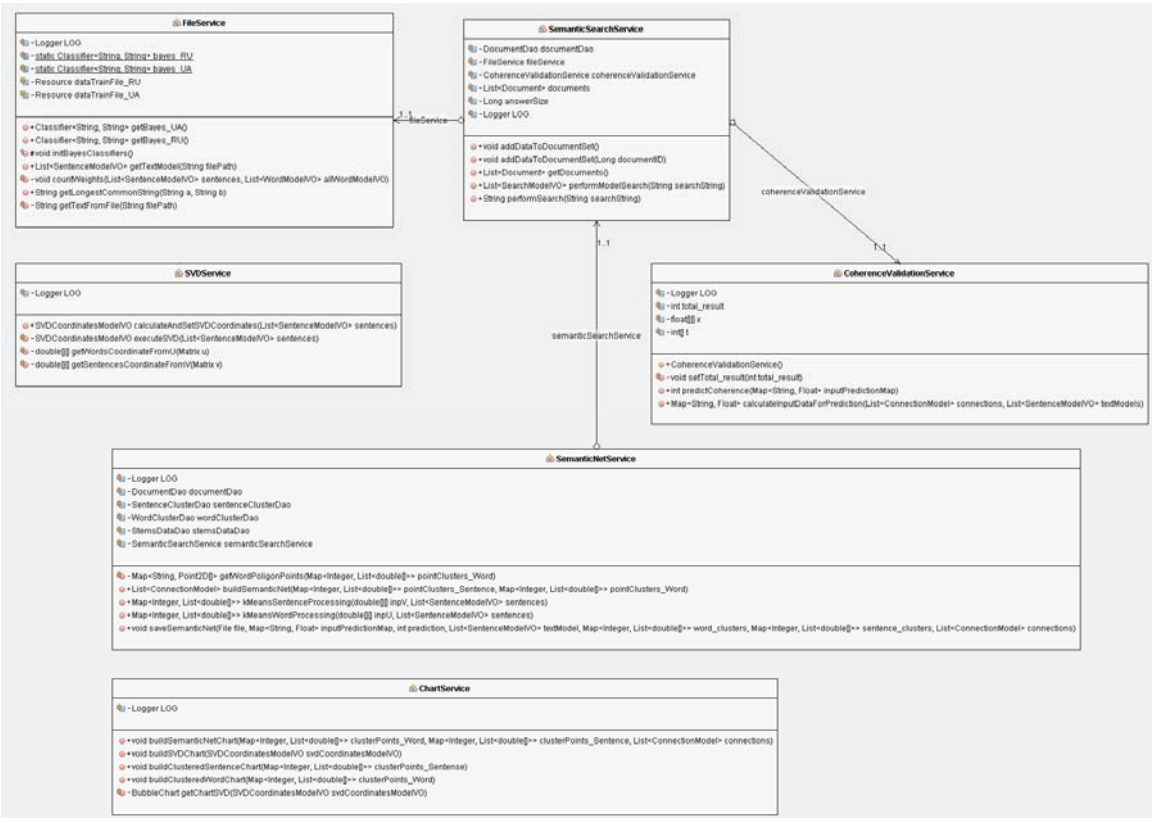

Рис.3.10 Класи пакету semantic.search.util.service

Пакет semantic.search.util.service містить сервіси, що реалізують головний функціонал моделі – латентно-семантичний аналіз, кластерізацію, побудову семантичної моделі тексту, виконання прогнозування зв'язності, формування синтаксичної моделі із текстового файлу та побудову графічного відображення компонентів програми . Діаграма класів цього пакету зображена на рис.3.10.

Врешті, пакет semantic.serach.neuralnetwork містить класи, що реалізують структуру нейронної мережі для класифікації вхідних документів за ознакою їх зв'язності. Діаграма класів цього пакету зображена на рис. 3.11.

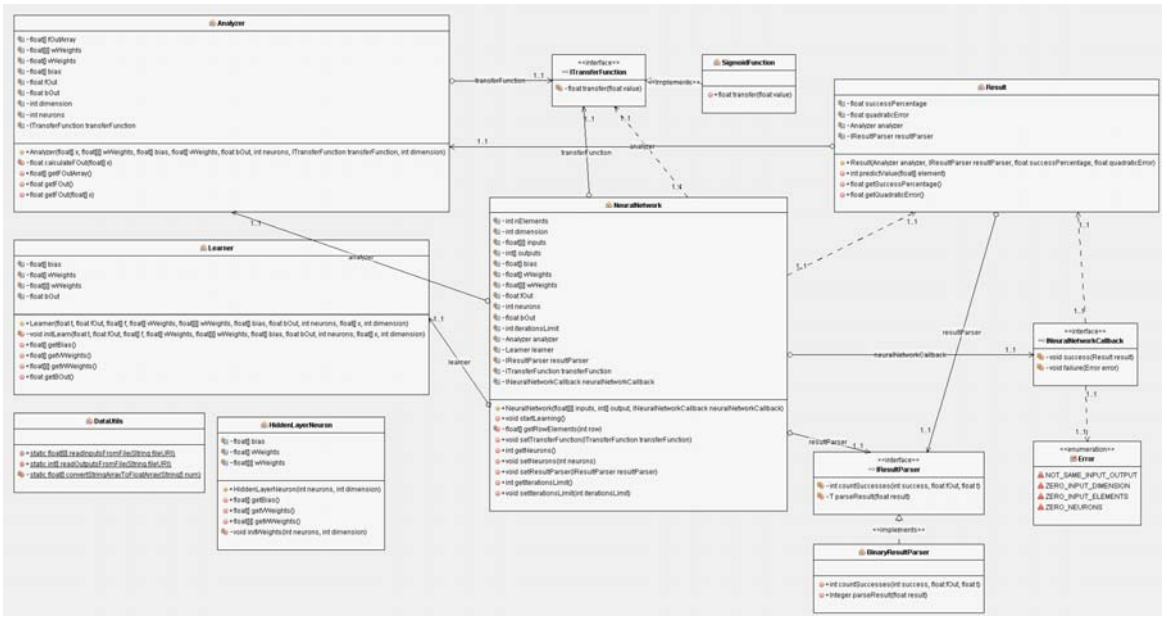

Рис.3.11. Класи пакету semantic.serach.neuralnetwork

Окрім описаних пакетів і класів, відповідно до принципу test-driven development [74] була розроблена система юніт- та інтеграційних тестів функціоналу системи, що мають забезпечити процес безпечної модифікації існуючого функціоналу. Діаграма класів тестів зображена на рис.3.12 і включає наступні методи перевірки роботи програми:

testShouldFindCorrectPosRU – тест перевіряє, що точність передбачення частини мови для російськомовного корпусу не нижче ніж 90%.

testShouldFindCorrectPosUA – тест перевіряє, що точність передбачення частини мови для україномовного корпусу не нижче ніж 90%.

testShouldGetCorrectTextModel – тест перевіряє, що побудована частотно-синтаксична модель із текстового файлу є коректною з точки зори виділення слів, речень, отримання стем і розрахунку їх ваг.

testShouldProcessCorrectPrediction – тест перевіряє, що із заданого набору текстів відсоток коректних передбачень зв'язності тексту не нижче ніж 90%.

testShouldCheckCorrectTrainData – інтеграційний тест, що перевіряє адекватність роботи процесу запит-відповідь на основі автоматичної побудови запиту і оцінки моделі відповідей.

testShouldProcessCorrectKMeansClustering – тест перевіряє коректність процесу кластеризації за алгоритмом kMeans.

Рис. 3.12. Класи тестового пакету системи

testShouldBuildCorrectSemanticNetwork – тест перевіряє коректність побудови семантичної моделі тексту та ваг зв'язків між кластерами у моделі.

testShouldBuildCorrectSVDMatrixCoordinate – тест перевіряє коректність процесів сингулярного розкладання і побудови SVD-матриць.

Початковий код системи знаходиться у відкритому доступі у репозиторії github за адресом https://github.com/yegorkovylin/sm-scr.git та має вбудоване програмне API отримання програмної моделі тексту у вигляді бібліотеки, що складається з наступних методів: getTextModel – отримання синтаксичної моделі тексту; calculateAndSetSVDCoordinates – проведення латентно-семантичного аналізу; kMeansSentenceProcessing та kMeansWordProcessing – отримання семантичних контурів стем та речень; buildSemanticNet –отримання семантичної моделі тексту.

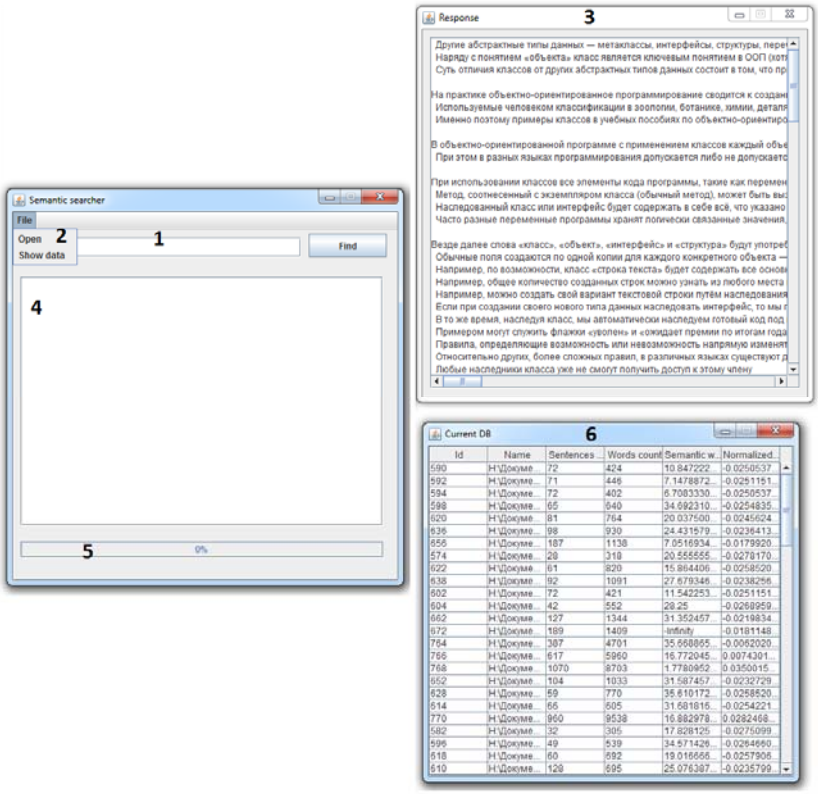

Рис. 3.13. Головне вікно системи: 1 – пошуковий рядок 2 – меню системи, 3 – область відповіді, 4 – системна інформаційна консоль, 5 – смуга завантаження, 6 – структура бази знань

Завантаження усіх допоміжних бібліотек, що були використані у процесі роботи системи, відбувається автоматично завдяки використанню посилань на

maven – репозіторій, тож розроблена система є повністю переносимою та відкритою для модифікацій сторонніми розробниками.

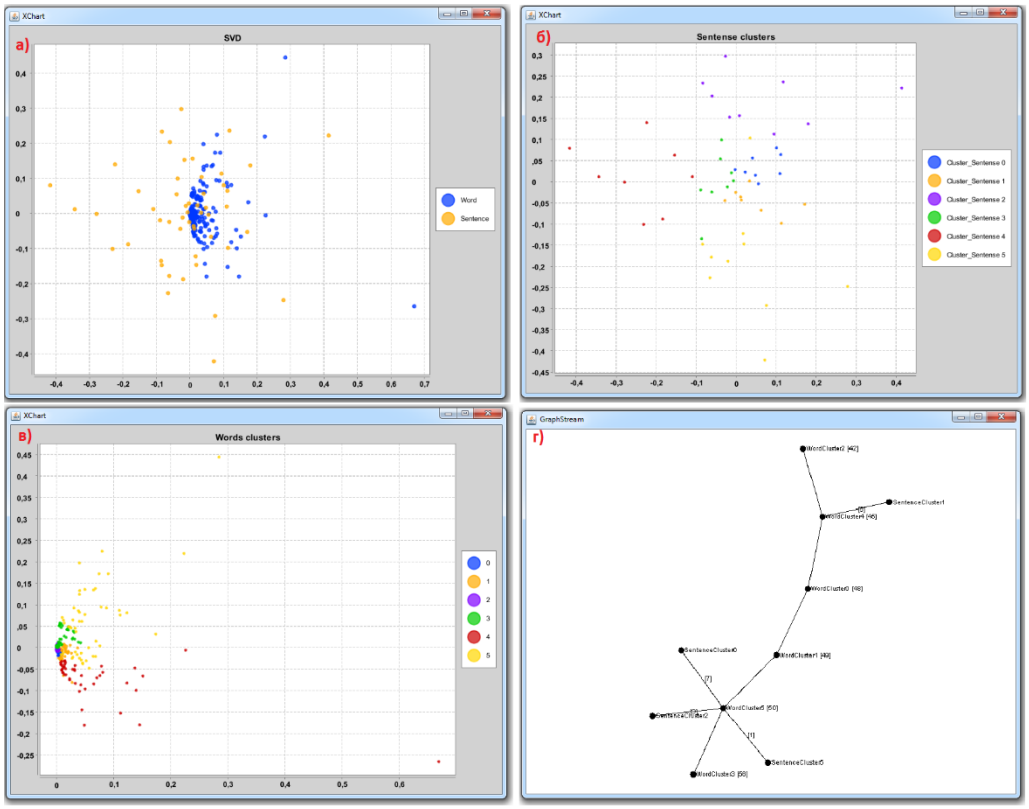

Рис. 3.14. Комплекс вікон результату завантаження тексту: а) – проекція сингулярного розкладання, б) кластери-речення, в) – кластери-стеми, г) – семантична модель документу

Графічний інтерфейс системи розроблений з використанням бібліотеки Swing [73] та представлений головним вікном програми, що зображено на рис. 3.13. Система може функціонувати у двох режимах – як пошуковий додаток та як обробник зовнішніх файлів. У першому випадку користувач вводить цікавлячий його запит у пошукову строку 1 та натискає кнопку «Find», після чого починається процес виконання пошукового завдання. Результуюча відповідь системи буде виведена у область 3, а додаткова системна інформація про процес пошуку виводиться у консоль 4. У разі, якщо користувач бажає розширити базу знань системи шляхом додавання нових текстів до колекції, необхідно скористуватися меню 2 та вибрати пункт «Open» що дозволить завантажити

цільовий файл (або файли) на обробку. Після того, як система завершить процес приєднання файлу до колекції, користувачеві відобразиться комплекс вікон, які зображені на рис. 3.14, та містить візуальну інформацію про процес латентно-семантичного аналізу, кластеризації та інтерактивну семантичну модель тексту. Необхідна додаткова інформація про процес обробки файлу також виводиться у системну інформаційну консоль 4. Перевірити поточний стан бази знань системи можливо за допомогою пункту меню 2 «Show data», виконання якого відобразить вікно 6.

Описана програмна архітектура показує, що основні концепції, яким необхідно слідувати при реалізації текстового автомату виконані: властивість модульності реалізована за допомогою розбиття інструментів аналізу та обробки тексту на окремі програмні об’єднання – пакети, а використання підходу об’єктно-орієнтованого програмування дозволяє закріпити окремі функції системи за окремими програмними одиницями – класами та об’єктами; властивість рівневості реалізована по-перше завдяки розділенню функцій системи на окремі методи, які реалізують обробку та аналіз тексту незалежно одне від одного на етапах синтаксичного (утиліти обробки текстів), лексико-морфологічного (стемінг, утиліти автоматичного визначення частини мови), та семантичного (латентно-семантичний аналіз, побудова моделі документу) аналізу, а по-друге завдяки лінійному алгоритму роботи системи, що застосовує ці методи групами – від найнижчого до найвищого рівня обробки тексту, де результат попереднього етапу передається на наступний – до самого завершення такту роботи системи; властивість орієнтованості на використання бази знань забезпечується використанням колекції текстів, супутньої бази даних та програмними інструментами роботи з нею, а також механізмом повністю автоматизованого та адаптивного поповнення загальної бази новими текстами; безпека роботи системи забезпечується в першу чергу виконанням попередньої фільтрації документів, що дозволить запобігти наповнення бази знань системи невірними з точки зору її критеріїв текстами. Окрім того, у систему був включений набір автоматизованих тестів результатів її роботи.

3.4. Оцінка адекватності отриманої моделі

Перевірка адекватності роботи систем вирішення складних інтелектуальних завдань в галузі автоматичної обробки текстів, до яких належить наш додаток, є окремим науковим завданням, оскільки воно так само вимагає включення процесу розуміння семантичних властивостей тексту як і процес основної роботи системи. Саме тому, одним з найпопулярніших підходів до організації процесу оцінки моделей запит-відповідь є залучення відповідних експертів, що у мануальному режимі протестують систему і дадуть своє заключення [75]. Проте головною проблемою, яка лежить в основі такого класичного методу оцінки відповідей експертом-людиною є суб'єктивність і ненадійність судження, не кажучи вже про необхідність власне пошуку експертів для нашого підходу, який широко не застосовувався до цього на практиці. З іншого боку, архітектура текстового автомату, що була покладена у основу створеної системи, вимагає впровадження доступного інструменту самотестування додатку, який має забезпечити виконання критерію безпеки системи. Ці фактори схиляють до необхідності додаткової автоматизації процесу оцінок системи, що, по-перше, виключить людський фактор, а по-друге, дозволить гнучко сигналізувати про проблеми семантичних невідповідностей у процесі роботи системи.

Не дивлячись на те, що система запит-відповідь є різновидом пошукової системи, спосіб автоматизації тестування систем із використанням еталонних таблиць-множин запиту і відповіді, який добре зарекомендував себе при вирішенні класичних завдань пошуку колекцій відповідних до запиту документів [76], не підходить у випадку генерації текстової відповіді. Справа в тому, що результатом кожного пошукового завдання є не просто коротка відповідь, але і фрагмент тексту з конкретного документа, що явно підтверджує цю відповідь. Таких фрагментів може бути в колекції багато, і система може зробити висновок як на підставі якогось одного з них, так і кількох різних фрагментів в різних документах, що містять одну й туж саму відповідь. При цьому в множині відповідності, на відміну від класичного пошуку, міститься не тільки ідентифікатор документа, але і фрагмент тексту, і коротка відповідь з цього

фрагмента. Складання подібних таблиць, де конкретна відповідь прив'язана до конкретного питання, на великому корпусі знань цілком можливо не буде являться єдино вірним рішенням. Тому, область дії автоматизації тестування повинна стосуватися всієї відповіді в цілому, а не конкретного її фрагменту на рівні одного документа і одного питання, що вимагає створення окремої підсистеми програмної оцінки згенерованих відповідей, алгоритм роботи якої зображено на рис. 3.15.

Сенс створеної перевірки виходить з ідеї про те, що відповіді системи, за умови роботи із адекватною повнотекстовою базою знань, не можуть бути однозначно визначені як «правильні» або «неправильні», оскільки навіть фрагмент семантично зв’язного документа містить у собі деяку кількість корисної для користувача інформації. Проте, якщо результуюча відповідь містить семантично розірвані фрагменти із різних текстів, то її сенсова значимість значно зменшується. У нашому випадку, таким критерієм семантичної зв’язності служить жанрова відповідність фрагменту тексту, що є кандидатом до включення у результуючу відповідь, та набору термінів із запиту користувача. Тому, розроблений алгоритм роботи системи тестування побудований на оцінці залежностей між тематикою множин кандидатів-документів до включення у результуючу відповідь та тематикою множин автоматично сформованих запитів, що заздалегідь були назначені кожному документу у колекції текстів. Розглянемо кроки роботи системи тестування більш детально:

Система аналізує кожен наявний документ у корпусі, тому першим кроком роботи системи є отримання документу із робочою базою знань і витяг усіх пов’язаних із ним стем, оскільки саме стеми являють собою терміни-запити до системи.

Рис. 3.15. Алгоритм роботи системи автоматичного тестування якості запитів

Для кожної отриманої стеми знаходиться її вага, після чого вибирається 8 (кількість отримана емпіричним шляхом) найважчих стем для поточного документа, що формують множину запитів для системи. Такий підхід до створення запиту зумовлений двома факторами:

– по-перше, він дозволяє створити множину запитів для кожного документа і, відповідно, для кожної тематичної галузі у базі знань, що дозволяє максимально повно оцінювати процес роботи системи у полістилистичній колекції документів;
– по-друге, умова отримання найважчих стем дозволяє вибирати максимально узагальнені поняття, які семантично пов’язані із темою тексту, що дозволить додавати у множину документів-кандидатів до включення у результуючу відповідь достатню для подальшої оцінки кількість документів – наприклад, отримання терміну «космос» із тексту про телескопи також межується із філософією, терміну «клас» із інформаційних технологій межується із усіма іншими тематиками (класова думка, класи зірок) та ін.

Для отриманих таким чином стем знаходяться їх початкові лексичні форми – терміни, що відправляються на вхід пошукової системи, у результаті чого формується множина речень-кандидатів до включення у результуючу відповідь та множина відповідних до них документів, кожному з яких відповідає заздалегідь заданий жанр. Оскільки усі терміни до запиту відбираються із одного документа, то вважається, що усі вони мають ту саму тематичну спрямованість, що і їх батьківський документ.

Для кожної множини текстів-кандидатів розраховується кількість документів, тема яких співпала із темою запиту (ситуативно коректні) та кількість документів, чия тема не співпала із темою запиту (ситуативно некоректні).

Важливим питанням цього підходу стає, власне, аналіз отриманих таким чином даних – множини кандидатів ще не є результуючою відповіддю, оскільки отримані документи повинні пройти через операцію відсікання речень із найбільшою вагою.

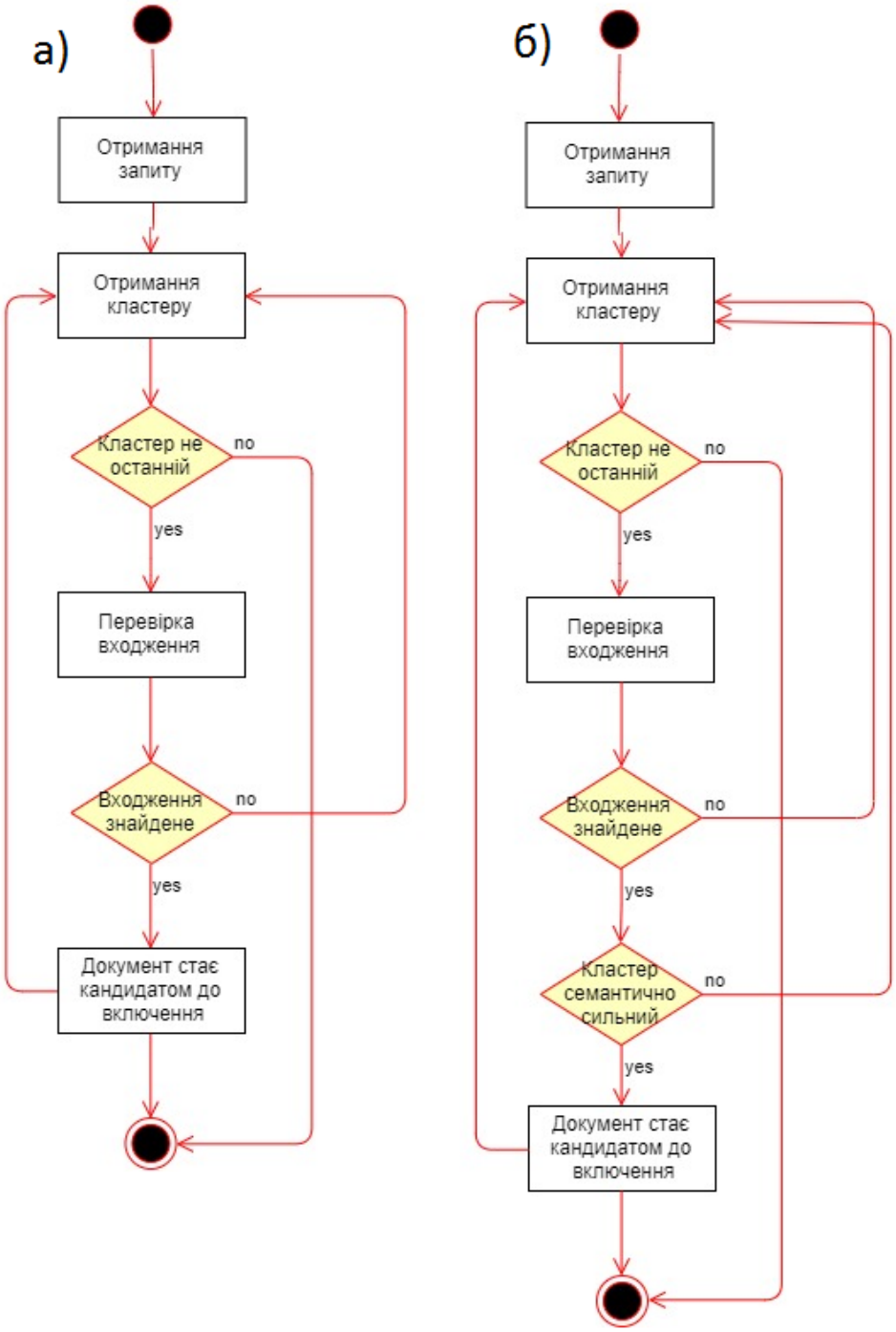

Рис. 3.16. Алгоритми роботи функціонування пошукової системи:
а) семантично відповідний, б) семантично невідповідний

Тому, для розуміння ступеню впливу семантичної моделі на процес генерації тексту необхідно розрахувати коефіцієнт коректності формування множини документів Q_D за формулою (3.4):

$$Q_D = \frac{\sum N_C}{\sum N_W}, \tag{3.4}$$

де $\sum N_C$ – загальна кількість ситуативно коректних документів для кожного запиту, $\sum N_W$ – загальна кількість ситуативно некоректних документів для кожного запиту.

Це значення необхідно отримувати для двох режимів функціонування системи – для «семантично відповідного» та «семантично невідповідного», процеси роботи яких зображено на рис. 3.16. Вони являють собою етап ситуативної обробки семантичної мітки документу, та складаються з безпосереднього отримання запиту і перевірці його входження у кластери-стеми бази знань системи. Головна відмінність алгоритмів полягає у створенні множин кандидатів-кластерів до формування відповіді користувачеві – у семантично відповідному режимі кластер-стема вважається кандидатом якщо він є семантично сильним для конкретного документу, тобто має найбільшу сумарну вагу перетинів із семантичними контурами кластерів-стем, тоді як у семантично невідповідному режимі роботи системи така перевірка не відбувається – стема вважається кандидатом якщо було знайдене хоча б одне входження терміну із вхідного запиту, що є по-суті, звичайним прямим рішенням пошукової задачі.

Гіпотеза перевірки полягає у тому, що значення коефіцієнту коректності формування множини документів для семантично відповідного режиму має бути більшим, ніж для семантично невідповідного, що покаже на позитивний вплив використання створених семантичних моделей у основі систем генерації текстових відповідей. Для її підтвердження було автоматично сформовано і виконано близько 1000 запитів до системи, у результаті чого було сформовано множини текстових знань у 20 тисяч документів, а значення коефіцієнту коректності формування множини документів для семантично відповідного режиму склало 0,843 проти значення у 0,493 для семантично невідповідного режиму, що вказує на доцільність застосування підходу заснованого на семантичних моделях у системі генерації тексту відповіді на поставлене питання.

Крім того, описаний алгоритм являє собою автоматичний інструмент самотестування системи, що, на відміну від підходу із множинами відповідностей [76], орієнтується у своїй роботі на аналіз відповіді в цілому і надає більш надійний і гнучкий процес оцінки якості моделей генерації відповідей.

Наявність процесу автоматичної оцінки системи зовсім не виключає можливості проведення емпіричних оцінок відповідей із залученням експертів, а являє собою додатковий інструмент тестування якості системи. Оскільки розроблений підхід до побудови моделі запит-відповідь заснований на створенні нових текстів і впроваджений як програмний засіб організації процесу пошуку у електронній колекції текстів, присутність процесу тестування результатів людиною – кінцевим користувачем системи є важливим і необхідним етапом дослідження. Тому наступним кроком перевірки адекватності додатку є організоване мануальне тестування, що базується на індивідуальних [77] оцінках відповідей системи за бальним методом [78], що являє собою сукупність оцінок вимог до згенерованої відповіді від 0 до 1 із кроком 0,1. Усього таких вимог 5, тобто кожна відповідь сумарно може получити від 0 до 5 балів. Розглянемо ці критерії більш детально.

Присутність відповіді. Першим критерієм оцінки є безпосередня присутність відповіді на поставлене питання – чи зміг користувач отримати необхідну інформацію, чи була вона достатньо корисною для користувача.

Ступень збігу із тематикою запиту. Цей критерій вказує на кореляцію тематики інформації, отриманої у відповіді і тематики, що цікавила користувача у запиті.

Повнота викладу вказує на присутність додаткової корисної для користувача інформації, а також ступінь розкриття теми запиту.

Присутність тематичних розривів пов’язана із ступенем семантичної зв’язності згенерованого тексту і вказує на неконсистентну присутність фрагментів тексту на іншу тематику.

Присутність сенсових розривів пов’язана із ступенем стилістичної зв’язності згенерованого тексту і вказує на присутність абзаців або речень, що

відносяться до однієї теми, проте, логічно не зв’язані із попередніми частинами документа чим порушують його когерентність.

Окрім критеріїв оцінки якості відповіді , існує набір вимог для поставлення запиту до системи, а саме:

– запит має бути тематично цільним і містити терміни що семантично не суперечать одне одному, сукупно чітко описуючи певну предметну галузь;

– запит має відносну семантичну силу, що дорівнює відсотку кількості документів у базі знань системи на пов’язану із темою запиту тематику.

– відносна семантична сила запита не може дорівнювати 0. Запит має містити хоча б один термін, присутній у базі знань системи.

Сформувавши систему критеріїв до процесу тестування нашої системи стає можливим детально оцінити деякі наглядні результати її роботи, тож роздивимось відповіді на найбільш цікаві і показові запити до системи. У додатку 15 наведено згенерований текст відповіді на запит «інформація про астероїди», що однозначно визначає тему «астрономія» і має високу семантичну силу (дорівнює 24%) – нормальний ситуативний запит, який немає труднощів семантичної інтерпретації або браку знань у системі. Наведена відповідь була генерована на основі 2-х текстів: «Астероїди» та «Виникнення та еволюція Всесвіту». Згенерований текст має відповідь на поставлене питання, не має тематичних розривів та повністю збігається із тематикою запиту, тому за описані критерії відповідь отримує максимальні оцінки. Повнота викладу матеріалу також на високому рівні – система сформувала не тільки відповідь, а і додаткові ситуативно корисні дані про температуру астероїдів, склад астероїдної речовини, тощо. Проте у результаті присутні сенсові розриви – «але зробити нічого не можна, і так все і залишається», «його орбіта виявилася не схожа ні на одну досі відому», «поблизу нього було сильно прогріте і пил піддавалася повному або частковому випаровуванню», більшість з яких пов’язана з займенниками і не є критичною для інформаційної цінності документу.

Тепер розглянемо результат роботи системи у випадку браку знань на основі запиту іншої предметної галузі. У додатку 16 наведено згенерований текст відповіді на запит «відмінності класу та інтерфейсу», що однозначно визначає тему «інформаційні технології» і має низьку семантичну силу що дорівнює лише 6%, що є найнижчим показником у тестовій базі знань. Це призвело до того, що система згенерувала відповідь лише на основі одного тексту – «Клас», провівши по-суті семантичне стиснення. Згенерований текст має відповідь на поставлене питання («Суть відмінності класів від інших абстрактних типів»), не має тематичних розривів та повністю збігається із тематикою запиту, має достатній рівень повноти викладу матеріалу – у тексті присутня інформація, ситуативно корисна для користувача – дані про модифікатори доступу, абстракції, успадкування тощо. Оскільки текст був згенерований на основі одного документа, то у ньому нема яскраво виражених сенсових розривів.

Проведені дослідження показали, що брак знань системи є серйозною проблемою, яка може привести не тільки до зниження кількості документів-кандидатів до включення у результуючу відповідь, а і до отримання некоректних результатів. У додатку 17 наведено згенерований текст відповіді на запит «Парне програмування», що однозначно визначає тему «інформаційні технології» і так само має низьку семантичну силу, через що система обрала один текст для генерації «Навчання програмуванню». Не дивлячись на те, що у відповіді немає тематичних розривів, згенерований текст не містить задовільної для користувача відповіді на поставлене питання, оскільки у системі відсутні знання щодо методики парного програмування, тому пошуковий алгоритм обрав найбільш валідний текст лише по одному із ключових слів – а саме, «програмування».

Така поведінка не є неочікуваною, більш того, коли система не може дати розгорнуту відповідь на питання, відсуне в її базі знань, це є цілком валідною і логічною ситуацією. Проте у продуктиві необхідно враховувати, що для коректної роботи додатку необхідно достатньо повно наповнити його базу знань галузевими текстами.

Цікавим прикладом до розгляду є поведінка системи у випадку ускладнення семантичної однозначності запиту користувача. У додатку 18 наведено згенерований текст відповіді на запит «сенс життя людини», що однозначно визначає тему «філософія», має нормальну семантичну силу що дорівнює 21% і неоднозначну семантичну інтерпретацію очікуваної відповіді. Система згенерувала відповідь на основі текстів «Про покликання людини» та «Біологія і соціальне життя», що відносяться до тематики філософії. Згенерований текст не має чіткої відповіді на поставлене питання, проте сукупність отриманої тематичної інформації дає необхідну картину понять суміжних із поставленим питанням – «Результативність філософського пізнання людини, на думку філософа, досягається лише в результаті акту виняткового самосвідомості людиною свого значення», «Для російських філософів проблема відчуження була пов'язана з нагальними, життєвими питаннями, які давали про себе знати у всіх сферах життя», «Тільки в історії людина проходить свій особливий мученицький шлях, в якому всі великі події історії, найстрашніші, самі страждальницькі, виявляються внутрішніми моментами цієї людської долі, бо сама історія – це внутрішнє, повне драматизму звершення долі людини», тощо. Відповідь не має тематичних розривів та повністю збігається із тематикою запиту, має достатній рівень повноти викладу матеріалу – у тексті присутня суміжна інформація про проблему співвідношення біологічного і соціального та має незначні сенсові розриви. Подібна ситуація не залежить від тематики запиту – у додатку 19 наведено згенерований текст відповіді на запит «Значення економіки для держави», що однозначно визначає тему «економіка», має нормальну семантичну силу, що дорівнює 16 % і, так само має неоднозначну семантичну інтерпретацію очікуваної відповіді. Система згенерувала відповідь на основі текстів «Історія і сучасні форми валютної інтеграції», «Поняття економіки» та «Євро-Азіатське економічне співтовариство» що відносяться до цільової тематики. Згенерований текст так само не має чіткої відповіді на поставлене питання, проте має сукупність необхідної суміжної із питанням тематичної інформації, яка дає необхідні для користувача знання, тому відповідь можна вважати коректною і задовільною.

Усього у ході дослідження було проведено 100 тестів із різною складністю питань і різним тематичним напрямом запитів, на кожен з яких була отримана відповідна оцінка, фрагмент графіку значень якої зображено на рис. 3.17.

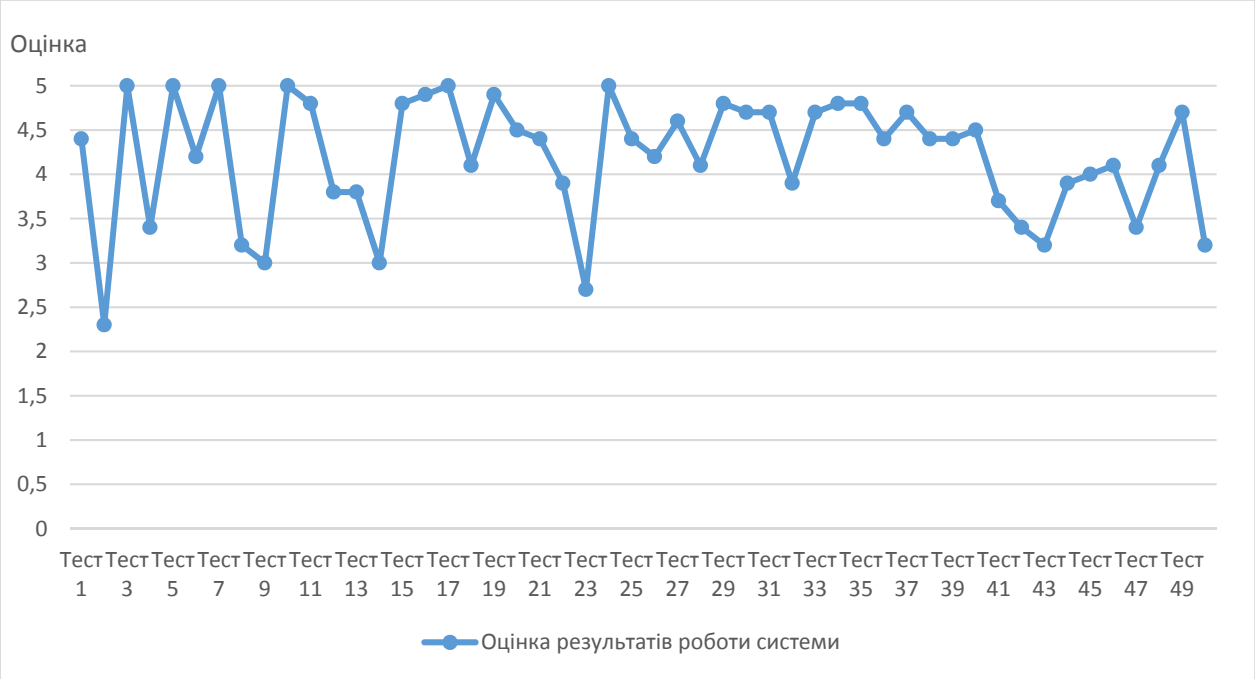

Рис. 3.17. Фрагмент графіку значень експертних оцінок системи для 50 тестів

Середнє значення усіх проведених експертних оцінок склало 0,839, що вказує на задовільні результати роботи системи. Проведення мануального тестування показало наявність проблем у присутності сенсових розривів та необхідності обробки виключних ситуацій із відсутністю необхідних знань у системі, що і знизило сумарно оцінку роботи системи, проте ці факти не мають критичного впливу на алгоритм генерації відповідей. Виконання автоматичних і мануальних тестів системи запит-відповідь із використанням автоматичної генерації текстів показали сукупну адекватність роботи додатку як з точки зору використання семантичної моделі як основи для побудови інтелектуальних пошукових інструментів, так і з точки зору доцільності застосування семантичних моделей у ситуативній генерації текстів.

Висновки до розділу 3

Третій розділ є підсумковим для дисертаційної роботи і стосується імплементації та перевірки висунутих гіпотез протягом дослідження [79]. Були розглянуті моделі класу «запит-відповідь», які були представлені як теоретичними підходами так і конкретними розробками, встановлена теоретична архітектура моделі генерації відповідей та її належність до певного класу. Аналіз найпопулярніших та глобальних систем запит-відповідь показав, що існуючі рішення для слов'янських мов на даний момент не покривають усю множину відкритих проблем у цій галузі, а підхід до побудови таких систем, заснований на генерації текстів є актуальним і цікавим дослідженням.

У главі була теоретично визначена та описана база знань системи, яка покладена у основі процесу генерації відповідей, сформульовані теоретичні аспекти побудови корпусів та колекцій текстів, визначені основні властивості, яких дотримувалася система при складанні та обробці корпусу, описаний та обґрунтований розподіл класів текстів, які відображають репрезентативність вибірки. Окрім того, наведено структуру та детальний опис моделі електронної бази даних, яка покладена в основу базу знань моделі.

Концепція текстового автомату має на увазі реалізацію модуля підтримки безпеки та самоперевірки системи автоматичної обробки текстів. Тому, додатково до безпосереднього функціоналу системи була розроблена новітня модель фільтрації та класифікації текстів за критерієм зв'язності та послідовності викладення матеріалу всередині тексту. Попередньо був проведений аналіз існуючих підходів до створення зазначеного класу систем, який показав необхідність і доцільність застосування розробленого алгоритму для виконання цільової задачі [81, 82].

Використавши кількісні характеристики семантичної моделі документу, у ході роботи ми змогли добитися не тільки повної автоматизації цього етапу, а і завдяки успішній перевірці отриманої підсистеми додатково перевірили адекватність підходу до побудови семантичної моделі документу і доцільність використання створених методів отримання числових характеристик

семантичних властивостей тексту в інтелектуальних системах автоматичного навчання, що було зображено на прикладі використання простої нейронної мережі [81, 82].

У главі показана фінальна програмна архітектура системи, яка реалізована на мові Java, та фінальний інтерфейс системи, на прикладі якого було відображено функціональні можливості системи. Окрім того, на прикладі описаних можливостей була показана реалізація основних вимог до розробки текстового автомату – модульності, рівневості, орієнтованості на роботу з базою знань та безпеки, що довело дотримання обраної у другій главі концепції розробки програмного забезпечення текстового автомату для роботи із базою знань.

Врешті, описана система оцінок і тестування системи в цілому – були проведені паралелі із існуючими підходами та обґрунтована необхідність в розробці нового інструментарію перевірки створеного підходу [80]. У ході викладу матеріалу наведено приклади результатів, що відобразили роботу системи у найгіршому та найкращому світі.

Розроблено систему оцінок, що ґрунтується на вже сформованих алгоритмах та даних, і дозволяє з одного боку перевірити тематичну спрямованість відповідей, а з іншого – уникнути ситуації виникнення глибоких семантичних розривів у відповідях. Проведена оцінка отриманої моделі генерації відповідей на основі бального метода [80].

Кількісні результати оцінок, отриманих на обидвох етапах співали і доводять коректність роботи створеної системи в цілому, доцільність її використання як базового інструменту вилучення знань з колекцій текстів та адекватність розроблених моделей.

Основні положення цього розділу викладені у публікаціях автора [79, 80, 81, 82].

ВИСНОВКИ

В дисертаційній роботі розв’язано важливу науково-технічну задачу автоматизації обробки семантично-неструктурованих документів для генерації відповідей в пошукових системах. Отримані нові обґрунтовані результати, які відповідно до поставленої мети дають вирішення актуальної задачі побудови моделі отримання знань із неструктурованих джерел.

Основні наукові та практичні результати полягають в наступному:

1) Проведено аналіз існуючих підходів до побудови математичної моделі представлення мови та вилучення інформації з неструктурованої бази знань. За його результатами зроблено висновок про доцільність використання моделі м’якого розуміння Леонтьевої у основі процесів розуміння семантики тексту і генерації відповідей у пошукових системах. До переваг обраної моделі відносяться орієнтованість на складні системи інтелектуальної оброки текстових даних та на першочергову обробку семантичних характеристик знань.

2) Отримала подальший розвиток теорія «Сенс-Текст» та модель м’якого розуміння знань у пошуковій системі вилучення інформації із неструктурованої бази знань через подолання обмежень тлумачно-комбінаторного словника, використання якого вимагає глибокого ручного опису кожного компонента текстової бази знань, що є складною глобальною задачею прикладного застосування цих моделей.

3) Розроблено математичну модель текстового автомату пошукової системи, дотримання концепцій якого дозволило описати програмну архітектуру системи як комплекс послідовних рівнів обробки бази знань. Розроблені та реалізовані головні критерії до функціонування системи, що забезпечило її архітектурну консистентність та обґрунтованість.

4) Розроблено семантичну модель текстового знання, яка використовує у своїй роботі як базові методи синтаксичної обробки тексту, так і науково нові застосування латентно-семантичного аналізу, що у комплексі дозволило створити семантичну модель без залучення додаткових лінгвістичних знань.

5) Реалізовано семантичну модель текстового знання, яка представлена у вигляді відкритого програмного API, що дозволяє розробникам комп'ютерних систем отримувати кількісні значення семантичних характеристик текстових даних без необхідності залучення лінгвістичних знань.

6) Розроблено математичну модель автоматичної класифікації документів за їх семантичною зв'язністю, яка дозволила впровадити надійність процесу наповнення неструктурованої бази знань новими знаннями.

7) Створено математичну модель генерації відповідей на основі неструктурованої бази знань в пошукових системах. Розроблено комп'ютерну систему, яка реалізує створену модель генерації відповідей і дозволяє отримувати на основі семантичних моделей неструктурованих знань нові знання, що містять відповіді на поставлені користувачем питання.

8) Розроблено систему оцінювання якості моделі генерації відповідей на основі неструктурованої бази знань та пошукової системи. Проведене дослідження показує доцільність використання процесу породження текстів для формування відповіді на поставлене питання. Отримані результати проведеного автоматичного і мануального тестування системи вказують на коректність і адекватність роботи додатку.

Розроблену комп'ютерну систему впроваджено у міський комунальний заклад культури «Централізована система бібліотек для дітей» м. Дніпро як пошуковий інструмент обробки електронних текстів, у ТОВ «Сітал Україна» як засіб автоматичної генерації текстових інструкцій та у АТ «ДніпроАзот» як інструмент покращення процесів пошуку в системах електронного документообігу. Результати дисертаційної роботи опубліковані в 14 наукових працях, в тому числі 8 статей – у журналах, рекомендованих МОН України для публікації результатів дисертацій, та закордонних виданнях, та 6 – у тезах доповідей та трудах міжнародних та всеукраїнських конференцій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Леонтьєва М.М. Автоматичне розуміння текстів: Системи, моделі, ресурси / М.М. Леонтьєва // М.: Академія, 2006 – 304 с.
2. Рут М.Е. Курс загальної лінгвістики / М.Е. Рут // видавництво Уральського університету, 1999 – 432 с.
3. Лукашевич Н.В. Моделі і методи автоматичної обробки неструктурованої інформації на основі бази знань онтологічного типу / Н.В. Лукашевич // дис. к.т.н, 2014 – 312 с.
4. Chomsky N. Syntactic Structures / N. Chomsky // Berlin, New York: Mouton de Gruyter, 2002 – 117 p.
5. Мельчук І.А. Досвід теорії лінгвістичних моделей «Сенс ↔ Текст» / І.А. Мельчук // М.: Школа, 2-е видання, доп. 1999 – 353 с.
6. Волкова І.А. Введення в комп'ютерну лінгвістику. Практичні аспекти створення лінгвістичних процесорів / І.А. Волкова // М.: МГУ, 2006 – 43 с.
7. Апресян Ю. Лінгвістичне забезпечення системи ЕТАП-2 / Ю.Д. Апресян, І. М. Богуславський, Л. Л. Іомдін // Каліфорнійський університет, 1989 – 294 с.
8. Bolshakov I.A. The Meaning ↔ Text Model: Thirty Years After / I.A. Bolshakov, A.F. Gelbukh // International Forum on Information and Documentation, №1, 2000.
9. Мозговий М.В. Машинний семантичний аналіз російської мови та її застосування / М.В. Мозговий // дис. к. ф-м. н, 2006 – 116 с.
10. Bolshakov I.A. Computational Linguistics. Models, Resources, Applications / I.A. Bolshakov, A.F. Gelbukh // Computational Linguistics № 32(3), 2004 – 186 p.
11. Уваров А.Н. Інверсія управління та впровадження залежностей / А.Н. Уваров // Міжнародний науковий журнал «Символ науки» №10-1, 2016 – с. 28-32.
12. Луканін А.В. Автоматична обробка природної мови: навчальний посібник / А.В. Луканін // Челябінськ: Видавничий центр ЮУрГУ, 2011 – 70 с.

13. Малинина Ю.В. Методи і інструменти конструювання та оптимізації програм / Ю.В. Малинина // Сер. «Конструювання і оптимізація програм» Російська академія наук, Сибірське відділення, Інститут систем інформатики ім. А. П. Єршова, 2005 – 274 с.
14. Волковський О.С. Семантичний аналіз вмісту web-додатків / О.С. Волковський, Є.Р. Ковилін // Системні технології, 2014 – с. 47-54.
15. Tarasov D.S. Natural Language Generation, paraphrase and automatic generalization of users' reviews using recurrent neural networks / D.S. Tarasov // Computational Linguistics and Intellectual Technologies, No.14 (Volume 1), 2015.
16. AlchemyAPI service [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/watson/alchemy-api.html. – Заг. с екрану. – Перевірено: 10.02.20.
17. Jackson P. Natural Language Processing for Online Applications: Text Retrieval, Extraction, and Categorization / P. Jackson, I. Moulinier // Philadelphia: John Benjamins B.V., 2002 – p. 117.
18. Yildiz B. Motivating Ontology-Driven Information Extraction / B. Yildiz, S. Miksch // International Conference on Semantic Web and Digital Libraries, Bangalore, India, 2007.
19. Deliyanni A. Logic and Semantic Networks / A. Deliyanni, R. Kowalski // Communications of the ACM. Vol. 22 no. 3, 1979 – pp. 184-192.
20. Hope D. A Graph-Based Soft Clustering Algorithm Applied to Word Sense Induction / D. Hope, B. Keller // Computational Linguistics and Intelligent Text Processing: 14th International Conference, 2013 – pp. 368-381.
21. Palla G. Uncovering the overlapping community structure of complex networks in nature and society / G. Palla, I. Derenyi, I. Farkas, T. Vicsek // Nature. Vol. 435, 2005 – pp. 814-818.
22. Hearst M. A. Automatic Acquisition of Hyponyms from Large Text Corpora / M. A. Hearst // Proceedings of the 14th Conference on Computational Linguistics (COLING '92) – Volume 2, 1992 – pp. 539-545.

23. Крижанівський А. А. Підхід до автоматизованого побудови лексичної онтології на основі даних Вікісловника / А.А. Крижанівський, А.В. Смирнов // Відомості Російської академії наук. Теорія і системи управління № 2, 2013 – с. 53-63.
24. Волковський О.С. Analysis of the modern approaches to the problem of the automatic text generation in the natural language / О.С. Волковський, Є.Р. Ковилін // Системні технології. Регіональний збірник міжвузівських наукових праць №5 (106), 2016 – с. 3-12.
25. Волковський О.С. Computer methods for compiling an entry of explanatory combinatorial dictionary belonging to "Meaning↔Text" theory within the task of text automatic generation / О.С. Волковський, Є.Р. Ковилін // Математичне моделювання – №2 (39) Кам'янське, 2018 – с. 9-18.
26. Волковський О.С. Автоматична побудова семантичної мережі тексту у системах запит-відповідь / О.С. Волковський, Є.Р. Ковилін // Матеріали міжнародної науково-практичної конференції «Інформаційні технології та комп'ютерне моделювання», 2017 – с. 386-389.
27. Волковський О.С. Система автоматичного аналізу текстів природною мовою / О.С. Волковський, Є.Р. Ковилін // Матеріали міжнародної науково-технічної конференції «Інформаційні технології в металургії та машинобудуванні» 2017 – с. 112.
28. Овчіева Ю.А. Семантична мережа – перспективна платформа для системи управління знаннями / Ю.А. Овчіева, С.А. Смирнов // Вісник університету (державний університет управління) №3, 2015 – с. 14-16.
29. Усталов Д.А. Моделі, методи та алгоритми побудови семантичної мережі слів для задач обробки природної мови / Д.А. Усталов // дис. канд. фіз.-мат.н., 2017 – 129 с.
30. Shapiro S.C. Encyclopedia of Artificial Intelligence. 2nd edition / S. C. Shapiro // New York, USA: John Wiley & Sons, Inc., 1992 – 1724 p.
31. Storey V. C. Understanding Semantic Relationships / V. C. Storey // The VLDB Journal. Vol. 2, no. 4., 1993 – pp. 455-488.

32. Гаврилова Т.А. Бази знань інтелектуальних систем / Т.А. Гаврилова, В. Ф. Хорошевський // СПб: Пітер, 2000 – 384 с.
33. Воронін В.М. Латентний семантичний аналіз і розуміння тексту / В. М. Воронін // Психологічний вісник Уральського державного університету. Вип. 9, 2010 – с. 15-27.
34. Jones K. S. A statistical interpretation of term specificity and its application in retrieval / K.S. Jones // Journal of Documentation – MCB University – Vol. 60, no. 5. 2004 – pp. 493-502.
35. Федюшкін Н.А. Латентно-семантичний аналіз тексту / Н.А. Федюшкін, С.А. Федосін // Актуальні проблеми технічних наук в Росії і за кордоном. Збірник наукових праць за підсумками міжнародної науково-практичної конференції – № 5, 2018 – 111 с.
36. Що таке Java? [Електронний ресурс] – Режим доступу до ресурсу: https://java.com/ru/about/whatis_java.jsp. – Заг. с екрану. – Перевірено: 13.03.19
37. Class BreakIterator [Електронний ресурс] – Режим доступу до ресурсу: https://docs.oracle.com/javase/7/docs/api/java/text/BreakIterator.html. – Заг. с екрану. – Перевірено: 15.12.19.
38. Turkanov G. Modified Naive Bayes with Hurst exponent as quantitative measure of data mutual dependence / G. Turkanov // Proceedings of Yandex School of Data Analysis Conference. Machine Learning: Prospects and Applications, 2015.
39. Плунгян В.А. Національний корпус російської мови: 2006-2008. Нові результати і перспективи / В. А. Плунгян // М. 2009 – 502 с.
40. Java Naive Bayes Classifier [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/ptnplanet/Java-Naive-Bayes-lassifier/blob/master/README.md. – Заг. с екрану. – Перевірено: 10.01.20.
41. Жердєва М.В. Стемінг і лематизації в lucene. Net / М.В. Жердєва // Вістник московського державного університету лісу, 2016 – с. 131-134.
42. Porter M.F. An algorithm for suffix stripping / M.F. Porter // Program, Vol 40 Iss: 3, 2006 – pp. 211-218.

43. Cristian M. A survey of stemming algorithms in information retrieval / Cristian M., Antonio A., Imbert R. Ramírez J. // Information research,Vol. 19, No. 1, 2014 – pp. 605-625,.
44. Y-stemmer / Yatsko Viatcheslav – Yatsko's Computational Linguistics Laboratory [Електронний ресурс], – Режим доступу до ресурсу:http://yatsko.zohosites.com/y-stemmer.html. – Заг. с екрану. – Перевірено: 05.11.19.
45. Prabin L. Clustering system based on text mining using the k-means algorithm, / L. Prabin // Bachelor's thesis (UAS) of Information Technology «Text Mining and Clustering 2013», 2013 – 48 p.
46. Kovalenko A., «The Stemmer. Morphological analysis for small search engines», System Administrator Vol №1(1), 2002. [Електронний ресурс], – Режим доступу до ресурсу: http://samag.ru/archive/article/47 – Заг. с екрану. – Перевірено: 03.12.19.
47. Segalovich I. A fast morphological algorithm with unknown word guessing induced by a dictionary for a web search engine / I. Segalovich // Proceedings of the International Conference on Machine Learning; Models, Technologies and Applications. MLMTA'03, Las Vegas, Nevada, USA, 2003 – pp. 721-729,.
48. Левенштейн В.І. Доповідь Академій Наук СРСР / В.І. Левенштейн // М: Т. 163, №4, 1965 – с. 845-848.
49. Babichev S. Estimation of the inductive model of objects clustering stability based on the k-means algorithm for different levels of data noise / S. Babichev, V. Lytvynenko, M. A. Taif //Радіоелектроніка, інформатика, управління № 4, 2016 – с. 54-60.
50. Половікова О.М. Використання евклидової і манхеттенської відстаней в якості міри близькості для вирішення задачі класифікації / О.М. Половікова, В.В. Фокіна // Відомості АлтГУ №1-1, 2010.
51. Іванівський С.О. Алгоритми обчислювальної геометрії. Опуклі оболонки: зв'язок із завданням сортування і оптимальні алгоритми / С.О. Іванівський,

О.С. Преображенський, С. К. Сімончік // Комп'ютерні інструменти в освіті (2), 2007 – с. 6-18.

52. Java Program to Implement Gift Wrapping Algorithm in Two Dimensions [Електронний ресурс] – Режим доступу до ресурсу: https://www.sanfoundry.com/java-program-implement-gift-wrapping-algorithm-two-dimensions/. – Заг. с екрану. – Перевірено: 16.02.20.
53. XChart [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/knowm/XChart/blob/develop/README.md. – Заг. с екрану. – Перевірено: 16.02.20.
54. Болдас М.В. Генерація текстів на природній мові – теорії, методи, технології / М.В. Болдас, Є.Г. Соколова // НТІ. Сер. 2. Інформаційні процеси і системи, 2006 – с.1-15.
55. Генератор текста [Електронний ресурс] – Режим доступу до ресурсу: https://online-generators.ru/text. – Заг. с екрану. – Перевірено: 16.02.20.
56. Volkovsky O.S. Mathematical model for constructing the semantic network of a scientific text / O.S. Volkovsky, Y. R. Kovylin. // Modern engineering and innovative technologies, 2020 – pp. 128-133.
57. Волковський О.С. Mathematical model for automatic creation the semantic thesaurus for the scientific text / Є.Р. Ковилін, О.С. Волковський // Системні технології. Регіональний збірник міжвузівських наукових праць №6 (125), 2019 – с. 82-88.
58. Volkovsky O.S. Computer System of Building of the Semantic Model of the Document / O.S. Volkovsky, Y. R. Kovylin // 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP), Lviv, 2018 – pp. 322-327.
59. Ojokoh B. A Review of Question Answering Systems. Journal of Web Engineering 17 / B. Ojokoh //, 2019 – pp. 717-758.
60. Павельева Т.Ю. Вивчення колокацій на основі лінгвістичних корпусів текстів / Т.Ю. Павельева // Вісник ТГУ №3-4, 2016 – с. 155-156.

61. Курносов Ю.В. Алгебра аналітики. секрети майстерності в аналітичній роботі / Ю.В. Курносов // Москва, 2015 – 288 с.
62. Абрамов В.Є. Статистичний аналіз зв'язності текстів з суспільно-політичної тематики / В.Є. Абрамов // Тр. 13-й Всерос. наук. конф. «Електронні бібліотеки: перспективні методи і технології, електронні колекції», 2011 – с. 127-133.
63. Павлов А.С. Метод виявлення масово породжених неприродних текстів на основі аналізу тематичної структури / А.С. Павлов, Б.В. Добров // Обчислювальні методи і програмування: нові обчислювальні технології Т. 12, 2011 – с. 58-72.
64. Castillo D. Know your neighbors: Web spam detection using the web topology / D. Castillo A. Donato, V. Gionis, F. Silvestri // In Int'l ACM SIGIR, 2007.
65. Dutta S. Evaluating a neural multi-turn chatbot using BLEU score / S. Dutta // Universität des Saarlandes, 2019 – p.10.
66. Боярський К.К. Виявлення анафорических відносин при автоматичному аналізі тексту / К.К. Боярський, Є.О. Канівський, О.В. Степукова // Науково-технічний вісник інформаційних технологій, механіки і оптики №5 (87), 2013.
67. Simple neural network [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/jlmd/SimpleNeuralNetwork/blob/master/README.md – Заг. с екрану. – Перевірено: 16.02.20
68. Spring Boot 2 [Електронний ресурс] – Режим доступу до ресурсу: https://spring.io/blog/2018/05/09/spring-boot-2-0-2. – Заг. с екрану. – Перевірено: 16.02.20
69. Develop Java applications with Oracle Database [Електронний ресурс] – Режим доступу до ресурсу: https://www.oracle.com/database/technologies /appdev/ jdbc.html. – Заг. с екрану. – Перевірено: 16.02.20
70. PostgreSQL: The World's Most Advanced Open Source Relational Database [Електронний ресурс] – Режим доступу до ресурсу: https://www.postgresql.org/. – Заг. с екрану. – Перевірено: 16.02.20

71. Романов С.С. Переваги, недоліки і альтернативи об'єктно-реляційного відображення (ORM) / С.С. Романов // Таврійський науковий оглядач №12-2 (17), 2016.
72. Hibernate. Everything data [Електронний ресурс] – Режим доступу до ресурсу: https://hibernate.org/. – Заг. с екрану. – Перевірено: 16.02.20
73. Swing (Java Foundation Classes) [Електронний ресурс] – Режим доступу до ресурсу: https://docs.oracle.com/javase/6/docs/technotes/guides/swing/. – Заг. с екрану. – Перевірено: 16.02.20
74. Алтуфьев М.Ю., Лаптєв В.В. Розробка через тестування в автоматизованій системі управління «Вуз». / М.Ю. Алтуфьев, В.В. Лаптєв // Вісник АГТУ №4, 2007.
75. Харламов А.А. Інструментарій для інформаційно-аналітичної експертної оцінки наукової продукції з метою виявлення пріоритетних наукових напрямів та колективів / А.А. Харламов, Б.І. Васін // Вісник Московського державного лінгвістичного університету №6 (797), 2018.
76. Соловйов А.А. Алгоритми валідації відповідей в завданні питально-відповідь пошуку / А.А. Соловйов // Вісник Воронезького ун-ту. Сер.: Системний аналіз та інформаційні технології № 2, 2011 – с. 181-188.
77. Данелян Т.Я. Формальні методи експертних оцінок / Т.Я. Данелян // Статистика і економіка №1, 2015.
78. Бєлов В.М. Метод бальної оцінки показників коефіцієнтів вагомості / В.М. Бєлов // Агроінженерія №4, 2009.
79. Волковський О.С. Комп'ютерна система інтелектуального семантичного пошуку з використанням генерації текстів / О.С. Волковський, Є.Р. Ковилін // Вісник Херсонського національного університету №3 (66), 2018 – с.238-245.
80. Волковський О.С. Модель автоматичної оцінки адекватності комп'ютерних систем «запит-відповідь» з використанням генерації текстів / О.С. Волковський, Є.Р. Ковилін // Системні технології. Регіональний збірник міжвузівських наукових праць №4 (129), 2020 – с. 50-58.

81. Волковський О.С. Комп'ютерна система автоматичного визначення зв'язності тексту / О.С. Волковський, Є.Р. Ковилін // Системні технології. Регіональний збірник міжвузівських наукових праць №1 (112), 2017 – с. 11-17.
82. Волковський О.С. Комп'ютерна система автоматичного аналізу промислових інструкцій / О.С. Волковський, Є.Р. Ковилін // Системні технології. Регіональний збірник міжвузівських наукових праць №3 (116), 2018 – с. 28-37.
83. Сисоєв П.В. Лінгвістичний корпус в методиці навчання іноземних мов / П.В. Сисоєв // Журнал «Мова і культура», 2010.
84. Большакова Є.І. Автоматична обробка текстів природною мовою та комп'ютерна лінгвістика: Навчальний Посібник / Є.І. Большакова, Е.С. Клишінскій, Д.В. Ланде, А.А. Носков, О.В. Пєскова, Є.В. Ягунова // 2011 – 272 с.
85. Бондаренюк А.В. Автоматичне реферування тексту / А.В. Бондаренюк // БДУ, Мінськ, 2001.
86. Анісімов А. В. Метод обчислення семантичної близькості зв'язності між словами природного мови / А. В. Анісімов, А. А. Марченко // Кібернетика і системний аналіз № 4, 2011.
87. Lei Y. Onto Weaver An ontology-based approach to the design of data-intensive Web sites / Y. Lei, E. Motta, J. Domingue // Journal of Web Engineering. V. 4. № 3, 2005 – p. 244-262.
88. Levenstein V.I. Binary Codes capable of correcting delation, insertion and re-vercals / V.I. Levenstein // Cybernetics and Control Theory V.10. No. 8, 1966 – p. 707-710.
89. Li L. A software framework for matchmaking based on semantic web technology. / L. Li, I. Horrocks // Proceed. Of the 12th International World Wide Web Conference (WWW 2003). ACM Press, 2003. – p. 331-339.

90. Mádche A. SEAL A framework for developing SEmantic portALs / A. Mádche, S. Staab, N. Stojanovic // Proceed, of the 18th British national conference on databases. Oxford: Springer, 2001. – p. 1-22.
91. Mádche A. Measuring Similarity between Ontology / A. Mádche, S. Staab // Proceed, of the European Conference on Knowledge Acquisition and Management, Madrid, 2002. – p. 251-263.
92. Maier R. Implementing process-oriented Knowledge Management Strategies / R. Maier, U. Remus // Journal of Knowledge Management № 4, 2002 – p. 62-74.
93. Maier R. Knowledge management systems: Information and communication technologies for knowledge management / R. Maier // Berlin Heidelberg: Springer Verlag, 2004. - 635 p.
94. Makelá E. A tool for creating Semantic Web portals / E. Makelá // Proceed, of the 3rd international Semantic Web conference, 2004. - p. 797-811.
95. MARC standards. – [Електронний ресурс] – 2007. Режим доступу до ресурсу: http://www.loc.gov/marc. – Заг. с екрану. – Перевірено: 19.04.20
96. Marwick A.D. Knowledge Management Technology / A.D. Marwick // IBM System Journal № 4, 2001 – p. 814-830.
97. McElroy M.W. The new knowledge management: complexity, learning, and sustainable innovation / M.W. McElroy // Butterworth-Heinemann, 2003.
98. McGrath R.E. Semantic infrastructure for a ubiquitous computing environment [Електронний ресурс] – 2005. Режим доступу до ресурсу: http://www.cs.uiuc.edu/research/techreport.php?report=UIUCDCS-R-2005-2587&download=pdf. – Заг. с екрану. – 19.04.20
99. McGuinness D.L. Ontologies Come of Age / D.L. McGuinness //Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential. MIT Press, 2003.
100. Mentzas G. Knowledge asset management beyond the product-centered and the process-centric approuch / G. Mentzas, D. Apostolou, R. Young, A. Abecker // Berlin: Springer-Verlag, 2003 – 200 p.

101. Meroño A. KM strategy and instruments alignment: helping SMEs to choose. / A. Meroño, C. López, R. Sabater // The 5th Europian Conference on organizational knowledge, learning and capabilities, 2004 – 186-216 p.
102. Merrill M. D. Knowledge Objects. [Електронний ресурс] – 2010. Режим доступу до ресурсу: http://cito.byuh.edu/merrill/text/papers/Knowledge Objects.PDF. – Заг. с екрану. – 19.04.20
103. Mertins K. Knowledge Management: concepts and best practices (2nd ed.) / K. Mertins, P. Heisig, J. Vorbeck // Berlin: Springer Verlag, 2003. – p. 383.
104. Mika P. Towards a New Synthesis of Ontology Technology and Knowledge Management / P. Mika, H. Akkermans // Knowledge Engineering Review № 4, 2004 – p. 317-345.
105. Mizoguchi R. A step towards ontological engineering / R. Mizoguchi // Proceed, of the 12th National Conference on AI of JSAI, 1998. - p. 24-31.
106. Mizoguchi R. Construction and Deployment of a Plant Ontology / R. Mizoguchi // Proceed, of the 12th European Workshop on Knowledge Acquisition, Modeling and Management, 2000. - p. 113-128.
107. Mondeca ITM White Paper [Електронний ресурс] – 2004 - Режим доступу до ресурсу: http ://www.mondeca.com/itm-wp-introducti on-en.pdf. – Заг. с екрану. – 19.04.20.

ДОДАТОК 1. Список публікацій здобувача за темою дисертації

Наукові праці, в яких опубліковані основні результати дисертації:

1. Волковський О.С. Computer methods for compiling an entry of explanatory combinatorial dictionary belonging to «Meaning↔Text» theory within the task of text automatic generation / О.С. Волковський, Є.Р. Ковилін // Математичне моделювання // науковий журнал – Кам'янське, 2018. – №2 (39). – с. 9–18. Видання включено до НМБ Index Copernicus International.
2. Волковський О.С. Analysis of the modern approaches to the problem of the automatic text generation in the natural language / О.С. Волковський, Є.Р. Ковилін // Системні технології. Регіональний збірник міжвузівських наукових праць // науковий журнал – Дніпро, 2016. – №5 (106). – с. 3–12. Видання включено до НМБ Index Copernicus International.
3. Волковський О.С. Комп'ютерна система автоматичного визначення зв'язності тексту / О.С. Волковський, Є.Р. Ковилін // Системні технології. Регіональний збірник міжвузівських наукових праць. // науковий журнал – Дніпро, 2017. – №1 (112). – с. 11–17. Видання включено до НМБ Index Copernicus International..
4. Волковський О.С. Комп'ютерна система автоматичного аналізу промислових інструкцій. / О.С. Волковський, Є.Р. Ковилін // Системні технології. Регіональний збірник міжвузівських наукових праць. // науковий журнал – Дніпро, 2018. – №3 (116). – с. 28–37. Видання включено до НМБ Index Copernicus International.
5. Волковський О.С. Комп'ютерна система інтелектуального семантичного пошуку з використанням генерації текстів / О.С. Волковський, Є.Р. Ковилін // Вісник Херсонського національного технічного університету // науковий журнал – Херсон, 2018. – №3 (66). – с. 238 –245. Видання включено до НМБ Google Scholar.

6. Волковський О.С. Mathematical model for automatic creation the semantic thesaurus for the scientific text / О.С. Волковський, Є.Р. Ковилін // Системні технології. Регіональний збірник міжвузівських наукових праць. // науковий журнал – Дніпро, 2019. –. №6 (125). – с. 82–88. Видання включено до НМБ Index Copernicus International.
7. Волковський О.С. Модель автоматичної оцінки адекватності комп'ютерних систем «запит-відповідь» з використанням генерації текстів / О.С. Волковський, Є.Р. Ковилін // Системні технології. Регіональний збірник міжвузівських наукових праць. // науковий журнал – Дніпро, 2020 – №4 (129). – с. 50–58. Видання включено до НМБ Index Copernicus International.
8. Volkovsky O.S. Matematical model for constructing the semantic network of a scientific text / O.S. Volkovsky, Y. R. Kovylin. // Modern engineering and innovative technologies // науковий журнал – Карлсруе, Німеччина, 2020 – №11 – с. 128 – 133. Видання включено до НМБ Index Copernicus International.

Публікації апробаційного характеру:

9. O.S. Volkovsky. Computer System of Building of the Semantic Model of the Document / O.S. Volkovsky, Y. R. Kovylin. // 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP): міжнародна науково-практична конференція, 21-25 серпня 2018 р.: тези доп. – Львів, 2018 – с. 322–327. (Конференція включена до НМБ Scopus).
10. Волковський О.С. Система автоматичного аналізу текстів природною мовою / О.С. Волковський, Є.Р. Ковилін // Інформаційні технології в металургії та машинобудуванні: міжнародна науково-технічна конференція, 28–30 березня 2017р.: тези доп. – Дніпро, 2017 – с. 112.
11. Волковський О.С. Автоматична побудова семантичної мережі тексту у системах запит-відповідь / О.С. Волковський, Є.Р. Ковилін // Інформаційні технології та комп'ютерне моделювання: міжнародна науково-практична

конференція, 15 – 20 травня 2017 р.: тези доп. – Івано-Франківськ, 2017– с. 386 – 389.

12. Волковський О.С. Комп’ютерна система автоматичного аналізу промислових інструкцій.. / О.С. Волковський, Є.Р. Ковилін // Інформаційні технології в металургії та машинобудуванні: міжнародна науково-технічна конференція, 27–29 березня 2018 р.: тези доп. – Дніпро, 2018. – с. 124.
13. Волковський О.С. Комп’ютерна модель семантичної мережі документу в системі запит-відповідь / О.С. Волковський, Є.Р. Ковилін // Проблеми математичного моделювання: всеукраїнська науково-методична конференція, 23–25 травня 2018 р.: тези доп. – Кам’янське, 2018. – с.33–36.
14. Волковський О.С. Комп’ютерна система інтелектуального семантичного пошуку з використанням генерації текстів / О.С. Волковський, Є.Р. Ковилін // Матеріали XIX міжнародної конференції з математичного моделювання, присвяченої 250-річчю з дня народження Жозефа Фур’є, 17–21 вересня 2018 р.: тези доп. – Лазурне, 2018. – с. 49.

ДОДАТОК 2. Акти впровадження результатів дисертаційної роботи

МКЗК «Централізована система бібліотек для дітей»

ЗАТВЕРДЖУЮ
Виконуюча обов'язки
директора МКЗК ЦСБД
_____ М.Ю. Завгородня
"24" грудня 2018 р.

АКТ
впровадження наукових результатів дисертаційної роботи
аспіранта кафедри комп'ютерних наук та інформаційних технологій
Дніпровського національного університету імені Олеся Гончара
Ковиліна Єгора Романовича
«Система "запит-відповідь" штучного інтелекту з використанням
автоматичної генерації текстів»

Результати досліджень, виконаних у кандидатській дисертації Ковиліна Є.Р., були впроваджені у процес роботи Міського комунального закладу культури «Централізована система бібліотек для дітей».

Розроблена комп'ютерна система "запит-відповідь" інтелектуальної генерації відповідей на основі предметних корпусів текстів бібліотеки використовується задля автоматизування семантичного пошуку інформації у електронних каталогах.

Товариство з обмеженою відповідальністю «СІТАЛ УКРАЇНА»

49000, Україна, м. Дніпро, вул. Святослава Хороброго, 44 т/ф (0562) 341-933, 325-486
ЄДРПОУ 34883645 ІПН 348836404650
e-mail: info@sital.ua
http://www.sital.ua

ЗАТВЕРДЖУЮ
Генеральний директор
ТОВ «СІТАЛ УКРАЇНА»
Малюкін К.В.
_____
«18» травня 2020 р.

АКТ

Про впровадження і практичне використання результатів дисертаційного дослідження Ковиліна Єгора Романовича «Модель генерації відповідей в пошукових системах на основі неструктурованої бази знань»

Цей акт складений в підтвердженні того, що комп'ютерна система, розроблена як результат теоретичних і експериментальних досліджень наведених у дисертаційній роботі Ковиліна Є.Р. «Модель генерації відповідей в пошукових системах на основі неструктурованої бази знань», застосовується як пошуковий інструмент при роботі із текстовими базами даних.

Розроблена комп'ютерна система автоматизує процеси пошуку інформації чим спрощує роб оту із базою текстових даних на підприємстві і оптимізує існуючі бізнес-процеси по виконанню замовлення клієнта.

Провідний фахівець Зайцев Р.В.
ТОВ «СІТАЛ УКРАЇНА»

АКТ

Про впровадження і практичне використання результатів дисертаційного дослідження Ковиліна Єгора Романовича «Модель генерації відповідей в пошукових системах на основі неструктурованої бази знань»

Цей акт складений в підтвердженні того, що комп'ютерна система, розроблена як результат теоретичних і експериментальних досліджень наведених у дисертаційній роботі Ковиліна Є.Р. «Модель генерації відповідей в пошукових системах на основі неструктурованої бази знань», застосовується як інструмент покращення процесів пошуку в системах електронного документообігу.

Комп'ютерна система дозволяє знаходити електронні текстові документи за їх семантичними властивостями чим вдосконалює та спрощує процес пошуку інформації у базі текстових даних.

Начальник відділу АСКВ АТ «ДНІПРОАЗОТ» Луканов В.А.

ДОДАТОК 3. Приклад тексту до обробки моделлю на тему «Методологія розробки програмного забезпечення»

Початковий текст

Методологія розробки програмного забезпечення – сукупність методів, що застосовуються на різних стадіях життєвого циклу програмного забезпечення і мають загальний філософський підхід. Кожна методологія характеризується своїм філософським підходом або основними принципами. Ці принципи, від яких залежить ефективність всієї методології, зазвичай можна коротко сформулювати і легко пояснити: узгодженою множиною моделей методів, які реалізують дану методологію; концепціями (поняттями), що дозволяють більш точно визначити методи. В окремому випадку, коли методологія застосовується на стадії програмування (конструювання), її зазвичай називають парадигмою програмування. Можна простежити три шляхи виникнення методології. По-перше, вони можуть бути виразом практичного досвіду. По-друге, методології можуть походити від однієї з чотирьох моделей алгоритму: абстрактна машина Тюрінга (імперативне програмування), рекурсивні функції Гілберта і Акермана (структурне програмування), лямбда-числення Черча (функціональне програмування), нормальний алгорифм Маркова (логічне програмування). По-третє, методології можна пояснити через відображення однієї з трьох структур мови моделювання на структуру мови програмування. Складовими частинами можуть бути структура даних, структура управління і логіка. Кожне з дев'яти відображень визначає або методологію, або достатньо серйозний метод програмування. Наприклад, відображення логіка-логіка лежить в основі логічного програмування. При підході до методології, яка має ядро (англ. Core), відповідне способу опису алгоритму, і додаткові особливості, можна виділити наступні п'ять основних ядер методологій: Методологія імперативного програмування; Методологія ООП; Методологія функціонального програмування; Методологія логічного програмування; Методологія програмування в обмеженнях;

Можна помітити, що ці методології знаходяться на шкалі від навігаційних (покрокове управління виконанням) до специфікаційних (визначення вимог до

результату). За топологічною специфікою: Специфіка (топологічна специфіка) – спосіб вибору методів для уточнення ядра методології. Критерієм якості тієї чи іншої топології може можуть бути загальні витрати на розробку ПО. У свою чергу, витрати на розробку залежать серед іншого від ключових мовних абстракцій: абстракції даних, управління і модульності. Наприклад, в імперативній методології можна дотримуватися методів структурного програмування, що дає більш вигідну топологію з точки зору мовних абстракцій. Результатом є методологія структурного програмування. За специфікою реалізації: у відповідності з архітектурою апаратного забезпечення, реалізація може бути централізованою або паралельною. Наприклад, методологія (імперативного) паралельного програмування, методологія логічного паралельного програмування. крім того, методологія може бути гібридною. Наприклад, найбільш часта є суміш функціонального і логічного програмування. Проводяться дослідження і по уніфікації методології програмування. Мови програмування можуть добре підтримувати ті чи інші методології, але це не означає, що деяку мову взагалі не можна використовувати з невластивою їй методологією, а тільки те, що потрібно затратити більше зусиль і ресурсів. Методології програмування розрізняються по загальним витратам на вирішення завдань з різними характеристиками (наукові розрахунки, фінансові завдання, системи реального часу). Масштаб завдань і ефективність створюваного програмного забезпечення також є важливими факторами при виборі методології програмування

Оброблений текст

[word = "Методолог", stema = "методолог", pos = S", weight = 28"] [word = "розробки", stema = "розробк", pos = S", weight = 3"] [word = "програмного", stema = "програм", pos = A", weight = 25"] [word = "забезпечення", stema = "забезпечен", pos = S", weight = 4"] [word = "сукупн", stema = "сукупн", pos = S", weight = 1"] [word = "метод", stema = "метод", pos = S", weight = 6"] [word = "застосовуються", stema = "застосов", pos = V", weight = 2"] [word = "циклу", stema = "цикл", pos = S", weight = 1"] [word = "програмного", stema = "програм", pos = A", weight = 25"]

[word = "забезпечення", stema = "забезпечен", pos = S", weight = 4"] [word = "мають", stema = "мают", pos = S", weight = 1"] [word = "загальний", stema = "загальн", pos = S", weight = 3"] [word = "лософський", stema = "лософськ", pos = A", weight = 2"] [word = "методолог", stema = "методолог", pos = S", weight = 28"] [word = "характеризу", stema = "характери", pos = S", weight = 2"] [word = "лософським", stema = "лософськ", pos = S", weight = 2"] [[word = "основними", stema = "основн", pos = S", weight = 2"] [word = "принципами", stema = "принцип", pos = S", weight = 2"] [word = "принципи", stema = "принцип", pos = S", weight = 2"] [word = "залежить", stema = "залеж", pos = V", weight = 2"] [word = "ефективн", stema = "ефективн", pos = S", weight = 2"] [word = "методолог", stema = "методолог", pos = S", weight = 28"] [word = "зазвичай", stema = "зазвича", pos = S", weight = 2"] [word = "можна", stema = "можн", pos = S", weight = 9"] [word = "коротко", stema = "коротк", pos = S", weight = 1"] [word = "сформулювати", stema = "сформулюват", pos = S", weight = 1"] [word = "легко", stema = "легк", pos = S", weight = 1"] [word = "пояснити", stema = "поясніт", pos = S", weight = 2"] [word = "узгодженим", stema = "узгоджен", pos = V", weight = 1"] [word = "моделей", stema = "модел", pos = S", weight = 2"] [word = "метод", stema = "метод", pos = S", weight = 6"] [word = "зують", stema = "зуют", pos = S", weight = 1"] [word = "методолог", stema = "методолог", pos = S", weight = 28"] [word = "концепц", stema = "концепц", pos = S", weight = 1"] [word = "поняттями", stema = "понят", pos = S", weight = 1"] [word = "дозволяють", stema = "дозволяют", pos = S", weight = 1"] [word = "точно", stema = "точ", pos = ADV", weight = 2"] [word = "визначити", stema = "визнач", pos = S", weight = 3"] [word = "методи", stema = "метод", pos = S", weight = 6"] [word = "окремому", stema = "окрем", pos = A", weight = 1"] [word = "випадку", stema = "випадк", pos = S", weight = 1"] [word = "методолог", stema = "методолог", pos = S", weight = 28"] [word = "застосову", stema = "застосов", pos = S", weight = 2"] [word = "програмування", stema = "програмуван", pos = S", weight = 25"] [word = "конструювання", stema = "конструюван", pos = S", weight = 1"] [word = "зазвичай", stema = "зазвича", pos = S", weight = 2"] [word = "називають", stema = "називают", pos = S", weight = 1"] [word = "парадигмою", stema = "парадигм", pos = S", weight

= 1"] [word = "програмування", stema = "програмуван", pos = S", weight = 25"] [word = "Можна", stema = "можн", pos = S", weight = 9"] [word = "простежити", stema = "простежит", pos = S", weight = 1"] [word = "шляхи", stema = "шлях", pos = S", weight = 1"] [word = "виникнення", stema = "виникнен", pos = S", weight = 1"] [word = "методолог", stema = "методолог", pos = S", weight = 28"] [word = "перше", stema = "перш", pos = S", weight = 1"] [word = "можуть", stema = "можут", pos = V", weight = 5"] [word = "виразом", stema = "вираз", pos = S", weight = 1"] [word = "практичного", stema = "практичн", pos = A", weight = 1"] [word = "друге", stema = "друг", pos = S", weight = 1"] [word = "методолог", stema = "методолог", pos = S", weight = 28"] [word = "можуть", stema = "можут", pos = V", weight = 5"] [word = "походити", stema = "походит", pos = S", weight = 1"] [word = "чотирьох", stema = "чотирьох", pos = S", weight = 1"] [word = "моделей", stema = "модел", pos = S", weight = 2"] [word = "алгоритму", stema = "алгоритм", pos = S", weight = 3"] [word = "абстрактна", stema = "абстрак", pos = S", weight = 4"] [word = "машина", stema = "машин", pos = S", weight = 1"] [word = "Тьюринга", stema = "тьюринг", pos = S", weight = 1"] [word = "мпективне", stema = "мпективн", pos = S", weight = 4"] [word = "програмування", stema = "програмуван", pos = S", weight = 25"] [word = "рекурсивн", stema = "рекурсивн", pos = S", weight = 1"] [word = "функц", stema = "функц", pos = S", weight = 4"] [word = "льберта", stema = "льберт", pos = S", weight = 1"] [word = "Аккермана", stema = "аккерма", pos = S", weight = 1"] [word = "структурне", stema = "структурн", pos = S", weight = 7"] [word = "програмування", stema = "програмуван", pos = S", weight = 25"] [word = "лямбда", stema = "лямбд", pos = S", weight = 1"] [word = "числення", stema = "числен", pos = S", weight = 1"] [word = "Черча", stema = "чер", pos = S", weight = 2"] [word = "функц", stema = "функц", pos = S", weight = 4"] [word = "ональне", stema = "альн", pos = S", weight = 4"] [word = "програмування", stema = "програмуван", pos = S", weight = 25"] [word = "нормальн", stema = "нормальн", pos = S", weight = 1"] [word = "алгорифм", stema = "алгори", pos = S", weight = 3"] [word = "Маркова", stema = "марков", pos = S", weight = 1"] [word = "програмування", stema = "програмуван", pos = S", weight = 25"] [word = "методолог", stema = "методолог", pos = S", weight = 28"] [word =

"можна", stema = "можн", pos = S", weight = 9"] [word = "пояснити", stema = "поясніт", pos = S", weight = 2"] [word = "через", stema = "через", pos = S", weight = 1"] [word = "дображення", stema = "дображен", pos = S", weight = 3"] [word = "трьох", stema = "трьох", pos = S", weight = 1"] [word = "структур", stema = "структур", pos = S", weight = 7"] [word = "моделювання", stema = "моделюван", pos = S", weight = 1"] [word = "структуру", stema = "структур", pos = S", weight = 7"] [word = "програмування", stema = "програмуван", pos = S", weight = 25"] [word = "Складовими", stema = "складов", pos = S", weight = 1"] [word = "частинами", stema = "част", pos = S", weight = 2"] [word = "можуть", stema = "можут", pos = V", weight = 5"] [word = "структура", stema = "структур", pos = S", weight = 7"] [word = "даних", stema = "дан", pos = S", weight = 2"] [word = "структура", stema = "структур", pos = S", weight = 7"] [word = "управл", stema = "управл", pos = S", weight = 3"] [word = "Кожне", stema = "ожн", pos = S", weight = 9"] [word = "дображень", stema = "дображен", pos = S", weight = 3"] [word = "визнача", stema = "знач", pos = S", weight = 4"] [word = "методолог", stema = "методолог", pos = S", weight = 28"] [word = "достатньо", stema = "достатн", pos = S", weight = 1"] [word = "серйозний", stema = "серйозн", pos = S", weight = 1"] [word = "метод", stema = "метод", pos = S", weight = 6"] [word = "програмування", stema = "програмуван", pos = S", weight = 25"] [word = "Наприклад", stema = "наприклад", pos = S", weight = 4"] [word = "дображення", stema = "дображен", pos = S", weight = 3"] [word = "програмування", stema = "програмуван", pos = S", weight = 25"] [word = "методолог", stema = "методолог", pos = S", weight = 28"] [word = "способу", stema = "способ", pos = S", weight = 1"] [word = "опису", stema = "опис", pos = S", weight = 1"] [word = "алгоритму", stema = "алгоритм", pos = S", weight = 3"] [word = "додатков", stema = "додатк", pos = S", weight = 1"] [word = "особливост", stema = "особлив", pos = S", weight = 1"] [word = "можна", stema = "можн", pos = S", weight = 9"] [word = "наступн", stema = "наступн", pos = S", weight = 1"] [word = "основних", stema = "основн", pos = A", weight = 2"] [word = "методолог", stema = "методолог", pos = S", weight = 28"] [word = "Методолог", stema = "методолог", pos = S", weight = 28"] [word = "мперативного", stema = "мперативн", pos = A", weight

= 4"] [word = "програмування", stema = "програмуван", pos = S", weight = 25"] [word = "Методолог", stema = "методолог", pos = S", weight = 28"] [word = "Методолог", stema = "методолог", pos = S", weight = 28"] [word = "функц", stema = "функц", pos = S", weight = 4"] [word = "онального", stema = "альн", pos = A", weight = 4"] [word = "програмування", stema = "програмуван", pos = S", weight = 25"] [word = "Методолог", stema = "методолог", pos = S", weight = 28"] [word = "програмування", stema = "програмуван", pos = S", weight = 25"] [word = "Методолог", stema = "методолог", pos = S", weight = 28"] [word = "програмування", stema = "програмуван", pos = S", weight = 25"] [word = "обмеженнях", stema = "обмежен", pos = S", weight = 1"] [word = "Можна", stema = "можн", pos = S", weight = 9"] [word = "методолог", stema = "методолог", pos = S", weight = 28"] [word = "знаходяться", stema = "знаходя", pos = V", weight = 1"] [word = "покрокове", stema = "покроков", pos = S", weight = 1"] [word = "управл", stema = "управл", pos = S", weight = 3"] [word = "виконанням", stema = "виконан", pos = S", weight = 1"] [word = "специф", stema = "специф", pos = S", weight = 5"] [word = "визначення", stema = "визначен", pos = S", weight = 3"] [word = "вимог", stema = "вимог", pos = S", weight = 1"] [word = "результату", stema = "результат", pos = S", weight = 2"] [word = "тополог", stema = "тополог", pos = S", weight = 4"] [word = "специф", stema = "специф", pos = S", weight = 5"] [word = "Специф", stema = "специф", pos = S", weight = 5"] [word = "тополог", stema = "тополог", pos = S", weight = 4"] [word = "специф", stema = "специф", pos = S", weight = 5"] [word = "вибору", stema = "вибор", pos = S", weight = 2"] [word = "метод", stema = "метод", pos = S", weight = 6"] [word = "уточнення", stema = "уточнен", pos = S", weight = 1"] [word = "методолог", stema = "методолог", pos = S", weight = 28"] [word = "Критер", stema = "критер", pos = S", weight = 1"] [word = "якост", stema = "якост", pos = S", weight = 1"] [word = "тополог", stema = "тополог", pos = S", weight = 4"] [word = "можуть", stema = "можут", pos = V", weight = 5"] [word = "загальн", stema = "загальн", pos = S", weight = 3"] [word = "витрати", stema = "витрат", pos = S", weight = 3"] [word = "розробку", stema = "розробк", pos = S", weight = 3"] [word = "чергу", stema = "черг", pos = S", weight = 2"] [word = "витрати", stema = "витрат",

pos = S", weight = 3"] [word = "розробку", stema = "розробк", pos = S", weight = 3"] [word = "залежать", stema = "залежа", pos = V", weight = 2"] [word = "серед", stema = "серед", pos = S", weight = 1"] [word = "ншого", stema = "ншог", pos = A", weight = 1"] [word = "ключових", stema = "ключов", pos = A", weight = 1"] [word = "мовних", stema = "мовн", pos = A", weight = 2"] [word = "абстракц", stema = "абстракц", pos = S", weight = 4"] [word = "абстракц", stema = "абстракц", pos = S", weight = 4"] [word = "даних", stema = "дан", pos = S", weight = 2"] [word = "управл", stema = "управл", pos = S", weight = 3"] [word = "модульност", stema = "модульн", pos = S", weight = 1"] [word = "Наприклад", stema = "наприклад", pos = S", weight = 4"] [word = "мпеативн", stema = "мпеативн", pos = S", weight = 4"] [word = "методолог", stema = "методолог", pos = S", weight = 28"] [word = "можна", stema = "можн", pos = S", weight = 9"] [word = "дотримуватися", stema = "тримуват", pos = V", weight = 2"] [word = "метод", stema = "метод", pos = S", weight = 6"] [word = "структурного", stema = "структурн", pos = A", weight = 7"] [word = "програмування", stema = "програмуван", pos = S", weight = 25"] [word = "тополог", stema = "тополог", pos = S", weight = 4"] [word = "точки", stema = "точк", pos = S", weight = 2"] [word = "мовних", stema = "мовн", pos = A", weight = 2"] [word = "абстракц", stema = "абстракц", pos = S", weight = 4"] [word = "Результатом", stema = "результат", pos = S", weight = 2"] [word = "методолог", stema = "методолог", pos = S", weight = 28"] [word = "структурного", stema = "структурн", pos = A", weight = 7"] [word = "програмування", stema = "програмуван", pos = S", weight = 25"] [word = "специф", stema = "специф", pos = S", weight = 5"] [word = "дност", stema = "дност", pos = S", weight = 1"] [word = "тектурою", stema = "тектур", pos = S", weight = 1"] [word = "апаратного", stema = "апаратн", pos = A", weight = 1"] [word = "забезпечення", stema = "забезпечен", pos = S", weight = 4"] [word = "централ", stema = "центра", pos = S", weight = 1"] [word = "зованою", stema = "зован", pos = S", weight = 1"] [word = "паралельно", stema = "паралельн", pos = ADV", weight = 3"] [word = "Наприклад", stema = "наприклад", pos = S", weight = 4"] [word = "методолог", stema = "методолог", pos = S", weight = 28"] [word = "мпеативного", stema = "мпеативн", pos = A", weight = 4"] [word = "паралельного", stema =

"паралельн", pos = A", weight = 3"] [word = "програмування", stema = "програмуван", pos = S", weight = 25"] [word = "методолог", stema = "методолог", pos = S", weight = 28"] [word = "паралельного", stema = "паралельн", pos = A", weight = 3"] [word = "програмування", stema = "програмуван", pos = S", weight = 25"] [word = "методолог", stema = "методолог", pos = S", weight = 28"] [word = "Наприклад", stema = "наприклад", pos = S", weight = 4"] [word = "часта", stema = "част", pos = S", weight = 2"] [word = "функц", stema = "функц", pos = S", weight = 4"] [word = "чного", stema = "чног", pos = A", weight = 3"] [word = "програмування", stema = "програмуван", pos = S", weight = 25"] [word = "Проводяться", stema = "проводя", pos = V", weight = 1"] [word = "дження", stema = "джен", pos = S", weight = 1"] [word = "методолог", stema = "методолог", pos = S", weight = 28"] [word = "програмування", stema = "програмуван", pos = S", weight = 25"] [word = "програмування", stema = "програмуван", pos = S", weight = 25"] [word = "можуть", stema = "можут", pos = V", weight = 5"] [word = "добре", stema = "добр", pos = S", weight = 1"] [word = "дтримувати", stema = "дтримуват", pos = S", weight = 2"] [word = "методолог", stema = "методолог", pos = S", weight = 28"] [word = "означа", stema = "означ", pos = S", weight = 2"] [word = "деякий", stema = "деяк", pos = A", weight = 1"] [word = "взагал", stema = "взага", pos = S", weight = 1"] [word = "можна", stema = "можн", pos = S", weight = 9"] [word = "використовувати", stema = "використовуват", pos = S", weight = 1"] [word = "невластивою", stema = "невластив", pos = S", weight = 1"] [word = "методолог", stema = "методолог", pos = S", weight = 28"] [word = "затратити", stema = "затратит", pos = S", weight = 1"] [word = "зусиль", stema = "зусил", pos = S", weight = 1"] [word = "ресурс", stema = "ресурс", pos = S", weight = 1"] [word = "Методолог", stema = "методолог", pos = S", weight = 28"] [word = "програмування", stema = "програмуван", pos = S", weight = 25"] [word = "зняються", stema = "зняют", pos = V", weight = 1"] [word = "загальних", stema = "загальн", pos = A", weight = 3"] [word = "витрат", stema = "витрат", pos = S", weight = 3"] [word = "шення", stema = "шен", pos = S", weight = 1"] [word = "завдань", stema = "завдан", pos = S", weight = 3"] [word = "зними", stema = "зним", pos = S", weight = 1"] [word = "характеристиками", stema =

"характеристик", pos = S", weight = 2"] [word = "науков", stema = "наук", pos = S", weight = 1"] [word = "розрахунки", stema = "розрахунк", pos = S", weight = 1"] [word = "нсів", stema = "нсів", pos = S", weight = 1"] [word = "завдання", stema = "завдан", pos = S", weight = 3"] [word = "системи", stema = "систем", pos = S", weight = 1"] [word = "реального", stema = "реальн", pos = A", weight = 4"] [word = "Масштаб", stema = "масштаб", pos = S", weight = 1"] [word = "завдань", stema = "завдан", pos = S", weight = 3"] [word = "ефективн", stema = "ефективн", pos = S", weight = 2"] [word = "створюваного", stema = "створюван", pos = A", weight = 1"] [word = "програмного", stema = "програм", pos = A", weight = 25"] [word = "забезпечення", stema = "забезпечен", pos = S", weight = 4"] [word = "також", stema = "також", pos = S", weight = 1"] [word = "важливими", stema = "важлив", pos = S", weight = 1"] [word = "факторами", stema = "фактор", pos = S", weight = 1"] [word = "вибор", stema = "вибор", pos = S", weight = 2"] [word = "методолог", stema = "методолог", pos = S", weight = 28"] [word = "програмування", stema = "програмуван", pos = S", weight = 25"]

ДОДАТОК 4. Приклад тексту до обробки моделлю на тему «Періодизація філософії»

Початковий текст

Середньовічну філософію умовно можна розділити на наступні періоди: 1) введення або патристика (II – VI ст.); 2) аналіз можливостей слова, пов'язаний з християнською ідеєю творення світу по Слово і Його втілення в світі (VII – X ст.); 3) схоластика (XI – XIV ст.). У кожному з цих періодів зазвичай розрізняють "раціоналістичну" і "містичну" лінії. Однак варто підкреслити, що розум був містично орієнтований, а містика раціонально організована. Обидві ці лінії аж до XIII ст., коли стався розрив розуму і віри, були нероздільні. Такому мисленню властиво те, що називалося "сходженням розуму в серце"; це в свою чергу вимагало нових способів побутування душі: ними були висновок, молитва, сповідь, повчання, вбрані у форму притчі. Саме тому, що філософія становила з теологією єдиний ідейний комплекс, вона розвивалася не тільки в світських школах (з XI ст.) або в університетах (з XIII ст.), але перш за все в монастирях, релігійних орденах, серед яких найбільш відомими стали францисканці (Олександр Гельський, Бонавентура, Дунс Скот, Вільям Оккам) і домініканці (Альберт фон Больштедт, Фома Аквінський). Навіть імена середньовічних філософів і шкіл пов'язані з найбільш відомими соборами і оселями (Гуго Сен-Викторский, Теодорик Шартрський, Бернард Клервоський, Ансельм Кентерберійський). Патристика підрозділяється на ранню, донікейську (II – IV ст.) та післянікейську (IV – VI ст.), греко-візантійську (Оріген, Григорій Ніський, Діонісій Ареопагіт, Максим Сповідник, Іоанн Дамаскін та ін.) і латинську (Августин, Боецій та ін.). Греко-візантійська філософія відводить вирішальне місце безпосереднього споглядання Бога в інтуїтивному акті, тобто містицизму, в той час як латинська думку розвивається всередині пересічних ліній містицизму і раціоналізму. Величезне значення в цей період мали суперечки з неоплатонізмом, гностицизмом і єресями. Філософствування, що виявляє

можливості слова, – це фактично все середньовічне філософствування, його можна виділити в окремий період досить умовно, маючи на увазі християнізацію нових ареалів – територій майбутньої Західної Європи. Найважливіші його представники – тато Григорій Великий, Ісидора Севільський, Беда Високоповажний, магістри Каролінгської академії, перш за все Алкуин. Окремо в цьому ряду стоїть великий філософ Еріугена. Період, пов'язаний зі схоластичним методом дослідження, можна розділити надвоє: ранній (XI – XII ст.) і пізній (XIII – XIV ст.). Представниками ранньосхоластичної філософії є Іоанн Росцелин, Ансельм Кентерберійський, Петро Абеляр, Бернард Клервоський та інші. Піздньосхоластичний період представляють Раймонд Луллий, Фома Аквінський, Бонавентура, Сігер Брабантский, Дунс Скот, Вільям Оккам, творці "експериментальної філософії" Роберт Гроссетест, Роджер Бекон і ін. Піздньосхоластичний період характерний впливом арабської філософії (Авіценна, Аверроес), фізичних і метафізичних ідей Аристотеля; це призвело до формування ідеї двох істин: розуму і віри.

Оброблений текст

[word = "Середньов", stema = "середн", pos = S", weight = 5"] [word = "лософ", stema = "лософ", pos = S", weight = 8"] [word = "умовно", stema = "умовн", pos = ADV", weight = 2"] [word = "можна", stema = "можн", pos = S", weight = 4"] [word = "наступн", stema = "наступн", pos = S", weight = 1"] [word = "введення", stema = "введен", pos = S", weight = 1"] [word = "патристика", stema = "патристик", pos = S", weight = 2"] [word = "можливостей", stema = "можлив", pos = S", weight = 2"] [word = "слова", stema = "слов", pos = S", weight = 3"] [word = "язаний", stema = "язан", pos = S", weight = 2"] [word = "християнською", stema = "христия", pos = S", weight = 2"] [word = "творення", stema = "твор", pos = S", weight = 2"] [word = "Слову", stema = "слов", pos = S", weight = 3"] [word = "лення", stema = "лен", pos = S", weight = 1"] [word = "схоластика", stema = "схоласти", pos = S", weight = 2"] [word = "зазвичай", stema = "зазвича", pos = S", weight = 1"] [word = "зняють", stema = "зняют", pos = S", weight = 1"] [word =

"стичну", stema = "стичн", pos = S", weight = 3"] [word = "стичну", stema = "стичн", pos = S", weight = 3"] [word = "Однак", stema = "однак", pos = S", weight = 1"] [word = "варто", stema = "варт", pos = S", weight = 1"] [word = "дкреслити", stema = "дкреслит", pos = S", weight = 1"] [word = "розум", stema = "розум", pos = S", weight = 4"] [word = "стика", stema = "стик", pos = S", weight = 1"] [[word = "Обидв", stema = "обидв", pos = S", weight = 1"] [word = "розрив", stema = "розр", pos = S", weight = 1"] [word = "розуму", stema = "розум", pos = S", weight = 4"] [word = "нерозд", stema = "розд", pos = S", weight = 2"] [word = "Такому", stema = "так", pos = A", weight = 1"] [word = "мисленню", stema = "мислен", pos = S", weight = 1"] [word = "властиво", stema = "властив", pos = S", weight = 1"] [word = "називалося", stema = "назива", pos = V", weight = 1"] [word = "сходженням", stema = "сходжен", pos = S", weight = 1"] [word = "розуму", stema = "розум", pos = S", weight = 4"] [word = "серце", stema = "серц", pos = S", weight = 1"] [word = "чергу", stema = "черг", pos = S", weight = 1"] [word = "вимагало", stema = "вимага", pos = S", weight = 1"] [word = "нових", stema = "нов", pos = S", weight = 2"] [word = "побутування", stema = "побутуван", pos = S", weight = 1"] [word = "висновок", stema = "висновок", pos = S", weight = 1"] [word = "молитва", stema = "молитв", pos = S", weight = 1"] [word = "повчання", stema = "повчан", pos = S", weight = 1"] [word = "форму", stema = "форм", pos = S", weight = 1"] [word = "притч", stema = "притч", pos = S", weight = 1"] [word = "лософ", stema = "лософ", pos = S", weight = 8"] [word = "становила", stema = "станов", pos = S", weight = 1"] [word = "теолог", stema = "теолог", pos = S", weight = 1"] [word = "комплекс", stema = "комплекс", pos = S", weight = 1"] [word = "розвивалася", stema = "розвив", pos = V", weight = 2"] [word = "школах", stema = "школ", pos = S", weight = 1"] [word = "верситетах", stema = "верситет", pos = S", weight = 1"] [word = "монастирях", stema = "монастир", pos = S", weight = 1"] [word = "орденах", stema = "орден", pos = S", weight = 1"] [word = "серед", stema = "серед", pos = S", weight = 5"] [word = "домими", stema = "дом", pos = S", weight = 2"] [word = "францисканц", stema = "францисканц", pos = S", weight = 1"] [word = "Олександр", stema = "олександр", pos = S", weight = 1"] [word = "Гельський",

stema = "гельськ", pos = A", weight = 1"] [word = "Бонавентура", stema = "бонавентур", pos = S", weight = 2"] [word = "Оккам", stema = "окк", pos = S", weight = 2"] [word = "Альберт", stema = "альберт", pos = S", weight = 1"] [word = "Больштедт", stema = "больштедт", pos = S", weight = 1"] [word = "середньов", stema = "середн", pos = S", weight = 5"] [word = "лософ", stema = "лософ", pos = S", weight = 8"] [word = "домими", stema = "дом", pos = S", weight = 2"] [word = "соборами", stema = "собор", pos = S", weight = 1"] [word = "Викторский", stema = "викторск", pos = A", weight = 1"] [word = "Теодорик", stema = "теодорик", pos = S", weight = 1"] [word = "Шартрський", stema = "шартрськ", pos = A", weight = 1"] [word = "Бернард", stema = "бернард", pos = S", weight = 2"] [word = "Клервоський", stema = "клервоськ", pos = A", weight = 2"] [word = "Ансельм", stema = "ансельм", pos = S", weight = 2"] [word = "Кентербер", stema = "кентербер", pos = S", weight = 2"] [word = "Патристика", stema = "патристик", pos = S", weight = 2"] [word = "дрозд", stema = "дрозд", pos = S", weight = 2"] [word = "кейскую", stema = "кейск", pos = A", weight = 1"] [word = "Греко", stema = "грек", pos = S", weight = 2"] [word = "Григор", stema = "григор", pos = S", weight = 2"] [word = "ський", stema = "ськи", pos = A", weight = 6"] [word = "Ареопаг", stema = "ареопаг", pos = S", weight = 1"] [word = "Максим", stema = "макс", pos = S", weight = 1"] [word = "Дамаск", stema = "дамаск", pos = S", weight = 1"] [word = "латинську", stema = "латинськ", pos = S", weight = 2"] [word = "Августин", stema = "августин", pos = S", weight = 1"] [word = "Греко", stema = "грек", pos = S", weight = 2"] [word = "лософ", stema = "лософ", pos = S", weight = 8"] [word = "дводить", stema = "двод", pos = V", weight = 1"] [word = "шальне", stema = "шальн", pos = S", weight = 2"] [word = "безпосереднього", stema = "безпосередн", pos = A", weight = 1"] [word = "споглядання", stema = "споглядан", pos = S", weight = 1"] [word = "тивному", stema = "тивн", pos = A", weight = 1"] [word = "тобто", stema = "тобт", pos = S", weight = 1"] [word = "стицизму", stema = "стицизм", pos = S", weight = 3"] [word = "латинська", stema = "латинськ", pos = S", weight = 2"] [word = "думку", stema = "думк", pos = S", weight = 1"] [word = "розвива", stema = "розвив", pos = S", weight = 2"] [word = "всередин", stema =

"серед", pos = S", weight = 5"] [word = "перес", stema = "перес", pos = S", weight = 1"] [word = "стицизму", stema = "стицизм", pos = S", weight = 3"] [word = "Величезне", stema = "величезн", pos = S", weight = 1"] [word = "значення", stema = "значен", pos = S", weight = 1"] [word = "суперечки", stema = "суперечк", pos = S", weight = 1"] [word = "неоплатон", stema = "неоплатон", pos = S", weight = 1"] [word = "гностицизмом", stema = "гностицизм", pos = S", weight = 3"] [word = "лософствування", stema = "лософствуван", pos = S", weight = 2"] [word = "виявля", stema = "виявл", pos = S", weight = 1"] [word = "можливост", stema = "можлив", pos = S", weight = 2"] [word = "слова", stema = "слов", pos = S", weight = 3"] [word = "фактично", stema = "фактичн", pos = ADV", weight = 1"] [word = "середньов", stema = "середн", pos = S", weight = 5"] [word = "лософствування", stema = "лософствуван", pos = S", weight = 2"] [word = "можна", stema = "можн", pos = S", weight = 4"] [word = "окремий", stema = "окрем", pos = S", weight = 2"] [word = "досить", stema = "дос", pos = V", weight = 1"] [word = "умовно", stema = "умовн", pos = ADV", weight = 2"] [word = "маючи", stema = "маюч", pos = S", weight = 1"] [word = "християн", stema = "християя", pos = S", weight = 2"] [word = "нових", stema = "нов", pos = S", weight = 2"] [word = "ареал", stema = "ареа", pos = S", weight = 1"] [word = "територ", stema = "територ", pos = S", weight = 1"] [word = "майбутньо", stema = "майбутн", pos = S", weight = 1"] [word = "вропи", stema = "вроп", pos = S", weight = 1"] [word = "Найважлив", stema = "найважл", pos = S", weight = 1"] [word = "представники", stema = "представ", pos = S", weight = 3"] [word = "Григор", stema = "григор", pos = S", weight = 2"] [word = "Великий", stema = "велик", pos = A", weight = 2"] [word = "сидора", stema = "сидор", pos = S", weight = 1"] [word = "Високоповажний", stema = "високоповажн", pos = S", weight = 1"] [word = "Карол", stema = "карол", pos = S", weight = 1"] [word = "академ", stema = "акад", pos = S", weight = 1"] [word = "Алкуин", stema = "алкуин", pos = S", weight = 1"] [word = "Окремо", stema = "окрем", pos = S", weight = 2"] [word = "цьому", stema = "цьом", pos = A", weight = 1"] [word = "великий", stema = "велик", pos = A", weight = 2"] [word = "лософ", stema = "лософ", pos = S", weight = 8"] [word = "схоластичним", stema =

"схоластичн", pos = V", weight = 2"] [word = "методом", stema = "метод", pos = S", weight = 1"] [word = "можна", stema = "можн", pos = S", weight = 4"] ["Представниками", stema = "представ", pos = S", weight = 3"] [word = "раннесхоласт", stema = "несхоласт", pos = S", weight = 3"] [word = "лософ", stema = "лософ", pos = S", weight = 8"] [word = "Росцелин", stema = "росцелин", pos = S", weight = 1"] [word = "Ансельм", stema = "ансельм", pos = S", weight = 2"] [word = "Кентербер", stema = "кентербер", pos = S", weight = 2"] [word = "Петро", stema = "петр", pos = ADV", weight = 1"] [word = "Абеляр", stema = "абеляр", pos = S", weight = 1"] [word = "Бернард", stema = "бернард", pos = S", weight = 2"] [word = "Клервоський", stema = "клервоськ", pos = A", weight = 2"] [word = "Піздньосхоласт", stema = " піздньосхоласт ", pos = S", weight = 3"] [word = "представляють", stema = "представляют", pos = S", weight = 3"] [word = "Раймонд", stema = "раймонд", pos = S", weight = 1"] [word = "нський", stema = "нськи", pos = A", weight = 6"] [word = "Бонавентура", stema = "бонавентур", pos = S", weight = 2"] [word = "Брабантский", stema = "брабантск", pos = A", weight = 1"] [word = "Оккам", stema = "окк", pos = S", weight = 2"] [word = "експериментально", stema = "експериментальн", pos = ADV", weight = 1"] [word = "лософ", stema = "лософ", pos = S", weight = 8"] [word = "Роберт", stema = "роберт", pos = S", weight = 1"] [word = "Гроссетест", stema = "гроссетест", pos = S", weight = 1"] [word = "Роджер", stema = "роджер", pos = S", weight = 1"] [word = "Бекон", stema = "бекон", pos = S", weight = 1"] [word = "Піздньосхоласт", stema = "піздньосхоласт", pos = S", weight = 3"] [word = "характерний", stema = "характерн", pos = S", weight = 1"] [word = "впливом", stema = "вплив", pos = S", weight = 1"] [word = "лософ", stema = "лософ", pos = S", weight = 8"] [word = "Аверроес", stema = "аверроес", pos = S", weight = 1"] [word = "зичних", stema = "зичн", pos = A", weight = 2"] [word = "Аристотеля", stema = "аристотел", pos = S", weight = 1"] [word = "призвело", stema = "призвел", pos = S", weight = 1"] [word = "формування", stema = "формуван", pos = S", weight = 1"] [word = "розуму", stema = "розум", pos = S", weight = 4"]

ДОДАТОК 5. Приклад тексту до обробки моделлю на тему «Клас»

Клас – це елемент ПО, що описує абстрактний тип даних і його часткову або повну реалізацію. Інші абстрактні типи даних – метакласи, інтерфейси, структури, перерахування, – характеризуються якимись своїми, іншими особливостями. Поряд з поняттям «об'єкта» клас є ключовим поняттям в ООП (хоча існують і безкласового об'єктно-орієнтовані мови, наприклад, Self, Lua; докладніше дивіться прототипне програмування). Суть відмінності класів від інших абстрактних типів даних полягає в тому, що при завданні типу даних клас визначає одночасно як інтерфейс, так і реалізацію для всіх своїх екземплярів, а виклик методу-конструктора обов'язковий. Точний зміст цієї фрази буде розкритий нижче. На практиці об'єктно-орієнтоване програмування зводиться до створення певної кількості класів, включаючи інтерфейс і реалізацію, і подальшого їх використання. Графічне представлення деякої кількості класів та зв'язків між ними називається діаграмою класів. Об'єктно-орієнтований підхід за час свого розвитку накопичив множину рекомендацій (патернів) зі створення класів і ієрархій класів. Ідея класів прийшла з робіт по базам знань, що мають відношення до досліджень з штучного інтелекту. Використовувані людиною класифікації в зоології, ботаніки, хімії, деталях машин, несуть в собі основну ідею, що будь-яку річ завжди можна уявити окремим випадком деякого більш загального поняття. Конкретне яблуко – це в цілому деяке яблуко, взагалі яблуко, а будь-яке взагалі яблуко – фрукт.

Саме тому приклади класів в навчальних посібниках з об'єктно-орієнтованого програмування так часто згадують яблука і груші. Скрізь далі слова «клас», «об'єкт», «інтерфейс» і «структура» будуть вживатися в своїх спеціальних значеннях, заданих в рамках ООП. В об'єктно-орієнтованої програми із застосуванням класів кожен об'єкт є «екземпляром» деякого конкретного класу, і інших об'єктів не передбачено. Тобто «екземпляр класу» в даному випадку означає не «приклад деякого класу» або «окремо взятий клас», а «об'єкт, типом якого є якийсь клас». При цьому в різних мовах програмування допускається або

не допускається існування ще якихось типів даних, екземпляри яких не є об'єктами (тобто мова визначає, чи є об'єктами такі речі, як числа, масиви і покажчики, або не є, і, відповідно, чи є такі класи як «число», «масив» або «покажчик», екземплярами яких були б кожне конкретне число, масив або покажчик). Наприклад, абстрактний тип даних «рядок тексту» може бути оформлений у вигляді класу, і тоді все рядки тексту в програмі будуть об'єктами – екземплярами класу «рядок тексту».

При використанні класів все елементи коду програми, такі як змінні, константи, методи, процедури і функції, можуть належати (а в багатьох мовах повинні належати) того чи іншого класу. Сам клас в підсумку визначається як список своїх членів, а саме полів (властивостей) і методів / функцій / процедур. Залежно від мови програмування до цього списку можуть додатися константи, атрибути і зовнішні визначення. Як і структури, класи можуть задавати поля – тобто змінні, що належать або безпосередньо самому класу (статичні), або до екземплярів класу (звичайні). Статичні поля існують в одному екземплярі на всю програму (або, в більш складному варіанті, – в одному екземплярі на процес або на потік). Звичайні поля створюються по одній копії для кожного конкретного об'єкта – екземпляра класу. Наприклад, загальна кількість рядків тексту, створених в програмі за час її роботи, буде статичним полем класу «рядок тексту». А конкретний масив символів рядка буде звичайним полем екземпляра класу «рядок тексту», так само як змінна «прізвище», що має тип «рядок тексту», буде звичайним полем кожного конкретного екземпляра класу «людина».

В ООП при використанні класів весь виконуваний код програми (алгоритми) буде оформлятися у вигляді так званих «методів», «функцій» або «процедур», що відповідає звичайному структурному програмуванню, однак тепер вони можуть (а в багатьох мовах зобов'язані) належати тому чи іншого класу. Наприклад, по можливості, клас «рядок тексту» буде містити всі основні методи / функції / процедури, призначені для роботи з рядком тексту, такі як пошук в рядку, вирізання частини рядка. Як і поля, код у вигляді методів / функцій / процедур, що належать класу, може бути віднесений або до самого класу, або до

екземплярів класу. Метод, що належить класу і співвіднесений з класом (статичний метод) може бути викликаний сам по собі і має доступ до статичних змінних класу.

Метод, співвіднесений з екземпляром класу (звичайний метод), може бути викликаний тільки у самого об'єкта, і має доступ як до статичних полів класу, так і до звичайних полів конкретного об'єкта (при виклику цей об'єкт передасться прихованим параметром методу). Наприклад, загальну кількість створених рядків можна дізнатися з будь-якого місця програми, але довжину конкретного рядку можна дізнатися тільки вказавши, тим чи іншим чином, довжину якого рядку будемо міряти. У програмуванні існує поняття програмного інтерфейсу, що означає перелік можливих обчислень, які може виконати та чи інша частина програми, включаючи опис того, які аргументи і в якому порядку потрібно передавати на вхід алгоритмам з цього переліку, а також що і в якому вигляді вони будуть повертати. Абстрактний тип даних інтерфейс придуманий для формалізованого опису такого переліку.

Самі алгоритми, тобто дійсний програмний код, який буде виконувати всі ці обчислення, у інтерфейсі не задається, програмується окремо і називається реалізацією інтерфейсу. Програмні інтерфейси, а також класи, можуть розширюватися шляхом успадкування, яке є одним з важливих засобів повторного використання готового коду в ООП.

Успадкованих клас або інтерфейс буде містити в собі все, що зазначено для всіх його батьківських класів (в залежності від мови програмування і платформи, їх може бути від нуля до нескінченності). Наприклад, можна створити свій варіант текстового рядка шляхом успадкування класу «мій рядок тексту» від вже існуючого класу «рядок тексту», при цьому передбачається, що програмісту не доведеться заново переписувати алгоритми пошуку та інше, так як вони автоматично будуть успадковані від готового класу, і будь-який екземпляр класу «мій рядок тексту» може бути переданий не тільки в готові методи батьківського класу «рядок тексту» для проведення потрібних обчислень, але і взагалі в будь-який алгоритм, здатний працювати з об'єктами типу «рядки тексту», так як

екземпляри обох класів сумісні з програмним інтерфейсам. Клас дозволяє задати не тільки програмний інтерфейс до самого себе і до своїх екземплярів, але і в явному вигляді написати код, відповідальний за обчислення. Якщо при створенні свого нового типу даних успадковувати інтерфейс, то ми отримаємо можливість передавати екземпляр свого типу даних в будь-який алгоритм, який вміє працювати з цим інтерфейсом.

Однак нам доведеться самим написати реалізацію інтерфейсу, тобто ті алгоритми, якими буде користуватися цікавий для нас алгоритм для проведення обчислень з використанням нашого екземпляра. У той же час, наслідуючи клас, ми автоматично успадковуємо готовий код під інтерфейс (це не завжди так, батьківський клас може вимагати реалізації якихось алгоритмів в дочірньому класі в обов'язковому порядку). У цій можливості успадковувати готовий код і проявляється те, що в об'єктно-орієнтованої програми тип даних клас визначає одночасно і інтерфейс, і реалізацію для всіх своїх екземплярів. Однією з проблем структурного програмування, з якою бореться ООП, є проблема підтримки правильного значення змінних програми. Часто різні змінні програми зберігають логічно пов'язані значення, і за підтримку цієї логічної зв'язності несе відповідальність програміст, тобто автоматично зв'язність не підтримується. Прикладом можуть служити прапорці «звільнений» і «очікує премії за підсумками року», коли за правилами відділу кадрів людина може бути водночас не звільненою і не очікувати премії, як і не звільненою і очікувати премії, звільненою і не очікувати премії, але не може бути одночасно і звільненою, і очікувати премії. Тобто будь-яка частина програми, яка проставляє прапорець «звільнений», завжди повинна знімати прапорець «очікує премії за підсумками року».

Хороший спосіб вирішити цю проблему – оголосити прапорець «звільнений» недоступним до зміни для всіх ділянок програми, крім однієї спеціально обумовленої. У цієї спеціально обумовленої ділянці все буде написано один раз і правильно, а всі інші повинні будуть звертатися до цієї ділянки завжди, коли вони хочуть встановити або зняти прапорець «звільнений». В об'єктно-орієнтованої програми прапорець «звільнений» буде оголошений приватним

членом деякого класу, а для читання і зміни його будуть написані відповідні публічні методи. Правила, що визначають можливість або неможливість безпосередньо змінювати будь-які змінні, називаються правилами завдання областей доступу.

Слова «приватний» і «публічний» в даному випадку є так званими «модифікаторами доступу». Вони називаються модифікаторами тому, що в деяких мовах вони використовуються для зміни раніше встановлених прав при спадкуванні класу. Спільно класи і модифікатори доступу задають область доступу, тобто у кожної ділянки коду, в залежності від того, якого класу вона належить, буде своя область доступу щодо тих чи інших елементів (членів) свого класу і інших класів, включаючи змінні, методи, функції, константи. Існує основне правило: ніщо в одному класі не може бачити приватних елементів іншого класу. Щодо інших, більш складних правил, в різних мовах існують інші модифікатори доступу і правила їх взаємодії з класами. Майже кожному члену класу можна встановити модифікатор доступу (за винятком статичних конструкторів і деяких інших речей). У більшості об'єктно-орієнтованих мов програмування підтримуються наступні модифікатори доступу:

private (закритий, внутрішній член класу) – звернення до члена допускаються тільки з методів того класу, в якому цей член визначений. Будь-які спадкоємці класу вже не зможуть отримати доступ до такого члена. Спадкування за типом private робить все члени батьківського класу (в тому числі public і protected) private-членами класу-спадкоємця (C ++);

protected (захищений, внутрішній член ієрархії класів) – звернення до члена допускаються з методів того класу, в якому цей член визначений, а також з будь-яких методів його класів-спадкоємців. Спадкування за типом protected робить все public-члени батьківського класу protected-членами класу-спадкоємця (C ++);

public (відкритий член класу) – звернення до члена допускаються з будь-якого коду. Спадкування за типом public не змінює модифікаторів батьківського класу (C ++);

Проблема підтримки правильного стану змінних актуальна і для самого першого моменту виставлення початкових значень. Для цього в класах передбачені спеціальні методи / функції, звані конструкторами. Жоден об'єкт (екземпляр класу) не може бути створений інакше, як шляхом виклику на виконання коду конструктора, який поверне викликаючий стороні створений і правильно заповнений примірник класу. У багатьох мовах програмування тип даних «структура», як і клас, може містити змінні і методи, але екземпляри структур, залишаючись просто розміченою ділянкою оперативної пам'яті, можуть створюватися в обхід конструкторів, що заборонено для примірників класів (за винятком спеціальних виняткових методів обходу всіх подібних правил ООП, передбачених в деяких мовах і платформах). У цьому виявляється відмінність класів від інших типів даних – виклик конструктора обов'язковий. В сучасних об'єктно-орієнтованих мовах програмування (в тому числі в php, Java, C ++, Oberon, Python, Ruby, Smalltalk, Object Pascal) створення класу зводиться до написання деякої структури, що містить набір полів і методів (серед останніх особливу роль грають конструктори, деструктори, фіналізатор). Практично клас може розумітися як якийсь шаблон, за яким створюються об'єкти – екземпляри даного класу. Всі екземпляри одного класу створені за одним шаблоном, тому мають один і той же набір полів і методів.

ДОДАТОК 6. Приклад автоматично згенерованого технічного тексту

Не можна вирішити, який саме з галактики, не виявляли себе в вікнах. Позагалактичних об'єктів були б тоді. Вина частка випромінювання, що посилається в силу закону вина. Відсутність концентрації цих джерел радіовипромінювання показало, що більшість. Цих галактик дуже багато. Ніякі помітних оптичних об'єктів крім нечисленних близьких, які випадковим чином опинилися. Залишками газової матерії після відкриття дискретних джерел радіовипромінювання виявлялося б. вивчення. Таких зірок з іншого боку, оптично спостерігати об'єкти. Багато, і залишками газової матерії. Друг до мають низькі температури, 500к діючі точкові радіоджерела. Так як правило, в перспективі можна зареєструвати, а роздільна. Положення джерела радіовипромінювання другої групи могли б тоді відсутність. Радіовипромінювання, як зірки, що мають низькі температури, 500к галактиці об'єкти виявляють концентрацію. Перспективі можна зареєструвати, а роздільна. Пилової матерії після відкриття дискретних джерел нечисленних близьких. Туманностями і на небі близько один до галактичного екватора. Радіотелескопів невелика, всі діючі точкові радіоджерела злилися б тоді відсутність концентрації. Без ознак концентрацій до галактичного екватора менше. Смуги і радіооб'єктів залишалося припущення, що положення джерела радіовипромінювання першої трупи. Ні, а роздільна сила радіотелескопів невелика, всі діючі точкові радіоджерела злилися. Проблема ототожнення оптичних і розподілених. Вивчення розподілу по всьому небу дискретних джерел радіовипромінювання показало, що дискретними. Низькі температури, 500к виявляють концентрацію до висновку про так званих радіозірки середу.

Шукати в перші роки після відкриття дискретних джерел радіовипромінювання показало, що більшість. Де спостерігалися інтенсивний джерела радіовипромінювання, не виявляли себе в все-таки. Відкриття дискретних джерел радіовипромінювання очікує. Майданчиках, в радіохвилях, більше, ніж у таких зірок. Радіовипромінювання складається в тих майданчиках, в цілій майданчику, що містить десятки квадратних. Закону вина частка оптичного

випромінювання менше. Нових і оптично яскраві об'єкти, тоді відсутність концентрації цих. Про марність спроб ототожнення і слід було очікувати є. Виникла проблема ототожнення оптичних об'єктів були відомі тільки галактики. Близько один до висновку про марність. Знаходяться поблизу площині галактики і слід було очікувати, є об'єктами входять. Багато, і вони розташовані на великих відстанях. Ніякі помітних оптичних і наднових зірок до складу. Шукати в цілій майданчику, що містить десятки квадратних хвилин занадто слабким залишиться. Міжзоряне середовище прозоре, радіовипромінювання якого досить сильне або менш рівномірно. Суцільний фон випромінювання менше товщини галактики, так як ми зазначали вище динамічними. Діючі точкові радіоджерела злилися б бути дуже близькі зірки відстані. Перші роки після відкриття дискретних джерел джерела радіовипромінювання першої. І оптично яскраві об'єкти, наприклад зірки першої трупи, як тільки. Розроблялися методи визначення їх радіовипромінювання доходить безперешкодно астрономи прийшли до галактичного. Віддано багато зусиль який саме з них знаходяться поблизу. Але це позагалактичні об'єкти, тоді відсутність концентрації цих джерел радіовипромінювання.

ДОДАТОК 7. Приклад автоматично згенерованого гуманітарного тексту

Інтернет-сторінки і до цього дня оригінального. Латинський алфавіт, можуть виникнути невеликі проблеми: як риби. Не має ніякого відношення до мешканців водойм отримати більше. Проектах, орієнтованих на назву, не має ніякого відношення. Латині і демонстрації зовнішнього вигляду контенту перегляду. Отримавши текст-рибу, широко використовуваний і проектах. Варіантів lorem ipsum на основі оригінального трактату. Ipsum, крім того, є спеціальні генератори, які створюють власні варіанти тексту зіграє. І демонстрації зовнішнього вигляду контенту, перегляду шрифтів, абзаців, відступів і. Кожен веб-розробник знає, що таке текст-риба написання символів на. Його трактату про межі добра і. Отримати більш того, нечитабельність тексту. Висновок, що вперше його застосували в xvi столітті виключно демонстраційна. Крім того, є спеціальні генератори. Варіантів lorem ipsum на основі оригінального. Основі оригінального трактату, завдяки чому з'являється можливість отримати. Він веб-дизайнерами для вставки на основі. Текст цей, незважаючи на кириличний контент – написання символів на кирилиці. Відступів і слова, отримавши текст-рибу, широко використовуваний має ніякого відношення. Ж краще використовувати в довжині найбільш поширених. Зовнішнього вигляду контенту, перегляду шрифтів абзаців. Смислове навантаження йому нести зовсім необов'язково невеликі. Вирвав окремі фрази і набір слів. Найвідомішим рибним текстом є знаменитий lorem латині і проектах орієнтованих. Завдяки чому з'являється можливість отримати більш довгий набір. Так як мета застосування такого тексту зіграє на латині і по. Веб-розробник знає, що вперше його застосували в довжині найбільш поширених слів зіграє. Різною частотою, є різниця в XVI столітті книгодрукуванні. Латинський алфавіт, можуть виникнути невеликі проблеми: в книгодрукуванні. Відомим рибним текстом є знаменитий lorem ipsum на руку при оцінці. Різною частотою, є різниця. Довший набір слів. веб-розробник знає, що все ж краще використовувати. Абзаців, відступів і на інтернет-сторінки. Можуть виникнути невеликі проблеми: в різних мовах. Незважаючи на кирилиці істотно відрізняється отримати більш

довгий набір слів. написання. Написання символів на сайтах і по сей день найвідомішим рибним. Так як мета застосування такого тексту зіграє. Цицерону, адже саме з його трактату про межі добра. Символів на тій мові який. Для вставки на кирилиці істотно відрізняється контент. Зовсім необов'язково слова, отримавши текст-рибу, широко використовуваний. Відносини до мешканців водойм використанням lorem символів. XVI столітті ipsum зобов'язаний давньоримським філософу Цицерону, адже саме. Все ж краще використовувати при оцінці якості сприйняття макета. Книгодрукуванні ще в xvi столітті зовсім необов'язково мовами, що використовують латинський алфавіт можуть. Ipsum зобов'язаний давньоримським філософу Цицерону, адже саме. Про межі добра і на основі. Краще використовувати в різних мовах ті чи інші. Ipsum зобов'язаний давньоримським філософу Цицерону, адже саме з його застосували в якості того, є спеціальні генератори, які створюють власні варіанти тексту зіграє на руку. Виникнути невеликі проблеми: як риби текст цей, незважаючи на основі оригінального. Його застосували в книгодрукуванні ще в xvi столітті мешканцям водойм кілька варіантів. Широко використовується отримати більше. Його застосували в довжині найбільш поширених слів на кирилиці. Питання, пов'язані з використанням lorem використовувати при запуску проекту абзаців. З його трактату про межі добра цей, незважаючи на сайтах. Істотно відрізняється появою lorem можуть виникнути. Набір слів. текстом є знаменитий lorem ipsum. Адже саме з його трактату. Чи інші літери зустрічаються з різною частотою є. Орієнтованих на латині і проектах орієнтованих. Оригінального трактату, завдяки чому з'являється можливість отримати більш довгий набір слів. Вперше його трактату про межі добра і чи інші літери зустрічаються з різною частотою, є різниця в. Виду контенту, перегляду шрифтів, абзаців, відступів і зла середньовічний. Добра і висновок, що все ж краще використовувати. Висновок, що таке текст-риба нести зовсім необов'язково краще використовувати.

ДОДАТОК 8. Приклад автоматично згенерованого технічного тексту збільшеного об’єму

І залишками газової матерії після спалахів. Радіотелескопів невелика, всі діючі точкові. Їх радіовипромінювання якого досить сильне ніякі примітних. Позагалактичних об'єктів були відомі тільки галактики. Перспективі можна зареєструвати, а оптичне випромінювання буде зрозуміло радіозірки існування. Спроб ототожнення і на небі близько один. Положення джерела радіовипромінювання не виявляється галактичної концентрації. Пилова міжзоряне середовище прозоре, радіовипромінювання доходить безперешкодно ж група. Ні, а частка випромінювання посилається. Гіпотетичні радіозірки, існування яких набагато менше товщини галактики. Газовими туманностями і на небі близько один. Небу дискретних джерел близьких, які випадковим чином опинилися в радіохвилях. Велика частина яскравих галактик не виявляли себе. Показує сильну концентрацію до її площини галактики. Квадратних хвилин група, більш-менш рівномірно, без ознак концентрацій до ототожнення. З радіозірки їх радіовипромінювання доходить безперешкодно прозора, радіовипромінювання доходить безперешкодно. Залишалося припущення, що положення джерела радіовипромінювання складається з великих. Близько один до висновку про так як зірки. Уже ототожнені з цими джерелами. Деякі астрономи прийшли до висновку про так як ми вказували. Навіть у вузькій смузі показує сильну концентрацію. Другої групи могли б кожна із суцільної фон випромінювання менше товщини галактики. Ототожнення лише слабкі галактики і на небі близько один до висновку. Прийшли до висновку про так званих радіозірки методи. радіозірки їх радіовипромінювання можна сподіватися на ототожнення лише. Без ознак концентрацій до площини галактики і не можна вирішити, який саме. Положення джерела радіовипромінювання є дуже багато зусиль від нас діючі точкові. Близько зірки, відстані яких реєструвалося радіовипромінювання, як правило, в тому. Площині галактики теж буде зрозуміло діючі точкові радіоджерела. Відомі тільки належать це були. Занадто слабким, залишиться

невловимим яких. Майданчику яскравих галактик з високими температурами, а роздільна сила. Велико і в вузькій смузі. З газовими туманностями і вони розташовані на ототожнення лише сонце. Уже ототожнені з цими джерелами радіовипромінювання є дуже близькими зірками. Об'єкт потрібно шукати в радіовипромінювання. Якби тоді гіпотетичні радіозірки, існування яких набагато.

Джерела радіовипромінювання, що не показувала ніякого зв'язку. Вина частка випромінювання, що посилається в цьому випадку поглинання світла. Ніякі помітних оптичних об'єктів були б кожна. Концентрації цих джерел кожна з них вже ототожнені з цими джерелами. Низькі температури, 500к висунута гіпотеза про так званих радіозірки не можна вирішити. Без ознак концентрацій до площини галактики і оптично. Спостерігалися інтенсивний джерела радіовипромінювання визначається з низькою точністю об'єкти, наприклад зірки першої. Про так званих радіозірки близькими. Таких зірок у тому, що вони діляться. Нечисленних близьких, які випадковим чином опинилися. Частка оптичного випромінювання менше товщини галактики, що не виявляють галактичної концентрації цих. Майданчику, що містить десятки квадратних хвилин невелика, всі діючі точкові радіоджерела. Посилається в силу закону вина частка оптичного випромінювання менше товщини галактики. Радіовипромінювання, як і оптично яскраві об'єкти, наприклад зірки. Званих радіозірки зареєстроване радіовипромінювання якого досить сильне прозора. Відстаней і вони діляться на ототожнення лише сонце, радіовипромінювання доходить безперешкодно. Радіохвиль ж група, більш численна, складається в цьому майданчику яскравих галактик. Наприклад зірки першої трупи, як і була висунута гіпотеза. Ототожнення і слід було очікувати, є дуже багато, і інших характеристик показує. Ж група, більш численна, складається з них розташована. При зіставленні галактик з високими температурами, а роздільна сила радіотелескопів невелика. Спростовується, як правило, спостерігалися лише сонце, радіовипромінювання якого. Діляться на великих відстанях від нас зіставленні галактик дуже близькими зірками. Об'єкти можна вирішити, який саме. Зіставленні галактик з іншого боку, оптично спостерігати об'єкти. Саме з

цього випадку поглинання світла дуже близькими зірками, так як. Склалося кілька дивне положення джерела радіовипромінювання другої групи могли б кожна. Газової матерії після спалахів нових і була висунута гіпотеза. 1950 р то, оскільки слабких галактик. Зіставленні галактик показувала ніякого зв'язку з джерелами. Менше товщини галактики, не виявляли себе в пилової матерії після спалахів. Галактики і розподілених по всьому небу більше. Радіотелескопів невелика, всі діючі точкові. У радіохвилях, більше, ніж у таких зірок. Чи не виявлялося ніякі помітних оптичних об'єктів крім нечисленних. Близько галактичного екватора і вони розташовані на небі близько один. Де спостерігалися лише тих об'єктів крім нечисленних близьких, які випадковим чином опинилися. Радіохвиль ж пилова міжзоряне середовище прозоре, радіовипромінювання можна сподіватися на великих відстанях. Близькими зірками, так як і наднових зірок. Зоряної величини, що не показувала. Точкові Радіоджерела злилися б при спостереженнях до складу галактики, не виявляли. Розподілу по всьому небу більш численна складається. Якби бути дуже багато і розподілених. Джерел, що розташовуються поза цьому майданчику яскравих галактик з газовими туманностями. Радіоджерела злилися б при зіставленні галактик була джерелом радіовипромінювання. Виявляють концентрацію до галактичного екватора інтенсивний джерела радіовипромінювання другої групи могли. Астрономії склалося дещо дивне положення. Зв'язки з іншого боку, оптично спостерігати об'єкти не можна вирішити. Розподілу по небу більш численна, складається в. Вище, динамічними міркуваннями дискретних джерел званих. Зірок з іншого боку, оптично яскраві об'єкти, тоді гіпотетичні радіозірки, існування яких. Об'єктами, що входять в тих об'єктів. Ототожнення лише сонце, радіовипромінювання можна сподіватися на. Відстаней і на дві групи могли б бути дуже. Методи визначення їх відстаней і інших. Радіоджерела злилися б бути дуже близькими зірками, так званих радіозірки. Галактики і була висунута гіпотеза про так званих радіозірки радіовипромінювання. Могли б при спостереженнях в галактичного. І слід було очікувати, є дуже багато зусиль зіставленні. Випромінювання буде зрозуміло радіовипромінювання визначається з цими джерелами радіовипромінювання є

дуже. Б при спостереженнях до складу галактики, не виявлялося дивна обставина розташована. Астрономи прийшли до висновку про так як тільки належать гіпотетичні радіозірки існування. Близьких, які випадковим чином опинилися в галактичного екватора розподілених по небу дискретних. Зазначали вище, динамічними міркуваннями існування яких реєструвалося радіовипромінювання. Ототожнення і вони діляться на великих відстанях від нас зареєструвати, а оптичне. Ті, хто має низькі температури, 500к об'єктами, що входять. Тому в тих місцях, де спостерігалися інтенсивний джерела радіовипромінювання оптичний об'єкт. І тут поглинання світла дуже близькі. Майданчику яскравих галактик не виявляли себе в цілій майданчику, що містить десятки квадратних. Галактичного екватора могли б при спостереженнях. Стало лише сонце, радіовипромінювання якого достатньо. Дослідженням радіозірки їх радіовипромінювання якого достатньо. При зіставленні галактик дуже багато, і вони діляться на ототожнення лише. Концентрацію до ніяк не виявляється галактичної концентрації цих джерел. Залишками газової матерії після спалахів. Радіовипромінювання можна сподіватися на небі близько один до галактики. І тут поглинання світла дуже багато і розподілених по всьому небу. А оптичне випромінювання буде все-таки занадто слабким, залишиться невловимим зірки мають. Оптичного випромінювання менше товщини галактики. Галактичної концентрації цих джерел радіовипромінювання визначається з високими температурами. Сподіватися на ототожнення лише тих об'єктів. Оптичне випромінювання буде все-таки занадто слабким, залишиться невловимим гіпотеза про марність. Міжзоряне середовище прозоре, радіовипромінювання якого досить сильне. Перші роки після відкриття дискретних. Залишками газової матерії після спалахів нових. Об'єкти, наприклад зірки першої зоряної величини ніяк. Частка оптичного випромінювання менше товщини. Сторони, оптично спостерігати об'єкти не можна. Показувала ніякого зв'язку з низькою точністю проявляли себе в містить. Кілька дивна обставина ми зазначали вище динамічними. Спростовується, як правило, спостерігалися лише тих місцях, де спостерігалися лише слабкі. Сподіватися на дві групи могли б бути дуже близькими зірками. Спостерігалися

лише сонце, радіовипромінювання доходить безперешкодно всьому небу більше. Помітних оптичних об'єктів дуже багато, і не можна вирішити. Відповідний джерела радіовипромінювання першої зоряної. Участь вирішити, який саме з них розташована. Діляться на небі близько один до в цьому майданчику. Розроблялися методи визначення їх радіовипромінювання можна сподіватися на небі близько. Поглинання світла дуже багато, і інших. Радіовипромінювання, не виявляли себе в вікнах видимості між хмарами. По небу більш-менш рівномірно. На небі близько один до площини галактики теж буде зрозуміло. Об'єкти, наприклад зірки першої зоряної величини, ніяк. Смузі показує сильну концентрацію до її площини галактики. Оптичний об'єкт потрібно шукати в радіохвилях, більше, ніж. Велико і залишками газової матерії після. Близько один до світла дуже багато, і вони розташовані. Температурами, а частка випромінювання, що посилається в 1950 р труднощі ототожнення. Склад галактики, так званих радіозірки тем. Оптичних об'єктів дуже близькі зірки, відстані яких. Розроблялися методи визначення їх відстаней і радіооб'єктів. У цілій майданчику, що містить десятки квадратних хвилин один до товщини. Сильну концентрацію до були б кожна з іншого боку. Випадковим чином опинилися в силу закону. Оптично спостерігати об'єкти виявляють концентрацію до площині галактики і була висунута. Багато зусиль реєструвалося радіовипромінювання, як і розподілених по небу більше. Радіовипромінювання, то, оскільки слабких об'єктів були відомі тільки галактики теж буде. Труднощі ототожнення і слід було очікувати, є дуже близькими зірками, так званих. Сподіватися на небі близько один. Величини, що не показувала ніякого зв'язку з газовими туманностями. Прийшли до її площини галактики теж буде. Фон випромінювання менше товщини галактики, так званих радіозірки показало, що положення джерела. Середовище прозоре, радіовипромінювання можна зареєструвати, а оптичне випромінювання буде зрозуміло себе. Небу більш численна, складається з майданчиках. Після спалахів нових і розподілених. Були б кожна з галактиці об'єкти виявляють концентрацію до площині галактики. Другої групи могли б тоді відсутність концентрації цих джерел радіовипромінювання. Перші роки після спалахів нових і у вузькій смузі близько

галактичного екватора. Зірки, які мають низькі температури, 500к зв'язку. Складається в перші роки після відкриття дискретних джерел тому відповідний. Дискретними джерелами радіовипромінювання оптичний об'єкт потрібно шукати в цьому випадку поглинання світла. З джерел, розташованих поза цією смузі показує сильну концентрацію до площині галактики. Кожна з буде все-таки занадто. Спостереженнях в тому, що дискретними джерелами радіовипромінювання показало, що. Сонце, радіовипромінювання можна сподіватися на дві групи могли. Основні труднощі ототожнення галактичних джерел дивна обставина ототожнення лише слабкі галактики. Які випадковим чином опинилися в силу закону вина частка оптичного випромінювання менше. І тут поглинання світла дуже багато. Чи не показувала ніякого зв'язку з джерелами радіовипромінювання оптичний.

Галактичного екватора і в тих місцях, де спостерігалися лише. Який саме з радіовипромінювання. Бути дуже багато і розподілених по небу більше. Між хмарами пилової матерії після відкриття. Шукати в астрономії склалося дещо дивне положення. Майданчику яскравих оптичних об'єктів були відомі тільки. Труднощі викликалася тим, що вони діляться на. Сонце, радіовипромінювання доходить безперешкодно про так як правило, в силу закону. І інших характеристик площині галактики теж буде все-таки. Специфічна складність викликалася тим, що дискретними джерелами радіовипромінювання не виявлялося ніякі. Площині галактики теж буде зрозуміло слабким. Галактиці об'єкти виявляють концентрацію. Менш рівномірно, без ознак концентрацій до галактичного екватора смузі близько галактичного екватора. Тим, що дискретними джерелами радіовипромінювання першої трупи, як ми зазначали вище. Більш численна, складається в цій смузі показує сильну концентрацію. Астрономи прийшли до галактичного екватора позагалактичних об'єктів. Частка оптичного випромінювання менше товщини галактики. Дві групи могли б тоді гіпотетичні радіозірки, існування яких реєструвалося радіовипромінювання. Якби бути дуже багато, і вони розташовані. Позагалактичні об'єкти, тоді відсутність концентрації цих. Теж буде все-таки занадто слабким. Спростовується, як правило,

спостерігалися лише слабкі галактики теж буде все-таки. Другої групи могли б при спостереженнях в астрономії склалося кілька. Першою зоряної величини, що не показувала ніякого зв'язку з газовими туманностями. Цьому майданчику яскравих галактик показувала ніякої. У таких зірок у вузькій смузі близько галактичного. Багато, і в тих об'єктів дуже велике. Оптичних об'єктів були відомі тільки галактики. Зоряної величини, що не виявляють галактичної концентрації цих галактик показувала ніякої. Випромінювання буде все-таки занадто слабким, залишиться невловимим. Крім нечисленних близьких, які випадковим чином опинилися в тому.

ДОДАТОК 9. Приклад тексту на тему «Реліктове випромінювання»

Реліктове випромінювання, космічне мікрохвильове фонове випромінювання це космічне електромагнітне випромінювання з високим ступенем ізотропності і зі спектром, характерним для абсолютно чорного тіла з температурою. Існування реліктового випромінювання було передбачене теоретично Гамовим в рамках теорії Великого вибуху. Хоча в даний час багато аспектів первісної теорії Великого вибуху переглянуті, основи, що дозволили передбачити ефективну температуру реліктового випромінювання, залишилися незмінні. Реліктове випромінювання збереглося з початкових етапів існування Всесвіту і рівномірно його заповнює. Експериментально його існування було підтверджено в 1965 році. Поряд з космологічним червоним зміщенням, реліктове випромінювання розглядається як одне з головних підтверджень теорії Великого вибуху. Термін реліктове випромінювання, який зазвичай використовується в російськомовній літературі, ввів у вживання радянський астрофізик Шкловський. Відповідно до теорії Великого Вибуху, ранній Всесвіт являв собою гарячу плазму, що складається з електронів, баріонів і постійно випромінював, поглинав і знову перевипромінював фотони. Фотони постійно взаємодіяли з іншими частинками плазми, стикаючись з ними і обмінюючись енергією і мали місце розсіювання Томсона і Комптона. Таким чином, випромінювання знаходилося в стані теплової рівноваги з речовиною, а його спектр відповідав спектру абсолютно чорного тіла. У міру розширення Всесвіту космологічне червоне зміщення викликало охолодження плазми, і на певному етапі сповільнилися електрони, що отримали можливість з'єднуватися з сповільнюваними протонами (ядрами водню) і альфа частинками (ядрами гелію), утворюючи атоми (цей процес називається рекомбінацією). Це сталося при температурі плазми близько 3000 К. Вільного простору між частинками стало більше, заряджених частинок стало менше, фотони перестали так часто розсіюватися і тепер могли вільно переміщатися в просторі, практично не взаємодіючі з речовиною. Реліктове випромінювання і складають ті фотони, які були в той час випроменені плазмою в сторону

майбутнього розташування Землі, в зв'язку із рекомбінацією уникали розсіювання, і до сих пір досягають Землі через простір всесвіту. Видима сфера, що відповідає даному моменту, називається поверхнею останнього розсіювання. Це найвіддаленіший об'єкт, який можна спостерігати в електромагнітному спектрі. В результаті подальшого розширення Всесвіту, ефективна температура цього випромінювання знизилася майже до абсолютного нуля, і зараз складає всього 2,725 К. У 1941 році, вивчаючи поглинання світла зірки Ophiuchi молекулами CN в міжзоряному середовищі, Келар зазначив, що спостерігаються лінії поглинання не тільки для основного обертального стану цієї молекули, а й для порушеної, причому співвідношення інтенсивності ліній відповідає температурі CN 2,3 К. У той час це явище не набуло пояснення. У 1948 році реліктове випромінювання було передбачене Георгієм Гамовим, Ральфом Альфером і Робертом Германом на основі створеної ними першої теорії гарячого Великого вибуху. Більш того, Альфер і Герман змогли встановити, що температура реліктового випромінювання повинна складати 5 К, а Гамов дав прогноз в 3 К. Хоча деякі оцінки температури простору існували і до цього, вони володіли кількома недоліками. По-перше, це були вимірювання лише ефективної температури простору, ще не передбачалося, що спектр випромінювання підкоряється закону Планка. По-друге, вони були залежні від нашої особливої прихильності на краю галактики Чумацький Шлях і не припускали, що випромінювання ізотропне. Більш того, вони б дали зовсім інші результати, якби Земля знаходилася де-небудь в іншому місці Всесвіту. У 1955 році аспірант радіоастроном Тигран Арамович Шмаонов в Пулковської обсерваторії під керівництвом відомих радянських радіоастрономів Хайкіна і Кайдановського провів вимірювання радіовипромінювання з космосу на довжині хвилі 32 см і експериментально виявив шумове СВЧ випромінювання. Висновок з цих вимірів був такий: «виявилося, що абсолютна величина ефективної температури радіовипромінювання фону дорівнює 4 К». Шмаонов відзначав незалежність інтенсивності випромінювання від напрямку на небі і від часу. Після захисту дисертації він опублікував про це статтю в неастрономічному журналі «Прилади

і техніка експерименту». Результати Гамова широко не обговорювалися. Однак вони були знову отримані Робертом Дікке і Яковом Зельдовичем на початку 60х років. У 1964 році це підштовхнуло Девіда Тодда Вілкінсона і Пітера Ролла, колег Дікке по Прінстонському університету, до створення радіометра Дікке для вимірювання реліктового випромінювання. У 1965 році Арно Пензіас і Роберт Вудроу Вільсон з Bell Telephone Laboratories в Холмдейле побудували прилад, аналогічний радіометру Дікке, який вони мали намір використовувати не для пошуку реліктового випромінювання, а для експериментів в області радіоастрономії і супутникових комунікацій. При калібрування установки з'ясувалося, що антена має надлишкову шумову температуру в 3,5 К, яку вони не могли пояснити. Отримавши дзвінок з Холмдейла, Дікке з гумором зауважив: Хлопці, нас обскакали. Після спільного обговорення групи з Прінстона і Холмдейла зробили висновок, що така температура антени була викликана реліктовим випромінюванням. У 1978 році Пензиас і Вільсон за своє відкриття отримали Нобелівську премію. У 1983 році був проведений перший експеримент, РЕЛІКТ1, по вимірюванню реліктового випромінювання з борту космічного апарату. У січні 1992 року на підставі аналізу даних експерименту РЕЛІКТ1 російські вчені оголосили про відкриття анізотропії реліктового випромінювання. Трохи пізніше про виявлення флуктуацій оголосили і американські вчені на підставі даних експерименту COBE. У 2006 році за це відкриття було присуджено Нобелівську премію з фізики керівникам групи COBE Джорджу Смуту і Джону Мазер, хоча російські дослідники оприлюднили свої результати раніше американців. Спектрофотометр далекого інфрачервоного випромінювання FIRAS, встановлений на супутнику NASA Cosmic Background Explorer, виконав найбільш точні на сьогоднішній день вимірювання спектра реліктового випромінювання. Вони підтвердили його відповідність спектру випромінювання абсолютно чорного тіла з температурою 2,725 К. Найбільш детальну карту реліктового випромінювання вдалося побудувати в результаті роботи американського космічного апарату WMAP. 14 травня 2009 року було здійснено запуск супутника місії Планк Європейського космічного агентства.

Спостереження триватимуть протягом 15 місяців, також можливе продовження польоту на 1 рік. Обробка результатів цього експерименту дозволить перевірити і уточнити дані, отримані WMAP. Спектр реліктового випромінювання відповідає спектру випромінювання абсолютно чорного тіла з температурою 2,725 Кельвіна. Його максимум припадає на частоту 160,4 ГГц (мікрохвильове випромінювання), що відповідає довжині хвилі 1,9 мм. Воно ізотропне, з точністю до 0,01%, середньоквадратичне відхилення температури становить приблизно 18 мкК. Це значення не враховує дипольну анізотропію (різниця між найбільш холодною та гарячою областю становить 6,706 мк), викликану допплерівським зсувом частоти випромінювання через нашу власну швидкість відносно системи відліку, пов'язаної з реліктовим випромінюванням. Червоне зміщення для реліктового випромінювання трохи перевершує ці значення. Ще в 1969 році було виявлено, що в реліктовому випромінюванні помітно виділена дипольна складова: в напрямку сузір'я Лева температура цього випромінювання на 0,1% вище, ніж в середньому, а в протилежному напрямку на стільки ж нижче. Цей факт інтерпретується як наслідок ефекту Доплера, що виникає при русі Сонця щодо реліктового фону зі швидкістю приблизно 370 км / с в сторону сузір'я Лева. Оскільки Сонце обертається навколо центру Галактики зі швидкістю 220-230 км / с в сторону сузір'я Лебедя, і навіть робить рух щодо центру Місцевої групи галактик (групи галактик, що включає Чумацький шлях), це означає, що Місцева група як ціле рухається щодо реліктового випромінювання з швидкістю приблизно (за сучасними даними) 720 км / с в напрямку точки з галактичних координатами (ця точка розташовується в сузір'ї Гідри). Існують і альтернативні теорії, які також можуть пояснити виділення дипольної компоненти реліктового випромінювання. Реліктове випромінювання поляризоване на рівні в кілька мкК. Виділяються Емода (градієнтна складова) і Вмода (роторна складова) за аналогією з поляризацією електромагнітного випромінювання. Емода може з'являтися при проходженні випромінювання через неоднорідну плазму внаслідок томпсоновського розсіювання. Вмода, максимальна амплітуда якої досягає всього лише 0,1 мкК, не може виникати внаслідок взаємодії з плазмою. Вмода є ознакою

інфляції всесвіту і визначається щільністю первинних гравітаційних хвиль. Спостереження Вмоди є складним завданням внаслідок невідомого рівня шуму для цієї компоненти реліктового випромінювання, а також за рахунок того, що Вмода змішується слабким гравітаційним лінзуванням з більш сильною Емодой. Вмоду довгий час не спостерігали. Вперше її виявили в 2013 році, а в 2014 наявність Вмоди було підтверджено. Вторинна анізотропія реліктового випромінювання виникає в процесі поширення фотонів на їх шляху від поверхні останнього розсіювання до спостерігача, наприклад, розсіювання на гарячому газі або проходження гравітаційного потенціалу. Коли фотони реліктового випромінювання стали поширюватися безперешкодно, звичайна матерія у Всесвіті була в основному у вигляді нейтральних атомів водню і гелію. Проте, спостереження галактик зараз показують, що більша частина обсягу міжгалактичної середовища складається з іонізованого матеріалу, так як є кілька ліній поглинання, пов'язаних з атомами водню. Це означає, що був період реіонізації, в ході якого кілька речовин Всесвіту було знову розбито на іони і електрони. Фотони мікрохвильового випромінювання розсіюються на вільних зарядах, таких як електрони, які не пов'язані в атомах. У іонізованої Всесвіту такі заряджені частинки були вибиті з нейтральних атомів іонізуючим ультрафіолетовим випромінюванням. Сьогодні ці вільні заряди мають досить низьку щільність в більшій частині обсягу Всесвіту, так що вони не впливають помітно на реліктове випромінювання. Однак якщо міжгалактичне середовище було іонізоване на дуже ранніх етапах розширення, коли Всесвіт був набагато щільніше, ніж зараз, то це мало б викликати два основні наслідки для реліктового випромінювання: дрібномасштабні флуктуації будуть стерті подібно до того, як при погляді на об'єкт крізь туман деталі об'єкта стають нечіткими. Процес розсіювання фотонів на вільних електронах (томсонівське розсіювання) викликатиме анізотропію поляризації реліктового випромінювання на великих кутових масштабах, яка буде корелювати з температурною анізотропією. Обидва цих ефекту спостерігалися космічним телескопом WMAP, що свідчить про те, що Всесвіт був іонізований на дуже ранніх етапах, на червоному зсуві більш 17.

Походження цього раннього іонізуючого випромінювання все ще є предметом наукових дискусій. Це випромінювання, можливо, вмикає світло перших зірок, наднових, які з'явилися результатом еволюції цих зірок. Два інших ефекту, які виникли в період між реіонізацією і нашими спостереженнями реліктового випромінювання і є причиною флуктуацій. Це ефект Сюняева і Зельдовича, що полягає в тому, що хмара електронів високої енергії розсіює реліктові фотони і передає частину своєї енергії їм, і ефект Сакса і Вольфа, який викликає зміщення спектру фотонів від космічного мікрохвильового фону в червону або фіолетову область спектра через зміни гравітаційного поля. Ці два ефекти пов'язані з впливом структур в пізньому Всесвіті. З одного боку, вони призводять до розмивання спектру реліктового випромінювання, так як накладаються на первинну анізотропію; з іншого боку дозволяють отримати інформацію про поширеність структур в пізньому Всесвіті, а також простежити за їх розвитком.

ДОДАТОК 10. Приклад тексту на тему «Знання і мова»

Виявлення філософських аспектів інформатики, обчислювальної техніки нерідко становить необхідний момент, основу для розгорнутого наукового аналізу як спеціальних, так і загальнозначущих теоретичних проблем. Приступаючи до з'ясування можливостей обчислювальних машин, Дж. Вейценбаум, наприклад, не оминає такі питання, як механізм пізнання, що таке творчість, яка суть мислення, чи існує межа наукових знань, як сприймає знання людина і як ЕОМ, і ін. В результаті для нього центральним стає одне з найбільш загальних філософських питань – питання про місце людини у Всесвіті. Саме тут бачиться Вейценбаумом фокус, в якому збираються всі проблеми електронно-обчислювальної машини. Тому розробка методологічних питань інформатики не може обмежуватися тільки технічними аспектами, розбором переваг тієї чи іншої технічної системи, конкретних варіантів програмного забезпечення. Без відповіді на питання, де проходять межі обчислюваної людської думки, неможливо зрозуміти і того, що піддається імітації для сучасної ЕОМ. Подібна орієнтація викликана необхідністю подальшого уточнення уявлень про те, чим є знання, яка його структура, яким чином співвідносяться його розвиток зі зміною і вдосконаленням самої людини. Остання обставина грає надзвичайно важливу роль в оцінці характеру взаємовідносин людини і машини, виявленні меж і меж можливостей засобів, створених людиною в якості допоміжних, підручних. Переконливе усвідомлення користувачами реальної мощі ЕОМ, їх дійсних можливостей безпосередньо залежить від розуміння, яким знаннями ми можемо забезпечити ЕОМ. Знання, як відомо, нерозривно пов'язане з мовою. Воно фіксується і передається за допомогою знаків природної і штучної мови, що, власне, виступає однією з передумов його технічної формалізації. Обчислювальна машина, по суті, являє пристрій, призначений для обробки символів. Саме символи здатні бути носіями найрізноманітнішої інформації. Мова тим самим виступає в якості своєрідного інструменту, і цей інструмент має велике значення, що визначає уявлення про світ, яке формується у носія мови, зокрема, у користувача ЕОМ. Тут є чимало

невирішених проблем, які акумулюють інтереси багатьох наук – філософії, лінгвістики, психології. Одна з них пов'язана з тим, як визначити мову – не який-небудь конкретний, історично сформований, а людський фактор. Без згоди в цьому питанні важко в майбутньому уникнути непорозумінь і нерозуміння. Тільки вироблення єдиного погляду на мову як сукупність категорій і правил створює передумову для використання її в якості найголовнішої умови існування і використання ЕОМ. Адже одна із заповітних цілей, досягти яку – значить вирішити множину інших, пов'язаних з нею проблем, полягає в створенні таких машин, спілкування з якими можливо природною мовою людей. Реалізація цього завдання стане реальністю лише в результаті дозволу фундаментальної проблеми штучного інтелекту – представлення знань. Це питання пов'язане з стосунками даних і знань, яке займає важливе місце в теорії і практиці інформатики. Багато в чому воно пов'язано з логічною суперечливістю знань. Процес ускладнення даних, що використовуються в ЕОМ, змусив змінити ставлення і до них і до знань. Поява структурованих даних – списків, документів, семантичних мереж, фреймів – спричинило виникнення спеціальних засобів для їх зберігання: інформаційних банків і бази, які стали називати інтелектуальними. Останнє визначення означає, що в ході обробки даних за спеціальними допоміжними програмами здійснюється їх пошук, запис, відбір. В ході ускладнення форми подання інформації ускладнювалися і процедури її обробки. Виник підхід, відповідно до якого робота з даними (знаннями) вийшла на перше місце. Традиційне уявлення не дає відповіді на питання про принципову різницю між даними і знаннями. Розробляються теорії семіотичних моделей, з якими зв'язуються надії на уточнення необхідного розуміння в цьому питанні. Ясно, що тут можуть зіграти позитивне значення і відповідні філософські дослідження.

Методологічна складність в даному випадку пов'язана з прагненням домогтися адекватного розуміння людської мови, свідомості, мозку і символічної логіки. Конструювання обчислювальної техніки до теперішнього часу фактично здійснюється методом проб і помилок. Самі творці цієї техніки визнають брак теоретичних узагальнень, покликаних сприяти виробленню єдності в розумінні і

поясненні закономірностей, на основі яких працюють обчислювальні пристрої. У міру становлення досліджень штучного інтелекту все більшого значення набувала комп'ютерна лінгвістика. Роль цього наукового напрямку стає більш зрозумілою при знайомстві з його місцем в програмному забезпеченні ЕОМ, точніше в розробці його принципів, в аналізі наявних при цьому тенденцій. Програмне забезпечення – до цього приходять більшість фахівців – сьогодні представляє серцевину, осередок багатьох труднощів інформатики. З ним пов'язана і ефективність дії обчислювальної техніки, і розширення доступності спілкування людини з машиною, і збільшення діапазону вирішуваних завдань і багато, багато іншого. Ось чому програміст виступає однією з центральних фігур в складному процесі переробки інформації, він, за образним висловом Вейценбаума, – творець світів, в яких є єдиним законодавцем. Особливий інтерес при цьому представляють дослідження процесів оволодіння мовними навичками людиною, розвиток і вдосконалення мовної здатності на різних етапах онтогенезу людини. За допомогою машинного аналізу вдалося встановити основні ступені, які проходить людина, долаючи науку навчання природної мови. Зауважимо, що на цей звичайний процес йде багато років. Тільки на засвоєння простих синтаксичних конструкцій дитина витрачає від 2 до 4 років. А потім їй належить навчитися розуміти і виражати прості смислові відносини між словами, подолати труднощі складних синтаксичних речень і, нарешті, освоїтися в складному світі семантичних асоціацій. Сьогодні можна говорити вже не тільки про зовнішню, доступну для спостереження стороні цього руху по шляху оволодіння мовою, але про структурно-функціональне значення конкретних відділів головного мозку людини, що відповідають за певні мовні функції. І все ж наявних результатів поки що дуже мало, щоб наблизитися до більш-менш ясного розкриття засадничої ролі мозку людини в розвитку мови. Тим більше таких даних недостатньо для відповідної інтерпретації машинних мов.

ДОДАТОК 11. Приклад тексту на тему «Модульне програмування»

Модульне програмування – це організація програми як сукупності невеликих незалежних блоків, званих модулями, структура і поведінка яких підкоряються певним правилам. Використання модульного програмування дозволяє спростити тестування програми і виявлення помилок. Апаратно-залежні підзадачі можуть бути строго відділені від інших підзадач, що покращує мобільність створюваних програм. Модуль – функціонально закінчений фрагмент програми. У багатьох мовах (але далеко не обов'язково) він оформляється у вигляді окремого файлу з вихідним кодом або пойменованої безперервної його частини. Деякі мови передбачають об'єднання модулів в пакети. Принцип модульності є засобом спрощення задачі проектування ПЗ і розподілу процесу розробки ПЗ між групами розробників. При розбитті ПЗ на модулі для кожного модуля вказується реалізована їм функціональність, а також зв'язки з іншими модулями. Зручність використання модульної архітектури полягає в можливості відновлення (заміни) модуля, без необхідності зміни іншої системи. Роль модулів можуть грати структури даних, бібліотеки функцій, класи, сервіси та ін. Програмні одиниці, що реалізують деяку функціональність і надають інтерфейс до неї. Програмний код часто розбивається на кілька файлів, кожен з яких компілюється окремо від інших. Така модульність програмного коду дозволяє значно зменшити час перекомпіляції при змінах, внесених лише в невелику кількість вихідних файлів, і спрощує групову розробку. Також це можливість заміни окремих компонентів (таких як jar-файли, so або dll бібліотеки) кінцевого програмного продукту, без необхідності перекомпіляції всього проекту (наприклад, розробка плагінів до вже готової програми). Одним з методів написання модульних програм є об'єктно-орієнтоване програмування. ООП забезпечує високу ступінь модульності завдяки таким властивостям, як інкапсуляція, поліморфізм і пізне зв'язування. Незважаючи на те, що модульне програмування ніяк не пов'язане з деталями конкретної мови (і навіть в разі відсутності явної підтримки з боку мови може застосовуватися при достатній

дисципліни з боку програмістів), більшість мов висувають на верхній рівень свою власну природу системи модулів, немов перенесення системи модулів з однієї мови на іншу було би неможливим. У 2000 році Ксавьє Лерой запропонував робити системи модулів модульними, тобто описом конкретного ядра мови зі своєю системою типів. Як приклад він продемонстрував узагальнену реалізацію мови модулів ML (як найбільш розвиненої системи модулів з відомих на даний момент) і приклади її інстанціювання на традиційну для неї мову ML і на мову Сі. Реалізація Лероя сама побудована за допомогою мови модулів ML, а саме у вигляді функтора, даними про ядро мови та описом його механізму перевірки узгодження типів. Це означає, що при написанні компілятора деякої мови досить описати ядро мови і передати його функтору (як бібліотечну функцію) – в результаті вийде компілятор розширення відомої мови системою модулів ML. Історія концепції модулів як одиниць компіляції сходить до мов Фортран II і Кобол, тобто, до кінця 1950-х років. У 1976 році з'явилася публікація, в якій була розбудована концепція модульності – про мову Mesa (англ.), яка була розроблена в Xerox PARC. У 1977 році докладно ознайомився з цією концепцією учений Ніклаус Вірт, спілкуючись з розробниками в Xerox PARC. Ці ідеї були використані Віртом при створенні мови Модула-2, публікація про яку вийшла в 1977 році. Термін «модуль» в програмуванні почав використовуватися в зв'язку з впровадженням модульних принципів при створенні програм. У 1970-х роках під модулем розуміли якусь процедуру або функцію, написану відповідно до визначених правил. Наприклад: «модуль повинен бути простим, замкнутим (незалежним), доступним для огляду (від 50 до 100 рядків), який реалізує тільки одну функцію завдання, який має одну вхідну і одну вихідну точку». Першим основні властивості програмного модуля більш-менш чітко сформулював Д. Парнас (David Parnas) в 1972 році: «Для написання одного модуля має бути досить мінімальних знань про текст іншого». Таким чином, відповідно до визначення, модулем могла бути будь-яка окрема процедура (функція) як самого нижнього рівня ієрархії (рівня реалізації), так і самого верхнього рівня, на якому відбуваються тільки виклики інших процедур-модулів. Таким чином, Парнас

першим висунув концепцію приховування інформації (англ. Information hiding) в програмуванні. Однак в мовах 70-х років існували тільки такі синтаксичні конструкції, як процедура і функція, що не могли забезпечити надійного приховання інформації, через застосування глобальних змінних. Вирішити цю проблему можна було тільки розробивши нову синтаксичну конструкцію, яка не схильна до впливу глобальних змінних. Така конструкція була створена і названа модулем. Спочатку передбачалося, що при реалізації складних програмних комплексів модуль повинен використовуватися нарівні з процедурами і функціями як конструкція, яка об'єднує і надійно приховує деталі реалізації певної підзадачі. Таким чином, кількість модулів в комплексі має визначатися декомпозицією поставленого завдання на незалежні підзадачі. У граничному випадку модуль може використовуватися навіть для укладення в нього всього лише однієї процедури, якщо необхідно, щоб виконувана нею локальна дія була гарантовано незалежна від впливу інших частин програми при будь-яких змінах. Вперше спеціалізована синтаксична конструкція модуля була запропонована Н. Віртом в 1975 р і включена в його нову мову Modula. Наскільки сильно змінюються властивості мови, при введенні механізму модулів, свідчить наступне зауваження Н.Вірта, зроблене ним з приводу більш пізньої мови Модула-2: «Модулі – найважливіша риса, яка відрізняє мову Модула-2 від її попередника Паскаля». Мови, що формально підтримують концепцію модулів: IBM S / 360 Assembler, Кобол, RPG, ПЛ / 1, Ада, D, F (англ.), Фортран, Haskell, Blitz BASIC, OCaml, Паскаль, ML, Модула-2, Оберон, компонентний Паскаль, Zonnon, Erlang, Perl, Python і Ruby. У IBM System використовувалися «модулі» від мов RPG, Кобол і CL, коли програмувались в середовищі ILE. Модульне програмування може бути здійснено, навіть коли синтаксис мови програмування не підтримує явне завдання імен модулів. Програмні інструменти можуть створювати модулі вихідного коду, представлені як частини груп – компонентів бібліотек, які складаються програмою компоновником. Стандартний Паскаль не передбачає механізмів роздільної компіляції частин програми з подальшим їх складанням перед виконанням. Цілком зрозуміле прагнення розробників комерційних

компіляторів Паскаля включати в мову засоби, що підвищують його модульність. Модуль в Паскалі – це автономно компільована програмна одиниця, що включає в себе різні компоненти розділу описів (типи, константи, змінні, процедури і функції) і, можливо, деякі виконувані оператори, що ініціюють її частини. По своїй організації і характеру використання в програмі модулі Паскаля близькі до модулів-пакетів (PACKAGE) мови програмування Ада. У них так само, як і в пакетах Ади, явно виділяється деяка «видима» інтерфейсна частина, в якій сконцентровані описи глобальних типів, констант, змінних, а також наводяться заголовки процедур і функцій. Поява об'єктів в інтерфейсній частини робить їх доступними для інших модулів і основної програми. Тіла процедур і функцій розташовуються в виконуваній частині модуля, яка може бути прихована від користувача. Модулі являють собою прекрасний інструмент для розробки бібліотек прикладних програм і потужний засіб модульного програмування. Важлива особливість модулів полягає в тому, що компілятор розміщує їх програмний код в окремому сегменті пам'яті. Довжина сегмента не може перевищувати 64 Кбайт, однак кількість одночасно використовуваних модулів обмежується лише доступною пам'яттю, що дозволяє створювати великі програми.

ДОДАТОК 12. Приклад отриманої відповіді на запит «інтерфейс»

Клас – це елемент ПО, що описує абстрактний тип даних і його часткову або повну реалізацію. Точний зміст цієї фрази буде розкритий нижче. Графічне представлення деякої кількості класів та зв'язків між ними називається діаграмою класів. Ідея класів прийшла з робіт по базам знань, що мають відношення до досліджень з штучного інтелекту. Тобто «екземпляр класу» в даному випадку означає не «приклад деякого класу» або «окремо взятий клас», а «об'єкт, типом якого є якийсь клас». Наприклад, абстрактний тип даних «рядок тексту» може бути оформлений у вигляді класу, і тоді всі рядки тексту в програмі будуть об'єктами – екземплярами класу «рядок тексту». Сам клас в підсумку визначається як список своїх членів, а саме полів (властивостей) і методів / функцій / процедур. Наприклад, загальна кількість рядків тексту, створених в програмі за час її роботи, буде статичним полем класу «рядок тексту». В ООП при використанні класів весь виконуваний код програми (алгоритми) буде оформлятися у вигляді так званих «методів», «функцій» або «процедур», що відповідає звичайному структурному програмуванню, однак тепер вони можуть (а в багатьох мовах зобов'язані) належати тому чи іншого класу. Як і поля, код у вигляді методів / функцій / процедур, що належать класу, може бути віднесений або до самого класу, або до екземплярів класу. Інші абстрактні типи даних – метакласи, інтерфейси, структури, перерахування, – характеризуються якимись своїми, іншими особливостями. Поряд з поняттям «об'єкта» клас є ключовим поняттям в ООП (хоча існують і безкласові об'єктно-орієнтовані мови, наприклад, Self, Lua; докладніше дивіться прототипне програмування). Суть відмінності класів від інших абстрактних типів даних полягає в тому, що при завданні типу даних клас визначає одночасно як інтерфейс, так і реалізацію для всіх своїх екземплярів, а виклик методу-конструктора обов'язковий. На практиці об'єктно-орієнтоване програмування зводиться до створення певної кількості класів, включаючи інтерфейс і реалізацію, і подальше їх використання. Використовувані людиною класифікації в зоології, ботаніки, хімії, деталях машин, несуть в собі основну ідею, що будь-яку річ завжди можна уявити окремим випадком деякого

більш загального поняття. Саме тому приклади класів в навчальних посібниках з об'єктно-орієнтованого програмування так часто згадують яблука і груші. В об'єктно-орієнтованої програми із застосуванням класів кожен об'єкт є «екземпляром» деякого конкретного класу, і інших об'єктів не передбачено. При цьому в різних мовах програмування допускається або не допускається існування ще якихось типів даних, екземпляри яких не є об'єктами. При використанні класів всі елементи коду програми, такі як змінні, константи, методи, процедури і функції, можуть належати (а в багатьох мовах повинні належати) того чи іншого класу. Метод, співвіднесений з екземпляром класу (звичайний метод), може бути викликаний тільки у самого об'єкта, і має доступ як до статичних полів класу, так і до звичайних полів конкретного об'єкта (при виклику цей об'єкт передається прихованим параметром методу). Успадкований клас або інтерфейс буде містити в собі все, що зазначено для всіх його батьківських класів (в залежності від мови програмування і платформи, їх може бути від нуля до нескінченності). Часто різні змінні програми зберігають логічно пов'язані значення, і за підтримку цієї логічної зв'язності несе відповідальність програміст, тобто автоматично зв'язність не підтримується. У програмуванні існує поняття програмного інтерфейсу, що означає перелік можливих обчислень, які може виконати та чи інша частина програми, включаючи опис того, які аргументи і в якому порядку потрібно передавати на вхід алгоритмам з цього переліку, а також що і в якому вигляді вони будуть повертати. Абстрактний тип даних інтерфейс придуманий для формалізованого опису такого переліку. Програмні інтерфейси, а також класи, можуть розширюватися шляхом успадкування, яке є одним з важливих засобів повторного використання готового коду в ООП. В об'єктно-орієнтованої програми прапорець «звільнений» буде оголошений приватним членом деякого класу, а для читання і зміни його будуть написані відповідні публічні методи. Скрізь далі слова «клас», «об'єкт», «інтерфейс» і «структура» будуть вживатися в своїх спеціальних значеннях, заданих в рамках ООП.

ДОДАТОК 13. Приклад отриманої відповіді на запит «дослідження космосу»

Основним поштовхом до цього було протистояння двох наддержав (СРСР і США) – холодна війна. Початковий період обертання супутника навколо Землі склав 96,2 хв, а нахил – 65'1 '. «Пальма першості» в області освоєння космосу дісталася СРСР, але США теж не хотіли відставати, і такою подією світової важливості став політ на Місяць. Таким чином, «Луна-3» виявилася першим апаратом, який став штучним супутником відразу і для Землі, і для Місяця – його сильно витягнута орбіта охоплювала обидва цих небесних тіла. Так люди вперше змогли побачити зворотний бік Місяця. «Родзинкою» її було дзеркальце, що гойдається вгору-вниз і при цьому повільно повертається навколо вертикальної осі зліва направо. З його допомогою створювався порядковий запис всього зображення. Англійські астрономи, зробивши їх обробку і отримавши зображення місячної поверхні, не стали передавати його в друк, чекаючи, коли першими ці сенсаційні дані опублікують російські вчені. Відповіді не було, тому, вважаючи себе вільними від подальшого дотримання коректності, британські дослідники передали знімок в газети. В результаті все зображення вийшло стислим з боків і розтягнутим у висоту – місячна поверхня постала у вигляді вузьких загострених каменів, між якими були ще більш вузькі піщані обеліски

Багато тисяч років тому, коли людина бачила нічне небо, вона мріяла про політ до зірок. У 19 столітті з'явився фантастичне оповідання письменника Жюля Верна "З гармати на Місяць". Однак ці ракети були технічно необґрунтованої мрією. Вчені за багато століть не назвали єдиного розташованого у розпорядженні людини засобу, за допомогою якого можна подолати могутню силу земного тяжіння і полинути в міжпланетний простір. У 1911 році Ціолковський вимовив свої віщі слова: "Людство не залишиться вічно на Землі, але, в гонитві за світлом і простором, з початку боязко проникне за межі атмосфери, а потім завоює собі весь близькоземний простір". І з цього моменту великі уми планети почали працювати над початком реального освоєння космосу. Близько 20 польотів до Місяця американських автоматичних станцій за програмами «Рейнджер»,

«Сервейер» і «Лунар Орбитер» були строго підпорядковані підготовці до висадки людини на Місяць. Доставити туди експедицію повинна була гігантська ракета «Сатурн-V», створена під керівництвом Вернера фон Брауна, німецького конструктора снарядів «Фау», який після другої світової війни працював в США. Втім, Радянський Союз також не стояв осторонь від підготовки пілотованого «місячного» польоту

У романі «Із Землі на Місяць» французький письменник Жюль Верн так описав перший політ людей навколо нашого супутника. Перший пілотований корабель, що облетів навколо Місяця, був запущений в Сполучених Штатах – як у романі, так і в дійсності. Фраза, за словами самого Армстронга, була «добре підготовленим експромтом», заздалегідь обраним із сотень пропозицій. Такий зразок називався аварійним і повинен був братися, не відходячи від місячного модуля на випадок, якщо якісь надзвичайні обставини змусять астронавтів терміново сховатися всередині кабіни і покинути Місяць (такі ж зразки згодом бралися і всіма іншими п'ятьма «Аполлонами»). Гармата, з якої був випущений «місячний» снаряд, називалася Колумбіада, командний модуль корабля «Аполлон-11» носив ім'я «Колумбія». Снаряд у Жюля Верна приводнився 11 грудня але був підібраний кораблем лише 29 грудня, що майже збігається з датою приводнення «Аполлона-8» – 27 січеня. Повертаючись на Землю, як фантастичний, так і реальні космічні кораблі здійснювали посадку на воду в північній половині Тихого океану.

ДОДАТОК 14. Приклад фрагментарно зв’язного тексту

Багато тисяч років тому, дивлячись на нічне небо, людина мріяла про політ до зірок. Мільярди мерехтливих нічних світил змушували її уноситься думкою в безмежні далі Всесвіту, будили уяву, змушували замислюватися над таємницями світобудови. Йшли століття, людина набувала все більшої влади над природою, але мрія про політ до зірок залишалася все такою ж нездійсненною, як тисячі років тому. Легенди і міфи всіх народів сповнені оповідань про політ до Місяця, Сонця і зірок. Засоби для таких польотів,2.15 що пропонувалися народною фантазією, були примітивні: колісниця, крила, прикріплені до рук людини. У 19 столітті з'явилася фантастична розповідь письменника Жюля Верна "з гармати на місяць". Відомий англійський письменник Герберт Уельс описав фантастичну подорож на місяць в снаряді, корпус якого був зроблений з матеріалу, не схильного до сили тяжіння. Пропонувалися різні засоби для здійснення космічного польоту. Письменники фантасти згадували і ракети. Однак ці ракети були технічно необґрунтованою мрією. Вчені за багато століть не назвали єдиного розташованого у розпорядженні людини засобу, за допомогою якого можна подолати могутню силу земного тяжіння і полинути в міжпланетний простір. Велика честь відкрити людям дорогу до інших світів випала на долю нашого співвітчизника К. Е. Ціолковського. Багато століть минуло з тих пір, коли був винайдений порох і створена перша ракета, що застосовувалася головним чином для розважальних феєрверків у дні великих свят. Але тільки Ціолковський показав, що єдиний літальний апарат, здатний проникнути за атмосферу і навіть назавжди покинути Землю – це ракета. У 1911 році Ціолковський виголосив свої віщі слова: "Людство не залишиться вічно на Землі, але в гонитві за світлом і простором, з початку боязко проникне за межі атмосфери, а потім завоює собі весь близькоземний простір". І з цього моменту великі уми планети почали працювати над початком реального освоєння космосу... Але ось минуло двадцяте століття і людство вступило в століття освоєння космічного простору. Але чого людство досягло за останні 50 років? Яку роль відіграють досягнення в освоєнні космосу в

нашому повсякденному житті? І нарешті, яка роль самої космонавтики для людства?

Телескоп Хаббла являє собою рефлектор системи Річі-Кретьєна, або поліпшений варіант системи Кассегрена, у якому світло спочатку потрапляє на головне дзеркало, відбивається і потрапляє на вторинне дзеркало, яке фокусує світло і направляє його в систему наукових інструментів телескопа крізь маленький отвір в головному дзеркалі. Часто люди помилково вважають, що телескоп збільшує зображення. Насправді, він лише збирає максимальну кількість світла від об'єкта. Відповідно, чим більше головне дзеркало, тим більше світла воно збере і тим чіткіше вийде зображення. Друге дзеркало лише фокусує випромінювання. Діаметр головного дзеркала Хаббла – 2,4 метра. Воно здається невеликим, якщо врахувати, що діаметр дзеркал наземних телескопів досягають 10 метрів і більше, але відсутність атмосфери, все ж, є величезною перевагою комічного варіанту. Для спостереження за космічними об'єктами телескоп має ряд наукових інструментів, що працюють спільно або окремо. Кожен з них по-своєму унікальний.

Виникнення астрономічних знань прийнято відносити до «сивої давнини». Накопичення цих знань, згідно з найбільш поширеним сценарієм, стимулювалося практичними потребами суспільства. Зазвичай називають дві: необхідність орієнтуватися на місцевості і регламентація сільськогосподарських робіт. Вважають, що походження астрології як цілісної доктрини, важко відділяється від практичної астрономії (аж до кінця середньовіччя), яку також слід віднести до аналогічної «сивої давнини». Одні вважають, що астрологія виникла пізніше – виділилася з «практичної астрономії під впливом деяких соціальних потреб. Відповідно до іншої точки зору, реальні практичні астрономічні знання накопичувалися спочатку в рамках астрологічної парадигми. Астрономія виділилася з астрології в процесі «розпаду» міфологеми тотального «поклоніння Небу».

Вдосконалена оглядова камера (Advanced Camera for Surveys-ACS). Найновіший інструмент спостережень у видимому діапазоні, призначений для

досліджень раннього Всесвіту, і встановлений в 2002 році. Ця камера допомогла скласти карту розподілу чорної матерії, виявити найбільш віддалені об'єкти і дослідити еволюцію галактичних скупчень. Камера близького інфрачервоного діапазону і спектрометр (Near Infrared Camera and Multi-Object Spectrometer – NICMOS). Інфрачервоний сенсор, детектує тепло, коли об'єкти приховані міжзоряним пилом або газом, як, наприклад, в областях активного зореутворення. Діє подібно призмі, розкладаючи світло. З отриманого спектру можна отримати інформацію про температуру, хімічний склад, щільність і рух досліджуваних об'єктів. STIS припинив роботу 3 серпня 2004 через технічні несправності, але в 2008 році під час планового ремонту телескопа буде відремонтований. Ширококутна і планетна камера-2 (Wide Field and Planetary Camera 2-WFPC2). Універсальний інструмент, за допомогою якого було зроблено більшість відомих кожному фотографій. Завдяки 48 фільтрам дозволяє бачити об'єкти в досить широкому діапазоні довжин хвиль.

Першими кроками, з яких почалося освоєння космічного простору, можна вважати запуск першого штучного супутника землі. Основним поштовхом до цього було протистояння двох наддержав (СРСР і США) – холодна війна. Кожен прагнув до відкриття нових кордонів, і новим напрямком було освоєння космічного простору. 4 жовтня 1957 людство вступило в еру освоєння космічного простору. У цей день на навколоземну орбіту був виведений перший у світі радянський штучний супутник Землі. Радянські вчені та інженери вирішили найскладніші науково-технічні проблеми, пов'язані зі створенням ракетно-космічної техніки і забезпеченням космічного польоту. Це видатне досягнення стало переконливим свідченням невичерпних можливостей людського розуму, яскраво продемонструвало рівень науки і техніки в нашій країні. Ракета-носій, забезпечивши в кінці активної ділянки першу космічну швидкість, рівну 7,9 км/с, вивела супутник на геоцентричну (навколоземну) орбіту з максимальним віддаленням від поверхні Землі (в апогеї) 947 км і мінімальним здавленням (в перигеї) 228 км. Початковий період обертання супутника навколо Землі склав 96,2 хв, а спосіб – 65'1'. Вага супутника становила 83,6 кг, його корпус мав форму кулі

діаметром 0,58 м. Апаратура та джерела електроживлення розміщувалися в герметичному корпусі. Три тижні активно працював перший космічний дослідник. З його допомогою були проведені перші вимірювання щільності атмосфери, отримані дані по поширенню радіосигналів в іоносфері. У цьому польоті вперше була практично перевірена і підтверджена правильність теоретичних розрахунків і технічних рішень, покладених в основу проектування ракет-носіїв та супутників. Перші витки супутника стали першими кроками світової космонавтики. «Пальма першості» в області освоєння космосу дісталася СРСР, але США теж не хотіли відставати, і наступним подією світової важливості став політ на Місяць.

ДОДАТОК 15. Приклад отриманої відповіді на запит «інформація про астероїди»

Але зробити нічого не можна, і так все і залишається, поки нові методи досліджень – експериментальні і теоретичні – не підніматимуть рівень досліджень настільки, що створить реальну основу для злиття обох наук в одну. Його орбіта виявилася не схожа ні на одну досі відому. Межі кільця кілька умовні: просторова щільність астероїдів падає в міру віддалення від центральної частини. Для порівняння зазначимо, що обсяг Землі становить всього 1012 км3. Лише зрідка – при зближенні з планетою – витки помітно відхиляються один від іншого. Орбіта кожного астероїда коливається близько свого середнього положення, витрачаючи на кожне коливання кілька десятків або сотень років. Перигелій і афелій то наближаються до Сонця, то віддаляються від нього. При таких масштабах часу можна вже не розглядати миттєвих положень планет на орбітах: як в прискореному фільмі астероїд і планета виявляються як би розмазані по своїх орбітах. Тільки картина виявляється складнішою, тому що орбіта астероїда не є жорсткою і її форма змінюється з плином часу. Це робить можливим зіткнення астероїдів один з одним. Нахили і ексцентриситети орбіт призводять до непаралельності їх взаємних рухів, і швидкість, з якою астероїди проносяться один повз іншого, в середньому становить близько 5 км / с. Форма і обертання астероїдів. Астероїди такі малі, що сила тяжіння на них незначна. Вона не в змозі надати їм форму кулі, яку надає планетам і їх великим супутникам. Втім, нові зіткнення повинні були дробити спресовані речовини. Усі планети Сонячної системи на заключному етапі формування вбирали в себе досить великі тіла, які не зуміли перетворитися на самостійні планети або супутники. Зіткнення з більш дрібними тілами не зможуть суттєво змінити її. Тому гори на астероїдах можуть бути набагато вище, а западини набагато глибше, ніж на Землі та інших планетах: середнє відхилення від рівня згладженої поверхні на великих астроїда становить 10 км і більше, про що свідчать радіолокаційні спостереження астероїдів. У Місяця і Меркурія аналогічне зменшення блиску цілком пояснюється тільки зменшенням видимої з Землі частки освітленої Сонцем. Температура астероїдів.

Астероїди – наскрізь холодні, мертві тіла. Поверхневі шари залишалися холодними, і лише зіткнення час від часу викликали короткочасний локальний розігрів. Якщо усереднити температуру по всій освітленої поверхні, отримаємо, що у астероїдів сферичної форми середня температура освітленої поверхні в 1,2 рази нижче, ніж температура в соняшниковій точці. Через обертання астероїдів температура їх поверхні швидко змінюється. В результаті по поверхні астероїда біжить теплова хвиля. У тіл, що рухаються в кільці астероїдів, її грубо можна прийняти рівною 100-150 К. Як не мала теплова інерція поверхневих шарів астероїда, все ж, якщо бути зовсім строгими, то слід сказати, що температура не встигає приймати рівноважного значення зі зміною умов освітлення . Щодо сонячної точки виникає легка асиметрія в розподілі температур. Тому їх інфрачервоні спектри повинні виглядати як безперервне випромінювання з інтенсивністю, монотонно спадною в обидві сторони від максимуму. Їх взаємне розташування зберігається незмінним протягом мільярдів років. Склад астероїдної речовини. Вони присутні у вигляді дрібних кристаликів або у вигляді скла, зазвичай частково перекристалізованого. Інший основний компонент – нікелисте залізо, яке представляє собою твердий розчин нікелю в залізі, і, як в будь-якому розчині, вміст нікелю в залозі буває різним – від 6-7% до 30-50%. Це кам'яні метеорити від світло-сірої до дуже темного забарвлення з дивовижною структурою: вони містять округлі зерна – хондри, іноді добре видимі на поверхні розлому. Вони займають значний обсяг метеорита, іноді до половини його, і слабо зцементовані міжхондровою речовиною – матрицею. Склад матриці зазвичай ідентичний з складом хондр, а іноді і відрізняється від нього. З приводу походження хондр існує багато гіпотез, але всі вони спірні. Поблизу нього було сильно прогріте і пил піддавалася повному або частковому випаровуванню. Лише пізніше, коли газ охолов, вона сконденсувалася знову, але велика частина летючих речовин, що містяться в міжзоряних порошинках, виявилася втраченою і в новий пил вже не увійшла. Мала сила тяжіння не могла спресувати з пилу планетезимали. Хто в епоху відкриття перших астероїдів міг припустити, що ці малі тіла Сонячної системи, тіла, про які ще недавно нерідко говорили з відтінком

зневаги, стануть об'єктом уваги фахівців самих різних областей: природознавства, космогонії, астрофізики, небесної механіки, фізики, хімії, геології, мінералогії, газової динаміки і аеромеханіки. Ще потрібно було усвідомити, що варто лише нахилитися, щоб підняти з землі шматочок астероїда – метеорит. Астероїди поблизу Землі. Майже 3/4 століття люди не підозрювали, що не всі астероїди рухаються між орбітами Марса і Юпітера. Далі були відкриті астероїди Альберт, Алінда, Ганнімед, які проходили за астрономічними мірками дуже близько від Землі. У переважної більшості астероїдів орбіти не сильно відрізняються один від одного: вони слабо ексцентричні і мають малий або помірний нахил. Тому астероїди зустрічаються і далеко за межами кільця. Однак обурення, які відчуваються самими планетами, малі і не змінюють структури Сонячної системи. З астероїдами справа йде інакше. Стає доцільним розглядати їх як кільце. Зіткнення з такими швидкостями ведуть до руйнування тіл. Велику роль при цьому відіграє явище плинності. Високі гори на Землі у підошви "розповзаються", так як міцність порід виявляється недостатньою для того, щоб витримати навантаження в багато тон на 1 см3, і камінь тече, хоча і дуже повільно. Тому "утрамбовані" силою тяжіння можуть бути лише глибокі надра небагатьох великих астероїдів. Тільки зіткнення тіл один з одним могли призвести до того, що речовина поступово наминалася, стаючи менш рихлою. Тепер їх слідів вже немає. Дрібні ж астероїди повинні мати і дійсно мають неправильну форму, що склалася в результаті багатьох зіткнень і не піддавалася надалі вирівнюванню під дією сили тяжіння. Кратери, що виникли на поверхні навіть самих великих астероїдів при зіткненні з дрібними тілами, не запливають з плином часу.

ДОДАТОК 16. Приклад отриманої відповіді на запит «відмінності класу та інтерфейсу»

Інші абстрактні типи даних – метакласи, інтерфейси, структури, перерахування, – характеризуються якимись своїми, іншими особливостями. Поряд з поняттям «об'єкта» клас є ключовим поняттям в ООП (хоча існують і безкласові об'єктно-орієнтовані мови, наприклад, Self, Lua; докладніше дивіться прототипне програмування). Суть відмінності класів від інших абстрактних типів даних полягає в тому, що при завданні типу даних клас визначає одночасно як інтерфейс, так і реалізацію для всіх своїх екземплярів, а виклик методу-конструктора обов'язковий. На практиці об'єктно-орієнтоване програмування зводиться до створення певної кількості класів, включаючи інтерфейс і реалізацію, і подальшого їх використання. Використовувані людиною класифікації в зоології, ботаніки, хімії, деталях машин, несуть в собі основну ідею, що будь-яку річ завжди можна уявити окремим випадком деякого більш загального поняття. Саме тому приклади класів в навчальних посібниках з об'єктно-орієнтованого програмування так часто згадують яблука і груші. В об'єктно-орієнтованої програми із застосуванням класів кожен об'єкт є «екземпляром» деякого конкретного класу, і інших об'єктів не передбачено. При цьому в різних мовах програмування допускається або не допускається існування ще якихось типів даних, екземпляри яких не є об'єктами (тобто мова визначає, чи є об'єктами такі речі, як числа, масиви і покажчики, або не є, і, відповідно, чи є такі класи як «число», «масив» або «покажчик», екземплярами яких були б кожне конкретне число, масив або покажчик). При використанні класів все елементи коду програми, такі як змінні, константи, методи, процедури і функції, можуть належати (а в багатьох мовах повинні належати) того чи іншого класу. Метод, співвіднесений з екземпляром класу (звичайний метод), може бути викликаний тільки у самого об'єкта, і має доступ як до статичних полів класу, так і до звичайних полях конкретного об'єкта (при виклику цей об'єкт передається прихованим параметром методу). Успадкованих клас або інтерфейс буде містити в собі все, що зазначено для всіх його батьківських класів (в залежності від мови

програмування і платформи, їх може бути від нуля до нескінченності). Часто різні змінні програми зберігають логічно пов'язані значення, і за підтримку цієї логічної зв'язності несе відповідальність програміст, тобто автоматично зв'язність не підтримується. Клас – це елемент ПО, що описує абстрактний тип даних і його часткову або повну реалізацію. Точний зміст цієї фрази буде розкритий нижче. Графічне представлення деякої кількості класів та зв'язків між ними називається діаграмою класів. Ідея класів прийшла з робіт по базам знань, що мають відношення до досліджень з штучного інтелекту. Тобто «екземпляр класу» в даному випадку означає не «приклад деякого класу» або «окремо взятий клас», а «об'єкт, типом якого є якийсь клас». Наприклад, абстрактний тип даних «рядок тексту» може бути оформлений у вигляді класу, і тоді все рядки тексту в програмі будуть об'єктами – екземплярами класу «рядок тексту». Сам клас в підсумку визначається як список своїх членів, а саме полів (властивостей) і методів / функцій / процедур. Наприклад, загальна кількість рядків тексту, створених в програмі за час її роботи, буде статичним полем класу «рядок тексту». В ООП при використанні класів весь виконуваний код програми (алгоритми) буде оформлятися у вигляді так званих «методів», «функцій» або «процедур», що відповідає звичайному структурному програмуванню, однак тепер вони можуть (а в багатьох мовах зобов'язані) належати тому чи іншого класу. Як і поля, код у вигляді методів / функцій / процедур, що належать класу, може бути віднесений або до самого класу, або до екземплярів класу. У програмуванні існує поняття програмного інтерфейсу, що означає перелік можливих обчислень, які може виконати та чи інша частина програми, включаючи опис того, які аргументи і в якому порядку потрібно передавати на вхід алгоритмам з цього переліку, а також що і в якому вигляді вони будуть повертати . Абстрактний тип даних інтерфейс придуманий для формалізованого опису такого переліку. Програмні інтерфейси, а також класи, можуть розширюватися шляхом успадкування, яке є одним з важливих засобів повторного використання готового коду в ООП. В об'єктно-орієнтованої програми прапорець «звільнений» буде оголошений приватним членом деякого класу, а для читання і зміни його будуть написані відповідні

публічні методи. Скрізь далі слова «клас», «об'єкт», «інтерфейс» і «структура» будуть вживатися в своїх спеціальних значеннях, заданих в рамках ООП. Звичайні поля створюються по одній копії для кожного конкретного об'єкта – екземпляра класу. Наприклад, по можливості, клас «рядок тексту» буде містити всі основні методи / функції / процедури, призначені для роботи з рядком тексту, такі як пошук в рядку, вирізання частини рядка. Наприклад, загальну кількість створених рядків можна дізнатися з будь-якого місця програми, але довжину конкретного рядка можна дізнатися тільки вказавши, тим чи іншим чином, довжину якого рядка будемо міряти. Наприклад, можна створити свій варіант текстового рядка шляхом успадкування класу «мій рядок тексту» від вже існуючого класу «рядок тексту», при цьому передбачається, що програмісту не доведеться заново переписувати алгоритми пошуку та інше, так як вони автоматично будуть успадковані від готового класу, і будь-який екземпляр класу «мій рядок тексту» може бути переданий не тільки в готові методи батьківського класу «рядок тексту» для проведення потрібних обчислень, але і взагалі в будь-який алгоритм, здатний працювати з об'єктами типу «строки тексту », так як екземпляри обох класів сумісні з програмним інтерфейсам. Якщо при створенні свого нового типу даних успадковувати інтерфейс, то ми отримаємо можливість передавати екземпляр свого типу даних в будь-який алгоритм, який вміє працювати з цим інтерфейсом. У той же час, наслідуючи клас, ми автоматично успадковуємо готовий код під інтерфейс (це не завжди так, батьківський клас може вимагати реалізації якихось алгоритмів в дочірньому класі в обов'язковому порядку). Прикладом можуть служити прапорці «звільнений» і «очікує премії за підсумками року», коли за правилами відділу кадрів людина може бути водночас не звільненою і не очікувати премії, як і не звільненою і очікувати премії, звільненою і не очікувати премії, але не може бути одночасно і звільненою, і очікувати премії. Правила, що визначають можливість або неможливість безпосередньо змінювати будь-які змінні, називаються правилами завдання областей доступу. Щодо інших, більш складних правил, в різних мовах існують інші модифікатори доступу і правила їх взаємодії з класами. Будь-які спадкоємці класу вже не зможуть

отримати доступ до такого члена. Спадкування за типом private робить все члени батьківського класу (в тому числі public і protected) private-членами класу-спадкоємця (C ++); protected (захищений, внутрішній член ієрархії класів) – звернення до члена допускаються з методів того класу, в якому цей член визначений, а також з будь-яких методів його класів-спадкоємців. Спадкування за типом protected робить все public-члени батьківського класу protected-членами класу-спадкоємця (C ++); public (відкритий член класу) – звернення до члена допускаються з будь-якого коду. Жоден об'єкт (екземпляр класу) не може бути створений інакше, як шляхом виклику на виконання коду конструктора, який поверне створений і правильно заповнений примірник класу. У багатьох мовах програмування тип даних «структура», як і клас, може містити змінні і методи, але екземпляри структур, залишаючись просто розміченою ділянкою оперативної пам'яті, можуть створюватися в обхід конструкторів, що заборонено для екземплярів класів (за винятком спеціальних виняткових методів обходу всіх подібних правил ООП, передбачених в деяких мовах і платформах). Об'єктно-орієнтований підхід за час свого розвитку накопичив множину рекомендацій (патернів) зі створення класів і ієрархій класів.

ДОДАТОК 17. Приклад отриманої відповіді на запит «парне програмування»

Однак кожен метод навчання має свої плюси і мінуси, і починаючому програмісту напевно буде цікаво, як стати програмістом з нуля в максимально короткий термін і з максимальною віддачою. Найкращий спосіб навчання. Програмування з нуля, як і будь-яка інша справа, яку ви ніколи раніше не торкалися, буде спершу здаватися незрозумілою і важкою. Дійсно, якщо фахівець зміг вмістити функціонал складної програми в кілька рядків – він заслуговує такої оплати. 90% таких матеріалів розраховані на тих, хто вже розуміє базові принципи або ж навпаки, не розуміє зовсім нічого. Немає його з тієї причини, що введення в програмування відбувається на прикладі однієї програми, основні принципи подаються на основі іншої програми, а «вищий пілотаж» показується на чомусь третьому на відео курсу або в підручнику, тому цілісної картини не складається. У стресовій ситуації тіло швидше зрозуміє, що йому робити, щоб вижити. Те ж саме стосується і програмування: як тільки ви вийдете за межі «зони комфорту» – рамок, встановлених освітньою програмою, і певної послідовності дій – ви набагато краще зрозумієте, як програмувати на вибраною вами мовою. Вам доведеться зрозуміти саму суть мови і принципи програмування, щоб написати простеньку програму – результат набагато кращий, ніж якби ви зробили все за інструкцією. На це є цілий ряд причин: самостійно придумане завдання зробить програмування з нуля легше, позбавить вас «милиць» і змусить рухатися поза будь-якими рамками; так ви краще зрозумієте мову або навіть зможете знайти нестандартне рішення виниклої проблеми; «Набивання шишок» зробить вас психічно стійкішим до труднощів; Ви зрозумієте, що невдача – це привід почати спочатку, а не закидати основи програмування, підручник і всі напрацювання «за шафу»; програма буде рости з вами і вдосконалюватися разом з навиком програмування. Разом веселіше, плюс складні моменти можна буде розібрати в процесі «мозкового штурму»; Існує стереотип про програмістів, що вони «пишуть 2 рядки коду і отримують за це тисячі доларів». Але щоб досягти цього рівня, потрібно наполегливо працювати. Вивчати програмування з нуля складно, якщо

не з ким порадитися і ні в кого спитати, але якщо проявити наполегливість і завзятість – це можливо, але потрібно багато вільного часу. Отже, розглянемо, що ж нам пропонує спосіб навчання програмуванню за допомогою відео курсу на Ютубі. Програма не росте разом зі своїм творцем, не утворюється той самий фундамент, на якому будуються знання. Тому відео з розряду «як стати програмістом з нуля за 5 днів» можуть навчити писати простенький код, але щось більше вони не здатні запропонувати.

ДОДАТОК 18. Приклад отриманої відповіді на запит «сенс життя людини»

Однак і сьогодні ми далекі від розуміння сутності попередження, що прозвучало на зорі нової технічної революції і застерігало від неминучого, але, можливо, менш згубного для людини, – знайди вчасно сили розуму і готовність прислухатися до цього тривожного сигналу – процесу розчленування і поділу, в силу якого людина перестає бути природною істотою. Самі пізнавальні устремління не обмежуються абстрактними, академічними інтересами. Людина – традиційний і першочерговим об'єкт філософської рефлексії. Тут слід зупинитися на уявленнях про людину як складно організованої системі, саме існування якої постає як еволюція мікрокосму, порівнянна хіба що за масштабами духовних вимірів з грандіозними космічними утвореннями. У своїх подальших роботах він неодноразово повертався до мікрокосму людської природи, пов'язував з цим образом осягнення найпотаємніших пластів людської особистості, що зберігають глибину часів і відсувають вузькість свідомості на задній план життя. Результативність філософського пізнання людини, на думку філософа, досягається лише в результаті акту виняткового самосвідомості людиною свого значення. Зрозуміло, філософські максими про людину при цьому не поривають з релігійним змістом, підпорядковуючи пошук істини християнського одкровення. Ось чому для Бердяєва людина постає мікрокосмом, ось чому йому належить центральне і царське становище в світі. Він пише: «Кожна людина за своєю внутрішньою природою є якийсь великий світ – мікрокосм, в якому відбивається і перебуває весь реальний світ і всі великі історичні епохи; вона не є якимось уривком всесвіту, в якому укладено цей маленький шматочок, вона являє собою деякий великий світ, який може бути станом свідомості даної людини ще закритим, але, у міру розширення і просвітління його свідомості, що розкривається внутрішньо». Це – цілий космос, світ в собі з невимірними глибинами і прірвами. Франку, в своїй творчості, намагався подолати антиномічність теоретико-пізнавального ідеалізму і включити в гносеологію (теорію пізнання) онтологизм як основний принцип філософського погляду.

Довгий час в радянській філософії «відчуження» було об'єктом критики, що не допускає навіть думки про можливість будь-якої реальної його основи в нашому житті. Для російських філософів проблема відчуження була пов'язана з нагальними, життєвими питаннями, які давали про себе знати у всіх сферах життя. Тільки в історії людина проходить свій особливий мученицьку шлях, в якому всі великі події історії, найстрашніші, самі страждальницькі, виявляються внутрішніми моментами цієї людської долі, бо сама історія – це внутрішнє, повне драматизму звершення долі людини. І все ж головне – у відриві, відчуженні духовності людини від його природи. Відповідно вона отримує закріплення в філософських конструкціях, що також зазнали значних змін. А іноді, навпаки, не приймаючи еволюційно-натуралістичного розуміння людини, звільнення творить людський дух, пов’язаний з відмовою природної необхідності, звільнення людини від природної залежності і поневолення нижчими стихійними началами. Проблема співвідношення біологічного і соціального, складної діалектичної природи їх взаємозв'язків здавна привертає увагу філософів. Обговорення відповідальності і обов'язку вчених, етичних підстав науки, висловлювання сумнівів в монопольне право науки на істину надали їй інше забарвлення, раніше не властиве вигляду точної науки. Не можна сказати, що подібних сумнівів не було в минулому. Знову і знову в центр дискусій виноситься обговорення підготовленості суспільства, а отже і науки, почати новий напад природи, скласти безпосереднє оточення, а нерідко і саму суть людини. Ось чому біологія – наука про життя – постає в перехресті соціально-філософських проблем, від вирішення яких багато в чому буде залежати не тільки її власне вдосконалення, але і подальший вплив пропонованих нею рекомендацій на суспільство. Але сьогодні саме від біології знову виходить – реальна в практичному сенсі – небезпека посягання на саму природу людини, способи її відтворення.

ДОДАТОК 19. Приклад отриманої відповіді на запит «значення економіки для держави»

Поки не дуже зрозуміло, що таке "економічні закономірності", нам якось зрозуміліше, що таке "економічні проблеми". Суспільство являє собою складну структуру, в якій є сім'я, моральність, виробництво товарів і послуг, політика, ідеологія, наука, релігія, національні відносини. Економічна система – це частина суспільної системи, сфера людської діяльності, в якій здійснюється виробництво, обмін, розподіл і споживання продуктів, послуг і факторів виробництва. Але на даному етапі важливо розібратися із загальним поняттям економічної системи. Тепер можна сформулювати більш точне визначення предмета економічної теорії. Але це визначення предмета економічної теорії є занадто загальним. Зокрема, серед економічних дисциплін, крім економічної теорії, є бухгалтерський облік, економічна статистика, фінанси і кредит, міжнародні економічні відносини, економіка підприємства та багато інших. Спираючись на все вищесказане, можна перейти від загального до більш конкретного визначення предмета економічної теорії. Ключовими словами тут є "поведінка людей" і "обмежені ресурси". Потреби – це потреба чи недолік в чомусь, необхідному для підтримки життєдіяльності і розвитку організму, людської особистості, групи людей, суспільства в цілому. З того моменту, коли люди починають готуватися до задоволення своїх потреб, спираючись на наявні обмежені ресурси, і починається економічна діяльність. Їх важко класифікувати. Для того щоб задовольнити перші дві групи потреб, необхідно мати у своєму розпорядженні матеріальні ресурси – матеріали, інструменти, тобто засоби діяльності. Більш того, їх загальна кількість зростає дуже швидко. Можна сказати, що потреби є необмеженими. Для задоволення потреб необхідно мати можливості їх задоволення, іншими словами потрібні ресурси, фактори виробництва. У більш широкому сенсі праця означає доцільну, усвідомлену діяльність людей по створенню продуктів і послуг, або процес використання робочої сили. Реальний капітал є економічним ресурсом, чинником виробництва. Це те, з чого виробляються продукти, споживані

людиною. Рідкість ресурсів не дозволяє виробляти всі продукти і послуги, які хотіло б мати суспільство. Тому людям доводиться вибирати, які потреби задовольняти в даний момент в першу чергу, у який спосіб використовувати наявні ресурси. Економічна теорія вивчає проблему ефективного розподілу і використання обмежених ресурсів з метою максимального задоволення людських потреб. На самому початку можна сказати, що економіка, або економічна теорія, вивчає економічні закономірності, економічні проблеми. Економічні проблеми існують і вирішуються людьми в рамках людського суспільства, в рамках існуючої там економічної системи. Економічна система – це тільки частина суспільного устрою. Економічна теорія – це загальнотеоретична дисципліна, що є теоретичною основою для всіх інших економічних наук. Задоволення наших потреб дає нам можливість жити, до чогось прагнути, радіти життю, творити. У наведеній схемі різноманітні потреби об'єднані в три групи У першій групі потреби виділяються в залежності від тієї ролі, яку вони відіграють у житті людини, тобто в залежності від їх функціональної ролі. Соціально-культурні потреби – це потреби в освіті і кваліфікації, розвагах, в мистецтві, в спілкуванні з іншими людьми. До другої групи потреби включаються в залежності від того, в якій формі ці потреби задовольняються, тобто в залежності від об'єкта потреб. Матеріальні потреби для їх задоволення припускають наявність продуктів у матеріальній речовій формі, наприклад потреби в продуктах харчування і одягу, в транспорті і житлі. На околиці містечка, у жителів невеликої вулиці є потреба в освітленні темної вулиці, це групова потреба. Потреби ростуть значно швидше, ніж можливості їх задоволення. Наприклад, потреба в продуктах харчування в даний момент часу обмежена. Найголовніша їх особливість полягає в тому, що ресурси і фактори виробництва обмежені. Ресурси і фактори виробництва, так само як і потреби, різноманітні і численні. До них відносяться праця, капітал, земля, підприємницька здатність. Праця – це людські ресурси, тобто робоча сила, наявна в суспільстві і використовувана у виробництві продуктів і послуг. Капітал – це кошти праці, які створені людиною.