УДК 004.04 https://doi.org/10.15407/pp2020.04.071

О.Р. Жежерун, О.Р. Смиш

АВТОМАТИЗАЦІЯ РОЗВ'ЯЗУВАННЯ ЗАДАЧ З ПЛАНІМЕТРІЇ, ЗАПИСАНИХ ПРИРОДНОЮ УКРАЇНСЬКОЮ МОВОЮ

У роботі досліджено й описано створення системи для розв'язування задач з планіметрії за допомогою сучасних можливостей обробки природної української мови та розробленої сукупності алгоритмів опрацювання тексту задачі. Розробка базується на аналізі текстів планіметричних задач та аналізі доступних засобів обробки живої української мови, що наразі наявні. Результатом роботи є кінцевий програмний продукт, написаний мовою Python, що дає змогу вирішувати прості завдання з планіметрії.
Ключові слова: обробка природньої мови, токенізація, лематизація, розмічування частин мови, сегментація тексту, видобування інформації, розмічений корпус.

Вступ

Мова – це той інструмент, за допомогою якого люди спілкуються та розуміють одне одного. Саме ці мови, які використовує людство у повсякденному житті між собою є природною, тобто такою, що виникла природнім шляхом серед людей. Проте, коли потрібно задати команди комп'ютеру, використовують формальну (штучну) мову – мову програмування. Мова програмування є тим ключем, що дає змогу командами (наборами інструкцій) створити зв'язок між людьми та комп'ютерами.

Природна мова має вагому, досі невирішену проблему, через що не годиться для взаємодії з комп'ютером, здебільшого через свої синтаксичні, смислові, відмінкові та референційні неоднозначності.

Обробка природної мови (англ. Natural language processing або NLP) – галузь в лінгвістиці, комп'ютерних науках, інформаційній інженерії та штучному інтелекті, яка спрямована на комп'ютерний аналіз та обробку природної (людської) мови. Загалом, метою NLP є надати можливість уніфікувати природну мову для розуміння її комп'ютером. Звісно, наразі машини не здатні розуміти українську мову, так само як розуміють її люди, проте вже нині вони мають досить великі можливості. Але, варто зазначити, що найбільші можливості все ж доступні лише для англійської мови. У цій роботі розглядається обробка природної української мови.

Розглянемо деякі ланки, що передбачає NLP, які будуть використовуватися в цій роботі.

Сегментація – це поділ тексту на певні значущі одиниці, такі як слова, речення, абзаци. Поділ на слова в українській мові, як і в інших багатьох мовах світу, що певною мірою використовують кирилицю чи латиницю, не є складним завданням, оскільки такі мови для поділу на слова застосовують пробіли, тобто пусті пропуски між словами, або знаки пунктуації. Дещо складніша ситуація з поділом на речення. Хоч речення в українській мові треба починати з великої літери та закінчувати крапкою (чи іншим символом пунктуації, що позначає закінчення речення, як-от знак оклику), все ж є слова, які граматично правильно писати з великої букви, наприклад імена людей. Також існують скорочення слів, де потрібно поставити крапку [1].

Розмічування частин мови (англ. Part-of-speech tagging) – це встановлення кожному слову з тексту відповідного теґу, який вказує, яка це частина мови зважаючи на визначення слова та на контекст у словосполученні, реченні чи абзаці. Розмічування частин мови ускладнюється тим, що в природній українській мові одне й те ж слово у різних реченнях може відповідати різним частинам мови, а отже й мати інше значення.

Стемінг (англ.Stemming) – процес скорочення слова до основи шляхом відкидання допоміжних частин, таких як закінчення чи суфікс.

Лематизація (англ. Lemmatization) – це процес отримання базової словникової форми слова – леми. На відміну від стемінгу лематизація використовує у процесі словник та морфологічний аналіз для цього.

Видобування інформації – це процес вилучення з неструктурованого або малоструктурованого тексту структурованої інформації, такої як сутності, зв'язки між ними, атрибути [2]. Як галузь в NLP, видобування інформації набуває все більшої зацікавленості, оскільки гостро постає необхідність у структуруванні інформації. Варто також додати, що під кожну задачу, надбудовується додаткова логіка на готовий інструмент роботи з текстом, оскільки охопити всі аспекти різних задач наразі не є можливим.

Різновиди геометричних задач

Геометрія, як наука, поділяється на певні галузі такі, як планіметрія, стереометрія, тригонометрія та інші. У цій роботі розглянуто планіметричні задачі, оскільки їхні умови легше піддаються опису в текстовій формі (можливо обійтись без зображення фігур та без тригонометричних рівнянь), також вивчення геометрії розпочинають саме з планіметрії, про що свідчать програми шкільної освіти. Слід провести аналіз видів задач та можливостей їхнього подання.

Існують різні типи задач, які відрізняються між собою, як складністю обрахунку, так і структурою побудови умови, питання (невідоме, що потрібно знайти) і можливостей відповіді на неї.

Наразі планіметричні задачі учні починають вивчати починаючи з 7 класу школи в Україні. Зважаючи на це, доцільно ознайомитися з підручниками школярів за 7 клас та з'ясувати структуру задач, які там використано. Використовуючи термінологію ЗНО [3], у цій роботі будемо розглядати завдання відкритого типу з короткою відповіддю. Далі розглянемо структуру та побудову таких задач, вивівши їхні закономірності шляхом статистичного аналізу, для цього взявши задачі потрібного типу зі шкільних підручників. Для демонстрації наведено приклад задачі відкритого типу з короткою відповіддю, що ілюструють більшість задач із підручника: «У трикутнику ABC відомо, що ∠A = 30°, ∠B = 45°, CM – висота, AC = 10 см. Знайдіть відрізок BM.». Надалі, саме цю задачу використано як приклад, на основі якого простежуватиметься хід обробки та розв'язку всіх задач.

Отже, з тексту задачі видно, що завдання подається здебільшого двома реченнями (рідше одним, ще рідше більш як два), де перше вказує, що дано (умова), а друге пояснює, що потрібно знайти. Як можна побачити, в підручниках також використані спеціальні математичні символи (знак кута, знак градуса та інші), що скорочують написання обсягу тексту, але, як буде видно далі в роботі, ускладнює роботу опрацювання такого тексту аналізатором. Усі іменування змінних (сутностей) подаються латинськими літерами, що очевидно та міжнародно прийнято. Спеціального слова, яке ідентифікувало би початок подання інформації умови (дано) немає, тому речення починаються прямо з подачі цієї умови. Проте, речення, що пояснює шукане, зазвичай починається зі слова «знайдіть», рідше «обчисліть» та «визначте», ще рідше зустрічаються питання, що починаються на «чому дорівнює». Розв'язком на такі типи задач зазвичай є одне число, наприклад, шуканий градус кута («Знайдіть ∠AMC.») чи довжина сторони («Знайдіть гіпотенузу AB.»), рідше зустрічається сукупність («Знайдіть бічні сторони трикутника.»).

Для проведення кількісного аналізу, яке зможе показати частоту повторень слів, що так само допоможе у визначенні на що саме варто сконцентрувати увагу, насамперед потрібно звести весь список зібраних задач до певного однакового виду. Тому для цього варто виконати лематизацію, а символи замінити на прописне слово. Процес методу реалізації лематизації є одним із ключових етапів попередньої обробки тексту, бо дає змогу вилучити закінчення й повертає основну чи словникову форму слова, яка й називається лемою. Виконавши процес лематизації поданого вище прикладу задачі, отримано таке речення: «у трикутник ABC відомо, що кут

А дорівнювати 30 градус, кут В дорівнювати 45 градус, СМ — висота, АС дорівнювати 10 см. знайти відрізок ВМ.». Очевидно, що метод зняв відмінкові форми зі слів, що допомагає при подальшому кількісному аналізі задач. Проведено лематизацію вибраних 76 задач, взятих з підручника. Фрагмент результату показано в таблиці.

Таблиця. Фрагмент результату кількісного аналізу частоти повторень слів у 76 задачах з планіметрії

№	Слово	Повторюваність
1	дорівнює	137
2	см	119
3	трикутник	95
4	кут	82
5	знайти	80
6	градус	54
7	і	45
8	сторона	44
9	висота	32
10	у	29
11	abc	27
12	основа	26
13	що	22
14	прямокутний	20

З таблиці видно, що на першому місці за повторюваністю є слово «дорівнює», потім одиниця виміру відрізків «см». На третьому місці розташоване слово «трикутник», з чого можна зробити висновок, що з більшості задач, вибраних випадково зі шкільних підручників, найбільше завдань, які стосуються саме трикутника (чотирнадцяте місце теж свідчить про трикутник, а саме на його різновид). Це очікувано, оскільки трикутник – це найменший за кількістю кутів багатокутник, а також трикутник вивчають більш поглиблено ніж інші фігури, через те, що саме трикутник лежить в основі тригонометрії, де вивчають взаємозв'язки між сторонами й кутами трикутників. Також цікаво, що на п'ятому місці слово «знайти», що доводить те, що шукане в завданні маркується цим словом. Одинадцяте місце, що зайняло «abc», вказує на іменування трикутника, тобто «АВС» та рідше кут з вершиною «В». Зазвичай, в задачах вказано назву фігури, оскільки це дозволяє легше описати умову, як-от з якого кута проведено відрізок.

Різновиди геометричних задач іншими мовами

Для подальшого створення універсального алгоритму опрацювання тексту, варто розглянути подання задач з планіметрії іншими мовами. Це також дасть змогу програмі працювати навіть змінюючи природну мову на іншу.

Розглянемо для початку східнослов'янські мови, оскільки українська мова входить до їхнього складу. До східнослов'янських мов ще входять білоруська та російська мови. Почнемо з білоруської. Варто зазначити, що оригінального підручника, написаного білоруською мовою не вдалось знайти. Знайдені два підручники, що використані для аналізу, є перекладом підручників з російської мови.

Оглянемо одну стандартну планіметричну задачу білоруською мовою з підручника з геометрії за 7 клас. Дано задачу: «У раўнабедраным трохвугольніку адна старана роўна 5 см, другая – 10 см. Знайдзіце перыметр трохвугольніка.» [4]. Можна помітити, що задача має таку ж структуру, як і задачі українською мовою. Оскільки, спершу йде опис того, що дано, а потім, у наступному реченні, зі словом «Знайдзіце», йде пояснення того, що потрібно знайти. Більшість задач подають опис одним складним реченням та одним простим реченням іде пояснення шуканого.

Отже, всі подібності, що існують між українською та білоруською мовами дають змогу у майбутньому переформатувати й використовувати створену програму з білоруською природною мовою.

Російська мова, входячи спільно з українською та білоруською до східнослов'янської підгрупи слов'янських груп мов, має багато однакових ознак, як граматичних, так і пунктуаційних, через що мо-

жна припустити можливість для використання створеного алгоритму роботи з текстом. Розглянемо типову планіметричну задачу, взяту з підручника з геометрії за 8 клас. Задача: «В равнобедренном треугольнике ABC с основанием AC проведена биссектриса AD. Найдите углы этого треугольника, если ∠ADB = 110°.» [5]. Очевидно, що структура тексту задачі подібна до тексту українських задач. Ідентично, перше речення пояснює умову задачі, друге ж речення описує, що потрібно знайти. Можна зробити висновок, що для алгоритму програми не буде важно перейти на російську мову, змінивши лише лексику.

Для повноти дослідження, варто також оглянути й задачі англійською мовою. Англійська є частиною германської групи, що входить в індоєвропейську сім'ю мов. Англійська та українська побудова тексту задач значно відрізняється, що є очевидним. Все ж розглянемо кілька задач для розуміння побудови, що хоч трохи подібні до структури тексту задач українською мовою. «Triangle ABC has side lengths of AB = 10, BC = 24, and AC = 26. Find the three angles of the triangle.». Такий тип задачі не розповсюджений, хоча й досить наближений до українського варіанту, оскільки є два речення, де в першому вказано умову задачі, а в другому те, що потрібно знайти, яке починається словом «Find» (можна перекласти як «Знайдіть»). Варто зазначити, що все ж більшість задач англійською мовою використовують різний опис для пояснення шуканого. Знайдено такі варіанти у книжці для тих, хто вчиться у коледжі: «find the measures of ∠B and ∠C», «How long is each leg?», «find the lengths of the two legs» [6]. Вищенаведені приклади опису шуканого цілком можуть бути вирішені в розробленому алгоритмі програми цієї роботи.

Аналіз готових рішень для розв'язку поставленої задачі

Розглядаючи проблему створення системи для розв'язування задач з геометрії, критичним аспектом у вирішенні залишається першочергово вибір готового рішення, яке змогло б задовольнити всім потребам в обробці природної української мови. Наразі список таких рішень досить малий, порівняно, до прикладу, з англійською мовою.

Це пояснюється й тим, що робота з NLP часто залежить від спеціального текстового корпусу. Текстовий корпус – це структурована та ретельно підібрана колекція текстів певною мовою. Найбільш вагомими й інформативними корпусами вважаються розмічені, оскільки вони несуть у собі морфологічну прописану до слів інформацію, як рід, число, відмінок та інше. Очевидно, що таких корпусів досить мало для будь-якої мови, тим більше для української, враховуючи те, що розмічення зазвичай відбувається в ручну командою людей-науковців.

Далі розглянемо наявні програмні рішення, що працюють з обробкою саме української живої мови.

Великий електронний словник української мови (ВЕСУМ) – це електронний зведений словник, що містить слова української мови з парадигмами відмінювання. Також, окрім граматичної інформації, словник пропонує заміни слів-покручів, надає розрізнення омонімів з відмінними парадигмами, позначки для рідковживаних слів тощо [7].

Морфологічний аналізатор та генератор для української та російської мов Pymorphy2. Аналізатор може переводити слово до нормальної форми, тобто надавати лему слова, переводити слово до потрібної форми та надавати граматичну інформацію про слово. Підтримка української мови у цьому аналізаторі є не основною, а експериментальною [8]. Можливості Pymorphy2 досить обмежені функціонально, що не є достатнім для цієї роботи. Цей створений аналізатор базується на словнику ВЕСУМ.

Розглянемо модель UDPipe, що навчена на золотому стандарті. «Золотий морфосинтаксовий стандарт» – це текстовий корпус, спеціально розроблений для універсальних залежностей, що розмічено повністю вручну у кілька шарів [9]. УЗ (UD) скорочення від «універсальні залежності» (Universal Dependencies). Це міжнародний проект, випущений у 2014 році спеціально для того, щоб описати синтак-

сичні зв'язки у природних мовах однією спільною метамовою, спільним набором понять. Основним поняттям синтаксичної теорії, на якій базується проект, є залежність, яка прописується для кожного слова (і не тільки) у реченні. Залежність – це зв’язок між двома словами у реченні, де одне з них є підрядним (залежник), а друге (голова) – керує залежником. Цю залежність можна проілюструвати графічно, поєднавши голову та залежник за допомогою стрілки, яка йде з голови до залежника. UDPipe – це здатний до навчання пайплайн для токенізації, маркування, лематизації та парсингу залежностей CoNLL-U файлів [10]. CoNLL-U формат – це перевірена та надійна версія формату CoNLL-X, анотації в якому кодуються у простий текстовий файл. Так, в їхні можливості входить: повернення леми слова; визначення частини мови; морфологічний розбір слова; номер до голови слова, що буде або номером голови, або ж нулем; зв’язок у реченні, який пов’язує слово з головою (якщо слово є головою, то його іменовано коренем у реченні).

Зрозуміло, що найбільше переваг і можливостей присутні в моделі UDPipe, що працює з розміченим корпусом. З боку швидкості, зручності й якості роботи він є найкращим інструментом. Звісно, у ньому є свої недоліки, як некоректне тегування чи розподіл на слова або речення, проте наразі це найоптимальніше рішення для роботи з обробкою української. Тому вирішено використовувати саме цей засіб.

Побудова класів та методів для опису планіметрії

Обираючи мову програмування, як інструмент для створення застосунку, вибір зроблений на користь Python. Оскільки це мова загального призначення, швидко набирає популярність останніми роками, одна з найвикористовуваніших мов для машинного навчання, існує багато готових математичних бібліотек та пакетів. А також через те, що UDPipe підтримує Python.

Визначивши у другому розділі роботи, що трикутник найчастіше зустрічається в задачах з планіметрії, вирішено приділити увагу саме цій фігурі.

Доцільно створити ієрархію класів, де головним буде клас «Багатокутник» («Polygon»), від якого унаслідуватимуться всі інші багатокутні опуклі фігури. Хоча зосередженість цієї роботи є на трикутнику, але опис вищого класу дасть змогу у майбутньому з легкістю додавати нові фігури, не змінюючи та не перероблюючи створену архітектуру. У цьому класі описано функцію для знаходження периметра, додаючи в циклі значення сторін багатокутника. Для подальшого створення класів, ще варто описати функції, що повертатимуть кількість відомих значень: так, для сторін, функція повертатиме значення, яке вказує на кількість відомих сторін, а для кутів, функція повертатиме значення, яке вказує на кількість відомих кутів. Відомі кути чи сторони, це ті, що вже вказані в умові чи знайдені у процесі розв’язання задачі. Це знадобиться у випадку, коли, наприклад, у трикутнику відомі два його кути, тоді знаючи цю інформацію, можна буде з легкістю знайти третій невідомий кут.

Клас «Трикутник» («Triangle») наслідуватиме клас «Багатокутник». Спираючись на таксономію онтології планіметрії, різновиди трикутника описуватимуться в окремих призначених класах, проте деякі властивості, притаманні для всіх трикутників, можливо запрограмувати й у цьому класі. Наприклад, властивість — змінну, що зберігатиме суму градусів кутів трикутника, що дорівнює 180. Існують задачі, в яких не вказано назви трикутника, тоді, для подальшого розв’язання задачі, варто самостійно надати фігурі назву, наприклад «ABC».

Далі, розглянемо функції, що створені для класу «Трикутник». Функція обрахунку площі трикутника, в основі якої закладена формула Герона, що дає змогу визначити площу трикутника за довжинами його сторін. Функція обрахунку сторони, яка шукає довжину сторони: за трьома кутами та однією відомою стороною; за периметром та двома відомими сторонами; за відомою площею та висо-

тою, що проведена до цієї сторони; за властивістю медіани, що проведена до цієї сторони. Функція обрахунку кутів, що шукає значення кута: за трьома сторонами, через арккосинус; за двома відомими кутами, через їхнє віднімання від суми кутів трикутника; за допомогою відношення сторони до синуса протилежного кута; за властивістю бісектриси, що проведена з цього кута; за властивістю медіани, що проведена до сторони.

Коли в задачі є висота, медіана чи бісектриса, тобто відрізки, що ділять основний трикутник, варто розуміти, що подальше розв'язання задачі має відбуватися з оглядом на утворені фігури, як на окремі трикутники. Очевидно, що для розв'язування таких типів задач потрібно весь час переходити з однієї фігури до іншої, а в деяких випадках це може бути потрібно й кілька разів. Отже, виникає проблема, яка полягає у тому, щоб синхронізувати отримувану інформацію з одного трикутника в інший і навпаки, поки не буде знайдено шукане. Для цього створено функцію у класі «Трикутник», що відповідає за синхронізацію даних між утвореними меншими трикутниками. Задача цієї функції у тому, щоб спершу присвоїти утвореним фігурам уже відомі значення та передавати нові значення під час знаходження кутів чи сторін у малих трикутниках основному трикутнику. Виклик функції здійснюється під час того, як проходить встановлення медіани, бісектриси чи висоти. Синхронізація відбувається кілька разів за все розв'язування задачі.

Встановлення медіани, бісектриси та висоти відбувається трьома окремими функціями, в яких є свої унікальні особливості, зважаючи на властивості цих відрізків у трикутнику. Виклик таких функцій здійснюється тоді, коли встановлена їхня наявність після опрацювання тексту.

Клас «Прямокутний Трикутник» наслідує вищий клас «Трикутник». У цьому класі створена змінна, що вказує на різновид трикутника, тобто зберігає значення «прямокутний». Також у класі прописано назви сторін трикутника, тобто перша сторона має назву «гіпотенуза», а інші дві – «катет». Одному з кутів трикутника присвоєно значення «90», цьому ж куту присвоєно назву «прямий». Іншим двом кутам присвоєно назву «гострий». Додатково додано функцію, що обчислює площу трикутника. Обрахунок здійснюється за формулою половини добутку катетів трикутника.

Клас «Рівнобедрений Трикутник» наслідує вищий клас «Трикутник». У класі створено змінну, що вказує на різновид трикутника, тобто зберігає значення «рівнобедрений». Також в цьому класі прописано назви сторін трикутника, тобто перша та третя сторона має назву «бічний», а друга сторона має назву «основа».

Клас «Рівносторонній Трикутник» наслідує вищий клас «Трикутник». У класі додано змінну, що вказує на різновид трикутника, тобто зберігає значення «рівносторонній». Усім кутам трикутника присвоєно значення «60».

Клас «Знайти» поєднує у собі кілька функцій. Функції для виводу інформації, в які входять: вивід кута за його іменем; пошук кута, де відбувається пошук шуканого кута за його іменем з усіх наявних кутів; вивід всіх кутів фігури; вивід сторони за його іменем; вивід сторони за його назвою; пошук сторони, де відбувається пошук шуканої сторони за іменем з усіх наявних сторін; перевірка числа шуканого (однина чи множина).

Проте, основний алгоритм роботи відбувається в конструкторі об'єкта класу. Він пов'язаний з синтаксовим аналізатором та буде розглянуто в наступному розділі.

Використання морфосинтаксового аналізатора в задачах планіметрії

Перед опрацюванням «сирого» тексту задачі потрібно спершу підготувати цей текст. Оскільки аналізатор, що використано в цій роботі, не здатен боротися з багатьма помилками. Тому першим кроком має бути очищення тексту від цих помилок. Помилки можуть бути синтаксичні, орфографічні, лексичні, тавтологічні тощо.

Наступним кроком є заміна математичних символів на прописний аналог. Аналізатор навчено на розмічених текстах, переважно художнього стилю. Це спричиняє не завжди коректне опрацювання текстів математичних задач. Тому варто автоматично замінювати такі символи: «°» на « градусів», «=» на «дорівнює», «∠» на «кут », «||» на «паралельна» та інші. На прикладі вибраної задачі отримано такий проміжний результат: «У трикутнику ABC відомо, що кут A дорівнює 30 градусів, кут B дорівнює 45 градусів, CM — висота, AC дорівнює 10 см. Знайдіть відрізок BM.»

Останнім кроком є заміна різноманітного формулювання для шуканого одним уніфікованим аналогом, оскільки визначення шуканого спиратиметься надалі на це слово. Відбувається автоматична заміна таких варіантів, як: «Обчисліть», «Визначте» та «Чому дорівнюють» на «Знайдіть». У випадку вибраної демонстраційної задачі такої заміни не потрібно.

Як можна помітити, майже всі заміни не є лемою цього ж слова. Це зроблено для того, щоб подальший аналіз тексту коректніше проставив залежності між словами, а на це впливає форма слова.

Налаштування аналізатора

Для використання UDPipe можна скористатись уже готовим прикладним програмним інтерфейсом. Формат тексту повинен бути UTF-8, а розмір одного запиту не повинен перевищувати одного мегабайта. Обробка тексту відбувається через файл.

Одна з проблем, яка виникає, коли в задачі задані числові значення з комою (не цілі числа), є досить суттєвою. Аналізатор створений так, щоб числа було поділено. Бо, саме синтаксично, це є окремими токенами. Оскільки, до прикладу, рахунок матчу 10:0 та позначення часу 15:30 (ще можуть записувати як 15.30) є різними токенами. Таке розбиття з одного числа, створить три окремі токени, де першим токеном є ціла частина, другим є знак коми, а третім дробовою частиною числа. Це призведе до некоректного створення залежностей між словами, що унеможливить подальшу обробку тексту. Для виправлення цього створено попередню перевірку ще не проаналізованого тексту на наявність чисел з комою. За допомогою можливостей Python та розробленого регулярного виразу відбувається пошук усіх чисел з комою, подальший їхній запис у масив, а потім заміна в тексті цих чисел на «00». Після виконання своєї роботи аналізатором, числа вписуються назад на своє місце з масиву. Оскільки аналізатор розпізнає число «00» як один числовий токен, то подальше повернення справжнього числа з масиву аж ніяк не впливає на результат, а проведений аналіз відповідає очікуваному.

Варто додати, що майже всі задачі містять у тексті або знак «=», або ж слово «дорівнює» (можлива й інша форма цього слова). Проте знак під час обробки буде замінено на слово, а надалі після лематизації отримано всюди одне уніфіковане слово «дорівнювати». Оскільки, більшість задач подають дані в умові за допомогою цього слова, доречно на основі нього створити пошук, який зможе відшукати в тексті всі значення.

Пошук базується на двох умовах. Спершу, перевірка чи токен перед словом «дорівнювати» є кутом, тобто одна велика або три великі латинські букви. Здійснюється за допомогою регулярного виразу: «^[A-Z]{1}$|^[A-Z]{3}$». Якщо умова виконана, то значення кута з його назвою заносяться у відповідні змінні. Наступна умова перевіряє чи перед словом «дорівнювати» є сторона і здійснюється за допомогою регулярного виразу: «^[A-Z]{2}$».

Додано й третє розгалуження, в разі, якщо перші дві умови не виконуються. Часто трапляється, що дані подано не до конкретної змінної, або ж в задачі не надано фігурі назви чи вказані дані описують не кут чи сторону. Прикладом можуть бути такі варіанти: «основа рівнобедреного трикутника дорівнює 12», «площа квадрата дорівнює 8», «більша сторона прямокутного трикутника дорівнює 5» тощо. Тоді пошук відбувається так: іде пошук слова вліво, доки не знайдеться слово, яке аналізатором визначено як підмет, це й буде змінною. Також ще відбувається пошук

означення до цього підмета, який зазвичай може стояти зразу перед ним.

Цей пошук здійснюється за допомогою обробленого тексту після синтаксового аналізатора, коли слова поділені на токени, проведена лематизація і визначені зв'язки між ними. Підмет у стовпці, що описує зв'язок, позначено як «nsubj», означення через стовпець частин мови як «ADJ».

Коли під час пошуку знайдено підмет, далі йде порівняння цього токена з переліком визначених слів. Якщо слово збігається із заготовленим можливим підметом, значення присвоюється відповідній змінній.

Пошук в умові задачі того, що потрібно знайти, відбувається в конструкторі об'єкта класу «Find». Обробка здійснюється за допомогою аналізатора. Орієнтиром, на який спирається пошук, є слово «знайти».

Алгоритм роботи такий: відбувається проходження всього тексту після слова «знайти», оминаючи токени, у яких визначено частину мови, як «PUNCT» (знаки пунктуації), «CCONJ» (різновиди сполучників), «VERB» (дієслова), «NUM» (числа), «SCONJ» (різновиди сполучників), «ADP» (прийменники), «DET» (детермінант), «ADV» (прислівниково-числівникові займенники). Це потрібно для відкидання інформації, яка не несе в собі значущі, важливі для обробки, значення.

Проходження припиняється, якщо на шляху трапляються такі токени, як: «.», «?», «якщо».

Після відбирання токенів, потрібно порівняти їх із тими, які заздалегідь визначені як важливі. Створено чотири масиви, які поділені за своїм смислом: масив різновиду шуканого (прикметники, як «найбільший»); масив, що означає сторону (іменники, як «відрізок» чи «відстань»); масив, що означає кут. Ще один масив, де зберігаються назви фігур. Також, може бути знайдено ім'я шуканого за допомогою регулярного виразу: «^[A-Z]{1,3}$». Іще перевірка, якщо токен дорівнює словам «площа» чи «периметр».

Насамперед потрібно для обробки задачі пройтися по лемах токенів, щоб знайти назву фігури. Для цієї перевірки створено масив, з назвами усіх багатокутних плоских фігур, а також масив з варіантами написання їхнього виду. Після знаходження токена назви фігури, перед ним завжди стоятиме токен з його видом.

Після збігу назви фігури з токеном, створюється екземпляр класу відповідної фігури чи класу відповідного виду цієї фігури.

Механізм розв'язування планіметричної задачі на прикладі

Увівши задачу: «У трикутнику ABC відомо, що ∠A = 30°, ∠B = 45°, CM – висота, AC = 10 см. Знайдіть відрізок BM.» у програму, спершу відбудеться заміна символів на їхні прописні слова-аналоги. Після цього кроку відбувається перевірка програмою на числа з комою. Наступний крок є одним з найосновніших, бо саме тут відбувається перевірка тексту задачі аналізатором. Саме тут текст аналізується, поділяється на токени, відбувається лематизація, проставляння залежностей між словами. Далі робота виконується лиш з цим підготовленим текстом.

Спершу визначається фігура, її вид (може бути відсутнє), її іменування (може бути відсутнє). Визначивши фігуру, відбувається виклик відповідного класу, що їй належить. Варто зазначити, якщо в задачі вказано вид фігури, до прикладу «прямокутний трикутник», відповідний і більш конкретний клас буде викликано. Отже, в задачі з прикладу, буде визначено фігуру «трикутник» та іменування фігури «ABC».

Далі відбувається пошук в тексті слів, таких як «бісектриса», «медіана» і «висота». Знайшовши таке слово, викликається відповідний метод в основному класі «Triangle». У прикладі є висота, тому цьому відрізку присвоюється ім'я «CM» з умови. У методі визначається: кут, з якого проведено висоту; сторони, що утворюють кут; до якої сторони проведена висота; відрізки, на які ділить висота, трикутники на які ділить висота.

За допомогою створених методів, які опрацьовують дані навколо слова «до-

рівнювати», у задачі визначено спершу значення кутів: куту «А» присвоєно значення «30», куту «В» присвоєно значення «45»; потім визначено значення відрізків: стороні «АС» присвоєно значення «10».

Тепер відбувається виклик класу «Знайти». Результатом пошуку шуканого є два масиви, де перший масив [«side», «name»] вказує на проставлені мітки знайденим токенам, а другий масив показує самі ж токени [«відрізок», «ВМ»]. Порядок елементів у першому та другому масивах відповідає один одному.

За допомогою методу для знаходження кутів, визначається невідомий кут «С».

Враховуючи, що в задачі є висота, яка ділить основний трикутник, утворені менші трикутники досліджуються окремо й розглядаються як повноцінні. Зважаючи на властивості висоти, менші трикутники належать до класу «Прямокутний Трикутник». Далі у трикутнику «САМ» знаходиться довжина відрізка «СМ» за допомогою відомих значень сторони та синуса кута. Відбувається виклик методу, що відповідальний за синхронізацію нових отриманих даних з трикутників. І в результаті у трикутнику «ВСМ» за допомогою значень сторони та тангенса кута отримано шуканий відрізок «ВМ».

Висновки

Отже, в роботі проведено аналіз наявних на сьогодні можливостей з обробки живої української мови.

Проведено аналіз задач з геометрії, який показує закономірності у формулюванні й структурі текстового подання задач.

Створена архітектура є гнучкою й це вказує на можливість додавання як опису нових фігур та їхніх властивостей, так і на створення додаткової логіки в застосунку.

Варто вказати на можливість переформатування програми для використання її з іншою природною мовою, до прикладу англійською, білоруською чи російською.

Література

1. Reynar J. C. A Maximum Entropy Approach to Identifying Sentence Boundaries. Reynar Jeffrey – Philadelphia, Pennsylvania, USA.
2. Sarawagi S. Information Extraction / Sarawagi Sunita – Mumbai. 2008.
3. ЗНО з математики: особливості тесту 2020 року [ЗНО з математики: особливості тесту 2020 року [Електронний ресурс]. 2019. Режим доступу до ресурсу: https://osvita.ua/test/training/5017/.
4. Казакоў В.У. ГЕАМЕТРЫЯ. Мінск: Народная асвета. 2017.
5. Геометрия. 7–9 классы [Л. С. Атанасян, В. Ф. Бутузов, С. Б. Кадомцев та ін.]. Москва: Просвещение. 2010. 384 с. (20).
6. Alexander D.C., Koeberlein G.M. Elementary Geometry for College Students. Belmont. (5).
7. Великий електронний словник української мови (ВЕСУМ) [Електронний ресурс]. 2017. Режим доступу до ресурсу: https://github.com/brown-uk/dict_uk/blob/master/doc/announcement.md.
8. Korobov M. Morphological Analyzer and Generator for Russian and Ukrainian Languages. Ekaterinburg. 2015.
9. Золотий морфосинтаксовий стандарт [Електронний ресурс]. Режим доступу до ресурсу: https://mova.institute/золотий_стандарт.
10. Straka M. Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe / Straka Milan Vancouver, 2017.

References

1. Reynar J. C. A Maximum Entropy Approach to Identifying Sentence Boundaries. Reynar Jeffrey – Philadelphia, Pennsylvania, USA.
2. Sarawagi S. Information Extraction / Sarawagi Sunita – Mumbai. 2008.
3. ZNO z matematyky: osoblyvosti testu 2020 roku. (2019). Retrieved June 12, 2020, from https://osvita.ua/test/training/5017/
4. Kazakow V.U. Geometryja. Minsk: Narodnaja asveta. 2017.
5. Geometriya: 7–9-e klassy: uchebnik dlya obshcheobrazovatelnykh uchrezhdeniy. 2010.

6. Alexander D.C., Koeberlein G.M. Elementary Geometry for College Students. Belmont. (5).
7. Velykyi elektronnyi slovnyk ukrainskoi movy (VESUM). (2017, November 30). GitHub Retrieved June 12, 2020, from https://github.com/brown-uk/dict_uk/blob/master/doc/announcement.md
8. Korobov M. Morphological analyzer and generator for Russian and Ukrainian languages. Communications in Computer and Information Science. 2015. P. 320–332. https://doi.org/10.1007/978-3-319-26123-2_31
9. Zolotyi morfosyntaksovyi standart. Laboratoriia ukrainskoi. Retrieved June 12. 2020, from https://mova.institute/золотий_стандарт.
10. Straka M. Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe / Straka Milan Vancouver. 2017. https://doi.org/10.18653/v1/k17-3009

Одержано 23.10.2020

Про авторів:

Жежерун Олександр Петрович,
кандидат фізико-математичних наук, старший науковий співробітник.
Кількість наукових публікацій в українських виданнях – 80.
Індекс Гірша – 4.
http://orcid.org/ 0000-0002-4034-6730,

Смиш Олег Русланович,
аспірант.
https://orcid.org/0000-0002-8074-9745.

Місце роботи авторів:

Національний університет «Києво-Могилянська академія», завідувач кафедри мультимедійних систем.
04655, Київ,
вул. Г. Сковороди 2.
Тел.: (044) 425-77-53
E-mail: zhezherun@ukma.edu.ua

Інститут програмних систем НАН України,
03187, Київ,
проспект Академіка Глушкова, 40.
Тел.: (044) 526-33-19
E-mail: o.smysh@ukma.edu.ua