Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мережних технологій факультету інформатики

СИСТЕМА ДЛЯ РОЗВ'ЯЗУВАННЯ ЗАДАЧ З ГЕОМЕТРІЇ

Текстова частина
магістерської роботи
за спеціальністю “Інженерія програмного забезпечення” 121

Керівник магістерської роботи
к.ф.-м.н., доц. _____
(прізвище та ініціали)
_____
(підпис)
“____” _____ 2020 р.
Виконав студент _____
_____
(прізвище та ініціали)
“____” _____ 2020 р.

Київ 2020

Зміст

Анотація

У роботі описано створення прикладу системи для розв’язування задач з геометрії за допомогою сучасних можливостей обробки природної української мови, на основі розробленого алгоритму опрацювання тексту, що базується на аналізі текстів геометричних задач та аналізі доступних засобів обробки живої мови.

Кінцевий програмний продукт дає змогу вирішувати прості завдання з планіметрії.

Вступ

Обробка природної мови є на сьогодні чи не найпопулярнішою з галузей математичної лінгвістики. Актуальність у створенні застосунків, які вміють взаємодіяти з текстом поданим природною мовою, набувають все більшого попиту.

Проте готового рішення, яке могло б допомогти у розв’язуванні математичних задач, задане природною українською мовою, наразі немає.

Мета дослідження полягає у тому, щоб проаналізувати наявні рішення обробки текстів українською мовою й розглянути можливість використання такого рішення для створення системи з розв’язування задач.

Завдання роботи ґрунтується на створенні алгоритму, що демонструє програмний застосунок, який дає змогу розв’язувати задачі, що подаються природною українською мовою.

Отримані результати дають змогу ознайомитися з сучасними аналізаторами текстів українською та демонструють їхні можливості. Створення гнучкого алгоритму опрацювання тесту живої мови для подальшого доповнення чи зміни мови без втрати працездатності.

Робота містить п’ять основних розділів.

Перший розділ дає змогу ознайомитись з обробкою природних мов, їхніми можливостями.

Другий розділ присвячено аналізу специфіки подання математичних задач природною українською мовою. Розглянуто також і інші мови.

Третій розділ освітлює наявні на сьогодні рішення для української мови. Показано їхні можливості та пояснено визначення найкращого рішення для цієї роботи.

Четвертий розділ знайомить з написанням класів та їхніх методів, що формально описують планіметрію, а саме багатокутні плоскі фігури та їхні властивості.

П’ятий розділ є завершальним та описує розроблений алгоритм, що базується на опрацюванні отриманих результатів роботи аналізатора тексту.

1 Обробка природної мови

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 ж 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 яку не годиться для взаємодії з комп'ютером, головним чином через свої синтаксичні, смислові, відмінкові та референційні неоднозначності.

NLP (Natural language processing) – галузь в лінгвістиці, комп'ютерних науках, інформаційній інженерії та штучному інтелекті, яка спрямована на комп'ютерний аналіз та обробку живої (людської) мови. Загалом, метою NLP є надати можливість уніфікувати природну мову для розуміння її комп'ютером. Звісно, наразі машини не здатні розуміти українську мови так само, як розуміють її люди, проте вже зараз вони мають досить великі можливості. Але, варто зазначити, що найбільші можливості все ж доступні лише для англійської мови, через свій статус засобу міжнародного спілкування. У цій дипломній роботі розглядається обробка живої української мови. Проблема полягає в тому, що наразі існує вичерпний список створених можливостей обробки української мови, що ускладнює роботу.

Далі розглянемо деякі ланки, що передбачає NLP, які вже певною мірою присутні в сучасних аналізаторах тексту та які використовуватимуться в цій роботі.

Сегментація тексту – це поділ тексту на певні значущі одиниці (токени), такі як слова, речення чи абзаци. Такий поділ не є легкою задачею для розв’язання, оскільки не у всіх мовах, як в українській, є спеціальні маркери, які б вказували на закінчення чи початок слова. 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. Дещо складніша ситуація з поділом на речення. Хоч і речення в українській мові заведено 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 (Part-of-speech tagging (POS tagging або PoS tagging або POST)) – це встановлення кожному слову з тексту відповідний теґ, який вказує, яка це частина мови зважаючи на визначення самого слова та на контекст у словосполученні, реченні чи абзаці. Розмічування частин мови ускладнюється тим, що в природній українській мові одне й те ж слово, проте у різних реченнях, може відповідати різним частинам мови, а отже й мати інше значення.

Стемінг (Stemming) – це виокремлення основи слова, що залишатиметься незмінною для ряду форм незалежно від відмінка та наявних суфіксів. При цьому отримана форма слова не обов’язково відповідатиме його кореню [2]. Зазвичай, на стемінг слова не впливає контекст чи значення самого слова.

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

Розпізнавання іменованих сутностей (Named entity (N.E.) recognition) – це форма вилучення певної інформації з тексту, де метою є встановлення класу кожному слову, як-от імена людей, географічні назви, грошові одиниці, час тощо [3].

Видобування інформації – це процес вилучення з неструктурованого або малоструктурованого тексту структурованої інформації, такої як сутності, зв’язки між ними, атрибути [4]. Як галузь в NLP, видобування інформації набуває все більшої зацікавленості. Оскільки гостро постає необхідність у структуруванні інформації. Варто також додати, що під кожну окрему задачу, надбудовується додаткова логіка на готовий інструмент роботи з текстом, оскільки охопити всі аспекти різних задач наразі не є можливим.

2 Аналіз задач з геометрії

2.1 Типи 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 текстовому виді (можливо обійтись без зображення фігур та без тригонометричних рівнянь), також вивчення геометрії розпочинають саме з планіметрії, про що свідчить 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. До прикладу, розглянемо задачі з математики з зовнішнього незалежного оцінювання (ЗНО), яке проходять абітурієнти для вступу до закладів вищої освіти.

Тест ЗНО з математики в останні роки сформований із завдань, що можна поділити на чотири типи: завдання, де потрібно обрати одну правильну відповідь з п’яти варіантів; завдання на встановлення відповідностей, де потрібно встановити пари чотирьом умовам з чотирма відповідниками (плюс є один зайвий відповідник); завдання відкритого типу, де потрібно надати коротку відповідь; завдання відкритого типу, де потрібно надати розгорнуту відповідь. Незалежно від типу задачі, деякі з них подають рисунок як умову або ж варіантом відповіді має бути рисунок [5].

Очевидно, що такі типи задач, які є в ЗНО не підхожі для опрацювання у цій роботі й не можуть використовуватись для комплексного аналізу, оскільки мають різну структуру та різні варіанти подання відповідей до неї.

Наразі планіметричні задачі учні починають вивчати починаючи з 7 класу школи. Зважаючи на це, доцільно буде ознайомитися з підручниками школярів за 7 клас та з’ясувати структуру задач, які там використано.

Взявши до прикладу український підручник з геометрії за 7 клас для загальноосвітніх шкіл автора А.Г.Мерзляк [6], можна простежити такі задачі, що потребують графічної відповіді, що використовують в умові рисунки, де розв’язком є доведення твердження та задачі з розгорнутою відповіддю. Аналогічні типи задач простежуються й в інших шкільних підручниках з геометрії (мається на увазі планіметричні задачі).

2.2 Задачі відкритого типу з короткою відповіддю

Для дослідження у цій роботі використовуватимуться 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. Задачі взято зі шкільних підручників за 7 та 8 класи.

Для наочної демонстрації приведено 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 самого 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, спершу потрібно привести 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 A дорівнювати 30 градус, кут B дорівнювати 45 градус, CM — висота, AC дорівнювати 10 см. знайти відрізок BM.”. Очевидно, що метод зняв відмінкові форми зі слів, що допомагає при подальшому кількісному аналізі задач.

Провівши лематизацію і заміну символів у вибраних 76 задачах, взятих з підручників, отримано такий результат, що наведено в таблиці 2.1.

№	Слово	Повторюваність
1	дорівнювати	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

Таблиця 2.1 – Фрагмент результату кількісного аналізу частоти повторень слів у 76 задачах з планіметрії

З таблиці 2.1 видно, що на першому місці за повторюваністю є слово “дорівнювати”, потім одиниця виміру відрізків “см”. На третьому ж місці розташоване слово “трикутник”, з чого можна зробити висновок, що з більшості задач, вибраних випадковим чином 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 “ABC” та рідше кут з вершиною “B”. Зазвичай, в задачах вказано назву фігури, оскільки це дозволяє легше описати умову, як-от з якого кута проведено відрізок.

2.3 Задачі з планіметрії на інших мовах

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.

2.3.1 Задачі з планіметрії білоруською мовою

Варто одразу відзначити, 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.” [7]. 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.

2.3.2 Задачі з планіметрії російською 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 можна 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°.” [8].

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.

2.3.3 Задачі з планіметрії англійською мовою

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 декілька 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 і 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” [9]. Приведені вище приклади опису шуканого цілком можуть бути вирішені в розробленому алгоритмі програми цієї роботи.

3 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, у порівнянні 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 в собі морфологічну прописану до слів інформацію, як-от рід, число, відмінок та інше. Очевидно, що таких корпусів досить мало для будь-якої мови, тим паче 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 всеохопний 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 [10].

Морфологічний аналізатор та генератор для української та російської мов Pymorphy2. Аналізатор може приводити слово до нормальної форми, тобто надавати лему слова, приводити слово до потрібної форми та надавати граматичну інформацію про слово. Підтримка української мови у цьому аналізаторі є не основною, а експериментальною [11]. 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 декілька шарів [12]. УЗ (UD) скорочення від “універсальні залежності” (Universal Dependencies). Це міжнародний проєкт, випущений у 2014 році спеціально для того, щоб описати синтаксичні зв'язки у природних мовах однією спільною метамовою, спільним набором понять. Основним поняттям синтаксичної теорії, на якій базується проєкт, 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 [13]. CoNLL-U формат – це перевірена та надійна версія формату CoNLL-X, анотації в якому кодуються в простий текстовий файл.

Також наявний проєкт стенфордських залежностей (СЗ) (Stanford Dependencies), який створено у 2005 році, що також навчений на золотому стандарті. В основі УЗ закладено СЗ. Ці проєкти досі підтримуються та активно оновлюються, особливо УЗ.

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 і якості роботи він є найкращим інструментом. Звісно, у ньому є свої недоліки, як-от некоректне тегування чи розподіл на слова або речення,

проте наразі це найоптимальніше рішення для роботи з обробкою української. Тому в цій роботі вирішено використовувати саме цей засіб.

4 Побудова класів та методів для опису планіметрії

4.1 Аналіз

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.

Розглянемо на рисунку 4.1 зв’язки між класами й підкласами понять у таксономічній ієрархії [14].

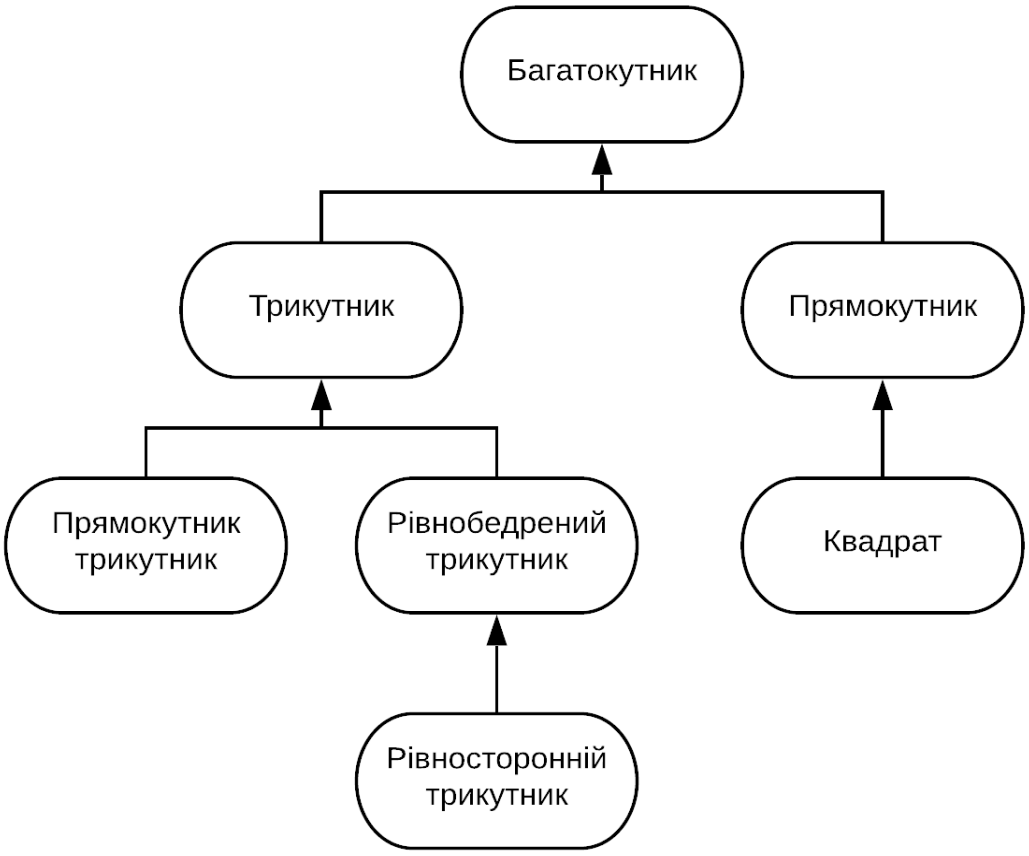

Рисунок 4.2 – Спрощений фрагмент таксономії для онтології «Планіметрія»

4.2 Клас “Багатокутник”

На основі аналізу можна створити ієрархію класів, де головним буде клас “Багатокутник” (“Polygon”), від якого наслідуватимуться 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.

Клас “Polygon” описує в собі такі поняття, як:

- ім’я багатокутника (наприклад, “трикутник”);
- назва багатокутника (наприклад, “ABC”);
- кількість сторін багатокутника (наприклад, “3”).

У цьому класі також можна описати й 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.

4.3 Допоміжні класи для поняття сторін і кутів

Правильним рішенням буде створити класи, що відповідатимуть за кути “Angle” та сторони “Side”, де можна прописати додаткові властивості, які є загальними, в незалежності від фігури.

У класі “Side” є такі властивості, як:

- назва сторони (наприклад, “AB”);
- обернута назва сторони (назва одного й того ж самого відрізка може бути, як “AB”, так і “BA”);
- довжина відрізка (наприклад, “12”);
- однакова сторона (у випадку, коли одна сторона є спільною для двох чи більше фігур).

У класі “Angle” є такі властивості, як:

- назва кута (наприклад, “ABC”);
- обернута назва кута (назва одного й того ж самого кута може бути, як “ABC”, так і “CBA”);
- значення кута (наприклад, “90”);
- однаковий кут (у випадку, коли один кут є спільним для двох чи більше фігур);
- коротка назва кута (наприклад, “A”);
- протилежна сторона до кута (наприклад, у випадку трикутника для визначення до якої сторони проведено певний відрізок).

4.4 Клас “Трикутник”

Клас “Трикутник” (“Triangle”) наслідуватиме клас “Багатокутник”. Спираючись на таксономію онтології “Планіметрія”, 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”.

Важливо зазначити, що присвоєння відбувається за допомогою стрічки літер, які почергово присвоюватимуться вершинам багатокутника, в залежності від їхньої кількості.

Після того, можна з легкістю запрограмувати назви кутів та сторін трикутника.

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 висотою, 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. До прикладу, розглянемо таку задачу: “У трикутнику ABC відомо, що ∠A = 30°, ∠B = 45°, CM — висота, AC = 10 см. Знайдіть відрізок BM.”. У цій задачі дано висоту CM, що автоматично ділить трикутник ABC на два прямокутні трикутники зі спільною стороною CM. У задачі потрібно знайти відрізок BM, що вже вказує на те, що потрібно оглядати утворений прямокутний трикутник CAM окремо.

Очевидно, що для розв’язування задачі потрібно весь час переходити з однієї фігури до іншої, а в деяких випадках це може бути потрібно й пару

разів. Отже, виникає проблема, яка полягає в тому, щоб синхронізувати отримувану інформацію з одного трикутника в інший і навпаки, допоки не буде знайдено шукане.

Для цього створено функцію в класі “Трикутник”, що відповідатиме за синхронізацію даних між утвореними меншими трикутниками. Задача цієї функції в 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 декілька разів за все розв’язування задачі.

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

4.4 Класи “Медіана”, “Бісектриса”, “Висота”

Ці три класи схожі за своєю побудовою. Кожен клас містить ім’я, в якому прописано, що це за відрізок; змінну для назви відповідного відрізка (наприклад “AB”); змінну для зберігання значення довжини відрізка; точку, з якої проведено відрізок (наприклад “A”), протилежну сторону, бо якої опущено відрізок.

Для класу “Бісектриса” додатково введено ще одну змінну, що відповідає за кут, який вона ділить навпіл.

Усі класи містять у собі створену функцію для встановлення значення довжини відрізка. Мається на увазі присвоєння значення відрізка сторонам, утворених двох трикутників.

4.5 Класи різновидів трикутника

4.5.1 Клас “Прямокутний Трикутник”

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.

4.5.2 Клас “Рівнобедрений Трикутник”

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”.

4.5.3 Клас “Рівносторонній Трикутник”

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”.

4.6 Клас “Знайти”

Клас “Знайти” (“Find”) поєднує в собі декілька функцій. Функції для виводу інформації, в які входять:

- функція для виводу кута за його іменем;
- функція пошуку кута, де відбувається пошук шуканого кута за його іменем з усіх наявних кутів;
- функція для виводу всіх кутів фігури;
- функція для виводу сторони за його іменем;
- функція для виводу сторони за його назвою;
- функція пошуку сторони, де відбувається пошук шуканої сторони за іменем з усіх наявних сторін;
- функція для перевірки 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.

5 Використання морфосинтаксового аналізатора в задачах планіметрії

5.1 Попереднє опрацювання тексту

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 повинно 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”, варто звернути увагу, що в задачах символ пишеться зразу після числового значення без відступу, тому заміна відбувається з урахуванням пробілу;
- “=” на “дорівнює”;
- “∠” на “кут ”, варто звернути увагу, що в задачах символ пишеться безпосередньо перед числовим значенням без відступу, тому заміна відбувається з урахуванням пробілу;
- “||” на “паралельна”.

Останнім кроком є заміна різноманітного формулювання для шуканого одним уніфікованим аналогом, оскільки визначення шуканого спиратиметься надалі на це слово. Відбувається автоматична заміна таких варіантів:

- “Обчисліть” на “Знайдіть”;
- “Визначте” на “Знайдіть”;
- “Чому дорівнюють” на “Знайдіть”.

Додані ще мінорні заміни, такі як:

- “завдовжки” на “дорівнює”;
- “см2” на “см”.

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.

5.2 Налаштування аналізатора

Для використання UDPipe можна скористатись уже готовим прикладним програмним інтерфейсом (Application Programming Interface, API), що надає можливість доступу до модуля для розробки програмного забезпечення. Працює зазначене API за допомогою cURL. cURL або Client for URLs – це розроблений проєкт для передачі чи отримання файлів в терміналі, використовуючи URL синтаксис.

Формат тексту повинен бути UTF-8, а розмір одного запиту не повинен перевищувати 1MB.

Для аналізу тексту на Mac чи Linux з файлу, що містить текст для перевірки, використано таку термінальну команду:

curl -F json=false -F data='@input.txt' -F tokenizer= -F tagger= -F parser= https://api.mova.institute/udpipe/process > output.txt,

де *input.txt* – файл з “сирим” текстом;
output.txt – готовий файл з опрацьованим текстом.

Отже, обробка тексту відбувається через файловий режим.

5.3 Числа з комою

Проблема, яка виникає, коли в задачі задані числові значення з комою, тобто не цілі числа, є досить суттєвою. Аналізатор створений таким чином, щоб числа були розбитими. Бо, саме синтаксично, це вважається 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 жодним чином не впливає на результат, а проведений аналіз відповідає очікуваному.

5.4 Задачі зі словом “дорівнює”

Варто зразу зазначити, 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.

Отже, в наскрізному прикладі задачі, що розглянуто в цій роботі, а саме: “У трикутнику ABC відомо, що ∠A = 30°, ∠B = 45°, CM — висота, AC = 10 см. Знайдіть відрізок BM.”, куту A буде присвоєно значення “30”, а куту B значення “45”.

Наступна умова перевіряє чи перед словом “дорівнювати” є сторона, тобто дві великі латинські букви. Здійснюється за допомогою регулярного виразу: “^[A-Z]{2}$”. Відповідно, в прикладі задачі стороні AC буде присвоєно значення “10”.

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. У такому разі пошук відбувається наступним чином: іде пошук слова вліво, доки не знайдеться слово, яке аналізатором визначено як підмет, це й буде змінною. Також з 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. Приклад перевірки такого збігу з наступним присвоєнням значення:

```
elif subject[2] == 'гіпотенуза' and adjective == 0:
        for side in figure.sides:
                if side.name == 'гіпотенуза':
                        figure.set_side(side, found_measures[0])
```

5.5 Задачі зі словом “проведений”

Часто задачі, в умові яких використано бісектрису, медіану чи висоту, потребують додаткового пояснення про їхнє розташування у фігурі (з якої вершини чи до якої сторони їх проведено) для полегшення розуміння. Такі варіанти написання не використовують назви фігури. Тоді, для цього в більшості задач використане слово “проведений”. Це задачі типу: “висота, проведена до основи трикутника,”, “бісектриса, проведена з вершини прямого кута,”, “медіана, проведена до бічної сторони,” тощо. Пояснення такого типу мають два розгалуження: на “проведений з …” або “проведений до …”. Варто передбачити ці нюанси, тобто присвоїти назву цьому відрізку, спираючись на цей опис. Запропоновано кінець відрізка позначити латинською буквою “Z”. Опрацьовані наразі такі варіанти подання:

- “до основа”;
- “до гіпотенуза”;
- “до катет АВ”;
- “до бічний”;

- “до сторона AB”;
- “до AB”;
- “до інший”;
- “з вершина прямий”;
- “з вершина ABC”;
- “з прямий”;
- “з кут ABC”;
- “з ABC”.

5.6 Опрацювання шуканого в тексті

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”. Крапка та знак оклику сигналізує про завершення речення. А після токена “якщо” в реченні зазвичай іде додаткова умова задачі. Це, наприклад, речення такого типу: “Знайдіть висоту AD, якщо площа рівностороннього трикутника дорівнює 36 $см^2$, а сторона AB дорівнює 10 см.”.

Після відбирання токенів, потрібно порівняти їх із тими, які заздалегідь визначені як важливі. Створено чотири масиви, які поділені за своїм смислом. Масив різновиду шуканого: “найбільший”, “найменший”, “менший”, “більший”, “невідомий”. Масив, що означає сторону: “сторона”, “відрізок”, “гіпотенуза”, “катет”, “відстань”, “бічний”, “основа”, “висота”, “медіана”, “бісектриса”, “радіус”, “діаметр”, “лінія”, “стежка”, “січна”, “діагональ”, “радіус”, “діаметр”, “дотична”. Масив, що означає кут: “кут”, “вершина”. Та останній масив, де розписані фігури: “трикутник”, “квадрат”, “прямокутник”, “трапеція”, “ромб”, “паралелограм”. Також, може бути знайдено ім’я шуканого за допомогою регулярного виразу: “^[A-Z]{1,3}$” (пошук великих латинських літер, де одна чи три літери означають кут, а дві літери – сторону). Іще перевірка, якщо токен дорівнює словам “площа” чи “периметр”.

Наступним кроком є зіставлення вищевказаних умов зі знайденими токенами та присвоєння їм визначених міток. Для прикладу, використавши задачі, яка розглядається в цій роботі, 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”, “BM”]. Порядок елементів у першому та другому масивах відповідає один одному.

5.7 Опрацювання інших умов у задачі

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.

Варто також передбачити можливість, коли фігура може належати одночасно декільком класам. Наприклад, трикутник з кутами “90”, “45” і “45” градусів одночасно є рівнобедреним прямокутним трикутником.

Здійснюється окремо пошук таких слів, як “бісектриса”, “медіана”, “висота”. Оскільки часто в умові ці відрізки вказано так: “AD — висота”

Висновки

Отже, в роботі проведено аналіз наявних на сьогодні можливостей з обробки живої української мови. Хоча значущою проблемою у створенні застосунку був саме пошук рішення для опрацювання українського тексту, показано, що ця галузь стрімко розвивається та отримує дедалі більше зацікавлення серед програмістів та вже сьогодні надає можливості, які дають вагомі результати, на базі яких можна створювати власні системи.

Проведено аналіз задач з геометрії, який показує закономірності у формулюванні й структурі текстового подання задач. Також показані особливості не лише задач українською живою мовою, а й білоруською, англійською та російською мовами.

Продемонстровано створення цілої системи, на базі практичних можливостей опрацювання мови, що приймає на вхід текст задачі, а на виході дає її розв’язок.

Програмний застосунок дає змогу надавати відповідь на прості планіметричні задачі, де основною фігурою є трикутник. Проте, 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.

Розглядаючи розроблений алгоритм, варто вказати на можливість переформатування програми для використання її з іншою природною мовою, до прикладу англійською, білоруською чи російською, оскільки алгоритм обробки тексту задачі є універсальним через всесвітньо прийняті умови подання такого типу математичних текстів. Тому можливі подальші перспективи розвитку системи.

Список літератури

1. Reynar J. C. A Maximum Entropy Approach to Identifying Sentence Boundaries / Reynar Jeffrey – Philadelphia, Pennsylvania, USA.

2. Голуб Т. В. Метод стемінгу україномовних текстів для класифікації документів на базі алгоритму Портера : дис. канд. техн. наук / Голуб Т. В. – Запоріжжя, 2017.

3. A Borthwick- Ph. D. Thesis New York University, 1999 - A Maximum Entropy Approach to Named Entity Recognition

4. Sarawagi S. Information Extraction / Sarawagi Sunita – Mumbai, 2007.

5. 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/.

6. Мерзляк А. Г. Геометрія: підручник 7 класу загальноосвітніх навчальних закладів / А. Г. Мерзляк, В. Б. Полонський, М. С. Якір., 2015.

7. Казакоў В. У. ГЕАМЕТРЫЯ / В. У. Казакоў. – Мінск: Народная асвета, 2017.

8. Геометрия. 7-9 классы / [Л. С. Атанасян, В. Ф. Бутузов, С. Б. Кадомцев та ін.]. – Москва: Просвещение, 2010. – 384 с. – (20).

9. Alexander D. C. Elementary Geometry for College Students / D. C. Alexander, G. M. Koeberlein. – Belmont. – (5).

10. 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.

11. Korobov M. Morphological Analyzer and Generator for Russian and Ukrainian Languages / Korobov Mikhail – Ekaterinburg, 2015.

12. золотий морфосинтаксовий стандарт [Електронний ресурс] – Режим доступу до ресурсу: https://mova.institute/золотий_стандарт.

13. Straka M. Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe / Straka Milan – Vancouver, 2017.

14. Хахалин Г. К. ПРЕДМЕТНАЯ ОНТОЛОГИЯ ДЛЯ ПОНИМАНИЯ ТЕКСТОВ ГЕОМЕТРИЧЕСКИХ ЗАДАЧ / Хахалин Г. К. – Москва.

Додаток А

(обов’язковий)

Термінальний вивід розв’язку задачі

Трикутник
Назва - ABC

Сторони:
AB = 13.66 - ----
BC = 7.07 - ----
CA = 10.0 - ----

Кути:
ABC (B) = 45.0 - , сторони: AB, BC; ----
BCA (C) = 105.0 - , сторони: BC, CA; ----
CAB (A) = 30.0 - , сторони: CA, AB; ----

Висота:
CM = 5.0

Трикутники, утворені від висоти:

Назва - BCM
Сторони:
BC = 7.07 - гіпотенуза ----
CM = 5.0 - катет ----
MB = 5.0 - катет ----

Кути:
BCM (C) = 45.0 - гострий, сторони: BC, CM; ----
CMB (M) = 90 - прямий, сторони: CM, MB; ----
MBC (B) = 45.0 – гострий, сторони: MB, BC; ---- ABC

Назва - CAM
Сторони:
CA = 10.0 - гіпотенуза ----
AM = 8.66 - катет ----
MC = 5.0 - катет ----

Кути:
CAM (A) = 30.0 - гострий, сторони: CA, AM; ---- CAB
AMC (M) = 90 - прямий, сторони: AM, MC; ----
MCA (C) = 60.0 - гострий, сторони: MC, CA; ----

['side', 'name']
['відрізок', 'BM']

Відповідь:
MB = 5.0

Додаток Б

(обов’язковий)

Перелік прийнятих скорочень

NLP (Natural language processing) – обробка природної мови

POS tagging (Part-of-speech tagging або POST) – розмічування частин мови

UD (Universal Dependencies) – універсальні залежності (УЗ)

SD (Stanford Dependencies) – стенфордські залежності (СЗ)

Додаток В

(обов’язковий)

Термінальний вивід задачі аналізатором тексту

['#', 'newdoc']
['#', 'newpar']
['#', 'sent_id', '=', '1']
['#', 'text', '=', 'У', 'трикутнику', 'ABC', 'відомо,', 'що', 'кут', 'A', 'дорівнює', '30', 'градусів,', 'кут', 'B', 'дорівнює', '45', 'градусів,', 'CM', '—', 'висота,', 'AC', 'дорівнює', '10', 'см.']
['1', 'У', 'у', 'ADP', '_', 'Case=Loc', '2', 'case', '_', '_']
['2', 'трикутнику', 'трикутник', 'NOUN', '_', 'Animacy=Inan|Case=Loc|Gender=Masc|Number=Sing', '4', 'obl', '_', '_']
['3', 'ABC', 'ABC', 'X', '_', 'Foreign=Yes', '4', 'nsubj', '_', '_']
['4', 'відомо', 'відомо', 'ADV', '_', 'Degree=Pos', '0', 'root', '_', 'SpaceAfter=No']
['5', ',', ',', 'PUNCT', '_', '_', '9', 'punct', '_', '_']
['6', 'що', 'що', 'SCONJ', '_', '_', '9', 'mark', '_', '_']
['7', 'кут', 'кут', 'NOUN', '_', 'Animacy=Inan|Case=Nom|Gender=Masc|Number=Sing', '9', 'nsubj', '_', '_']
['8', 'A', 'A', 'X', '_', 'Foreign=Yes', '7', 'nmod', '_', '_']
['9', 'дорівнює', 'дорівнювати', 'VERB', '_', 'Aspect=Imp|Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin', '4', 'ccomp', '_', '_']
['10', '30', '30', 'NUM', '_', 'Case=Acc|NumType=Card|Uninflect=Yes', '11', 'nummod:gov', '_', '_']
['11', 'градусів', 'градус', 'NOUN', '_', 'Animacy=Inan|Case=Gen|Gender=Masc|Number=Plur', '9', 'obj', '_', 'SpaceAfter=No']
['12', ',', ',', 'PUNCT', '_', '_', '15', 'punct', '_', '_']
['13', 'кут', 'кут', 'NOUN', '_', 'Animacy=Inan|Case=Nom|Gender=Masc|Number=Sing', '15', 'nsubj', '_', '_']
['14', 'B', 'B', 'X', '_', 'Foreign=Yes', '13', 'nmod', '_', '_']
['15', 'дорівнює', 'дорівнювати', 'VERB', '_', 'Aspect=Imp|Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin', '9', 'parataxis', '_', '_']
['16', '45', '45', 'NUM', '_', 'Case=Acc|NumType=Card|Uninflect=Yes', '17', 'nummod:gov', '_', '_']
['17', 'градусів', 'градус', 'NOUN', '_', 'Animacy=Inan|Case=Gen|Gender=Masc|Number=Plur', '15', 'obj', '_', 'SpaceAfter=No']
['18', ',', ',', 'PUNCT', '_', '_', '21', 'punct', '_', '_']
['19', 'CM', 'CM', 'X', '_', 'Foreign=Yes', '21', 'nsubj', '_', '_']
['20', '—', '—', 'PUNCT', '_', 'PunctType=Dash', '21', 'punct', '_', '_']
['21', 'висота', 'висота', 'NOUN', '_', 'Animacy=Inan|Case=Nom|Gender=Fem|Number=Sing', '15', 'parataxis', '_', 'SpaceAfter=No']
['22', ',', ',', 'PUNCT', '_', '_', '24', 'punct', '_', '_']
['23', 'AC', 'AC', 'X', '_', 'Foreign=Yes', '24', 'nsubj', '_', '_']
['24', 'дорівнює', 'дорівнювати', 'VERB', '_', 'Aspect=Imp|Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin', '21', 'parataxis', '_', '_']
['25', '10', '10', 'NUM', '_', 'Case=Acc|NumType=Card|Uninflect=Yes', '26', 'nummod:gov', '_', '_']
['26', 'см', 'см', 'NOUN', '_', 'Abbr=Yes|Animacy=Inan|Case=Gen|Gender=Masc|Number=Plur|Uninflect=Yes', '24', 'obj', '_', 'SpaceAfter=No']
['27', '.', '.', 'PUNCT', '_', '_', '24', 'punct', '_', '_']
[]
['#', 'sent_id', '=', '2']
['#', 'text', '=', 'Знайдіть', 'відрізок', 'BM.']
['1', 'Знайдіть', 'знайти', 'VERB', '_', 'Aspect=Perf|Mood=Imp|Number=Plur|Person=2|VerbForm=Fin', '0', 'root', '_', '_']
['2', 'відрізок', 'відрізок', 'NOUN', '_', 'Animacy=Inan|Case=Acc|Gender=Masc|Number=Sing', '1', 'obj', '_', '_']
['3', 'BM', 'BM', 'X', '_', 'Foreign=Yes', '2', 'nmod', '_', 'SpaceAfter=No']
['4', '.', '.', 'PUNCT', '_', '_', '1', 'punct', '_', 'SpaceAfter=No']
[]

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
Зав. кафедри інформатики,
к.ф.-м.н.
_____ С. С. Гороховський
(підпис)
7 листопада 2019 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на дипломну роботу

студенту 2 року МП ІПЗ факультету інформатики

Смишу Олегу Руслановичу

Розробити Систему для розв'язування задач з геометрії

Зміст ТЧ до магістерської роботи:

Зміст

Анотація

Вступ

1. Обробка природної мови
2. Аналіз тексту геометричних задач
3. Аналіз готових рішень для розв’язку поставленої задачі
4. Побудова класів та методів для опису планіметрії
5. Використання морфосинтаксового аналізатора в задачах планіметрії

Висновки

Список літератури

Додатки

Дата видачі 25 жовтня 2019 р. Керівник _____
(підпис)
Завдання отримав _____
(підпис)

Тема:

_____

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проєкту (роботи)	Початок виконання етапу	Примітка
1	Отримання завдання на дипломну роботу	25.10.2019	
2	Огляд технічної літератури за темою роботи	30.10.2019	
3	Аналіз сучасних методів обробки природної мови	05.11.2019	
4	Аналіз задач з геометрії	12.12.2019	
5	Програмування класів та методів для опису планіметрії	08.01.2020	
6	Застосування розробленого алгоритму з морфосинтаксовим аналізатором в задачах планіметрії	22.03.2020	
7	Написання пояснювальної записки	15.04.2020	
8	Створення слайдів для доповіді та написання доповіді	01.05.2020	
9	Аналіз отриманих результатів з керівником	11.05.2020	
10	Корегування роботи за результатами попереднього захисту	30.05.2020	
11	Остаточне оформлення пояснювальної записки та слайдів	10.06.2020	
12	Захист магістерської роботи	16.06.2020	

Студент _____

Керівник _____

“____” _____ 2020 р.