МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ

«ФАХОВИЙ КОЛЕДЖ ІНЖЕНЕРІЇ ТА УПРАВЛІННЯ

НАЦІОНАЛЬНОГО АВІАЦІЙНОГО УНІВЕРСИТЕТУ»

ДОПУСТИТИ ДО ЗАХИСТУ

Заступник директора з НР

_____ О.В. Родіонова

«____»_____ 2021 р.

КВАЛІФІКАЦІЙНА РОБОТА

ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ

«БАКАЛАВР»

Тема: «Програмний модуль для аналізу тексту з використанням штучного інтелекту»

Автор: Борисенко Владислав Миколайович

Керівник проєкту: К.Б. Куц

Нормконтролер: В.М. Кругляк

Київ 2021

ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ФАХОВИЙ КОЛЕДЖ ІНЖЕНЕРІЇ ТА УПРАВЛІННЯ
НАЦІОНАЛЬНОГО АВІАЦІЙНОГО УНІВЕРСИТЕТУ»

Циклова комісія: Інженері програмного забезпечення

Освітнього ступеня: «Бакалавр»

Спеціальність: 123 Комп'ютерна інженерія

Освітньо-професійна програма: «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Голова циклової комісії

_____ Н.А. Рябчук

«____»_____ 2021 р.

ЗАВДАННЯ

на виконання дипломного проєкту

Борисенка Владислава Миколайовича

1. Тема роботи: «Програмний модуль для аналізу тексту з використанням штучного інтелекту»

затверджена наказом від «15» березня 2021 року № 28-Ст.

2. Термін виконання: з 12.04.2021р. по 20.06.2021р.

3. Вихідні дані: Розробити програму яка буде аналізувати коментарі та визначатиме, чи відноситься коментар до позитивних відгуків чи до негативних

4. Зміст пояснювальної записки:

У 1 розділі проаналізувати завдання на проєкт та основні методи його розв'язання.

У 2 розділі наведести опис основних етапів розробки.

У 3 розділі описати основні вимоги охорони праці.

КАЛЕНДАРНИЙ ПЛАН

виконання дипломної роботи

№ п/п	Етапи виконання кваліфікаційної роботи	Термін виконання етапів	Примітка
1.	Уточнення постановки задачі	12.04.2021	Виконано
2.	Аналіз літературних джерел	15.04.2021	Виконано
3.	Обґрунтування рішення	21.04.2021	Виконано
4.	Збір інформації	01.05.2021	Виконано
5.	Аналіз існуючих методів. Обгрунтування вибору мови програмування	06.05.2021	Виконано
6.	Виконання проєкту	08.06.2021	Виконано
7.	Оформлення і друк пояснювальної записки	15.06.2021	Виконано
8.	Оформлення презентації	17.06.2021	Виконано
9.	Отримання рецензій	20.06.2021	Виконано
10.	Захист проєкту		

Дипломник Борисенко В.М.

(підпис, дата) (П.І.Б.)

Дипломний керівник Куц К.Б.

(підпис, дата) (П.І.Б.)

Консультант з охорони праці Пешков І.В.

(підпис, дата) (П.І.Б.)

Зміст

Вступ

Зародження, а потім широке використання штучного інтелекту в різних сферах людської діяльності, і, перш за все, в технологіях інтернету речей викликало численні дискусії в різних галузях знань: філософії, соціології, машинобудуванні, приладобудуванні, комп'ютерних технологіях, робототехніці, медицині, освіті, військовій справі, юриспруденції тощо.

Дійсно, в останні роки з'являється все більше повідомлень, що свідчать про різке підвищення ефективності комплексів і систем інтернету речей в разі використання технологій штучного інтелекту (ШІ). Ці повідомлення відноситься до автономно керованого транспорту (літаків, кораблів, автомобілів), різноманітних промислових, медичних, будівельних, освітніх, професійних і побутових роботів, роботів військового та спеціального призначення тощо.

Штучний інтелект поступово входить у наше повсякденне життя. Роботи замінюють людину на виробництві, у обслуговуючій сфері, а системи штучного інтелекту, у тому числі – системи комп'ютерного зору, є основними системами керування роботизованими системами. Штучний інтелект – найперспективніший напрямок у машинобудуванні, у тому числі – у проектуванні, найперспективніший напрямок для досліджень та відкриттів, бізнесу.

Отримання значущої інформації – це і є задача інтелектуального аналізу тексту, який є процесом отримання якісної інформації з напів- та неструктурованих даних. Інформація може бути отримана спеціальним програмним забезпеченням шляхом виявлення певних шаблонів чи трендів на основі статистичної обробки тексту.

Тема дипломного проекту – «Програмний модуль для аналізу тексту з використанням штучного інтелекту».

Суть роботи програми полягає у інтелектуальному аналізі даних з використанням штучного інтелекту, метою якого є отримання інформації з тексту, ґрунтуючись на застосуванні ефективних, у практичному плані, методів машинного навчання та обробки природної мови.

1 Загальна частина

1.1 Постановка задачі

Метою даної дипломної роботи є розробка програмного модулю для аналізу тексту з використанням штучного інтелекту.

Процес аналізу відбувається відповідно до наступної послідовності кроків:

1. збір (gathering) текстових документів з різних джерел з наступним виконанням попередньої обробки (preprocessing) (усунення реклами; токенизація (tokenization), тобто розділення його на складові частини; усунення незначущих слів (артиклів, сполучників тощо); вирівнювання (stemming), тобто зведення всіх однокореневих слів до основної форми і, зменшення загальної кількості слів; перетворення тексту до вигляду, придатного для подальшого аналізу, наприклад, у вигляді гістограми частоти появи слів);

2. вибірка потрібних та усунення непотрібних властивостей тексту;

3. застосування одного чи декількох методів інтелектуального аналізу тексту з метою отримання патернів (шаблонів);

4. оцінювання патернів відповідно до критеріїв пошуку.

Завдання, які повинні бути виконані в процесі розробки системи, наступні:

1. можливість вводу тексту за допомогою пристроїв вводу;
2. можливість стирання частини тексту;
3. виконання зчитування тексту;
4. аналіз зчитаного тексту;
5. вивід результату аналізу.

Мінімальні системні вимоги до програмного та апаратного забезпечення для коректного функціонування програмного засобу:

1. програмна платформи: Windows XP/ 7/ 8/ 8.1/ 10;

2. 32-розрядний (x86) або 64-розрядний (x64) процесор з тактовою частотою 1 ГГц;

3. відеокарта з 500 Мб відеопам'яті;
4. процесор з частотою 1,0 ГГц і 2 ядрами;
5. 100 Мб оперативної пам'яті;
6. 100 Мб місця на жорсткому диску.

1.2 Теоретичні відомості

Інтелектуальні інформаційні системи проникають у всі сфери нашого життя, тому важко провести сувору класифікацію напрямів, у яких ведуться активні і чималі дослідження у сфері штучного інтелекту. Розглянемо коротко окремі.

Розробка інтелектуальних інформаційних систем чи систем, заснованих на виключно знаннях - це один з головних напрямів штучного інтелекту. Основною метою побудови таких систем є виявлення, дослідження й застосування їх знань висококваліфікованими експертами за для вирішення складних завдань. При побудові систем, заснованих на виключно знаннях, використовуються знання, накопичені експертами у конкретних правилах розв'язання тих чи інших завдань. Цей напрям своєї має на меті імітації людського мистецтва аналізу неструктурованих і слабоструктурованих проблем. У цій сфері досліджень здійснюється розробка моделей уявлення, вилучення й структурування знань, і навіть вивчаються проблеми створення баз знань. Приватним випадком систем, заснованих на виключно знаннях є експертні системи.

Розробка мовних інтерфейсів і машинний переклад. Проблеми комп'ютерної лінгвістики і машинного перекладу розробляють штучний інтелект з 1950-х рр. Системи машинного перекладу з природної мови забезпечують швидкість і систематичність доступу до інформації, оперативність і однаковість перекладу великих потоків, зазвичай, науково-технічних текстів. Системи машинного перекладу будуються як інтелектуальні системи, у основі яких лежать бази знань певної предметної області й складні моделі, щоб забезпечити додаткову трансляцію "вихідна мова оригіналу - мова сенсу - мова перекладу". Вони

базуються на структурно-логічному підході, що включає послідовний аналіз політики та синтез звичайно-мовних повідомлень. В цих системах здійснюється асоціативний пошук аналогічних фрагментів тексту та його перекладів у спеціальних базах даних. Даний напрям охоплює також методи й розробку систем, які забезпечують реалізацію процесу спілкування людини з комп'ютером на природній мові.

Генерація розпізнавання промови. Системи мовного спілкування створюють у цілях підвищення швидкості входження інформації ЕОМ, розвантаження зору, рук, і навіть для реалізації мовного спілкування на значній відстані відстані. У цих системах під текстом розуміють фонемний текст (як чується).

Обробка візуальної інформації. У цьому науковому напрямі вирішуються завдання обробки, аналізу та синтезу зображень. Завдання обробки зображень пов'язані з трансформуванням графічних образів, результатом якого є нові зображення. У задачі аналізу вихідні зображення перетворюються на дані іншого типу, наприклад, у текстові описи. При синтезі зображень на вхід системи надходить алгоритм побудови зображення, а вихідними даними є графічні об'єкти (системи машинної графіки).

Навчання і самонавчання. Ця актуальна область штучного інтелекту включає моделі, методи лікування й алгоритми, зорієнтовані на автоматичне накопичення, процес формування знань з допомогою процедур аналізу та узагальнення даних. До цього напряму відносятся системи видобутку даних і системи пошуку закономірностей в комп'ютерних базах даних розпізнавання образів. Це один з ранніх напрямів штучного інтелекту, у якому розпізнавання об'єктів базується на підставі застосування спеціального математичного апарату, забезпечує розподілення об'єктів на класи, а класи описуються сукупностями певних значень та ознак.

Ігри й машинна творчість. Машинна творчість охоплює творення комп'ютерної музики, віршів, інтелектуальні системи для винаходу нових об'єктів. Створення інтелектуальних ігор один із найрозвиненіших комерційних напрямів у сфері розробки програмного забезпечення.

Програмне забезпечення систем штучного інтелекту. Інструментальні кошти на розробку інтелектуальних систем включають спеціальні мови програмування, зорієнтовані на обробку символьної інформації, мови логічного програмування, мови уявлення знань, інтегровано-програмні середовища, і навіть оболонки експертних систем, що дозволяють створювати прикладні ЕС, не вдаючись до програмування.

Нові архітектури комп'ютерів. Це новий напрям пов'язаний з створенням комп'ютерів не фон-неймановської архітектури, орієнтованих на обробку символьної інформації. Відомі вдалі промислові рішення паралельних і векторних комп'ютерів, однак у наш час вони теж мають дуже високу вартість, і навіть недостатню сумісність з обчислювальними засобами.

Інтелектуальні роботи. Створення інтелектуальних роботів становить кінцеву мету робототехніки. Нині переважно використовуються програмовані маніпулятори з жорсткою схемою управління, названі роботами першого покоління. Попри очевидні успіхи окремих розробок, ера інтелектуальних автономних роботів доки не настала. Основними стримуючими чинниками з розробки автономних роботів є невирішені проблеми, у області інтерпретації знань, машинного зору, адекватного збереження і обробки тривимірної візуальної інформації.

Штучний інтелект буде реалізовано лише тоді, коли нежива машина зможе розв'язати завдання, які досі не вдавалося вирішити людині.

Штучний інтелект як наука існує близько півстоліття. Цей напрям інформатики - наймолодший, він виник у середині 1970-х років. Першою інтелектуальною системою вважається програма "Логик-Теоретик", призначена як доказ теорем і обчислення висловлювань. Її робота уперше була продемонстровано 9 серпня 1956 р. У створенні програми брали участь такі вчені, як А. Ньюелл, А.Тьюринг, До. Шеннон, Дж. Лоу, Р. Саймон та інших. Відтоді, та по наш час у області штучного інтелекту розроблено безліч комп'ютерних систем, які заведено називати інтелектуальними. Області їх застосування охоплюють практично всі галузі людської діяльності, пов'язані з обробкою інформації.

Існують різні методи створення систем штучного інтелекту. У наш час можна виділити 4 досить різних методи:

1. Логічний підхід. Основою для вивчення логічного підходу слугує алгебра логіки. Кожен програміст знайомий з нею з того часу, коли він вивчав оператор IF. Свого подальшого розвитку алгебра логіки отримала у вигляді числення предикатів — в якому вона розширена за рахунок введення предметних символів, відношень між ними. Крім цього, кожна така машина має блок генерації цілі, і система виводу намагається довести дану ціль як теорему. Якщо ціль досягнута, то послідовність використаних правил дозволить отримати ланцюжок дій, необхідних для реалізації поставленої цілі (таку систему ще називають експертною системою). Потужність такої системи визначається можливостями генератора цілей і машинного доведення теорем. Для досягнення кращої виразності логічний підхід використовує новий напрям, його назва — нечітка логіка. Головною відмінністю цього напряму є те, що істинність вислову може приймати окрім значень «так»/«ні» (1/0) ще й проміжні значення — «не знаю» (0,5), «пацієнт швидше живий, ніж мертвий» (0,75), «пацієнт швидше мертвий, ніж живий» (0,25). Такий підхід подібніший до мислення людини, оскільки вона рідко відповідає «так» або «ні».

2. Під структурним підходом ми розуміємо спроби побудови ШІ шляхом моделювання структури людського мозку. Однією з перших таких спроб був перцептрон Френка Розенблатта. Головною моделюючою структурною одиницею в перцептронах (як і в більшості інших варіантах моделювання мозку) є нейрон. Пізніше виникли й інші моделі, відоміші під назвою нейронні мережі (НМ) і їхні реалізації — нейрокомп'ютери. Ці моделі відрізняються за будовою окремих нейронів, за топологією зв'язків між ними і алгоритмами навчання. Серед найвідоміших на початку 2000-х років варіантів НМ можна назвати НМ зі зворотнім поширенням помилки, мережі Кохонена, мережі Гопфілда, стохастичні нейронні мережі. У ширшому розумінні цей підхід відомий як конекціонізм. Відмінності між логічним та структурним підходом не стільки принципові, як це здається на перший погляд. Алгоритми спрощення і вербалізації нейронних

мереж перетворюють моделі структурного підходу на явні логічні моделі. З іншого боку, ще 1943 року Воррен Маккалох і Волтер Піттс показали, що нейронна мережа може реалізувати будь-яку функцію алгебри логіки.

3. Еволюційний підхід. Під час побудови системи ШІ за даним методом основну увагу зосереджують на побудові початкової моделі і правилах, за якими вона може змінюватися (еволюціонувати). Причому модель може бути створено за найрізноманітнішими методами, це може бути і НМ, і набір логічних правил, і будь-яка інша модель. Після цього ми вмикаємо комп'ютер і він на основі перевірки моделей відбирає найкращі з них, і за цими моделями за найрізноманітнішими правилами генеруються нові моделі. Серед еволюційних алгоритмів класичним вважається генетичний алгоритм.

4. Імітаційний підхід. Цей підхід є класичним для кібернетики з одним із її базових понять чорний ящик. Об'єкт, поведінка якого імітується, якраз і являє собою «чорний ящик». Для нас не важливо, які моделі у нього всередині і як він діє, головне, щоб наша модель в аналогічних ситуаціях поводила себе без змін. Таким чином тут моделюється інша властивість людини — здатність копіювати те, що роблять інші, без поділу на елементарні операції і формального опису дій. Часто ця властивість економить багато часу об'єктові, особливо на початку його життя.

У рамках гібридних інтелектуальних систем намагаються об'єднати ці напрямки. Експертні правила висновків, можуть генеруватися нейронними мережами, а побіжні правила отримують за допомогою статистичного вивчення. Багатообіцяльний новий підхід, який ще називають підсиленням інтелекту, розглядає досягнення ШІ у процесі еволюційної розробки, як поточний ефект підсилення людського інтелекту технологіями.

Як наукова дисципліна ШІ має кілька основних напрямів:

1. машинне мислення (англ. machine reasoning, охоплює процеси планування, представлення знань і міркування, пошук та оптимізацію);

2. машинне навчання (умовно поділяється на глибоке навчання (англ. deep learning) і навчання з підкріпленням (англ. reinforcement learning)),

3. робототехніка (включає в себе управління, ситуаційне сприйняття, датчики і приводи, а також інтеграцію усіх інших методів в кібер-фізичні системи).

Якщо проаналізувати історію ШІ, можна виділити такий обширний напрям як моделювання міркувань (англ. Model-based reasoning). Багато років розвиток науки ШІ просувався саме таким шляхом, і зараз це одна з найрозвиненіших областей в сучасному ШІ. Моделювання міркувань має на увазі створення символьних систем, на вході яких поставлена деяка задача, а на виході очікується її розв'язок. Як правило, запропонована задача уже формалізована, тобто переведена на математичну форму, але або не має алгоритму розв'язання, або цей алгоритм занадто складний, трудомісткий тощо. В цей напрям входять: доведення теорем, прийняття рішень і теорія ігор, планування і диспетчеризація, прогнозування.

Таким чином, на перший план виходить інженерія знань, яка об'єднує завдання отримання знань з простої інформації, їх систематизацію і використання. Досягнення в цій області зачіпають майже всі інші напрями дослідження ШІ. Тут також необхідно відзначити дві важливі підобласті. Перша з них — машинне навчання — стосується процесу самостійного отримання знань інтелектуальною системою під час її роботи. Другу пов'язано зі створенням експертних систем — програм, які використовують спеціалізовані бази знань для отримання достовірних висновків щодо довільної проблеми.

Великі і цікаві досягнення є в області моделювання біологічних систем. Сюди можна віднести кілька незалежних напрямків. Нейронні мережі використовуються для розв'язання нечітких і складних проблем, таких як розпізнавання геометричних фігур чи кластеризація об'єктів. Генетичний підхід заснований на ідеї, що деякий алгоритм може стати ефективнішим, якщо відбере найкращі характеристики у інших алгоритмів («батьків»). Відносно новий підхід, де ставиться задача створення автономної програми — агента, котрий співпрацює з довкіллям, називається агентний підхід. А якщо належним чином примусити

велику кількість «не дуже інтелектуальних» агентів співпрацювати разом, то можна отримати «мурашиний» інтелект.

Задачі розпізнавання об'єктів вже частково розв'язуються в рамках інших напрямків. Сюди відносяться розпізнавання символів, рукописного тексту, мови, аналіз текстів. Особливо слід згадати комп'ютерне бачення, яке пов'язане з машинним навчанням та робототехнікою.

Робототехніка і штучний інтелект часто поєднуються одне з одним. Об'єднання цих двох наук, створення інтелектуальних роботів, можна вважати ще одним напрямом ШІ.

Окремо тримається машинна творчість, через те, що природа людської творчості ще менше вивчена, ніж природа інтелекту. Тим не менше, ця область існує, і тут стоять проблеми написання комп'ютером музики, літературних творів (часто — віршів та казок), образотворче мистецтво.

Нарешті, існує безліч програм штучного інтелекту, кожна з яких утворює майже самостійний напрямок. Як приклади, можна навести програмування інтелекту в комп'ютерних іграх, нелінійному керуванні, інтелектуальні системи безпеки. Наприклад, у 2018 році дослідники з Корнуельського університету зробили те, що зможе кардинально змінити процес розробки нових відеоігор. Вони створили пару нейронних мереж, що змагаються (генеративних змагальних мереж), і навчили їх на прикладі найпершої гри-шутера, Doom-а. В процесі навчання нейронні мережі визначили основні принципи побудови рівнів цієї гри і після цього вони стали здатні генерувати нові рівні без найменшої допомоги з боку людей.

Не важко бачити, що більшість областей дослідження перетинаються. Це властиво для будь-якої науки. Але в штучному інтелекті взаємозв'язок між, задавалося б, різними напрямами, виражено дуже сильно, і це пов'язано з філософською суперечкою про сильний і слабкий ШІ.

1.3 Аналіз засобів реалізації та обґрунтування вибору мови програмування

Засобами реалізації програмного продукту є програмне середовище розробки та мова програмування програмного засобу.

Для вибору найбільш доцільної мови програмування було обрано такі об’єктно-орієнтовані мови як C# та Python.

Серед середовищ розробки було порівняно інтегроване середовище Microsoft Visual Studio та Project Rider.

Переваги мови Python:

Безсумнівним плюсом є те, що інтерпретатор Python реалізований практично на всіх платформах і операційних системах. Першою такою мовою була Сі, проте її типи даних на різних машинах могли займати різну кількість пам'яті і це дійсно служило деякою перешкодою при написанні програм. Python ж таким недоліком не володіє.

Наступна важлива риса – розширюваність мови, цьому надається велике значення і, як пише сам автор, мова була задумана саме як розширювана. Це означає, що є можливість вдосконалення мови всіма зацікавленими програмістами. Інтерпретатор написаний на Сі і вихідний код доступний для будь-яких маніпуляцій. У разі необхідності, можна вставити його в свою програму і використовувати як вбудовану оболонку. Або ж, написавши на Сі свої доповнення до Python і скомпілювавши програму, отримати "розширений" інтерпретатор з новими можливостями.

Гідність – наявність великого числа модулів, що підключаються до програми, які забезпечують різні додаткові можливості. Такі модулі пишуться на Сі і на самому Python і можуть бути розроблені усіма досить кваліфікованими програмістами. Як приклад можна навести такі модулі:

Numerical Python – розширені математичні можливості, такі як маніпуляції з цілими векторами і матрицями;

Tkinter – побудова додатків з використанням графічного інтерфейсу користувача (GUI) на основі широко розповсюдженого на X-Windows Tk-інтерфейсу;

OpenGL – використання великої бібліотеки графічного моделювання дво- і тривимірних об'єктів Open Graphics Library фірми Silicon Graphics Inc.

Єдиним недоліком є порівняно невисока швидкість виконання Python-програми, що обумовлено її інтерпретованістю. Однак, це з лишком окупається перевагами мови при написанні програм не дуже критичних до швидкості виконання.

Перевагами використання мови C# є:

1. підтримка корпорацією Microsoft. На відміну від Java, якої не пішов на користь перехід у власність Oracle, C# добре розвивається завдяки зусиллям Microsoft;

2. останнім часом багато вдосконалюється. Так як C # був створений пізніше, ніж Java та інші мови, то потрібно дуже багато доопрацювати. Також це стосується популяризації і безкоштовності - було обіцяно відкрити вихідний код, а інструменти (Visual Studio, Xamarin) стали безкоштовними для приватних осіб і невеликих компаній;

3. багато конструкцій, які створені для полегшення написання і розуміння коду (особливо якщо це код іншого програміста) і не грають ролі при компіляції;

4. середній поріг входження. Синтаксис схожий на C, C ++ або Java полегшує перехід для інших програмістів. Для новачків це також одна з найперспективніших мов для вивчення;

5. додано функціональне програмування (F #);

Незначними недоліками є:

1. орієнтованість, в основному, тільки на .NET (на Windows платформу);

2. безкоштовність тільки для невеликих компаній, учнів і програмістів-одинаків. Для великих команд покупка ліцензій обійдеться недешево;

3. зберегли оператор go to.

Особливості Microsoft Visual Studio:

Офіційна. Так як і мова, і середовище розробки створені в Microsoft, логічно припустити, що нічого більш функціонального ви не знайдете в усьому Інтернеті.

У деяких випадках без Visual Studio не обійтися - наприклад, при використанні технологій UWP і WPF.

Безкоштовна. Версії «Community edition» для рядового користувача буде досить. Тим більше, тепер можна підключати плагіни (на відміну від старої версії Express).

Функціональна. У Visual Studio безліч якісних плагінів. З їх допомогою можна розширити функціональність програми і підключити інші мови.

Підтримує платформи .NET. Visual Studio має широкі можливості по розробці додатків під Windows, в тому числі в .NET-сегменті.

Хмарні сховища. Увійдіть в співтовариство Visual Studio - і отримаєте доступ до хмарного сховища, де зможете розташовувати файли проектів.

Корпоративність. Технологія беклога дозволяє членам команди взаємодіяти при гнучкою методології розробки.

Баги при переходах з тріал-версії. При переході на платну версію можуть губитися настройки і порушуватися робота корпоративного сервера.

Складність. Самостійно освоїти Visual Studio новачкові буде непросто - надто багато доступних функцій, захованих в підрозділах меню.

Project Rider - середовище, випущене JetBrains для работи с платформою .NET.

ReSharper. Це плагін, спочатку розроблений для підвищення продуктивності Visual Studio. Тепер на його основі випущена IDE.

Підтримка повного циклу. Фірмова риса продуктів JetBrains, втілена і в Project Rider. З ним ви зможете організувати весь цикл створення ПО: від ідеї до підтримки.

Функціональність. Project Rider дозволяє підключити MSBuild і XBuild, працювати з CLI-проектами і організувати налагодження додатків .NET and Mono. Безліч опцій для швидкого створення коду покращує продуктивність.

Multiple runtime. Підтримка декількох запущених програм.

Кросплатформеність. Project Rider працює з Windows, Linux і MacOS.

Контроль версій. Вбудований інструмент дозволяє безпосередньо організувати роботу з Git, Mercurial і TFS.

Недоліками є:

Молодість. Частина функціональності ще в розробці, не всі стартові баги виправлені.

Вартість. Найдешевша версія Project Rider обійдеться в 139 доларів за перший рік використання. Але є тріал-версія і спеціальні пропозиції для студентів і непрофільних організацій.

В результаті порівняння основних функціональних можливостей середовищем розробки програмного засобу було обрано Microsoft Visual Studio завдяки основним перевагам: безкоштовність, функціональність, складність, корпоративність.

Серед вище оглянутих мов програмування C# є найбільш перспективною завдяки вдосконаленню, наявності різних конструкцій, що спрощують написання коду, середньому порогу входження та іншим функціональним перевагам.

2 Спеціальна частина

2.1 Опис алгоритму створення програмного засобу

Алгоритм - це однозначно певна послідовність дій, записана на зрозумілій виконавцеві алгоритмічній мові і визначає процес переходу від вихідних даних до результату.

У цьому визначенні вже вказані основні властивості алгоритму. По-перше, алгоритм складається з кінцевого набору інструкцій або кроків, по-друге, кожен крок трактується виконавцем єдиним чином, що дозволяє гарантовано отримати рішення для деякого набору вхідних даних, по-третє, алгоритм завжди зводиться до деякого перетворення вихідних даних в результат або результати.

У цьому сенсі формули для вирішення квадратного рівняння або навіть чітко складену інструкцію по варінню кави можна вважати алгоритмами, здійсненними виконавцем-людиною.

Для машини, зрозуміло, потрібно більш чітка формалізація завдання, ніж для людини, розуміти природну мову комп'ютери поки не здатні, звідси необхідність врахування при складанні алгоритму обмеженого набору інструкцій ЕОМ.

Створення алгоритму програмного засобу передбачає графічне представлення інформації у вигляді геометричних фігур.

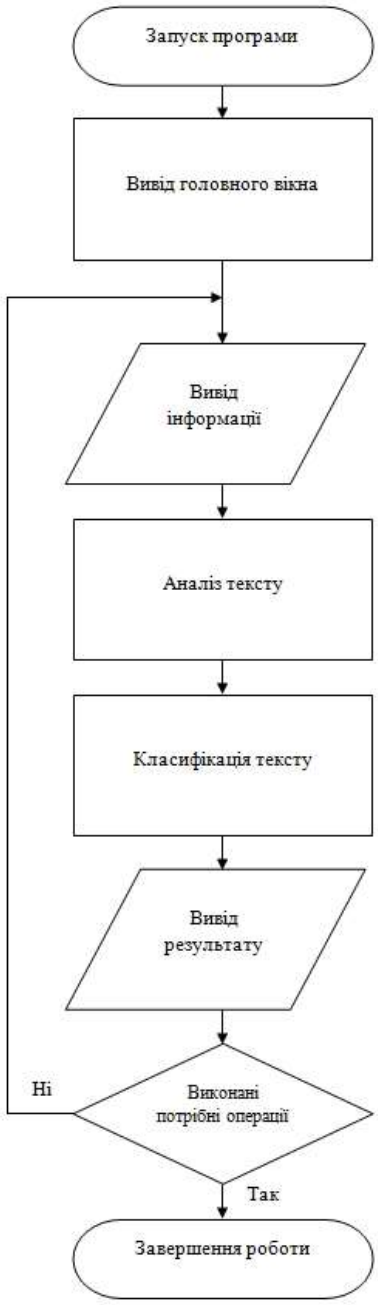


Рисунок 2.1 – Блок-схема алгоритму програми аналізу тексту

Мета дипломного проекту передбачає створення програмного засобу для аналізу тексту з використанням штучного інтелекту.

На рисунку 2.1 зображені основні функції, які виконує програмний засіб:

1. Відкриття вікна програми.
2. Ввід інформації:
3. Аналіз тексту.
4. Класифікація тексту
5. Вивід результату.

2.2 Функціональні та нефункціональні вимоги

Специфікація вимог до програмного забезпечення (англ. Software Requirements Specification (SRS)) - процес формалізованого опису функціональних і нефункціональних вимог, вимог до характеристик, які будуть імплементуватися, втілюватися на етапах ЖЦ ПЗ. Це повний опис поведінки системи, що розробляється.

Функціональні вимоги - це перелік функцій або сервісів, які повинна надавати система, а також обмежень на дані і поведінку системи при їхньому виконанні. Специфікація функціональних вимог - опис функцій та їхніх властивостей, які не містять у собі протиріч і виключень.

Нефункціональні вимоги визначають умови виконання функцій (наприклад, захист інформації у БД, аутентифікація доступу до ПС тощо), середовища, що безпосередньо не пов'язані з функціями, а відбивають потреби користувачів щодо їх виконання. Ці вимоги характеризують принципи взаємодії із середовищами або іншими системами, а також визначають показники часу роботи, захисту даних і досягнення якості з урахуванням рекомендацій використовуваного стандарту. Вони можуть встановлюватися як числові значення (наприклад, час чекання відповіді, кількість клієнтів, що обслуговуються і ін.) у різних одиницях виміру, включаючи, наприклад, ймовірність (значення ймовірності безвідмовної роботи системи – показника її надійності).

Функціональні вимоги програмного засобу аналізу тексту наступні:

1. Користувач повинен мати можливість вводу тексту.
2. Система повинна виконувати аналіз тексту та його класифікацію.
3. Можливість виводу результату аналізу.

До нефункціональних вимог відносяться:

1. можливості інтерфейсу користувача;
2. час відгуку;
3. надійність;
4. пристосованість до інсталяції;

5. інформаційна підтримка;

6. пристосованість до супроводження;

7. інші фактори.

Вимоги до зовнішнього інтерфейсу:

1. сумісність з потребами та можливостями користувача;

2. реакція системи на всі типи запитів також повинна бути однозначною і зрозумілою, простою;

3. максимальна простота його використання і готовність в повній мірі задовольнити запити користувача;

4. вивід різних компонентів системи повинен здійснюватися в новому вікні.

Системні вимоги:

1. сумісний з операційними системами Windows 7, Windows 8 та Windows 10;

2. портативний модуль (з вихідними кодами) повинен займати не більше 100 МБ дискового простору;

3. модуль повинен коректно працювати на апаратному забезпеченні з обмеженими ресурсами.

2.3 Структура програмного модуля

Структура програми - штучно виділені програмістом взаємодіючі частини програми. Використання раціональної структури усуває проблему складності розробки; робить програму зрозумілою людям; підвищує надійність роботи програми при скороченні терміну її тестування і термінів розробки взагалі.

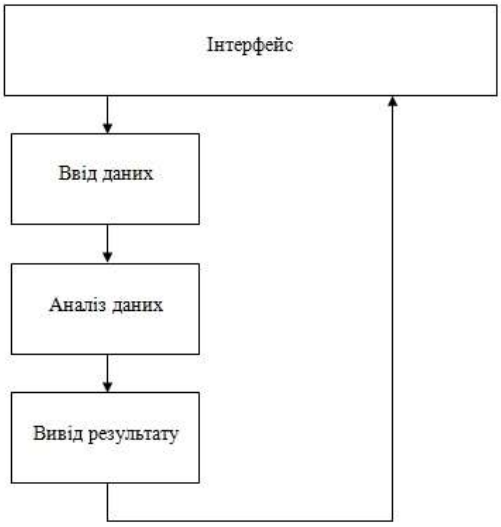


Рисунок 2.2 – Структурна схема програми аналізу тексту

Компоненти схеми:

1. інтерфейс, забезпечує зручний ввід та вивід тексту для користувача;
2. ввід даних в програму;
3. аналіз даних;
4. вивід результату аналізу.

Універсальна мова моделювання (Unified Modelling Language або UML) — це мова позначень або побудови діаграм, призначена для визначення, візуалізації і документування моделей зорієнтованих на об’єкти систем програмного забезпечення. UML не є методом розробки, іншими словами, у конструкціях цієї мови не повідомляється про те, що робити першим, а що останнім, і не надається інструкцій щодо побудови вашої системи, але ця мова допомагає вам наочно переглядати компонування системи і полегшує співпрацю з іншими її розробниками. Розробкою UML керує Object Management Group (OMG). Ця мова є загальноприйнятим стандартом графічного опису програмного забезпечення.

UML розроблено для розробки структури зорієнтованого на об’єкти програмного забезпечення, ця мова має дуже обмежену користь для програмування на основі інших парадигм.

Конструкції UML створюються з багатьох модельних елементів, які позначають різні частини системи програмного забезпечення. Елементи UML

використовуються для побудови діаграм, які відповідають певній частині системи або точці зору на систему.

Діаграма класів системи штучного аналізу тексту представлена на рис. 2.3.

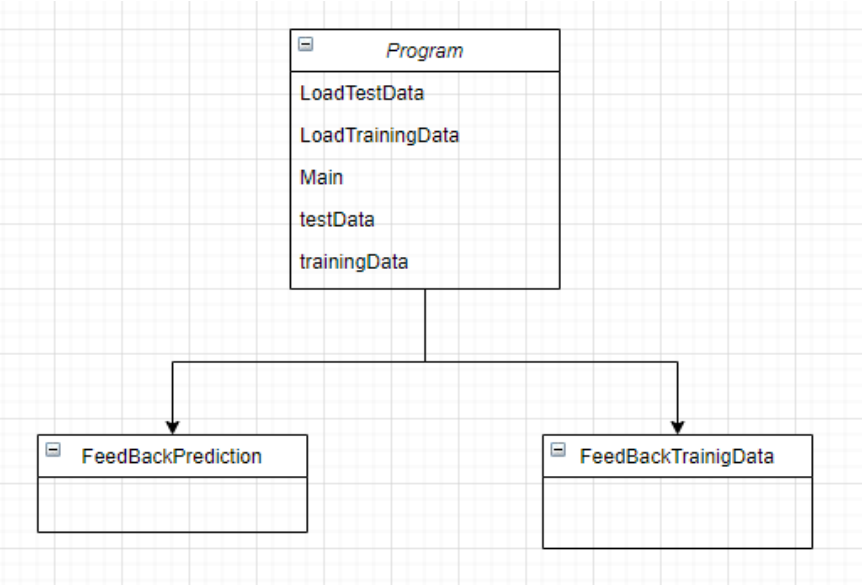


Рисунок 2.3 – Діаграма класів

Діаграма класів дозволяє формалізувати логічну модель програми, на рівні структурних елементів системи, тобто класів. Вона надає статичне представлення про структуру програми – з яких класів вона складається, які зв’язки між цими класами, з чого складається кожний клас.

Діаграми варіантів використання (usecase diagrams) використовуються для відображення сценаріїв використання системи (usecases) та користувачів системи (actors), які використовують її функції. Актори на діаграмі варіантів використання позначаються символом людини, а варіанти використання – еліпсом. Актори та варіанти використання поєднуються напрямленою асоціацією (unidirectional association) – стрілкою, що спрямована від актора до варіанта використання. Також актори можуть поєднуватися з використанням зв’язків узагальнення.

Діаграма варіантів використання модулю зображена на рисунку 2.4.

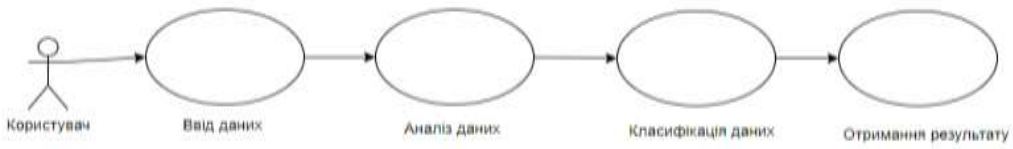


Рисунок 2.4 – Діаграма прецедентів програмного засобу

Діаграма послідовностей використовується для уточнення діаграми прецедентів, більш детального опису логіки сценаріїв використання. Це відмінний засіб документування проекту з точки зору сценаріїв використання. Діаграми послідовностей зазвичай містять об'єкти, які взаємодіють у рамках сценарію, повідомлення, якими вони обмінюються, і які повертаються результати, які пов'язані з повідомленнями.

На діаграмі послідовності зображуються тільки ті об'єкти, які безпосередньо беруть участь у взаємодії..

Діаграма послідовності (sequence diagram) є основним способом відображення взаємодії об'єктів в часі. Незважаючи на те, що діаграма послідовності спочатку орієнтована на програмні проекти з використанням ООП, вона застосовується значно ширше, в тому числі у випадках, коли не передбачені відповідні методи опису «динаміки» програми. У подібних випадках повинна здійснюватися адаптація поняття об'єкта.

Діаграма послідовності, як і інші діаграми, що виділяються стандартом UML, повинна зображуватися відповідно до нього.

Основні моменти, на які слід звернути увагу при роботі над діаграмою послідовності наступні.

Взаємодіючі об'єкти (objects), як правило, зображуються у вигляді прямокутників з суцільними межами і розміщуються по горизонталі.

Усередині прямокутника вказується ім'я об'єкта, за якими через двокрапку може слідувати ім'я класу даного об'єкта. Імена об'єкта і класу підкреслюються.

Серед об'єктів можуть виділятися так звані актори (actors), Актори повинні зображувати не особливим чином за допомогою символу «чоловічка», а так, як звичайні об'єкти.

Лінія життя (lifeline) об'єкта зображується за допомогою штрихованої лінії, яка проводиться вертикально вниз від середини його нижньої межі.

За допомогою лінії життя показується період часу, протягом якого об'єкт існує в системі і, отже, може потенційно брати участь у взаємодіях.

Всі об'єкти існують «в одному часі». Початок тимчасового відліку відповідає рівню примикання колій життя до об'єктів, розташованих на діаграмі вище всіх інших об'єктів (створених перед взаємодією або вже існуючих в системі). При створенні деякого об'єкта пізніше створення інших взаємодіючих об'єктів він показується на відповідному рівні. При видаленні об'єкта, тобто при звільненні займаних їм ресурсів, його лінія життя переривається символом «X».

Активність (activity) об'єкта, тобто період часу, протягом якого він бере участь у взаємодії, може збігатися з фокусом управління (focus of control) і відображається тонким вертикальним прямокутником відповідної тривалості на лінії його життя. Ширина прямокутників повинна дорівнювати 10 мм і повинна бути однаковою в межах креслення. Об'єкти - ініціатори взаємодій рекомендується зображати на кресленні лівіше.

Повідомлення (messages), якими обмінюються об'єкти в процесі взаємодії, показуються різними лініями зі стрілками між лініями життя об'єктів, спрямованими в бік передачі.

Термін повідомлення має максимально широкий сенс і може означати будь-який вид передачі управління або даних.

Діаграма послідовності модулю аналізу тексту представлена на рисунку 2.5.

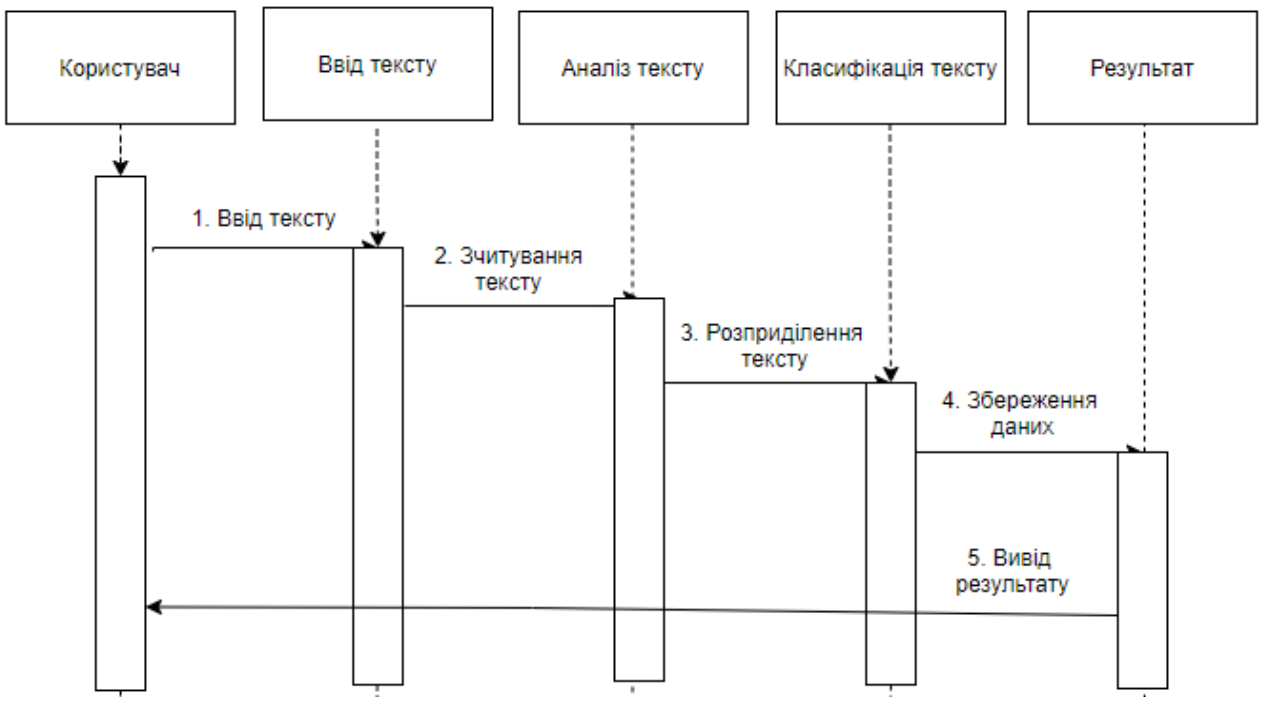


Рисунок 2.5 – Діаграма послідовності системи

2.4 Опис засобів реалізації

Даний модуль являє собою консольну програму. Консольна програма – програма, що не має графічного інтерфейсу. В програмі можна здійснювати введення та виведення текстової інформації в звичайному вікні, яке називають консольним.

Рисунок 2.6 – Консольне вікно вводу/виводу

Приклади основних методів, які використовуються для реалізації системи аналізу тексту описані нижче.

Метод здійснює створення нового об’єкту структури List.

```
static List<FeedBackTrainigData> trainingData = new List<FeedBackTrainigData>();
```

Наступний метод аналогічний попередньому.

```
static List<FeedBackTrainigData> testData = new List<FeedBackTrainigData>();
```

Наступний метод виконує додавання даних до структури.

```
static void LoadTestData()
{
testData.Add(new FeedBackTrainigData()
{
   FeedBackText = "Good",
   IsGood = true
```

```
});

testData.Add(new FeedBackTrainigData()
{
   FeedBackText = "not",
   IsGood = false
});

testData.Add(new FeedBackTrainigData()
{
   FeedBackText = "no",
   IsGood = false
});

testData.Add(new FeedBackTrainigData()
{
   FeedBackText = "no good",
   IsGood = false
});

testData.Add(new FeedBackTrainigData()
{
   FeedBackText = "not bad",
   IsGood = true
});

testData.Add(new FeedBackTrainigData()
{
   FeedBackText = "not nice",
   IsGood = false
});

testData.Add(new FeedBackTrainigData()
{
   FeedBackText = "sweet",
```

```
    IsGood = true
});

testData.Add(new FeedBackTrainigData()
{
    FeedBackText = "Bad",
    IsGood = false
});

testData.Add(new FeedBackTrainigData()
{
    FeedBackText = "nice and good",
    IsGood = true
});

testData.Add(new FeedBackTrainigData()
{
    FeedBackText = "Bad and horrible",
    IsGood = false
});

testData.Add(new FeedBackTrainigData()
{
    FeedBackText = "pretty good",
    IsGood = true
});

testData.Add(new FeedBackTrainigData()
{
    FeedBackText = "pretty bad",
    IsGood = false
});

testData.Add(new FeedBackTrainigData()
{
```

```
    FeedBackText = "awesome",
    IsGood = true
});

testData.Add(new FeedBackTrainigData()
{
    FeedBackText = "shit",
    IsGood = false
});

testData.Add(new FeedBackTrainigData()
{
    FeedBackText = "nice",
    IsGood = true
});

testData.Add(new FeedBackTrainigData()
{
    FeedBackText = "shittiest shit",
    IsGood = false
});

testData.Add(new FeedBackTrainigData()
{
    FeedBackText = "beautiful",
    IsGood = true
});

testData.Add(new FeedBackTrainigData()
{
    FeedBackText = "ow so bad",
    IsGood = false
});

testData.Add(new FeedBackTrainigData()
```

```
        {
            FeedBackText = "average but ok",
            IsGood = true
        });

        testData.Add(new FeedBackTrainigData()
        {
            FeedBackText = "could be better",
            IsGood = false
        });
    }
```

Наступний метод представляє собою роботу штучного інтелекту для аналізу тексту.

```
        static void Main(string[] args)
    {
        LoadTrainingData();

        var mlContext = new MLContext();

        IDataView dataView = mlContext.CreateStreamingDataView<FeedBackTrainigData>(trainingData);

        var pipeline = mlContext.Transforms.Text.FeaturizeText("FeedBackText", "Features").Append(mlContext.BinaryClassification.Trainers.FastTree(numLeaves: 50, numTrees: 50, minDatapointsInLeaves: 1));

        var model = pipeline.Fit(dataView);

        LoadTestData();

        IDataView dataView1 = mlContext.CreateStreamingDataView<FeedBackTrainigData>(testData);

        var predictions = model.Transform(dataView1);
```

```
        var metrics = mlContext.BinaryClassification.Evaluate(predictions, "Label");

        Console.WriteLine(metrics.Accuracy);

        while (true)
        {
            Console.WriteLine("Input a feedback string: ");

            string feedBackStr = Console.ReadLine().ToString();

            var PredFunc = model.MakePredictionFunction<FeedBackTrainigData, FeedBackPrediction>(mlContext);

            var feedBackInput = new FeedBackTrainigData();

            feedBackInput.FeedBackText = feedBackStr;

            var feedBackPredicted = PredFunc.Predict(feedBackInput);

            Console.WriteLine("Predicted feed back is: " + feedBackPredicted.IsGood);
        }
    }
```

2.5 Порівняльний аналіз реалізованого програмного засобу та програм-аналогів

На даний час існує потреба об’єктивного аналізу змісту інформаційних потоків, які в свою чергу вимагають певних методів дослідження. Одним із таких методів є контент-аналіз. Під поняттям контент-аналізу розуміють об’єктивний аналіз змісту будь-якого тексту.

Програми кількісного контент-аналізу надають можливість аналізу текстів за різними критеріями, здатні працювати з документами різних форматів і здійснюють класифікацію словників.

Особливістю програм якісного контент-аналізу є те, що вони призначені не тільки для аналізу текстів, але й для аудіо та відео фрагментів кодування, також допомагають в розвитку теорій і надають можливість перевірки цих теорій.

Серед найбільш відомих програм аналізу тексту є: Textanz, TextQuest, Kwali-tan.

Textanz є дійсно унікальним і повинен мати інструмент для всіх, що беруть участь в написанні текстів. Ця програма аналізує всі види змісту тексту і надає вам зі списком або словником слів, фраз, граматики і форм, і частоти їх використання в тексті. Інформація, яку ви отримаєте від аналізу Textanz дозволяє перевірити надмірне використання або повторення слів або фраз в будь-якому документі. Textanz забезпечує життєво важливу функцію редагування, яка відсутня в більшості стандартних граматичних-пропускних і редагування програм.

Інтерфейс роботи програмного засобу зображений на рис. 2.7.

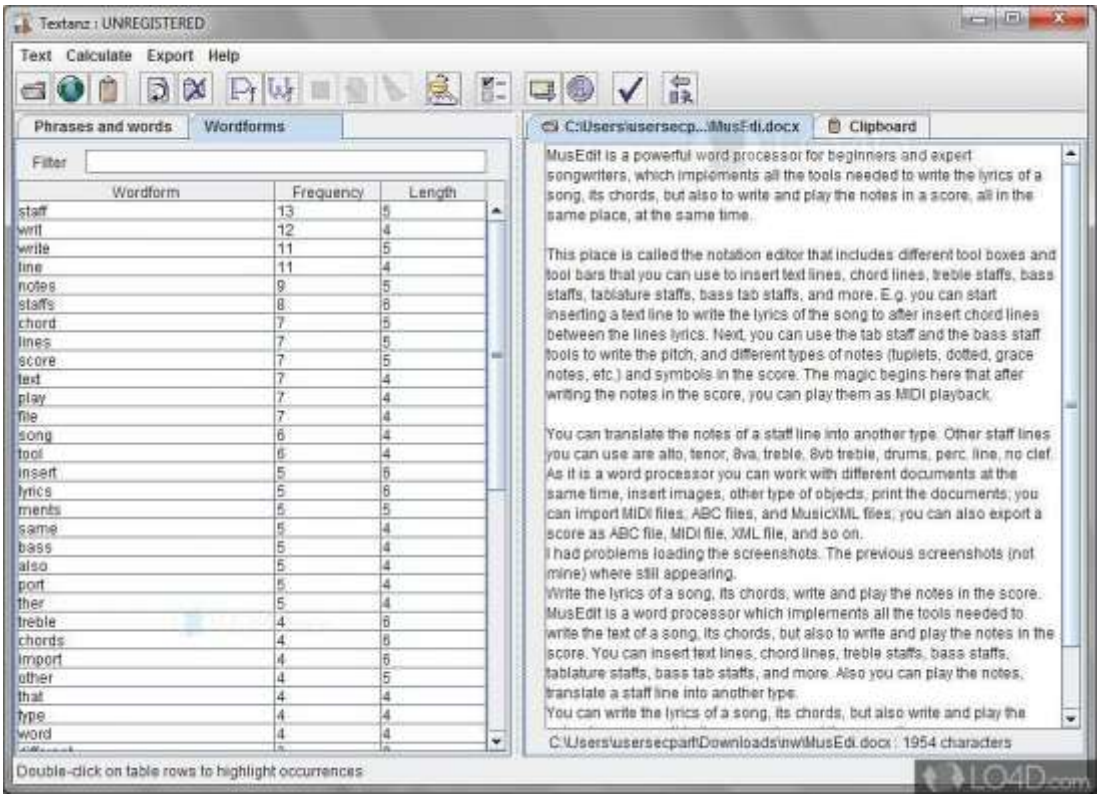
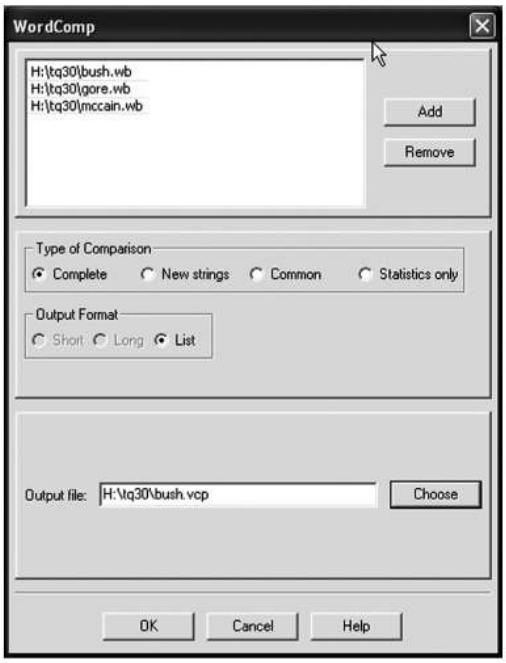


Рисунок 2.8 – Програма Textanz

TextQuest - це незалежна від мови програма, яка підтримує кількісний і якісний аналіз текстів: списків слів, списків фраз і збігів слів. Всі вони відсортовані за алфавітом.



Source: Copyright Harald Klein, Social Science Consulting, Osnabrück, Germany; used by permission.

Рисунок 2.9 – Програма TextQuest

Kwalitan - це програма, яка забезпечує ефективне зберігання даних та пропонує безліч інструментів для аналізу матеріалу, таких як кодування, вибір та пошук фрагментів тексту, категоризація кодів, огляд кодів (або слів) у тексті, ключові слова в контексті та написання пам'яток. За допомогою Kwalitan можна аналізувати тексти, зображення, аудіо та відеокліпи..

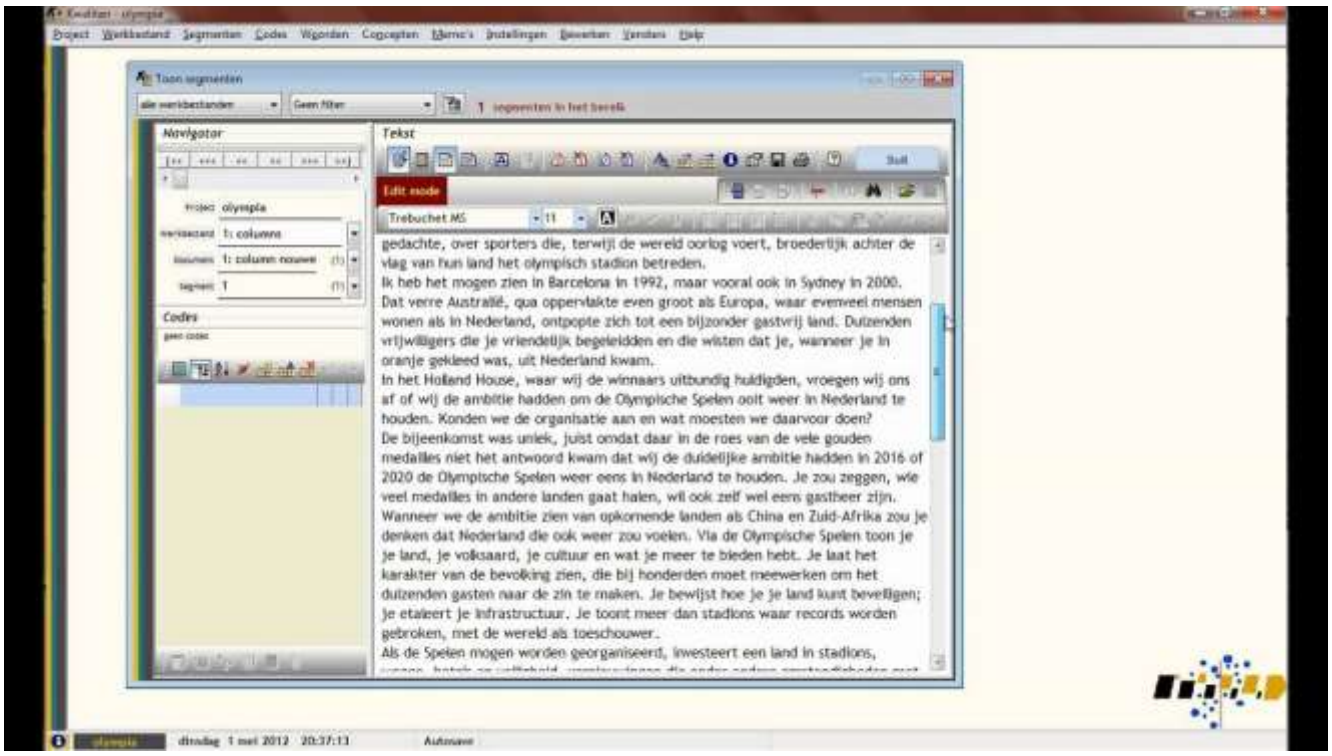


Рисунок 2.10 – Програма Kwalitan

«Програмний модуль для аналізу тексту з використанням штучного інтелекту» здійснює зчитування тексту, який вводиться за допомогою пристроїв вводу, виконує аналіз за допомогою штучного інтелекту та вивід результату аналізу.

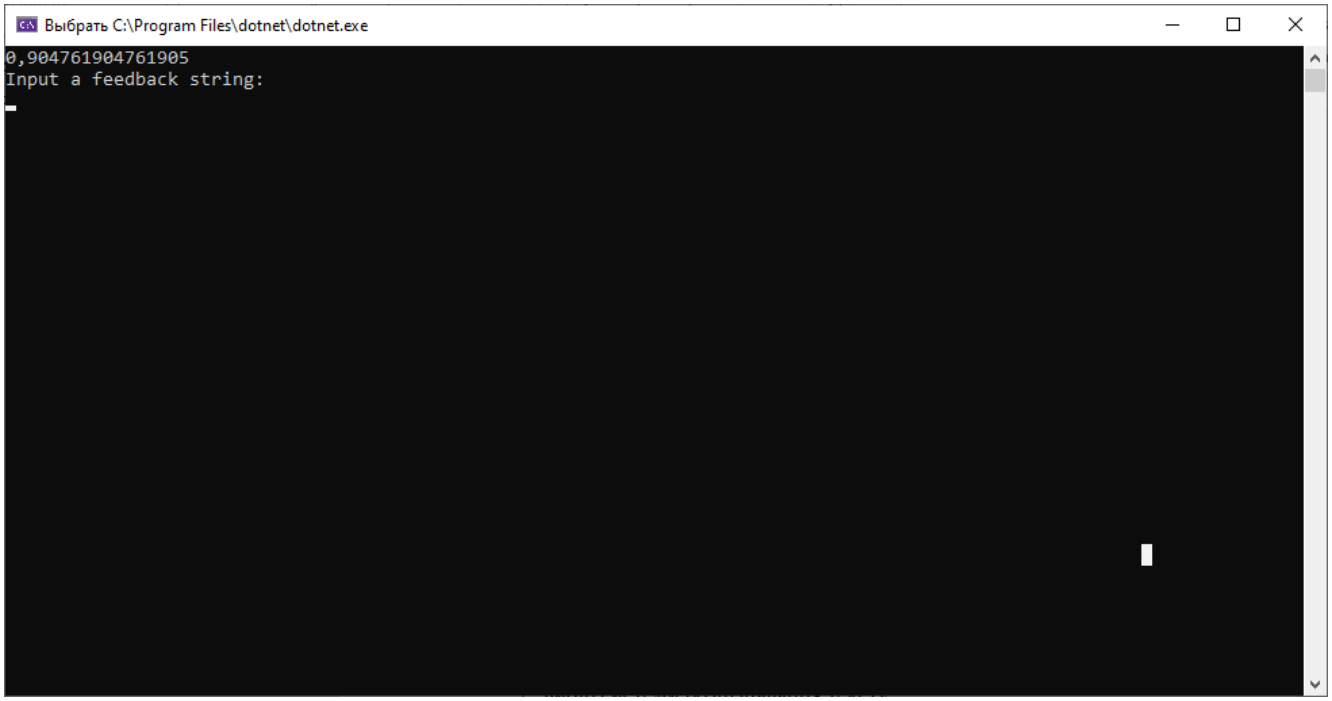


Рисунок 2.10 – Програма аналізу тексту «Програмний модуль для аналізу тексту з використанням штучного інтелекту»

При порівнянні вище перелічених програм для аналізу тексту із створеним програмними засобом можна зробити висновок, що «Програмний модуль для аналізу тексту з використанням штучного інтелекту» коректно виконує аналіз тексту, дозволяє скопіювати друкований текст, що зручно для подальшого використання.

Програмний засіб є безкоштовним та доступним для користувачів через мережу Інтернет. Для користування програмою не потрібно завантажувати непотрібні файли та модулі.

2.6 Керівництво користувача програмного модуля

Програмний засіб запускається при подвійному натисненні на ярлик. На екрані з’являється головне вікно програмного засобу (рис. 2.11).

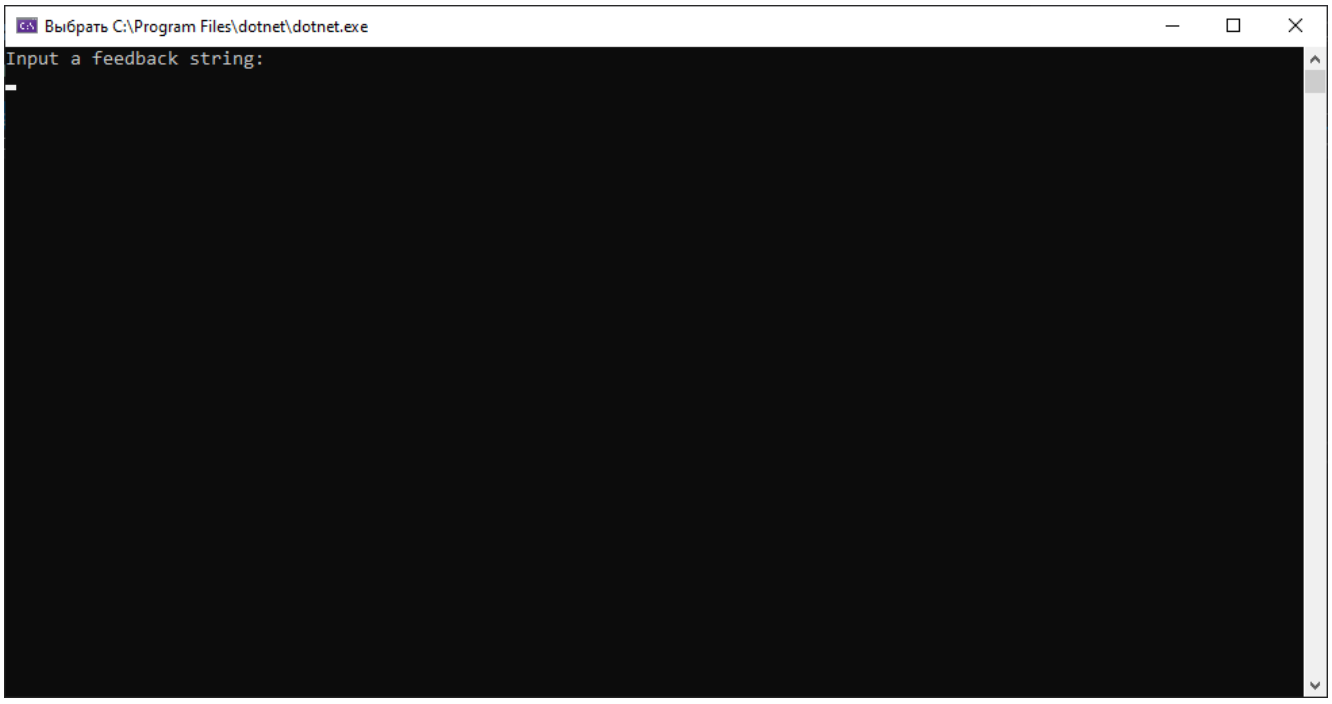


Рисунок 2.11 – Головне вікно програми

У головному вікні за замовчуванням виведено наступний текст (рисунок 2.12).

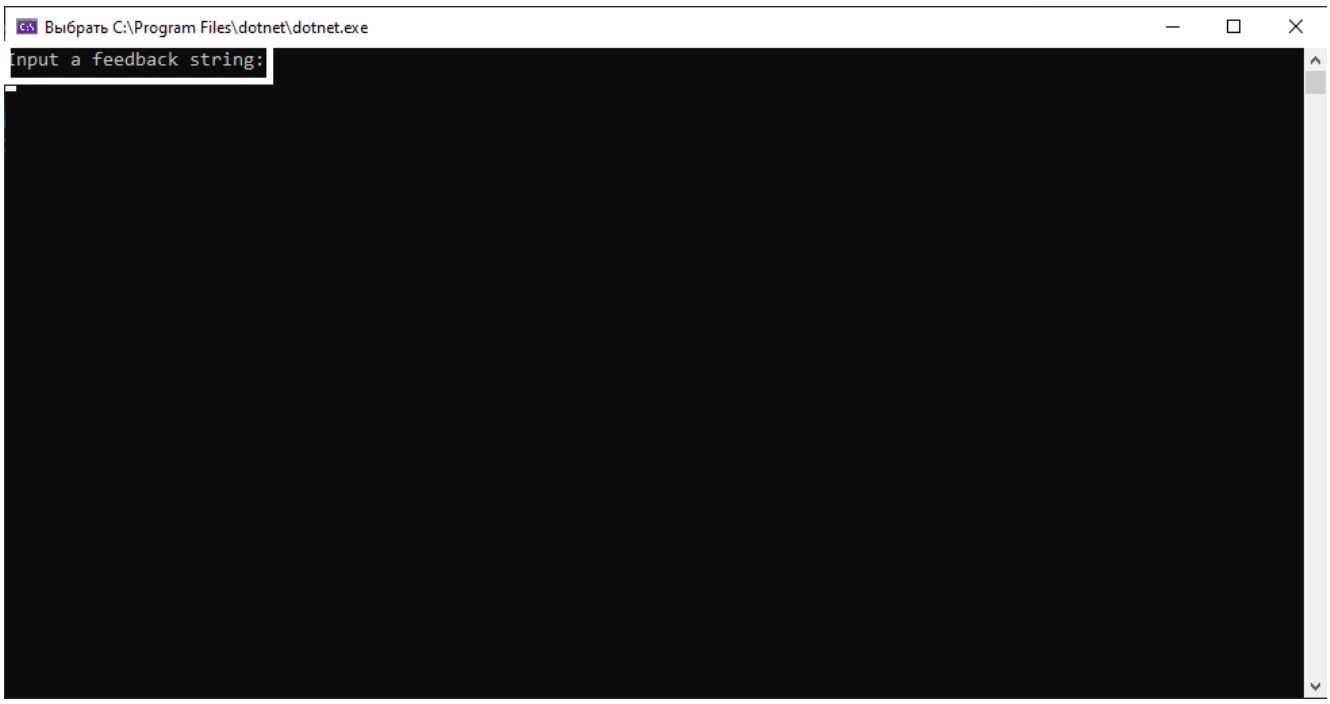


Рисунок 2.12 – Виведений за замовчуванням текст

Після цього програма доступна для вводу тексту (рисунок 2.13).

Рисунок 2.13 – Ввід тексту

Після вводу програма виконує аналіз тексту за допомогою штучного інтелекту та виводить відповідне повідомлення. При позитивному відгуку виводиться наступне повідомлення (рисунок 2.14).

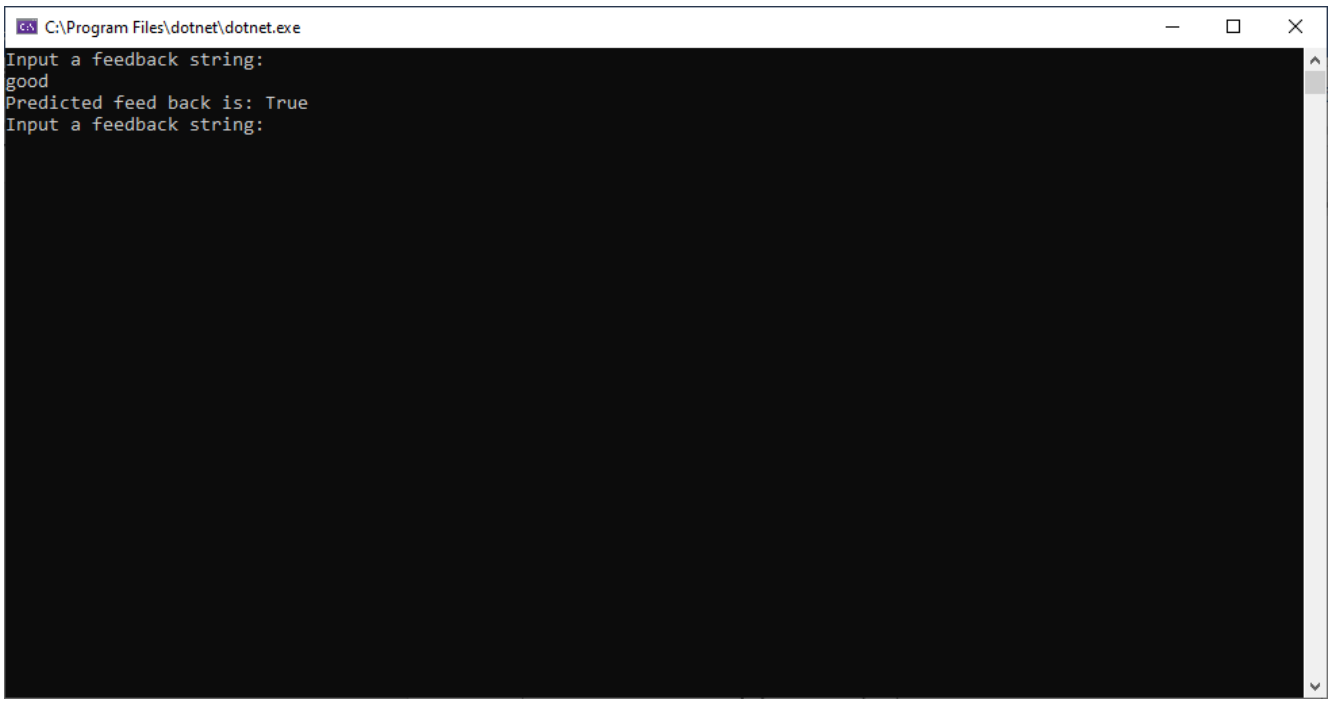


Рисунок 2.14 – Аналіз позитивного відгуку

Приклад негативного відгуку представлений на рисунку 2.15.

Рисунок 2.15 – Негативний відгук

Для коректного завершення програмного засобу потрібно натиснути на червоний хрестик в правому верхньому куті вікна програми (рисунок 2.16).

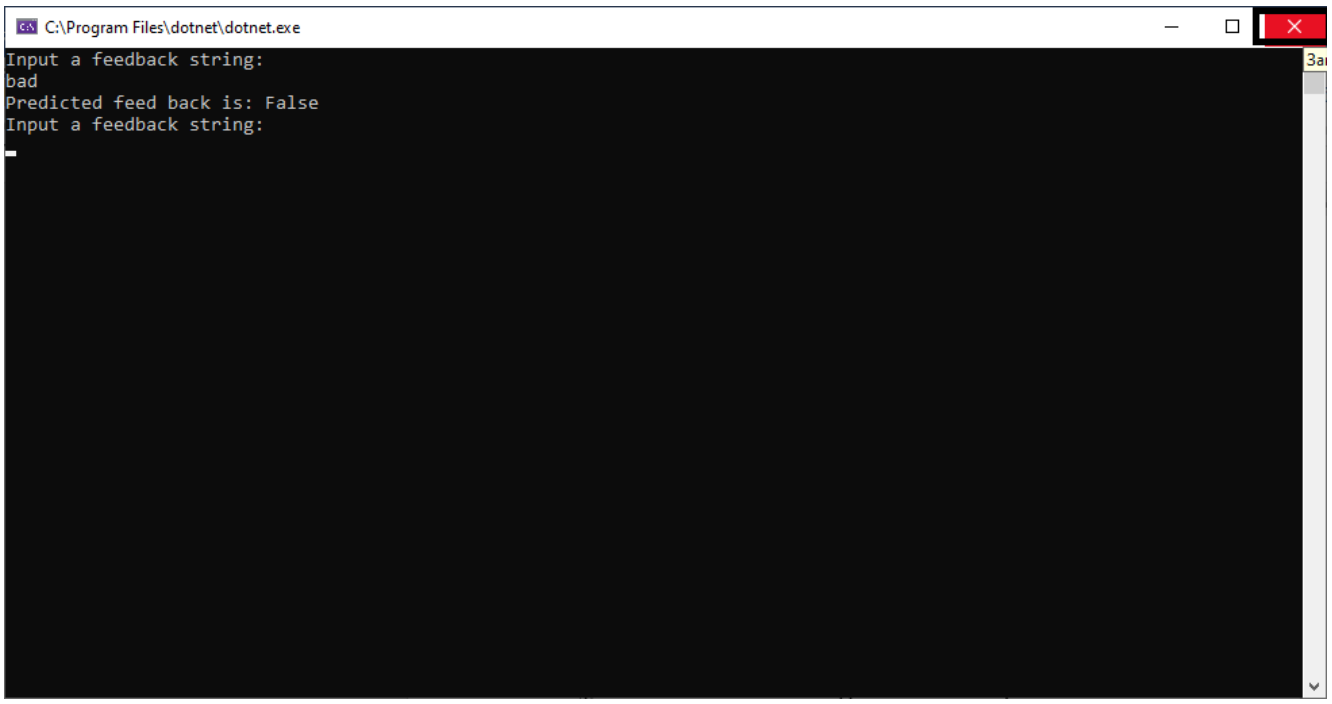


Рисунок 2.16 – Вихід із програми

2.7 Тестування роботи програмного модуля

Мета: перевірка якості розроблюваного програмного забезпечення на виконання технічного завдання, і призначений для вияву інформації про якість продукту відносно контексту, в якому він має використовуватись.

Рівень плану: Тест План (*Test Plan*).

Склад документа: опис тестувальних робіт, об'єкта тестування, стратегії тестування, розкладу, критеріїв початку і закінчення тестування.

План проекту: тестування програмного засобу як повноцінного продукту.

Опис тестованого продукту: Програмний засіб аналізу тексту з використанням штучного інтелекту. Програму написано мовою C# в середовищі Microsoft Visual Studio.

Відповідний стандарт: *IEEE 829-1998 Format*.

Тестові завдання. Тестування якості всього програмного продукту.

Повинно бути проведено тестування таких частин:

1. Програмний засіб.
2. Коректність вводу тексту.
3. Коректність аналізу тексту.
4. Коректність виводу результату аналізу.

До аспектів, що перевіряються, з точки зору користувачів, основні функції програмного засобу повинні тестуватися на:

1. Функціональні можливості (правильність, здатність до взаємодії).
2. Надійність (стабільність, стійкість до помилки).
3. Практичність (зрозумілість, простота використання).
4. Ефективність (характер зміни в часі, характер зміни ресурсів).
5. Супроводжуваність (аналізованість).
6. Мобільність (адаптованість, простота впровадження).

Нетестовані аспекти. Модулі, які рідко перевіряються користувачами:

1. Узгодженість.
2. Захищеність.

3. Відновлюваність.

4. Стійкість.

5. Взаємозамінність.

Підхід до тестування. Рівень тестування: системний, з точки зору кінцевого користувача.

Спеціальні засоби тестування: відсутні, тестування буде проводитися вручну.

Метрики: в рамках даного плану передбачається створити комплект тестів, повний щодо метрики за тестовими випадками (*Test Cases*) (відповідно до міжнародного стандарту *ISO 14598*).

Особливі вимоги до тестування: відсутні, тестування проводиться в звичайному режимі.

Сегмент компонентів: певний сегмент компонентів повинен бути протестований разом.

Обмеження для тестування: істотним обмеженням є проведення тестування вручну.

Рекомендована методика тестування: ручна, тому що зменшує матеріальні та програмні витрати на проведення тестування.

Критерії успішності та припинення тестування. Критерії успішності тестування:

1. система передається в експлуатацію, коли розроблений повний комплект тестів і всі розроблені тести виконуються без помилок;

2. вдале тестування на виправлену помилку при повторній передачі на тестування.

Критерії припинення тестування:

1. після виконання тестового сценарію.

2. після завершення кінцевого терміну перевірки (дедлайну).

3. програма має серйозні недоліки, що подальше тестування просто не має жодного сенсу.

4. основні баги знайдені, шукати далі економічно не вигідно.

Тест кейси зображені на таблицях 2.1, 2.2, 2.3.

Таблиця 2.1 – Тест кейс для вводу тексту

Дія	**Очікуваний результат**	**Фактичний результат**
Передумова		
Відкрити програмний засіб	Програмний засіб аналізу тексту відкритий та доступний для використання	Пройдено
Кроки тесту		
Ввести текст	Текст успішно введено	Пройдено
Редагування тексту	Текст успішно редаговано	Пройдено
Постумова		
Завершити роботу програмного засобу	Робота програмного засобу завершена	Пройдено

Таблиця 2.2 – Тест кейс для основних функцій модуля

Дія	**Очікуваний результат**	**Фактичний результат**
Передумова		
Відкрити програмний засіб	Програмний засіб аналізу тексту відкритий та доступний для використання	Пройдено
Кроки тесту		
Виконати ввід тексту	Текст введено коректно	Пройдено

Виконати аналіз тексту	Аналіз тексту виконано	Пройдено
Вивід результату	Результат аналізу виведено успішно	Пройдено
Постумова		
Завершити роботу програмного засобу	Робота програмного засобу завершена	Пройдено

Таблиця 2.3 – Тест кейс додаткових функцій для користувача

Дія	**Очікуваний результат**	**Фактичний результат**
Передумова		
Відкрити програмний засіб	Програмний засіб аналізу тексту відкритий та доступний для використання	Пройдено
Кроки тесту		
Визначення позитивних відгуків	Позитивний відгук успішно визначено	Пройдено
Визначення негативних відгуків	Негативний відгук успішно визначено	Пройдено
Постумова		
Завершити роботу програмного засобу	Робота програмного засобу завершена	Пройдено

Висновком виконання даних тестових випадків (табл. 2.1, 2.2, 2.3) є те, що програмний засіб виконує відкриття вікна програми, дозволяє виконати ввід відгуків, виконує його аналіз, визначає чи відгук є позитивним чи негативним та виводить результат аналізу на екран.

3 Охорона праці

З розвитком науково-технічного прогресу важливу роль грає можливість безпечного виконання людьми своїх трудових обов'язків. У зв'язку з цим була створена і розвивається наука про охорону праці і життєдіяльності людини.

Безпека життєдіяльності (БЖД) - це комплекс заходів, спрямованих на забезпечення безпеки людини в середовищі проживання, збереження її здоров'я, розробку методів і засобів захисту шляхом зниження впливу шкідливих і небезпечних факторів до допустимих значень, вироблення заходів по обмеженню збитку в ліквідації наслідків надзвичайних ситуацій мирного і воєнного часу.

Охорона здоров'я трудящих, забезпечення безпеки умов праці, ліквідація професійних захворювань і виробничого травматизму складає одну з головних турбот людського суспільства. Звертається увага на необхідність широкого застосування прогресивних форм наукової організації праці, зведення до мінімуму ручної, малокваліфікованої праці, створення обстановки, що виключає професійні захворювання і виробничий травматизм.

Даний розділ дипломного проекту присвячений розгляду наступних питань:

1. визначення оптимальних умов праці техніка-програміста;
2. розрахунок освітленості;
3. розрахунок рівня шуму.

3.1 Характеристика умов праці програміста

Науково-технічний прогрес вніс серйозні зміни в умови виробничої діяльності робітників розумової праці. Їх праця стала більш інтенсивним, напруженим ділом, яке вимагає значних витрат розумової, емоційної і фізичної енергії. Це зажадало комплексного рішення проблем ергономіки, гігієни і організації праці, регламентації режимів праці та відпочинку.

В даний час комп'ютерна техніка широко застосовується у всіх областях діяльності людини. При роботі з комп'ютером людина піддається дії ряду небезпечних і шкідливих виробничих факторів: електромагнітних полів (діапазон радіочастот: ВЧ, УВЧ і СВЧ), інфрачервоного і іонізуючого випромінювань, шуму і вібрації, статичної електрики і ін.

Робота з комп'ютером характеризується значною розумовою напругою і нервово-емоційним навантаженням операторів, високою напруженістю зорової роботи і достатньо великим навантаженням на м'язи рук при роботі з клавіатурою ЕОМ. Велике значення має раціональна конструкція і розташування елементів робочого місця, що важливо для підтримки оптимальної робочої пози людини-оператора.

У процесі роботи з комп'ютером необхідно дотримуватись правильного режиму праці та відпочинку. В іншому випадку у персоналу наголошуються значна напруга зорового апарату з появою скарг на незадоволеність роботою, головні болі, дратівливість, порушення сну, втому і хворобливі відчуття в очах, в поясниці, в області шиї і руках.

3.2 Вимоги до виробничих приміщень

Забарвлення і коефіцієнти віддзеркалення. Забарвлення приміщень і меблів повинні сприяти створенню сприятливих умов для зорового сприйняття, гарного настрою.

Джерела світла, такі як світильники і вікна, які дають віддзеркалення від поверхні екрану, значно погіршують точність знаків і тягнуть за собою перешкоди фізіологічного характеру, які можуть виразитися в значній напрузі, особливо при тривалій роботі. Віддзеркалення, включаючи віддзеркалення від вторинних джерел світла, повинне бути зведено до мінімуму. Для захисту від надмірної яскравості вікон можуть бути застосовані штори і екрани.

Освітлення. Правильно спроектоване і виконане виробниче освітлення покращує умови зорової роботи, знижує стомлюваність, сприяє підвищенню продуктивності праці, благотворно впливає на виробниче середовище, надаючи

позитивну психологічну дію на працюючого, підвищує безпеку праці і знижує травматизм.

Недостатність освітлення приводить до напруги зору, ослабляє увагу, приводить до настання передчасної стомленості. Надмірно яскраве освітлення викликає засліплення, роздратування і різь в очах. Неправильний напрямок світла на робочому місці може створювати різкі тіні, відблиски, дезорієнтувати працюючого. Всі ці причини можуть призвести до нещасного випадку або профзахворювань, тому так важливий правильний розрахунок освітленості.

Існує три види освітлення - природне, штучне і поєднане (природне і штучне разом).

Природне освітлення - освітлення приміщень денним світлом, що потрапляє через світлові прорізи в зовнішніх огороджуючих конструкціях приміщення. Природне освітлення характеризується тим, що змінюється в широких межах залежно від часу дня, пори року, характеру області і ряду інших чинників.

Штучне освітлення застосовується при роботі в темний час доби і вдень, коли не вдається забезпечити нормовані значення коефіцієнта природного освітлення (похмура погода, короткий світловий день). Освітлення, при якому недостатнє за нормами природне освітлення доповнюється штучним, називається змішаним освітленням.

Згідно СНіП II-4-79 в приміщень обчислювальних центрів необхідно застосувати систему комбінованого освітлення.

При виконанні робіт категорії високої зорової точності (найменший розмір об'єкту розрізнення 0,3 ... 0,5 мм) величина коефіцієнта природного освітлення (КЕО) повинна бути не нижче 1,5%, а при зоровій роботі середньої точності (найменший розмір об'єкту розрізнення 0,5 ... 1,0 мм) КЕО повинен бути не нижче 1,0%. В якості джерел штучного освітлення звичайно використовуються люмінесцентні лампи типа ЛБ, або ДРЛ, які попарно об'єднуються в світильники, які повинні розташовуватися рівномірно над робочими поверхнями.

Вимоги до освітленості в приміщеннях, де встановлені комп'ютери, наступні: при виконанні зорових робіт високої точності загальна освітленість

повинна складати 300лк, а комбінована - 750лк; аналогічні вимоги при виконанні робіт середньої точності - 200 і 300лк відповідно.

Параметри мікроклімату. Параметри мікроклімату можуть мінятися в широких межах, у той час як необхідною умовою життєдіяльності людини є підтримка постійності температури тіла завдяки терморегуляції, тобто здатності організму регулювати віддачу тепла в навколишнє середовище. Принцип нормування мікроклімату - створення оптимальних умов для теплообміну тіла людини з навколишнім середовищем.

Обчислювальна техніка є джерелом істотних тепловиділень, що може привести до підвищення температури і зниження відносної вологості в приміщенні. У приміщеннях, де встановлені комп'ютери, повинні дотримуватися певні параметри мікроклімату. У санітарних нормах СН-245-71 встановлені величини параметрів мікроклімату, що створюють комфортні умови. Ці норми встановлюються в залежності від пори року, характеру трудового процесу і характеру виробничого приміщення (табл. 3.1).

Об'єм приміщень, в яких розміщені працівники обчислювальних центрів, не повинен бути меншим 19,5 м ^{3} / людини з урахуванням максимального числа одночасно працюючих в зміну. Норми подачі свіжого повітря в приміщення, де розташовані комп'ютери, приведені в табл. 3.2.

Таблиця 3.1 - Параметри мікроклімату для приміщень, де встановлені комп'ютери

Період року	**Параметр мікроклімату**	**Величина**
Холодний	Температура повітря в приміщенні Відносна вологість Швидкість руху повітря	22 ... 24 ° С 40 ... 60% до 0,1 м / с
Теплий	Температура повітря в приміщенні Відносна вологість Швидкість руху повітря	23 ... 25 ° С 40 ... 60% 0,1 ... 0,2 м / с

Таблиця 3.2 - Норми подачі свіжого повітря в приміщення, де розташовані комп'ютери

Характеристика приміщення	**Об'ємна витрата подається в приміщення свіжого повітря, м ^{3} / на одну людину в годину**
Об'єм до 20м^3 на особу 20 ... 40м ^{3} на особу Більш 40м ^{3} на особу	Не менше 30 Не менше 20 Природна вентиляція

Для забезпечення комфортних умов використовуються як організаційні методи (раціональна організація проведення робіт залежно від пори року і доби, чергування праці і відпочинку), так і технічні засоби (вентиляція, кондиціювання повітря, опалювальна система).

Шум і вібрація. Шум погіршує умови праці надаючи шкідливу дію на організм людини. Працюючі в умовах тривалої шумової дії викликають дратівливість, головні болі, запаморочення, зниження пам'яті, підвищену

стомлюваність, зниження апетиту, біль у вухах і т.д. Такі порушення в роботі ряду органів і систем організму людини можуть викликати негативні зміни в емоційному стані людини аж до стресових. Тривала дія інтенсивного шуму [вище 80 дБ (А)] на слух людини приводить до його часткової або повної втрати.

У табл. 3.3 вказані граничні рівні звуку залежно від категорії тяжкості і напруженості праці, що є безпечними відносно збереження здоров'я і працездатності.

Таблиця 3.3 - Граничні рівні звуку, дБ, на робочих місцях.

Категорія напруженості праці	**Категорія важкості праці**			
	I. Легка	**II. Середня**	**III. Важка**	**IV. Дуже важка**
I. Мало напружений	80	80	75	75
II. Помірно напружений	70	70	65	65
III. Напружений	60	60	-	-
IV. Дуже напружений	50	50	-	-

Ергономічні вимоги до робочого місця. Проектування робочих місць, забезпечених відеотерміналами, відноситься до числа важливих проблем ергономічного проектування в області обчислювальної техніки.

Робоче місце і взаємне розташування всіх його елементів повинне відповідати антропометричним, фізичним і психологічним вимогам. Велике значення має також характер роботи.

Ергономічними аспектами проектування відеотермінальних робочих місць, зокрема, є: висота робочої поверхні, розміри простору для ніг, вимоги до розташування документів на робочому місці (наявність і розміри підставки для

документів, можливість різного розміщення документів, відстань від очей користувача до екрану, документа, клавіатури і т.д.), характеристики робочого крісла, вимоги до поверхні робочого столу, регульованість елементів робочого місця.

Головними елементами робочого місця програміста є стіл і крісло. Основним робочим положенням є положення сидячи.

Оптимальне розміщення предметів праці і документації в зонах досяжності:

1. дисплей розміщається в зоні а (у центрі);
2. системний блок розміщається в передбаченій ніші столу;
3. клавіатура – у зоні г/д;
4. «миша» – в зоні в справа;
5. сканер в зоні а/б (зліва);
6. принтер знаходиться в зоні а (праворуч);
7. документація: необхідна при роботі – в зоні легкої досяжності долоні – в, а у висувних ящиках столу – література, невикористовувана постійно.

Для комфортної роботи стіл повинен задовольняти наступним умовам:

1. висота столу повинна бути вибрана з урахуванням можливості сидіти вільно, в зручній позі, при необхідності спираючись на підлокітники;
2. нижня частина столу повинна бути сконструйована так, щоб програміст міг зручно сидіти, не був змушений підбирати ноги;
3. поверхня столу повинна мати властивості, що виключають появу відблисків у поле зору програміста;
4. конструкція столу повинна передбачати наявність висувних ящиків (не менше 3 для зберігання документації, лістингів, канцелярських приналежностей);
5. висота робочої поверхні рекомендується в межах 680-760мм. Висота поверхні, на яку встановлюється клавіатура, повинна бути близько 650мм.

Велике значення надається характеристикам робочого крісла. Так, рекомендована висота сидіння над рівнем підлоги перебуває в межах 420-550мм. Поверхня сидіння м'яка, передній край закруглений, а кут нахилу спинки - регульований.

Велике значення також надається правильній робочій позі користувача. При незручній робочій позі можуть з'явитися болі в м'язах, суглобах і сухожиллях. Вимоги до робочої пози користувача відеотермінала наступні:

1. голова не повинна бути нахилена більш ніж на 20^0;

2. плечі повинні бути розслаблені;

3. лікті - під кутом 80 ° ... 100^0;

4. передпліччя і кисті рук - в горизонтальному положенні.

Причина неправильної пози користувачів обумовлена наступними чинниками: немає хорошої підставки для документів, клавіатура знаходиться дуже високо, а документи - низько, нікуди покласти руки і кисті, недостатній простір для ніг.

3.3 Розрахунок освітленості і рівня шуму

Розрахунок освітленості робочого місця зводиться до вибору системи освітлення, визначенню необхідного числа світильників, їхнього типу і розміщення. Виходячи з цього, розрахуємо параметри штучного освітлення.

Зазвичай штучне освітлення виконується за допомогою електричних джерел світла двох видів: ламп накалювання і люмінесцентних ламп. Використовуватимемо люмінесцентні лампи, які порівняно з лампами розжарювання мають ряд істотних переваг:

1. за спектральним складом світла вони близькі до денного, природного світла;

2. володіють більш високим ККД (у 1,5-2 рази вище, ніж ККД ламп розжарювання);

3. мають підвищену світловіддачею (в 3-4 рази вище, ніж у ламп розжарювання);

4. більш тривалий термін служби.

Розрахунок освітлення проводиться для кімнати площею 15м 2, ширина якої 5м, висота - 3 м. Для визначення кількості світильників визначається світловий потік, де

F - розраховується світловий потік, Лм;

E - нормована мінімальна освітленість, Лк (визначається за таблицею). Роботу програміста, відповідно до цієї таблиці, можна віднести до розряду точних робіт, отже, мінімальна освітленість E = 300лк;

S - площа освітлюваного приміщення (у нашому випадку S = 15м 2);

Z - відношення середньої освітленості до мінімальної (звичайно приймається рівним 1,1 ... 1,2, нехай Z = 1,1);

K - коефіцієнт запасу, враховує зменшення світлового потоку лампи в результаті забруднення світильників у процесі експлуатації (його значення залежить від типу приміщення й характеру проведених у ньому робіт і в нашому випадку K = 1,5);

n - коефіцієнт використання, (виражається відношенням світлового потоку, що падає на розрахункову поверхню, до сумарного потоку всіх ламп і обчислюється в частках одиниці; залежить від характеристик світильника, розмірів приміщення, фарбування стін і стелі, які характеризуються коефіцієнтами відображення від стін (Р С) і стелі (Р П)), значення коефіцієнтів Р С і Р П були зазначені вище: Р С = 40%, Р П = 60%. Значення n визначимо по таблиці коефіцієнтів використання різних світильників. Для цього обчислимо індекс приміщення:

S - площа приміщення, S = 15 м 2;

h - розрахункова висота підвісу, h = 2.92 м;

A - ширина приміщення, A = 3 м;

B - довжина приміщення, B = 5 м.

Знаючи індекс приміщення I, за таблицею 7 [23] знаходимо n = 0,22

Для освітлення вибираємо люмінесцентні лампи типу ЛБ40-1, світловий потік яких F = 4320 Лк.

Розрахуємо необхідну кількість ламп:

N - обумовлений число ламп;

F - світловий потік, F = 33750 Лм;

F л - світловий потік лампи, F л = 4320 Лм.

При виборі освітлювальних приладів використовуємо світильники типу ОД. Кожен світильник комплектується двома лампами.

Розрахунок рівня шуму.

Одним з несприятливих факторів виробничого середовища в ІОЦ є високий рівень шуму, створюваний друкованими пристроями, обладнанням для кондиціонування повітря, вентиляторами систем охолодження в самих ЕОМ.

Для вирішення питань про необхідність і доцільність зниження шуму необхідно знати рівні шуму на робочому місці оператора.

Рівень шуму, що виникає від декількох некогерентних джерел, що працюють одночасно, підраховується на підставі принципу енергетичного підсумовування випромінювань окремих джерел:

L_i - рівень звукового тиску і-го джерела шуму;

n - кількість джерел шуму.

Отримані результати розрахунку порівнюється з допустимим значенням рівня шуму для даного робочого місця. Якщо результати розрахунку вище допустимого значення рівня шуму, то необхідні спеціальні заходи щодо зниження шуму. До них відносяться: облицювання стін і стелі залу звукопоглинальними матеріалами, зниження шуму в джерелі, правильне планування обладнання і раціональна організація робочого місця оператора.

Рівні звукового тиску джерел шуму, що діють на оператора на його робочому місці представлені в табл. 3.4.

Таблиця 3.4 - Рівні звукового тиску різних джерел.

Джерело шуму	**Рівень шуму, дБ**
Жорсткий диск	40

Вентилятор	45
Монітор	17
Клавіатура	10
Принтер	45
Сканер	42

Зазвичай робоче місце оператора оснащене наступним обладнанням: вінчестер в системному блоці, вентилятор (и) систем охолодження ПК, монітор, клавіатура, принтер і сканер.

Підставивши значення рівня звукового тиску для кожного виду обладнання у формулу, отримаємо:

$L_{\Sigma} = 10 \cdot \lg (10^{4} + 10^{4,5} + 10^{1,7} + 10^{1} + 10^{4,5} + 10^{4,2)} = 49,5$ дБ

Отримане значення не перевищує допустимий рівень шуму для робочого місця оператора, рівний 65 дБ (ГОСТ 12.1.003-83). І якщо врахувати, що навряд чи такі периферійні пристрої як сканер і принтер будуть використовуватися одночасно, то ця цифра ще нижчою. Крім того при роботі принтера безпосередню присутність оператора необов'язково, тому що принтер обладнаний механізмом автоподачі аркушів.

3.4 Заходи та засоби протипожежного захисту

Під пожежною безпекою розуміють такий стан промислового або цивільного об’єкта, за якого з регламентованою ймовірністю виключається можливість виникнення і розвитку пожеж та впливу на людей небезпечних чинників пожежі, а також забезпечується захист матеріальних цінностей та довкілля.

Пожежна безпека об’єкта – доволі складне і багатоаспектне завдання, тому для його вирішення потрібно підходити комплексно. Комплекс заходів та засобів щодо пожежної безпеки складається із відповідних систем, зокрема:

1. системи запобігання пожеж, що містить підсистеми запобігання утворенню горючого середовища та виникненню в горючому середовищі джерела запалювання;

2. системи протипожежного захисту, що, своєю чергою, містить такі підсистеми: підсистема обмеження розвитку пожежі; підсистема забезпечення безпечної евакуації людей та майна; підсистема створення умов для успішного гасіння пожежі;

3. системи організаційно-технічних заходів, що передбачає організаційні, технічні, режимні та експлуатаційні заходи.

Організаційні заходи пожежної безпеки передбачають організацію пожежної охорони на об’єкті, проведення навчань з питань пожежної безпеки (інструктажі та пожежно-технічні мінімуми), застосування наочних засобів протипожежної пропаганди та агітації, організацію ДПД та ПТК, проведення перевірок, оглядів стану пожежної безпеки приміщень, будівель, об’єкта загалом та ін.

До технічних заходів належать суворе дотримання правил і норм, визначених чинними нормативними документами при реконструкції приміщень, будівель та об’єктів, технічному переоснащенні виробництва, експлуатації чи можливому переобладнанні електромереж, опалення, вентиляції, освітлення тощо.

Заходи режимного характеру передбачають заборону куріння та застосування відкритого вогню у недозволених місцях, недопущення появи сторонніх осіб у вибухонебезпечних приміщеннях чи об’єктах, регламентацію пожежної безпеки при проведенні вогневих робіт тощо.

Експлуатаційні заходи передбачають своєчасне проведення профілактичних оглядів, випробувань, ремонтів технологічного та допоміжного устаткування, а також інженерного господарства (електромереж, електроустановок, опалення, вентиляції).

Система запобігання пожежі – це комплекс заходів і технічних засобів, які запобігають виникненню пожежі.

Керівники підприємств повинні визначити обов'язки посадових осіб (у тому числі заступників керівника) з забезпечення пожежної безпеки, призначити відповідальних за пожежну безпеку окремих будівель, споруд, приміщень, дільниць, технологічного та інженерного обладнання, а також за зберігання та експлуатацію технічних засобів протипожежного захисту.

Обов'язки осіб, відповідальних за забезпечення пожежної безпеки, утримання та експлуатації засобів протипожежного захисту слід відобразити у відповідних документах.

Керівник підприємства зобов'язаний вживати (в межах наданих йому повноважень) відповідних заходів реагування на факти порушень чи невиконання іншими працівниками підприємства встановленого протипожежного режиму, вимог правил пожежної безпеки та нормативних актів, що діють у цій сфері.

Керівники підприємств повинні:

1. організувати розроблення комплексних заходів для забезпечення пожежної безпеки, впроваджувати на підприємстві досягнення науки і техніки, позитивний досвід;

2. відповідно до нормативних актів з пожежної безпеки розробляти і затверджувати положення, інструкції та інші нормативні акти, що діють у межах підприємства, здійснювати постійний контроль за їх додержанням;

3. забезпечувати додержання протипожежних вимог стандартів, норм, правил, а також виконання вимог приписів і постанов органів держпожнагляду;

4. організовувати навчання працівників правилам пожежної безпеки та пропаганду заходів для їх забезпечення;

5. в разі відсутності в нормативних актах вимог, потрібних для гарантування пожежної безпеки - вживати відповідних заходів, узгоджуючи їх з органами держпожнагляду;

6. тримати у справному стані засоби протипожежного захисту і зв'язку, пожежну техніку, обладнання та інвентар, не допускати їх використання не за призначенням;

7. створювати в разі потреби відповідно до встановленого порядку підрозділи пожежної охорони та потрібну для їх функціонування матеріально-технічну базу;

8. подавати на вимогу Державної пожежної охорони відомості та документи про стан пожежної безпеки підприємства (об'єкта) і продукції, яку підприємство виробляє;

9. вживати заходів з впровадження автоматичних засобів виявлення і гасіння пожеж та використання з цією метою виробничої автоматики;

10. своєчасно інформувати пожежну охорону про несправність пожежної техніки, систем протипожежного захисту, водопостачання, а також завчасно інформувати про закриття доріг і проїздів на своїй території;

11. проводити службове розслідування випадків пожеж.

До первинних засобів пожежогасіння належать:

1. вогнегасники;
2. пожежні крани-комплекти, ручні насоси
3. лопати, ломи, сокири, гаки, пили, багри;
4. ящики з піском, бочки з водою;
5. азбестові полотнища, повстяні мати та ін.

Первинні засоби пожежогасіння розміщують на пожежних щитах, які встановлюють на території об'єкта з розрахунку один щит на 5000 м^2. Вони мають бути пофарбовані у червоний колір, а пожежний інструмент у чорний.

Серед первинних засобів пожежогасіння найважливішу роль відіграють вогнегасники різних типів: водяні, водо-пінні, порошкові, вуглекислотні, газові.

Залежно від способу транспортування вони бувають: переносні (до 20 кг) та пересувні (до 450 кг).

Залежно від об'єму вогнегасники бувають малолітражні (до 5л), ручні (до 10 л), пересувні (понад 10л).

Вогнегасники маркують буквами, що означає їх вид та цифрами, що визначають їх об'єм.

Найбільш перспективними є порошкові вогнегасники, які застосовують для гасіння лужних металів, ЛЗР і ТР, електрообладнання, що горить під напругою до 1000В, твердих та газоподібних речовин.

Найбільш розповсюдженими є:

1. ОП-1, ОП-2, ОП-9, ОП-10 — переносні;
2. ОПА-50, ОПА-100 — пересувні.

Вони відрізняються між собою лише складом порошку та пристроєм для його подачі.

Вуглекислотні вогнегасники застосовуються для гасіння загорянь на машинах, автомобілях і для невеликих об'ємів нафтопродуктів, а також електроустановок під напругою до 1000В.

У корпусі вогнегасника міститься вуглекислий газ у рідкому стані під високим тиском 6мПа (ручні) і 15 мПА (переносні). У горловині балону змонтований спеціальний пусковий пристрій із сифонною трубкою, який приводиться у дію за допомогою вентильного або пістолетного пристрою. Виходячи з балону назовні, зріджений двооксид вуглецю перетворюється на снігоподібну масу за температури - 80°С.

Вибір типу вогнегасника визначається розмірами загоряння і можливих осередків пожеж.

Утворенню горючого середовища запобігають застосуванням герметичного виробничого устаткування, максимально можливою заміною в технологічних процесах горючих речовин та матеріалів негорючими, обмеженням кількості пожежо-, вибухонебезпечних речовин та матеріалів під час використання та зберігання, а також правильним їх розміщенням, ізоляцією горючого та вибухонебезпечного середовища, організацією контролю за складом повітря в приміщенні та контролю за станом середовища в апаратах, застосуванням робочої та аварійної вентиляції, відведенням горючого середовища в спеціальні пристрої та безпечні місця, застосуванням в установках з горючими речовинами пристроїв

від пошкодження та аварій, використанням інгібувальних (хімічно активні компоненти, що сприяють припиненню пожеж) та флегматизаційних (інертні компоненти, що роблять середовище негорючим) речовин.

Виникненню в горючому середовищі джерела запалювання запобігають використанням устаткування та пристроїв, при роботі яких не виникає джерел запалювання, використанням електроустаткування, що відповідає за досягнення класу пожежо- та вибухонебезпеки приміщеннями та зонами груп і категорій вибухонебезпечної суміші, виконанням вимог щодо сумісного зберігання речовин та матеріалів, використанням устаткування, що задовольняє вимоги електростатичної іскробезпеки, улаштуванням блискавкозахисту, організацією автоматичного контролю параметрів, що визначають джерела запалювання, використанням швидкодіючих засобів захисного вимкнення, заземленням устаткування, видовжених металоконструкцій, використанням при роботі з ЛЗР інструментів, що не допускають іскроутворення, ліквідацією умов для самоспалахування речовин і матеріалів, усуненням контакту з повітрям пірофорних речовин, підтриманням температури нагрівання поверхні устаткування, пристроїв, речовин та матеріалів, які можуть контактувати з горючим середовищем нижче гранично допустимої (80 %) температури займання.

У даному розділі дипломного проекту були викладені вимоги до робочого місця техніка-програміста. Створені умови повинні забезпечувати комфортну роботу. На підставі вивченої літератури з даної проблеми, були зазначені оптимальні розміри робочого столу і крісла, робочої поверхні, а також проведено вибір системи і розрахунок оптимального освітлення виробничого приміщення, а також розрахунок рівня шуму на робочому місці. Дотримання умов, що визначають оптимальну організацію робочого місця інженера - програміста, дозволить зберегти гарну працездатність протягом усього робочого дня, підвищить як в кількісному, так і в якісному відносинах продуктивність праці програміста, що в свою чергу сприятиме якнайшвидшій розробці і налагодженню програмного продукту.

Висновки

Технологія інтелектуального аналізу тексту в даний час широко застосовується до широкого кола урядових, дослідницьких та бізнес-потреб. Всі три групи можуть використовувати інтелектуальний аналіз тексту для управління документами та пошуку документів, що стосуються їх повсякденної діяльності.

Дипломний проект «Програмний модуль для аналізу тексту з використанням штучного інтелекту» створений для виконання аналізу коментарів на основі штучного інтелекту. Для розробки програмного модулю було використано мову програмування C#, інтегроване середовище розробки Microsoft Visual Studio.

Створений програмний засіб виконує аналіз кожного введеного коментаря та визначає, чи відноситься коментар до позитивних відгуків чи до негативних та виводить результат на екран.

Розробка проекту включає такі основні етапи, як створення алгоритмів програмного засобу, розроблено ряд діаграм, що доповнюють структурну схему та розкривають загальні принципи програмної реалізації як окремих блоків, так і засобу в цілому, визначення основних вимог, визначення основних засобів реалізації, розробка інструкції функціонування програмного засобу, ручне тестування, визначено системні вимоги до програмного засобу, .

Вимоги до робочого місця техніка-програміста описані у розділі «Охорона праці». У вимогах до виробничих приміщень були зазначені оптимальні розміри робочого столу і крісла, робочої поверхні, а також проведено вибір системи і розрахунок оптимального освітлення виробничого приміщення, а також розрахунок рівня шуму на робочому місці.

Програмний засіб в подальшому може бути вдосконалений шляхом додавання додаткових модулів, або інтеграції даного модуля в програмний засіб, а також вдосконаленням алгоритмів штучного інтелекту, реалізацією зручного інтерфейсу користувача та можливістю роботи із великими обсягами тексту.

Програмний модуль може використовуватися для навчальних та професійних цілей.

Перелік використаних джерел

1. Аббасов, И.Б. Двухмерное и трехмерное моделирование в *3ds MAX* / И.Б. Аббасов. - М.: ДМК, 2012. - 176 с.

2. Ганеев, Р.М. 3D-моделирование персонажей в *Maya*: Учебное пособие для вузов / Р.М. Ганеев. - М.: ГЛТ, 2012. - 284 с.

3. Климачева, Т.Н. *AutoCAD.* Техническое черчение и 3D-моделирование. / Т.Н. Климачева. - СПб.: BHV, 2008. - 912 с.

4. Пекарев, Л. Архитектурное моделирование в *3ds Max* / Л. Пекарев. - СПб.: BHV, 2007. - 256 с.

5. Петелин, А.Ю. 3D-моделирование в *Google Sketch Up* - от простого к сложному. Самоучитель / А.Ю. Петелин. - М.: ДМК Пресс, 2012. - 344 с.

6. Погорелов, В. *AutoCAD* 2009: 3D-моделирование / В. Погорелов. - СПб.: BHV, 2009. - 400 с.

7. Полещук, Н.Н. *AutoCAD* 2007: 2D/3D-моделирование. / Н.Н. Полещук. - М.: Русская редакция, 2007. - 416 с.

8. Тозик, В.Т. 3ds Max Трехмерное моделирование и анимация на примерах / В.Т. Тозик. - СПб.: *BHV,* 2008. - 880 с.

9. Швембергер, С.И. *3ds Max*. Художественное моделирование и специальные эффекты / С.И. Швембергер. - СПб.: *BHV,* 2006. - 320 с.

10. Еріх Гамма, Річард Хелм, Джон Вліссідес, Ральф Джонсон. Паттерны проектирования. – Питер, 2001, – 395 с.

11. А. Пол. Объектно-ориентированое программирование в действии.– СПБ.: Питер, 1997, 447 с.

12. Дж. Ликнесс Приложения для *Windows 8* на *C#* и *XAML*. – Питер, 2013, – 368 с.

13. Святослав Куликов – Тестирование программного обеспечения Базовый курс (2-е издание), 2018.

14. Сью Блэкман, *Beginning 3D Game Development with Unity/* Сью Блэкман; *Apress*, 2011, 992 с.

15. Фохт, Д. Проектирование и программная реализация систем на персональных ЭВМ/ Д. Фохт СПб.: БХВ - Петербург, 2005. - 96 с.

16. Крейтон, *P.X. Unity Game Development Essentials / P.X.* Крейтон *Packt Publishing*, 2010, 83 с.

17. Мартынов, Н.Н. *C#* для начинающих / Н.Н. Мартынов. - Москва; Кудиц-пресс, 2007. - 272 с.

18. Павловская, Т.А. *C#.* Программирование на языке высокого уровня / Т.А. Павловская - Спб; ПИТЕР, 2009 432 с.

19. Фленов, М. Библия *C#*/Фленов М. СПб.: БХВ - Петербург, 2011. -560с.

20. Фленов, М. *C#.* Секреты программирования / М. Фленов - СПб, ПИТЕР, 2006. - 457с

21. Дубовці В.А. Безпека життєдіяльності. / Учеб. посібник для дипломників. - К.: вид. Кірпи, 1992.

22. Мотузко Ф.Я. Охорона праці. - М.: Вища школа, 1989. - 336с.

23. Безпека життєдіяльності. / Под ред. Н.А. Бєлова - М.: Знання, 2000 - 364с.

24. Самгін Е.Б. Освітлення робочих місць. - М.: МІРЕА, 1989. - 186с.

25. Довідкова книга для проектування електричного освітлення. / Под ред. Г.Б. Кнорринга. - Л.: Енергія, 1976.

26. Боротьба з шумом на виробництві: Довідник / Є.Я. Юдін, Л.А. Борисов; За заг. ред. Є.Я. Юдіна - М.: Машинобудування, 1985. - 400с., Іл.Зінченко В.П. Основи ергономіки. - М.: МГУ, 1979. - 179с.

27. Методичні вказівки з виконання дипломної роботи (проекту) для студентів спеціальності 5.05010301«Розробка програмного забезпечення».

Додаток А – Код модуля MainWindow

```
using System;
using System.Collections.Generic;
using Microsoft.ML;
using Microsoft.ML.Data;
using Microsoft.ML.Runtime.Api;
using Microsoft.ML.Runtime.Data;

namespace CommentML
{
    class FeedBackTrainigData
    {
        [Column(ordinal: "0", name: "Label")]
        public bool IsGood { get; set; }

        [Column(ordinal: "1")]
        public string FeedBackText { get; set; }
    }

    class FeedBackPrediction
    {
        [ColumnName("PredictedLabel")]
        public bool IsGood { get; set; }
    }


    class Program
    {
        static List<FeedBackTrainigData> trainingData = new List<FeedBackTrainigData>();

        static List<FeedBackTrainigData> testData = new List<FeedBackTrainigData>();

        static void LoadTestData()

        {
            testData.Add(new FeedBackTrainigData()
            {
                FeedBackText = "Good",
                IsGood = true
```

```
});

testData.Add(new FeedBackTrainigData()
{
   FeedBackText = "not",
   IsGood = false
});

testData.Add(new FeedBackTrainigData()
{
   FeedBackText = "no",
   IsGood = false
});

testData.Add(new FeedBackTrainigData()
{
   FeedBackText = "no good",
   IsGood = false
});

testData.Add(new FeedBackTrainigData()
{
   FeedBackText = "not bad",
   IsGood = true
});

testData.Add(new FeedBackTrainigData()
{
   FeedBackText = "not nice",
   IsGood = false
});

testData.Add(new FeedBackTrainigData()
{
   FeedBackText = "sweet",
   IsGood = true
});

testData.Add(new FeedBackTrainigData()
{
   FeedBackText = "Bad",
   IsGood = false
```

```
});

testData.Add(new FeedBackTrainigData()
{
   FeedBackText = "nice and good",
   IsGood = true
});

testData.Add(new FeedBackTrainigData()
{
   FeedBackText = "Bad and horrible",
   IsGood = false
});

testData.Add(new FeedBackTrainigData()
{
   FeedBackText = "pretty good",
   IsGood = true
});

testData.Add(new FeedBackTrainigData()
{
   FeedBackText = "pretty bad",
   IsGood = false
});

testData.Add(new FeedBackTrainigData()
{
   FeedBackText = "awesome",
   IsGood = true
});

testData.Add(new FeedBackTrainigData()
{
   FeedBackText = "shit",
   IsGood = false
});

testData.Add(new FeedBackTrainigData()
{
   FeedBackText = "nice",
   IsGood = true
```

```
    });

    testData.Add(new FeedBackTrainigData()
    {
        FeedBackText = "shittiest shit",
        IsGood = false
    });

    testData.Add(new FeedBackTrainigData()
    {
        FeedBackText = "beautiful",
        IsGood = true
    });

    testData.Add(new FeedBackTrainigData()
    {
        FeedBackText = "ow so bad",
        IsGood = false
    });

    testData.Add(new FeedBackTrainigData()
    {
        FeedBackText = "average but ok",
        IsGood = true
    });

    testData.Add(new FeedBackTrainigData()
    {
        FeedBackText = "could be better",
        IsGood = false
    });
}

static void LoadTrainingData()

{
    trainingData.Add(new FeedBackTrainigData()
    {
        FeedBackText = "Good",
        IsGood = true
    });
```

```
testData.Add(new FeedBackTrainigData()
{
   FeedBackText = "no good",
   IsGood = false
});

trainingData.Add(new FeedBackTrainigData()
{
   FeedBackText = "not",
   IsGood = false
});

trainingData.Add(new FeedBackTrainigData()
{
   FeedBackText = "not bad",
   IsGood = true
});

trainingData.Add(new FeedBackTrainigData()
{
   FeedBackText = "not nice",
   IsGood = false
});

trainingData.Add(new FeedBackTrainigData()
{
   FeedBackText = "Bad",
   IsGood = false
});

trainingData.Add(new FeedBackTrainigData()
{
   FeedBackText = "nice and good",
   IsGood = true
});

trainingData.Add(new FeedBackTrainigData()
{
   FeedBackText = "Bad and horrible",
   IsGood = false
});
```

```
trainingData.Add(new FeedBackTrainigData()
{
   FeedBackText = "pretty good",
   IsGood = true
});

trainingData.Add(new FeedBackTrainigData()
{
   FeedBackText = "pretty bad",
   IsGood = false
});

trainingData.Add(new FeedBackTrainigData()
{
   FeedBackText = "awesome",
   IsGood = true
});

trainingData.Add(new FeedBackTrainigData()
{
   FeedBackText = "shit",
   IsGood = false
});

trainingData.Add(new FeedBackTrainigData()
{
   FeedBackText = "nice",
   IsGood = true
});

trainingData.Add(new FeedBackTrainigData()
{
   FeedBackText = "shittiest shit",
   IsGood = false
});

trainingData.Add(new FeedBackTrainigData()
{
   FeedBackText = "beautiful",
   IsGood = true
});
```

```
            trainingData.Add(new FeedBackTrainigData()
            {
                FeedBackText = "ow so bad",
                IsGood = false
            });

            trainingData.Add(new FeedBackTrainigData()
            {
                FeedBackText = "average but ok",
                IsGood = true
            });

            trainingData.Add(new FeedBackTrainigData()
            {
                FeedBackText = "could be better",
                IsGood = false
            });
        }

        static void Main(string[] args)
        {
            LoadTrainingData();

            var mlContext = new MLContext();

            IDataView dataView = mlContext.CreateStreamingDataView<FeedBackTrainigData>(trainingData);

            var pipeline = mlContext.Transforms.Text.FeaturizeText("FeedBackText", "Features").Append(mlContext.BinaryClassification.Trainers.FastTree(numLeaves: 50, numTrees: 50, minDatapointsInLeaves: 1));

            var model = pipeline.Fit(dataView);

            LoadTestData();

            IDataView dataView1 = mlContext.CreateStreamingDataView<FeedBackTrainigData>(testData);

            var predictions = model.Transform(dataView1);

            var metrics = mlContext.BinaryClassification.Evaluate(predictions, "Label");

            Console.WriteLine(metrics.Accuracy);
```

```
            while (true)
            {
                Console.WriteLine("Input a feedback string: ");

                string feedBackStr = Console.ReadLine().ToString();

                var PredFunc = model.MakePredictionFunction<FeedBackTrainigData,
FeedBackPrediction>(mlContext);

                var feedBackInput = new FeedBackTrainigData();

                feedBackInput.FeedBackText = feedBackStr;

                var feedBackPredicted = PredFunc.Predict(feedBackInput);

                Console.WriteLine("Predicted feed back is: " + feedBackPredicted.IsGood);
            }
        }
    }
}
```