Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка
Навчально-науковий інститут філології
Кафедра української мови та прикладної лінгвістики

Автоматичне визначення стилю тексту

Кваліфікаційна робота
освітнього ступеня «бакалавр»
студентки 4 курсу
освітньої програми
«Прикладна (комп'ютерна) лінгвістика та англійська мова»,
спеціальність – 035 Філологія (035.10 Прикладна лінгвістика)
Ксенія Олегівна СИКЛИТЕНКО

Науковий керівник:
д.філол.н., проф. Наталія ДАРЧУК

«Допущено до захисту»
Протокол засідання
кафедри української мови та прикладної лінгвістики
протокол № 15 від «06» 06 2024 року
завідувач кафедри _____(підпис)
к.філол.н., доц. Сергій РІЗНИК

Київ
2024

Анотація

Темою кваліфікаційної роботи бакалавра є “Автоматичне визначення стилю тексту”. Дослідження з цієї теми є актуальним, оскільки автоматизація процесів класифікації та аналізу текстів дозволяє зменшити витрати часу та ресурсів, підвищити точність і ефективність роботи з великими обсягами текстової інформації. Створення комп’ютерного інструменту для автоматичного стилістичного аналізу текстів має практичне значення для: збирання текстових даних різного обсягу у корпуси та навчальні набори; обробки текстових матеріалів для різних академічних установ; досліджень у академічних сферах. Метою дипломного проєкту є створення нейронної мережі для автоматичного визначення стилю тексту. Об'єкт дослідження – українськомовні тексти різних стилів. Предмет дослідження полягає у визначенні ефективної архітектури нейронної мережі та методів обробки даних для стильової диференціації.

Вступ надає інформацію про актуальність, мету, предмет та об’єкт дослідження. Також в ньому представлені завдання та методи, застосовані у дипломному проєкті.

Перший розділ складається з 10 підрозділів, вони містять аналіз теоретичної літератури, що стосується стилістики та машинного навчання;

Другий розділ містить 8 підрозділів, які демонструють практичне застосування отриманих теоретичних знань, а саме: розробку програми для створення та обробки датасету, розробку програми для навчання нейронної мережі та розробку програми для тестування нейронної мережі.

Розділ «Висновки» містить стислий виклад реалізації мети дослідження, підрахунки використаних даних та отриманих результатів, оцінювання результатів дослідження.

У результаті було створено рекурентну нейронну мережу, що є ефективним інструментом для автоматичного визначення стилю тексту. Отримані результати продемонстрували, що розроблена нейронна мережа може ефективно класифікувати тексти, що не мають ознак інших стилів, а також ті тексти, що мають ознаки різних стилів.

Summary

The topic of the bachelor's thesis is "Automatic text style detection". The research on this topic is relevant because automation of text classification and analysis processes allows to reduce time and resources, increase the accuracy and efficiency of working with large amounts of textual information. The creation of a computer tool for automatic stylistic analysis of texts is practically important for: collecting textual data of various sizes into corpora and training sets; processing textual materials for various academic institutions; research in academic fields. The aim of the diploma project is to create a neural network for automatic text style detection. The object of research is Ukrainian texts of different styles. The subject of the research is the determination of the effective structure of the neural network and data processing methods for stylistic differentiation.

The introduction provides information about the relevance, purpose, subject and object of the study. It also presents the tasks and methods used in the thesis project.

The first section consists of 10 subsections, which include an analysis of theoretical literature related to stylistics and machine learning;

The second section contains 8 subsections that demonstrate the practical application of the theoretical knowledge gained, namely: development of a program for creating and processing a dataset, development of a program for training a neural network, and development of a program for testing a neural network.

The “Conclusions” section contains a brief summary of the research objective, calculation of the data used and the results obtained, and evaluation of the research results.

As a result, a recurrent neural network has been created, due to the efficiency of automatic text style detection. The obtained results demonstrated that the developed neural network can effectively classify texts that do not have features of other styles, as well as those texts that have features of different styles.

ЗМІСТ

ВСТУП

Об’єм текстової інформації збільшується кожного дня, тож існує попит на створення нових комп’ютерних інструментів, здатних класифікувати та упорядковувати текстовий матеріал. Створення автоматичного інструменту для визначення стилю тексту допоможе заощадити час у: збиранні текстових даних різного обсягу у корпуси або навчальні набори; обробці текстових матеріалів для таких установ, як архіви; дослідженнях у академічних сферах, особливо у сфері стилістики, авторизації тексту.

У галузі обробки природної мови, нейронні мережі демонструють високу точність через свою здатність враховувати складні залежності. Вони здатні навчитися розпізнавати не тільки сталі та конкретні характеристики, а й специфічні мовні випадки через їхнє вміння встановлювати контекст текстових даних.

Мета дипломного проєкту - створити нейронну мережу, що буде автоматично визначати стиль тексту.

Завдання дослідження:

- опрацювати теоретичну літературу, що допоможе визначити відмінні характеристики наукового, офіційно-ділового та художнього стилів українськомовних текстів;
- опрацювати теоретичну літературу, яка допоможе ознайомитися з найефективнішим типом нейронної мережі та підходом до її створення для здійснення мети дослідження;
- з Корпусу української мови, витягнути необхідні тексти, що представлені трьома стилями української мови;
- створити навчальний набір для нейронної мережі на основі отриманих із корпусу текстів;
- використовуючи мову програмування Python, розробити нейронну мережу для автоматичного визначення стилів українськомовних текстів;
- провести навчання нейронної мережі на основі зібраного датасету;
- протестувати функціонування та результати нейронної мережі;

- використати готову нейронну мережу, яка була навчена та протестована, для обробки шістьох текстів, що представляють три функціональні стилі української мови.

Об'єкт дослідження – українськомовні тексти різних стилів. **Предмет** дослідження полягає у визначенні ефективної архітектури нейронної мережі та методів обробки даних для стильової диференціації.

Методи дослідження: класифікація, аналіз, експеримент.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ЗАСАДИ АВТОМАТИЧНОГО СТИЛЮ УКРАЇНСЬКОМОВНИХ ТЕКСТІВ

У дипломній роботі будуть досліджуватимуться три функціональні стилі, а саме: науковий, офіційно-діловий та художній.

В українській мові виокремлюють п’ять основних функціональних стилів: науковий, публіцистичний, художній, розмовний та офіційно-діловий. Функціональний стиль — це система засобів літературної мови, що представляє окрему сферу мовної взаємодії на лексичному, морфологічному та синтаксичному рівнях. Функціональний стиль використовується для забезпечення функцій мови та мовлення — спілкування, повідомлення та впливу [14, с. 6]. П. Дудик також виділяв два додаткові стилі - епістолярний та конфесійний [21, с. 8]. Деякі науковці ставляться досить скептично до самостійності цих двох стилів та вважають їх підстилями основних стилів [22, 23]. Функціональний стиль є аналогічним до поняття стиль, але слід зазначити, що стиль — це об’єктивне поняття, що усвідомлено всіма носіями літературної мови за замовчуванням [7, с. 12-13].

Теоретичні засади для автоматичного стилістичного аналізу українськомовних текстів базуються на двох головних принципах:

- у встановленні частотних та функціональних характеристик трьох функціональних стилів української мови;
- у застосуванні найефективніших підходів до створення нейронної мережі, яка здійснюватиме лінгвостилістичну класифікацію.

За допомогою лінгвістичного аналізу необхідно встановити основні дистинктивні характеристики наукового, офіційно-ділового та художнього стилів та вибрати тип нейронної мережі, що може найефективнішим чином класифікувати тексти за стилями.

1.1. Науковий стиль

Науковий стиль — це різновид літературної мови, що використовується у технічних та наукових галузях і виконує функцію повідомлення. Цьому стилю притаманна логічність, об’єктивність та однозначність. Тексти містять наукову

інформацію, тому в них наявні такі явища як синтез, аналіз, доказова база, аргументація та висновки [21, с. 56].

Науковий стиль має такі підстилі та жанри (За Т. Б. Маслова) [3, с. 24]:

1. власне науковий — орієнтований на фахівців окремої галузі: монографія, дисертація, наукова стаття, автореферат, рецензія;
2. науково-інформативний - опрацювання результатів наукових досліджень: реферат, анотація, тези, конспект, коментар, довідка;
3. науково-довідниковий — притаманна висока точність та доступність наукової інформації: словник, енциклопедія, довідник;
4. науково-популярний — орієнтований на широку аудиторію, у тому числі неспеціалістів, для заохочення та посилення уваги нефахівців до науки: журнальна стаття, нарис, книга, повідомлення наукового змісту в ЗМІ;
5. науково-навчальний — спрямований на набуття знань та вмінь: підручник, тренажер, практикум, навчально-методичний посібник, лекція, реферат, семінарська доповідь;
6. науково-технічний — орієнтований на спеціалістів технічних галузей: технічна документація, інструкція, науково-технічна реклама, патент.

Науковий стиль має чіткий та відмінний від інших стилів лексичний склад. Науковим текстам притаманна термінологічна лексика. Дудик П. С. розмежовував в лексиці два види: терміни, що належать окремим науковим галузям (підмовам) та загальнонаукові терміни [21, с. 76]. Слід відзначити, що терміни зазвичай походять від іншомовних слів, їх називають інтернаціоналізмами [14, с. 71]. Тому у текстах наукового стилю часто можна зустріти у дужках, поряд із терміном, написаним українською мовою, його відповідник мовою оригіналу, зазвичай грецькою або латинською. З власних спостережень, грецькі та латинські терміни мають високу частоту вживання у галузях біології та медицини. У всіх підмовах наукового стилю слова, як правило, вживаються тільки у прямому значенні, проте у науково-популярному підстилі досить часто зустрічаються у переносних значеннях для більш ефективного донесення специфічної інформації широкій аудиторії. Окремі

автори монографій, наукових статей, дисертацій також іноді використовують метафори та фразеологізми для наведення аналогій, зіставляючи одні явища з іншими або для позначаючи об'єкти за допомогою використання образних засобів [21, с. 36].

На морфологічному рівні, науковий стиль також має свої особливості. По суті, морфологічні особливості є наслідком специфічності лексичного складу текстів. У наукових текстах найчастотнішою частиною мови є іменник через термінологічність лексики. Як правило, у науковому стилі іменники демонструють абстрактність лексичного складу тексту та найчастіше мають суфікси: -ізм, -изм, -от(а). Також високочастотними є субстантивовані частини мови з метою вираження опредметненості дії та ознаки [17, с. 36]:

1. віддієслівні іменники, що мають суфікси -анн(я), -енн(я), -інн(я), -тт(я);
2. відприкметникові іменники, що мають суфікси -ість, -ин(а), -изм(-ізм).

Серед іменних частин мови прикметники зазвичай посідають друге місце за частотою вживання у текстах наукового стилю [17, с. 33]. Іменники представляють об'єкти, а прикметники описують властивості або характеристики цих об'єктів. Прикметники можуть стосуватися як конкретної галузі науки, так і позначати більш загальні характеристики. Можна виділити такі найчастотніші лексико-семантичні групи вживання прикметників:

1. ті, що стосуються конкретної наукової галузі: *бронхіальний, вегетативний, міжклітинний* тощо;
2. ті, що описують розмір, форму, стан та характеристики об'єкту: *великий, круглий, критичний, первинний* тощо;
3. ті, що порівнюють/зіставляють явища або результати досліджень між собою: *менший, більший* тощо;
4. ті, що описують міру та ступінь об'єкту: *високий, максимальний* тощо;
5. ті, що стосуються оцінки результатів дослідження або важливості досліджуваних явищ: *задовільний, важливий* тощо.

У наукових текстах дієслова, як правило, використовуються з метою називання дій або станів. Можна відмітити високу частоту вживання зворотних

дієслів у пасивному стані та безособових дієслів на -но, -то. Науковий матеріал не викладається від першої особи однини, його замінюють першою або третьою особою в множині [12, с. 84]. Вважається, що таким чином науковий текст сприйматиметься об’єктивно. Тому на синтаксичному рівні, у текстах наукового стилю переважають безособові та неозначено-особові речення, а також складнопідрядні реченням з причинно-наслідковими зв'язками, які найбільш відповідають потребам наукового викладу [14, с. 9-10]. Необхідно відзначити, що досить часто зустрічаються складнопідрядні речення з підрядним умови, зазвичай вони мають таку конструкцію: сполучник "якщо" у підрядній частині речення та частка "то" у головній. Також слід включити такі синтаксичні особливості наукового стилю:

1. висока частота вживання складених іменних присудків, де іменними частинами, в основному, виступають іменники та дієприкметники;
2. однорідні члени речення;
3. віддієслівні іменники – основний спосіб вираження підмета;
4. частотне вживання моделі словосполучення – іменник у називному відмінку + іменник у родовому відмінку [13, с. 27].

1.2. Офіційно-діловий стиль

Офіційно-діловий стиль – це різновид літературної мови, що використовується для забезпечення ділового спілкування на державному, законодавчому, дипломатичному, політичному, громадському, адміністративному рівні. Він виконує інформативну функцію. Найголовніша його відмінність від інших функціональних стилів – чітка структура тексту, яка позбавлена емоційності та будь-якого вияву індивідуальності автора. Як правило, має форму документів [21, с. 69].

Виділяють такі підстилі та жанри офіційно-ділового стилю (за Кузнецова Г. П.) [8, с. 128]:

1. законодавчий – регулює відносини приватних або службових осіб, громадян, держави, приватних та державних інституцій на законодавчому

рівні. Використовується у таких жанрах: закон, законопроект, нормативний акт, указ, статут;

2. юридичний – застосовується у юриспруденції: судочинстві, арбітражі, нотаріальній діяльності тощо. Цей стиль регулює правові та конфліктні відносини між різними суб'єктами, зокрема державою, підприємствами, організаціями та приватними особами. Виділяються такі жанри: заява, постанова, протокол, вирок, акт, судовий виклик (повістка у суд), запит;
3. дипломатичний – стосується ділових відносин між державами у сферах політики, економіки, освіти та культури. Найтиповіші жанри: угода, конвенція, пакт, декларація, дипломатична нота, меморандум;
4. адміністративно-канцелярський – функціонує у трудовій, підприємницькій сфері та діловодстві. В текстах цього стилю регулюються трудові/ділові відносини між підприємствами, структурними підрозділами, державними установами, а також відносини між приватними особами та організаціями. Представлений такими жанрами: контракт, залікова книжка, довідка, розпорядження тощо.

На лексичному рівні, офіційно-діловий стиль суттєво відрізняється від інших стилів. Йому притаманне обов’язкове дотримання єдиної структури тексту, адже неправильно оформлений документ вважається недійсним. Офіційно-діловий стиль має свою термінологію, яку не використовують інші стилі через її специфіку та однозначність. Наприклад, більшість жанрів офіційно-ділового стилю апелюють законами (Конституцією), тому одне з найчастотніших слововживань — "стаття". Відповідно до правил оформлення офіційно-ділової документації, у кожному типі документів, як правило, повинна бути дата створення або схвалення документу. Місяць записується саме літерами. Також одна з головних ознак офіційно-ділового стилю — встановлення часових меж. Майже у кожному офіційному документі буде згаданий часовий термін для виконання обов’язків: місяць, години. Всі слова, без виключень, повинні бути вжиті тільки у одному, прямому значенні. Офіційно-діловий стиль є єдиним функціональним стилем, що вимагає

дотримання мовних норм, іншими словами використання канцеляризмів, майже у всіх жанрах. Відхилення від мовних норм є неприпустимим, адже саме вони надають тексту нейтральності та офіційності.

На морфологічному рівні, офіційно-діловий стиль є подібним до наукового. Обом стилям притаманна термінологічність, а отже найчастотнішою частиною мови є іменник. Віддієслівні абстрактні іменники із суфіксами -ання присутні майже у кожному реченні. Відприкметникові та абстрактні іменники із суфіксом -ість також мають високу частоту вживання у текстах. Іменниковість законодавчих та юридичних текстів також зумовлена тим, що у текстах офіційно-ділового стилю часто згадуються:

1. назви посад;
2. прізвища, імена, імена по-батькові;
3. ідентифікація людей за статтю, громадянством, статусом (чоловік/жінка, юридична/фізична особа, особа, громадяни/іноземні громадяни тощо);
4. назва місяцю, записана літерами.

У офіційно-діловому тексті жодне речення не обходиться без дієслова. Найчастотнішою формою дієслова є форма теперішнього часу, що не позначає процес або певний проміжок часу, а встановлює постійні характеристики чи ознаки явищ [9, с. 152]. Інфінітиви та безособові форми дієслів на -но, -то є також типовими для цього стилю.

Прикметники є менш вживаними, ніж іменники та дієслова. У офіційно-діловому стилі використовуються всі граматичні категорії прикметників: відносні, якісні та присвійні. Однак найбільш поширеними є відносні прикметники, які підкреслюють сталість характеристик іменників у відношенні до інших предметів або дій [9, с. 144]. Наприклад: *фіксований, підприємницький, міжнародний, освітній, адміністративний, податковий*. Примітним є те, що віддієслівні прикметники, які часто зустрічаються у текстах офіційно-ділового стилю та виконують такі функції в тексті (за Кузнецова Г. П.) [9, с. 145]:

1. позначають ознаки осіб, що виконують дію (мають суфікси -льн(ий), -ч(ий), -івн(ий)): *накопичувальний, виборчий, керівний* тощо;
2. позначають ознаки осіб, що є об'єктами дії (мають суфікси -н(ий), -овн(ий), -енн(ий)): *визначений, основний, денний* тощо;
3. дійові ознаки осіб, які не поширюються на інші об’єкти (мають суфікси -лив(ий), -к(ий)): *сприятливий, короткий* тощо.

В усіх підстилях вживаються числівники, як у форматі цифр, так і в текстовому вигляді. Запис числівників текстовим чином майже завжди стосується встановлення номеру абзацу та частини конкретної статті. У законодавчому, юридичному, іноді в адміністративно-канцелярському підстилях апелюють статтями і їхніми номерами, записаними у вигляді цифр. Як було вже зазначено, у кожному типі документів, як правило, повинна бути дата створення цього текстового документу. У законах і кодексах в обов’язковому порядку встановлюється від якого числа, місяця та року окремий закон був схвалений. Якщо це стосується, наприклад, законопроекту, слід встановити число, місяць та рік, коли він повинен набрати чинності.

На синтаксичному рівні науковий та офіційно-діловий стилі також мають подібні риси. Офіційно-діловому стилю, як і науковому, притаманні безособові або неозначено-особові речення. Цьому стилю не притаманна емоційність, тож питальні та окличні речення відсутні майже у всіх жанрах. Типовим є вживання розщеплених присудків, що мають таку модель: дієслово + іменник (*контролювати - здійснювати контроль; погодитися - надавати згоду* тощо) [9, с. 168]. Часто спостерігається означення перед іменником, виражене відносним прикметником, що має пряме відношення до іменника та надає йому сталу ознаку: *надати правову допомогу, викладати у письмовій формі* тощо. Щодо дієслова, він часто має форму інфінітива. Складені дієслівні присудки є також досить частотним явищем для цього стилю, зокрема у моделях:

1. допоміжне дієслово + безособова форма дієслова на -но, -то;
2. дієслово-зв’язка + пасивний дієприкметник із суфіксами -н(ий), -ен(ий), -єн(ий), -т(ий) [9, с. 154];

3. допоміжне дієслово + інфінітив.

Можна також часто зустріти означення, виражені дієприкметниковими зворотами. Загалом, це стосується строгих мовних норм, іншими словами кліше, які необхідно використовувати відповідно до виду документу/жанру офіційно-ділового стилю: діючи в межах; обґрунтовуючи висновки; здійснюючи правосуддя тощо. Для синтаксису офіційно-ділового стилю типовим є вживання однорідних членів речення. Вони сприяють розширенню інформації про об'єкти, дозволяють детально класифікувати та уточнювати поняття. Найпоширеніший тип речень є складнопідрядні з підрядним означальним [16, с. 183]. Підрядні означальні частини додаються до головних частин за допомогою сполучників і сполучних слів, таких як "який", "що", "коли" [16, с. 183]. Також частотними можна вважати складні речення з підрядними умови, до головного речення додаються за допомогою сполучника "якщо".

1.3. Художній стиль

Художній стиль — це літературна мова, що використовується у творчих працях, виконуючи функцію впливу, адже впливає на психіку людини, як негативним, так і позитивним чином, здатний формувати ідеологічні та естетичні погляди. Цей стиль вважається особливим та найпоширенішими з усіх функціональних стилів української мови. Він вбирає у себе характеристики кожного стилю, він не має, по суті, строгої структури тексту чи вимог. Художній стиль можна вважати основою для усіх інших стилів, адже він і є літературною мовою, яку використовують всі інші стилі, тільки у відповідній, до їхніх вимог, формі.

Художній стиль має чотири підстилі та досить широку низку жанрів [4, с. 54-55]:

1. епічний (проза) — розповідні твори, в якому зображують побут, різні життєві обставини, ситуації та події. Поділяється на такі жанри: оповідання, роман, повість, новела, казка тощо;

2. ліричний (поезія) — емоційні твори, що наштовхують на роздуми про сенс людського існування через зображення переживань. Цей підстиль має такі жанри: вірш, поема, дума, сонет, пісня, ода тощо;
3. драма — літературний твір, здебільшого у формі п’єси, що передає емоції, конфлікти та проблеми через діалоги між особами. Можна виділити такі жанри: комедія, трагедія, трагікомедія, мелодрама тощо;
4. комбінований — твір, що поєднує елементи трьох інших підстилів, створюючи суміжні жанри. Найпоширеніші жанри цього підстилю: роман у віршах, драма-феєрія, усмішка, ліро-епічний твір.

У дипломній роботі передбачений аналіз тільки прозових творів.

В художньому стилі дуже широкий лексичний та фразеологічний склад. Йому притаманна динамічність лексики. Слова вжиті не тільки в прямому значенні, їм притаманна багатозначність. Оскільки два попередніх стилі мали досить строгу структуру тексту, художні тексти не мають відповідної структури. Хоч у художньому стилі важко визначити особливості на лексичному рівні, адже він охоплює широку низку тем, все ж таки можна виділити більш частотні характеристики, ніж у інших стилів, зокрема: широкий вжиток метафор та інших художніх засобів, фразеологізмів, власних імен та назв, застарілої лексики (архаїзмів, історизмів, старослов’янська лексика), неологізмів, діалектизмів, емоційно-експресивної лексики, варваризмів тощо.

Прозовим творами притаманний опис явищ, що оточують людину або мають відношення до об’єкту, тому найчастотнішими частинами мови вважаються прикметники, прислівники та дієслова. Іменник також є поширеною частиною мови у тексті, проте не найхарактернішим для цього стилю. Іменник апелює поняттями, тому здебільшого він характерний для наукового та офіційно-ділового стилів. Проте є закономірність вживання певних лексико-семантичних груп, частотних тільки в художньому стилі:

1. позначення особи, що стосуються ідентифікації її статі, соціального класу, характеристик зовнішності, вчинків у минулому: *дівчина/хлопець, багатій/бідняк, білявка/красуня, гуляка/крадій* тощо;

2. власні назви для позначення імен, прізвищ, прізвиськ, назв місць, міст та інших географічних об’єктів: *Марія/Олекса, Павленко/Кайдаш, Шпала (стосується людей високого зросту) /Макароніна (стосується людей, які мають кучеряве волосся), Хрещатик/Старовокзальна, Чернігів/Семигори, Дніпро/Феофанія* тощо;
3. загальні назви для опису фізичних об’єктів та місцевості: *базар, берег* тощо;
4. збірні іменники використовують для позначення сукупності об’єктів, що є однаковими або подібними: *хлопці/дівчата, містяни* тощо;

Окремо слід відмітити іменники, що надають емоційного забарвлення тексту. Вони відображають суб’єктивне ставлення до об’єктів. Як правило, ці іменники класифікують так:

1. зменшено-пестливі. З метою ідентифікації розміру чи віку об’єкту та демонстрації як позитивного ставлення до об’єкту, так і зневажливого: *дівчинка, хлопчик, дідусь вітерець* тощо;
2. збільшено-згрубіле. Як правило, використовують для гіперболізації певного явища або для демонстрації негативного ставлення до об’єкту: *дівчисько, хлопчисько, дідуган, вітрище* тощо.

Висока частотність прикметників та прислівників зумовлюється широким використанням епітетів та характерним явищем для художніх творів — описовості. Епітети розкривають ознаки об’єкта через прикметники та прислівники, що характеризують явища емоційним, образним та суб’єктивним чином. В той же час для опису подій чи оточення, як правило, використовують більш нейтральні та фактичні прикметники і прислівники. Як і у випадку з іменниками, ці частини мови мають зменшено-пестливі та збільшено-згрубілі форми, що зустрічаються у художніх творах дуже часто. Збільшено-згрубілі форми прикметників та прислівників часто зустрічаються у гіперболах. Слід також відзначити широке використання ступенів порівняння прикметників та прислівників простої форми, адже вони також здатні додати відтінок суб’єктивності та емоційності художнім творам [14, с. 152-153].

У художньому стилі дієслова, здебільшого, є динамічними, підкреслюють процесуальність [14, с. 157]. На відміну від інших двох стилів, у прозі використовують особові дієслова та дієслова наказового способу. Останній тип дієслова є характерним для цього стилю саме через наявність діалогів або емоційного монологу у художніх творах. Художньому стилю притаманні дієслова усіх трьох часів: теперішнього, минулого і майбутнього. Проте характерне явище описовості для художнього стилю передають дієслова теперішнього часу, що описують події в минулому [14, с. 159].

Щодо службових частин мов, необхідно відзначити використання прийменників. Прийменники є однаково високочастотними для кожного функціонального стилю української мови, адже вони забезпечують логічні зв'язки між іншими частинами мовами у тексті. Проте у художньому стилі вони так часто використовуються також для опису місцевості та оточення в цілому.

На синтаксичному рівні художній стиль відрізняється, знову ж таки, через свою динамічність, нестабільність та емоційність. Якщо для наукового та офіційно-ділового стилів характерні тільки розповідні та неокличні речення, то художньому стилю притаманні всі види речень: за метою висловлення — питальні, розповідні та спонукальні речення; за емоційним забарвленням — окличні та неокличні. Речення, здебільшого, означено-особові та узагальнено-особові. Підмет часто виражений займенником або власним іменником. Означення, здебільшого, виражені прикметниками, але також зустрічаються дієприкметники, що виконують функцію саме означення, а не присудка. У прозовій творчості можна спостерігати наявність неповних речень через діалогічність [14, с. 168]. У художньому стилі, особливо у прозових творах, висока частотність залучення безсполучникових речень, вони вважаються оптимальним типом зв'язку для описовості місць або спогадів [11, с. 153]. Висока частота використання безсполучникового типу зв'язку може бути зумовлена вживанням прямої мови у прозі [11, с. 157]. Однорідні члени речення також використовуються у цьому стилі, але не з метою розширити уявлення про певне явище, а для більш емоційного сприйняття тексту. Тож

однорідні члени речення можуть представляти із себе зовсім різні, не пов’язані між собою характеристики або явища [14, с. 191]. Щодо складнопідрядних речень варто зазначити, що вони є у кожному прозовому тексті, причому художні тексти не мають чіткої закономірності щодо вживання конкретних видів підрядних частин. Проте можна відмітити, що у творчих працях автори надають перевагу більш простим реченням, неускладненим, на відміну від авторів, що представляють науковий стиль [21, с. 82].

1.4. Загальна інформація про нейронні мережі

Нейронна мережа — це математична модель, що складається з великої кількості взаємопов'язаних вузлів (штучних нейронів), які функціонують разом для обробки інформації. Вона подібна до нейронів людського мозку: нейронні мережі покращують сприйняття та якість обробки інформації через встановлення нових зв’язків між нейронами [29, с. 1].

Нейронні мережі здатні самостійно навчатися і виявляти ключові характеристики даних без попереднього програмування [50]. Це означає, що замість того, щоб наперед визначати правила і закономірності для розпізнавання об'єктів чи виконання завдань, ми даємо мережі багато прикладів і вона сама вчиться розуміти, які особливості або ознаки важливі для правильного результату [50]. Найпоширеніші типи нейронних мереж [21]: 1) нейронна мережа прямого поширення; 2) рекурентна нейронна мережа; 3) згорткова нейронна мережа; 4) модулярна нейронна мережа.

Нейронна мережа включає такі компоненти:

- нейрони — основні обчислювальні елементи нейронної мережі, які отримують і обробляють вхідні дані;
- з'єднання — мости між нейронами, через які передається інформація. Кожне з'єднання має вагу, яка визначає силу значення та напрямок сигналу, що передаються між нейронами;
- ваги — числові коефіцієнти, що відповідають за обчислення сили зв'язку між нейронами. Визначають ступінь впливу одного нейрона на інший [29, с. 8]. Ваги обчислюються під час навчання нейронної мережі;

- функція активації нейронів (activation function) — математична функція, що використовується для передачі інформації через нейрони [29, с. 9]. Виконує обчислення, необхідні для перетворення вхідних даних у вихідні значення нейронів [24];
- правила навчання нейронних мереж — механізм, за допомогою якого нейронна мережа навчається на основі вхідних даних. Відповідає за корегуавання ваг і зміщень для мінімізації помилок нейронної мережі, тим самим підвищуючи якість її результатів [30, с. 2].

1.5. Рекурентна нейронна мережа

Рекурентна нейронна мережа (RNN) — це тип нейронної мережі, що дозволяє впливати попереднім вихідним даним на наступні вхідні дані [33, с. 1]. Цей тип нейронної мережі ефективно працює з текстовим матеріалом та у сфері обробки природної мови [33, с. 1]. В той же час, у інших типів нейронних мереж усі дані є незалежними один від одного, а для якісного автоматичного визначення стилю тексту необхідно, щоб програма запам’ятовувала попередні дані та на їхній базі передбачала наступне слово в реченні. Для вирішення проблеми врахування контексту було створено рекурентну нейронну мережу, яка використовує прихований шар — стан, що здатний запам'ятовувати інформацію про послідовність даних [39].

Основна відмінність RNN полягає у способі передачі інформації всередині мережі. RNN отримують дані на вході та видають результати на виході, як і будь-яка інша нейронна мережа, проте у інших типів глибоких нейронних мереж кожен шар має свої власні ваги (параметри, які визначають, як інформація передається між нейронами). Це означає, що кожен шар працює незалежно від інших. В той же час, RNN використовує одну й ту саму групу ваг на всіх етапах обробки даних. Тобто, коли дані передаються з одного етапу на інший, ваги не змінюються, що дозволяє краще працювати з такими послідовними даними, як текстові матеріали, оскільки вони можуть зберігати інформацію про попередні дані під час обробки нових [39].

Алгоритм навчання рекурентної нейронної мережі:

1. мережа отримує вхідні дані та обробляє їх послідовно, одиницю даних за один раз;
2. використовуючи вхідні та попередні дані, обчислюється новий стан мережі. Таким чином мережа враховує як нові дані, так і те, що вона запам'ятала з попередніх станів;
3. мережа оновлює свою пам'ять, щоб бути готовою до обробки наступного елементу, згідно послідовності;
4. мережа продовжує цей процес для кожного наступного елементу, збираючи інформацію з усіх попередніх даних. Це дозволяє мережі враховувати контекст всієї послідовності;
5. після того, як всі дані оброблені, мережа використовує свій поточний стан для обчислення кінцевого результату;
6. мережа порівнює результат свого передбачення із заздалегідь правильним результатом і порівнює наскільки вони відрізняються;
7. мережа використовує дані про помилкові передбачення, щоб скорегувати ваги і покращити точність на наступних ітераціях. Таким чином, мережа навчається і стає більш точною.

Рекурентна нейронна мережа має різні архітектури, які можуть суттєво змінювати принцип її роботи. Для дослідження було вибрано довготривалу короткочасну пам’ять (Long Short Term Memory).

1.6. Довготривала короткочасна пам'ять

Архітектура довготривалої короткочасної пам'яті була розроблена для подолання проблеми зникання градієнтів, що може виникати в стандартних рекурентних нейронних мережах при навчанні довгих послідовностей [32, с. 10]. Мережа коригує свої ваги на основі сили впливу кожної ваги на помилку. Цей вплив вимірюється за допомогою градієнта. Проблема зникання градієнту полягає в тому, що іноді ці градієнти стають дуже маленькими, що призводить до незначної зміни ваг і мережа практично перестає навчатися [29, с. 115-117]. У найгіршому випадку це може зупинити процес навчання зовсім, і мережа не зможе поліпшувати свої результати. Необхідно, щоб коригування ваг були

достатньо великими, адже мережа коригує їх на основі помилок, які вона робить.

За допомогою спеціального функціоналу, а саме вентилів, архітектура довготривалої короткочасної пам'яті має здатність зберігати інформацію протягом тривалого часу [33, с. 3]. Вентилі здатні керувати даними, що знаходяться у пам’яті. Існують такі типи вентилів [27, с. 14]:

- вентиль вхідних даних (англ. input gate) — додає нову інформацію у пам’ять;
- вентиль видалення (англ. forget gate) — видаляє інформацію із пам’яті;
- вентиль вихідних даних (англ. output gate) — визначає вихідні дані на основі поточного стану пам'яті і передає їх до наступного блоку пам’яті.

Таким чином, довготривала короткочасна пам’ять допомагає рекурентній нейронній мережі запам'ятовувати важливу інформацію протягом довгого часу і видаляти непотрібні дані, що допомагає мережі ефективно працювати з послідовними даними, особливо при врахуванні контексту. За допомогою цих трьох вентилів, LSTM передає через мережу тільки необхідну інформацію, що дозволяє їй навчатися на важливих даних, тим самим забезпечуючи максимально точні результати.

1.7. Вкладання слів

В останні роки з’явилась нова ідея розробки та навчання нейронних мереж, що змінила спосіб представлення та розуміння даних — вкладання слів.

Вкладання слів (англ. Embedding) — це метод відображання слів, словосполучень та навіть речень [31, с. 1] у вигляді векторів дійсних чисел. В основі вкладання лежить ідея про те, що слова, які зазвичай використовуються в схожих контекстах, повинні мати схожі представлення у векторному просторі, для цього алгоритми враховують контекст слова в тексті під час навчання моделі [31, с. 1]. Вкладання було створено з метою поліпшення роботи з текстовою інформацією, що в свою чергу дозволяє нейронній мережі ефективніше вирішувати завдання, пов'язані з природною мовою, включаючи класифікацію даних [29, с. 83-84]. Іншими словами, вкладання допомагає

нейронній мережі працювати з текстом, розуміючи його з більш високою точністю.

Основні та найпоширеніші функції Embedding:

- фіксує семантичні зв’язки між слововживаннями. Текстові одиниці зі схожими значеннями у векторному вигляді відображаються ближче одна відносно одної, ніж слова із семантичною відмінністю, що допомагає нейронній мережі у розумінні контексту представлених даних [31, с. 3];
- є ефективним методом для обробки великих масивів даних, адже текстові одиниці, представлені у векторній формі, займають менше місця і потребують менше ресурсів для обробки, при цьому залишаючи за собою важливі зв’язки та інформацію про функціонування даних [31, с. 1];
- використання інформації про семантичну подібність допомагає нейронній мережі робити прогнози на нових прикладах, що відсутні у вхідних даних для навчання мережі;
- зменшує обчислювальну складність процесів навчання через компактність даних.

1.8. Повнозв'язний шар

Нейрони повнозв'язного шару (англ. Dense layer) з'єднані з усіма нейронами попереднього шару, тобто кожен нейрон повнозв'язного шару може взаємодіяти з будь-яким нейроном попереднього шару. Dense layer обробляє кожний елемент попереднього шару, виконуючи матричне перемноження цих елементів зі своїми вагами. Отримані дані передаються на наступний шар. Dense layer є одним з основних компонентів багатьох нейронних мереж і використовується для здійснення складних нелінійних перетворень вхідних даних, таких як зображення, текст або звук, що допомагає моделі вирішувати такі завдання як: класифікація, прогнозування або розпізнавання об'єктів [49].

Основна причина використання повнозв'язних шарів полягає у тому, що вони здатні фіксувати складні залежності в даних, дозволяючи кожному нейрону взаємодіяти з усіма нейронами попереднього шару. Недоліком використання повнозв’язних шарів є те, що вони можуть створювати проблему

перенавчання, коли модель запам’ятовує тільки навчальні дані і не використовує інформацію з нових — набутих даних, отриманих під час навчання [49].

1.9. Проблема перенавчання нейронної мережі

Перенавчання - це ситуація, коли модель машинного навчання враховує тільки вхідні дані і не використовує нову інформацію, яку отримує під час навчання. Зазвичай, причини виникнення такого явища є взаємопов’язаними між собою та призводять до однакового результату — отримання відмінних результатів на навчальних даних, але незадовільні результати на нових [45]. Найпоширеніші причини:

- наявність великої кількості шарів або нейронів, що робить модель занадто складною;
- надмірна кількість вхідних даних для навчання;
- наявність шумів та неточних даних.

У результаті, модель машинного навчання занадто довго фіксується на навчальних даних і не встановлює нові явища, що призводить до низької ефективності роботи над новими даними.

Техніка виключення (dropout) допомагає усувати проблему перенавчання. Вона працює за принципом випадкового вилучення частини нейронів або з'єднань між ними під час тренування моделі [29, с. 132]. При цьому на кожному кроці навчання, дропаут робить неактивними певну частину нейронів, що допомагає моделі машинного навчання узагальнювати дані — вона навчається шукати найважливіші характеристики та загальні принципи взаємодії даних, оскільки нейрони вчаться взаємодіяти з різними нейронами, а не запам'ятовувати конкретні навчальні приклади. У результаті, модель стає менш схильною до перенавчання [29, с. 132].

1.10. Висновки до розділу 1

Було проаналізовано три функціональних стилів української мови на лексичному, морфологічному та синтаксичному рівнях. З огляду на теоретичний

матеріал, можна дійти висновку, що для створення програми, яка буде ефективно визначати стиль тексту, ознак тільки синтаксичного та морфологічного рівнів недостатньо. Офіційно-діловий та науковий стилі є дуже подібними на цих рівнях, а у художнього стилю майже відсутні типові граматичні ознаки та чітко визначена структура тексту і речень. Тексти художнього стилю хоч і мають найбільше відмінностей у морфологічному та синтаксичному плані, проте ці тексти не мають чітких вимог до їх написання, тож граматичні характеристики завжди варіюються в залежності від багатьох факторів: авторів, жанрів, підстилів, тем. При синтаксичному та морфологічному аналізі увага зосереджується саме на структурі речень та формах слів, а отже поза увагою залишається зміст тексту.

На лексичному рівні, лексичний склад кожного стилю є дійсно індивідуальним. Проте на цьому рівні нейронна мережа також може стикнутися з певними труднощами. Найпоширеніші з них такі:

1. терміни можуть вживатися, наприклад, як у науковому стилі, так і в офіційно-діловому;
2. наявність таких підмов, в яких спостерігаються ознаки різних функціональних стилів – на лексичному рівні в тому числі;
3. відсутність мінімальної обмеженості словника у текстах художнього стилю;
4. наявність полісемії та омонімії.

Для якісного та ефективного функціонування нейронної мережі необхідно знайти спосіб, щоб дозволити програмі «зрозуміти» контекст вибірки, а значить правильно визначити стиль всього тексту. Згідно теоретично опрацьованого матеріалу, який стосується типів нейронних мереж, рекурентна нейронна мережа здатна враховувати семантику тексту. Щоб забезпечити максимально точний результат, необхідно використати одну з найефективніших архітектур рекурентної нейронної мережі, а саме — довготривалу короткочасну пам’ять (англ. Long Short-Term Memory). LSTM-мережі можуть зберігати інформацію на довготривалій основі, що дозволяє краще обробляти залежності між текстовими

елементами. Для підвищення ефективності обробки тексту також можна використовувати метод Embedding, який перетворює слова у вектори фіксованої розмірності, що відображають їхні семантичні властивості. Це дозволяє моделі розуміти рівень семантичної близькості між словами. Шари Dense забезпечують повне з'єднання нейронів з іншими нейронами, що сприяє більш глибокій обробці інформації. Dropout, у свою чергу, допомагає уникнути перенавчання моделі, вимикаючи певний відсоток нейронів під час навчання, що робить модель більш стійкою та здатною до встановлення нових принципів функціонування текстових даних між собою.

РОЗДІЛ 2. РОЗРОБКА РЕКУРЕНТНОЇ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ АВТОМАТИЧНОГО ВИЗНАЧЕННЯ СТИЛЮ ТЕКСТУ УКРАЇНСЬКОЮ МОВОЮ

У попередньому дослідженні [18] була представлена нейронна мережа з моделлю bag of words, яка оцінювала наявність слова у тексті та визначала частоту його використання. Використовувалися лише шари dense і dropout, проте незважаючи на передачу зв'язків між нейронами за допомогою цих шарів, контекст вхідних текстових даних не враховувався, що призводило до неточних результатів. Вихідні дані могли бути правильно визначені нейронною мережею, проте не у кожному випадку. Причому навіть за правильної відповіді точність була низькою.

Робота над помилками та аналіз результатів показав, що основною проблемою була відсутність врахування контексту. Врахування одних тільки слововживань не забезпечило максимально точним результатом, адже якщо нейронна мережа виділятиме тільки словоформи, не враховуючи при цьому семантику тексту, є високий ризик того, що програма визначатиме стиль не з максимальною точністю або видаватиме неправильний результат. Після опрацювання літератури зі стилістики та теоретичного аналізу функціональних стилів української мови стало зрозуміло, що необхідно впровадити розпізнавання контексту.

Попередня модель bag of words [18] мала обмежену ефективність також через великий датасет, що спричинило проблеми з ресурсами та великою тривалістю процесу навчання. Тож для обробки більших датасетів необхідно використовувати іншу модель. Було досліджено різні типи нейронних мереж [21] та встановлено ті, що можуть розпізнавати контекст та шари, які також для цього використовують. На основі опрацьованої літератури з’ясовано, що для створення компактного навчального набору даних необхідно представити речення у вигляді індексів, де кожне унікальне слово представляє собою номер, починаючи з одиниці. Такий підхід не тільки зберігає, але й підвищує якість даних, які подаються нейронній мережі.

Після опрацювання наукової літератури щодо машинного навчання, було встановлено, що для врахування контексту ефективними є рекурентні нейронні мережі [29, с. 162-164], зокрема з використанням шарів довготривалої короткочасної пам'яті (LSTM) [29, с. 160] та Embedding [31, 29]. Однак шари dense [49] і dropout [29, с. 132] також було вирішено використати, оскільки вони можуть ефективно працювати у комбінації з новими шарами. Таким чином, майбутня нейронна мережа включатиме як нові шари, так і ті, що використовувалися у попередньому дослідженні [18].

2.1. Опис навчального набору для створення нейронної мережі

Навчальний набір складався з текстів, що представляють три функціональні стилі української мови: науковий, офіційно-діловий та художній. Тексти було вивантажено з лінгвістичного порталу mova.info [26]. Текстові матеріали були вже промарковані за стилями. На порталі матеріали розподілені згідно таких принципів:

- тексти наукового стилю розподілені за науковими галузями;
- тексти офіційно-ділового представлені категорією "Законодавчі тексти" та розподілені за законодавчими сферами відповідно;
- тексти художнього стилю представлені двома підстилями — "Художня проза" та "Поетична мова". Тексти були зібрані з категорії "Художня проза", де вони розподілені згідно власних імен та прізвищ окремих письменників.

Науковий стиль представляє тексти з таких галузей: алергологія, біологія, ботаніка, зоологія, імунологія, медицина, науково-технічні тексти, психологія. Набір з текстів наукового стилю складає 2 244 964 слововживань. Тексти з перерахованих галузей були вибрані згідно того, що вони найкраще ілюструють специфічні лінгвістичні явища та семантику наукового стилю.

Офіційно-діловий стиль на порталі mova.info [26] представлений законодавчим підстилем, зокрема такими сферами законодавства: Закони і Кодекси України; Законодавство України (з 2000 року); Конституція України; Сімейний Кодекс України, Судочинство. Набір з текстів офіційно-ділового

стилю сягає 2 737 766 слововживань. Вибір законодавчих текстів є вдалим через те, що він демонструє строгу структуру та відсутність полісемії у текстах офіційно-ділового стилю.

Художній стиль представляють тексти різних українських митців. Було обрано тексти таких авторів: Валер'ян Підмогильний, Василь Стефаник, Микола Хвильовий, Іван Франко, Іван Нечуй-Левицький. Кількість слововживань складає 1 846 126. Вибір текстів різних письменників зумовлений тим, що кожний з них має індивідуальний авторський стиль, а тексти художнього стилю не мають чітко-визначеної структури та тематики. Тому твори різних письменників демонструють масштабність лексичного складу, різних підходів до написання текстів та відсутність сталих закономірностей, як у попередніх двох стилів.

Тексти були вивантажені у форматі txt з лінгвістичного порталу mova.info. [26]. Після того як тексти були отримані, почався процес їхньої обробки. Процес складався з двох частин, кожна з яких представлена окремою програмою:

- Обробка даних та створення навчального набору.
- Підготовка навчального набору для розробки нейронної мережі.

2.2. Обробка даних та створення навчального набору

Програма DIPLOMA_DatasetCreation, розміщена у Додатку 1, створена для виконання конкретного набору задач, а саме: видалення стоп-слів та розділових знаків, зчитування вмісту кожного файлу, поділ текстів із файлів на частини, визначення стилю тексту за назвою папки, де він знаходиться, створення .csv файлу, наповнення його частиною конкретного тексту та інформацією про відповідний стиль.

Для роботи програми створено нову папку DIPLOMA, в ній були наявні 3 нові папки з назвами стилів до яких відносяться вхідні текстові дані: Науковий, Офіційно-Діловий, Художній. Відповідно тексти розподілено за цими стилями у відповідних папках..

Програма була створена за допомогою мови Python у програмному середовищі PyCharm. Для її роботи було використано стандартні бібліотеки: os - для роботи з операційною системою; pandas – для обробки даних; re – для використання і роботи з регулярними виразами; random – для використання випадкового вибору даних.

На рисунку 2.1 зображена основна функція read_files_from_folders, що виконує такі дії:

- заходить в папку DIPLOMA, що містить три папки з текстами, кожна з яких названа відповідно до стилю текстів, що в ній знаходяться;
- проходить по всім папкам всередині папки DIPLOMA (а саме Науковий, Офіційно-Діловий, Художній);
- проходить по всіх текстових файлах у кожній папці;
- зчитує вміст кожного файлу та виконує попередню обробку тексту;
- розбиває текст на речення;
- випадково об'єднує речення у один текстовий фрагмент (від 1 до 6 речень — це допоможе нейронній мережі краще оцінювати тексти не зважаючи на їхні розміри у майбутньому);
- видаляє пунктуацію з отриманих фрагментів;
- додає отримані фрагменти до набору даних (data) разом з відповідним стилем.

```python
def read_files_from_folders(root_folder, stop_words):
    data = set()
    for style_folder in os.listdir(root_folder):
        style_path = os.path.join(root_folder, style_folder)
        if os.path.isdir(style_path):
            for filename in os.listdir(style_path):
                file_path = os.path.join(style_path, filename)
                if os.path.isfile(file_path) and filename.endswith('.txt'):
                    with open(file_path, 'r', encoding='utf-8') as file:
                        lines = file.read()
                        lines = preprocess_text(lines, stop_words)
                        sentences = re.split(r'(?<!\w\.\w.)(?<![A-Z][a-z]\.)(?<=\.|\?)\s', lines)
                        previous_el = 0
                        while True:
                            if previous_el >= len(sentences):
                                break
                            num_of_el = random.randint(1,6)
                            sentence = ' '.join(sentences[previous_el: min(previous_el+num_of_el, len(sentences)-1)])
                            sentence = re.sub(r'[\_,\\/:;?!(){}\[\]<>]', '', sentence)
                            if sentence is not None and sentence != '' and not sentence.isspace() and len(sentence.split(' ')) > 3:
                                data.add((sentence, style_folder))
                            previous_el = previous_el+num_of_el
    return data
```

Рисунок 2.1 Функція read_files_from_folders

Функція save_to_csv, що зображена на рисунку 2.2, за допомогою pandas, перетворює набір наших оброблених даних (data) у датасет та зберігає його у файл CSV, що має вигляд таблиці з 2 колонками:

1. "Text", що містить частини текстів випадкового розміру;
2. "Style", де зазначено стиль тексту, частину із якого представлено в тому ж рядку.

```
def save_to_csv(data, output_file):
    df = pd.DataFrame(data, columns=['Text', 'Style'])
    df.to_csv(output_file, index=False)
```

Рисунок 2.2 Функція save_to_csv

Функція read_stopwords, що зображена на рисунку 2.3, зчитує стоп-слова з файлу stopwordsUkrainian.txt [37] та залишає в пам’яті для подальшого використання у функції preprocess_text.

```
def read_stopwords(stopwords_file):
    with open(stopwords_file, 'r', encoding='utf-8') as file:
        stop_words = set(file.read().splitlines())
    return stop_words
```

Рисунок 2.3 Функція read_stopwords

Функція попередньої обробки тексту preprocess_text, що зображена на рисунку 2.4, використовується для того, щоб відформатувати та очистити текст за допомогою регулярних виразів:

- видалити всі розділові знаки та переноси слів на новий рядок;
- замінити усі великі букви на маленькі;
- видалити стоп-слова (за допомогою функції read_stopwords, зображену на рисунку 2.3).

```python
def preprocess_text(text, stop_words):
    text = re.sub(r'\r\n', '', text)
    text = re.sub(r'\n\r', '', text)
    text = re.sub(r'\r', '', text)
    text = re.sub(r'\n', '', text)
    text = re.sub(r'\s', ' ', text)
    text = text.lower()
    text = ' '.join(word for word in text.split() if word not in stop_words)
    return text
```

Рисунок 2.4 Функція preprocess_text

2.3. Підготовка навчального набору для розробки нейронної мережі

Програма DIPLOMA_DatasetPreprocessing (Додаток 2) присвячена підготовці датасету, що був збережений у CSV файл, для використання його нейронною мережею, а саме:

- токенізує текст;
- перетворює слова в індекси;
- заповнює послідовності;
- створює словник, де збережено відповідний індекс кожного слова та словник, де збережено відповідний номер кожного стилю;
- розбиває на тренувальну і тестову вибірки.

Функція tokenize_text, що зображена на рисунку 2.5, виконує такі дії:

- розбиває кожен рядок текстових даних (датасету, що зберігається у CSV файлі), які знаходяться у колонці під назвою "Text", на слова;
- приписує кожному унікальному слову свій індекс та зберігає слово з його індексом у словник — файл word_index.csv.

```python
def tokenize_text(df):
    word_index = {}
    tokens_list = []
    global MAX_TOKENS
    for index, text in enumerate(df['Text']):
        tokens = text.split()
        tokens_list.append(tokens)
        for word in tokens:
            if word not in word_index:
                word_index[word] = len(word_index) + 1
        MAX_TOKENS = max(len(tokens), MAX_TOKENS)
    return word_index, tokens_list
```

Рисунок 2.5 Функція tokenize_text

Функції encode_labels та save_mapping_styles, які зображені на рисунку 2.6, використовуються для кодування назв стилів із колонки "Style" у датасеті, збереженому у CSV файлі, на відповідні номери від 0 до 2 (Науковий — 0; Офіційно-Діловий — 1; Художній — 2). Вони зберігаються у файлі styles_mapping.csv.

```python
def encode_labels(df):
    label_encoder = LabelEncoder()
    labels_encoded = label_encoder.fit_transform(df['Style'])
    return label_encoder, labels_encoded

1 usage
def save_mapping_styles(label_encoder, output_file):
    styles_mapping_df = pd.DataFrame({'Style': label_encoder.classes_, 'Encoded_Style': np.arange(len(label_encoder.classes_))})
    styles_mapping_df.to_csv(output_file, index=False)
```

Рисунок 2.6 Функції encode_labels та save_mapping_styles

Функція save_preprocessed_data, зображена на рисунку 2.7, використовується для збереження оброблених даних у два файли — навчальний та тестовий датасети.

```python
def save_preprocessed_data(train_df, test_df, train_output_file, test_output_file):
    train_df.to_csv(train_output_file, index=False)
    test_df.to_csv(test_output_file, index=False)
```

Рисунок 2.7 Функція save_preprocessed_data

Функція preprocess_data, зображена на рисунку 2.8, використовує усі попередньо описані функції для виконання роботи усієї програми, а саме:

- завантажує набір даних з CSV файлу;
- токенізує текст, розбиваючи кожну частину тексту на окремі унікальні слова, присвоює кожному токену унікальний номер — індекс;
- зберігає індекс слів у CSV файл;
- кодує мітки (стилі) у відповідні числові значення та зберігає їх у CSV файл;
- розділяє набір даних на тренувальний і тестовий набори;
- зберігає оброблені дані у CSV файли.

```
def preprocess_data(dataset_path, train_output_file, test_output_file):
    df = pd.read_csv(dataset_path)

    word_index, tokens_list = tokenize_text(df)
    global MAX_TOKENS
    max_sequence_length = MAX_TOKENS
    padded_sequences = pad_sequences(tokens_list, word_index, max_sequence_length)

    word_index_df = pd.DataFrame(word_index.items(), columns=['Word', 'Index'])
    word_index_df.to_csv('word_index.csv', index=False)

    label_encoder, labels_encoded = encode_labels(df)

    save_mapping_styles(label_encoder, 'styles_mapping.csv')

    X_train, X_test, y_train, y_test = train_test_split(padded_sequences, labels_encoded, test_size=0.2, random_state=42)

    train_df = pd.DataFrame({'Text': X_train, 'Style': y_train})
    test_df = pd.DataFrame({'Text': X_test, 'Style': y_test})
    save_preprocessed_data(train_df, test_df, train_output_file, test_output_file)
```

Рисунок 2.8 Функція preprocess_data

2.4. Процес розробки нейронної мережі

Нейронна мережа створена у процесі виконання дипломної роботи складається з двох частин. Перша частина – відповідає за навчання, друга частина – відповідає за тестування. Рішення щодо розподілу на дві частини було прийнято для того, щоб забезпечити незалежне функціонування та використання обох частин. Таким чином, процес тестування ніяк не використовує код, що був створений для навчання нейронної мережі і може бути використаний для тестування різних текстів поспіль. Це значно скорочує час виконання задач, що поставлених перед нейронною мережею та зменшує ресурсозатратність системи. Незалежність двох частин допомогла підвищити

швидкість навчання та якість внесених змін до коду після тестування**.** Обидві програми використовують Tensorflow та Keras для забезпечення їхнього функціонування.

Вибираючи між бібліотеками для створення та навчання нейронної мережі було розглянуто такі варіанти:

- PyTorch [46] — це відкрита бібліотека машинного та глибокого навчання. Вона базується на мові програмування Python і надає широкий набір інструментів для реалізації нейронних мереж та інших моделей глибокого навчання.
- MXNet [36] — це відкрита бібліотека глибокого навчання, призначена для розробки та навчання нейронних мереж. Вона надає інструменти для розробки складних моделей глибокого навчання, в тому числі мовлення.
- Tensorflow [47] – відкрита програмна бібліотека для машинного навчання, розроблена компанією Google.

Для розробки нейронної мережі було вибрано TensorFlow через деякі переваги бібліотеки, а саме:

- надає великий спектр інструментів та функцій для розробки різноманітних типів нейронних мереж, включаючи рекурентні нейронні мережі;
- для дослідження є важливим, щоб ця бібліотека підтримувала роботу на графічних процесорах (GPU); [34, с. 1]
- має високорівневий інтерфейс Keras [44], що робить використання необхідних функцій більш доступним та простим; [34, с. 2]
- надає можливість працювати з розподіленими обчисленнями, що дозволяє працювати з великими обсягами даних та високошвидкісними обчисленнями;
- надає функціонал врахування послідовностей, для здійснення цієї задачі він використовує шар LSTM; [34, с. 9]
- працює з мовою програмування Python.

Як уже було згадано, Tensorflow має інтеграцію з Keras, тому його функціонал також було використано у процесі створення та навчання нейронної мережі. Keras – це високорівневий прикладний програмний інтерфейс (англ. API) для створення та тренування моделей нейронних мереж, який був інтегрований у TensorFlow як його офіційний високорівневий API. Keras робить використання TensorFlow простіше, надаючи великий набір вбудованого функціоналу та спрощуючи процес створення та навчання нейронної мережі. При цьому, слід зазначити, інтерфейс не втрачає рівні гнучкості та продуктивності. Для дослідження було використано такий набір інструментів Keras:

1. вхідний (англ. Input) та вихідний (англ. Output) шари, функціональні шари Embedding, Dense, LSTM, Dropout;
2. функцію для заповнення послідовностей pad_sequences;
3. функцію callback [41] допомагає змінювати поведінку моделі під час навчання, є ефективним інструментом при використанні великих датасетів. Було використано метод callbacks.ModelCheckpoint [43] – він виконує функцію збереження контрольних точок (стадій навченості, за проходженням кожного циклу навчання) моделі через регулярні інтервали, вказавши шлях та формат для збереження вагів. Параметр save_best_only=True [43] відповідає за збереження найкращого результату із циклів, тобто того, що має найменші втрати та найбільшу точність за навчання;
4. метод compile [42], який використовується для налаштування процесу навчання та використовує такі функції:
   - функцію Loss [42] встановлює здатність моделі до прогнозування правильного результату. Вона також використовує стандартну функцію categorical_crossentropy, її використовують, як правило, у випадку, коли варіантів для передбачення більше, ніж два (в нашому випадку, маємо три варіанти, відповідно до трьох стилів);

- Optimizer [42], який керує процедурою навчання, використовуючи оптимізатор tf.keras.optimizers.Adam [48] зі швидкістю 7e-7. Така швидкість вважається низькою, проте низький темп навчання запобігає перенавчанню, мінімізує функцію втрат і підвищує точність.
- Metrics [42], що використовується для моніторингу навчання, викликається з модуля tf.keras.metrics. Демонструє точність навченості нейронної мережі під час самого навчання.

Спочатку, і код для навчання, і код для тестування було вирішено запустити на хмарному сервісі Google Colab [38] – ресурсі, що дозволяє запускати код Python прямо у браузері Google, не вимагаючи додаткових налаштувань, дозволяє використовувати своє власне хмарне сховище, що є зручним для розміщення та збереження як вхідних навчальних даних, так і вагів, що будуть створені у процесі навчання. Також сервіс надає доступ до лімітованого використання центральних (англ. CPU) та графічних процесорів (англ. GPU), що суттєво допомагає роботі програми для навчання, адже об’єми, що потрібні для обробки даних та навчання, зазвичай недоступні при використанні стандартного домашнього комп’ютера. Проте під час навчання було виявлено проблему пов’язану з об’ємом пам’яті виділеної Google для навчання, а саме — низька швидкість обробки датасету та навчання нейронної мережі на центральному процесорі. На GPU обробка даних була швидшою. Графічні процесори здатні виконувати значну кількість паралельних обчислень. Вони мають тисячі обчислювальних ядер, що дозволяє виконувати багато операцій одночасно. Центральний процесор має менше ядер, що обмежує його можливості для паралельної обробки. Повний процес навчання на центральному процесорі вимагав би 760 діб на навчання, що було неприйнятним, тож було прийнято рішення використовувати графічний процесор. При всіх перевагах Google Colab, він надає всього 12 гігабайтів відеопам’яті, яких не вистачило для обробки датасету і навчання нейронної мережі. Спроба розбиття вхідних даних на декілька частин та впровадження послідовного навчання цих частин окремо виявилось невдалим рішенням,

оскільки результат першого тестування виявив проблеми з точністю результатів. Було встановлено такі причини:

- навчання на окремих частинах датасету може призвести до того, що модель не побачить всі випадки функціонування даних одночасно. Це знижує її здатність встановлювати складні залежності та робити правильні передбачення на нових даних;
- навчання на невеликих частинах даних може призвести до перенавчання, коли модель ефективно працює на навчальних даних, але погано на тестових або нових даних. Це відбувається тому, що модель запам'ятовує конкретні деталі окремих частин даних і не встановлює загальні принципи функціонування даних;

На цьому етапі було з’ясовано, що підхід використання окремих частин датасету для навчання може призвести до результатів низької якості, втрати важливої інформації та збільшення витрат ресурсів і часу. Тому було вирішено використовувати методи, які дозволяють моделі навчатися на повному датасеті під час навчання. Було вирішено змінити середовище для навчання та тестування моделі. Як альтернативу Google Colab, було вибрано Kaggle [40]. Він пропонує хмарний сервіс та подібний функціонал, проте надає більший об’єм відеопам’яті, а саме 16 гігабайтів графічної пам’яті NVIDIA TESLA P100, з обмеженням у 30 годин на тиждень на використання. Емпіричним методом було визначено, що цього об’єму відеопам’яті та часу вистачить для навчання та тестування нейронної мережі у цьому дослідженні.

2.5. Програмний код для навчання нейронної мережі

Код, розміщений у Додатку 3, використовує низку бібліотек та пакетів для функціонування програми, а саме:

- json - для роботи з JSON форматами даних;
- pandas - для роботи з даними у форматі таблиць;
- tensorflow та keras - для створення та тренування нейронної мережі;

Спочатку нейронна мережа завантажує створені та підготовлені для опрацювання датасети для навчання та для тестування відповідно до рисунку 2.9:

```
train_df = pd.read_csv('/kaggle/input/diploma-text-stuff/train_dataset.csv')
test_df = pd.read_csv('/kaggle/input/diploma-text-stuff/test_dataset.csv')
```

Рисунок 2.9 Завантаження датасетів для навчання та тестування

Так як дані було збережено як JSON об’єкти всередині .csv файлів необхідно конвертувати їх у списки токенів, відповідно до рисунку 2.10.

```
train_df['Text'] = train_df['Text'].apply(lambda x: json.loads(x))
test_df['Text'] = test_df['Text'].apply(lambda x: json.loads(x))
```

Рисунок 2.10 Конвертація даних у списки токенів

Потім необхідно зробити всі рядки однаковими за розміром. Для цього програма заповнює послідовності нулями на місцях пропуску, відповідно до рисунку 2.11.

```
max_sequence_length = 7538
train_sequences = pad_sequences(train_df['Text'], maxlen=max_sequence_length, padding='post')
test_sequences = pad_sequences(test_df['Text'], maxlen=max_sequence_length, padding='post')
```

Рисунок 2.11 Заповнення місць пропуску нулями

Слід зауважити, що заповнення послідовностей можна робити як під час створення датасету, так і під час його обробки нейронною мережею. У роботу було впроваджено другий спосіб, адже він сприяє зменшенню розмірів датасетів і зменшенню навантаження. Це також було необхідно як і через розміри сховища та ліміт виділених ресурсів, так і через процес завантаження датасетів на хмарний сервіс, а саме — скорочення часу завантаження.

Функція build_model виконує побудову моделі для навчання, відповідно до рисунку 2.13 та використовує такі шари:

- input_layer — вхідний шар, що приймає наші послідовності;
- embedding_layer — перетворює індекси слів у вектори фіксованого розміру;
- lstm_layer1 — перший LSTM шар з 256 нейронами;
- lstm_layer2 — другий LSTM шар з 128 нейронами;

- dense_layer1 — повнозв'язний шар з 128 нейронами;
- Dropout1 — вимикає 15% випадково вибраних нейронів, використовуємо для запобігання перенавчанню;
- dense_layer3 — повнозв'язний шар з 32 нейронами;
- output_layer — вихідний шар з функцією softmax — використовується для перетворення вектора чисел у вектор ймовірностей, тобто для прогнозування результату.

```
def build_model(input_shape, vocab_size, num_styles):
    input_layer = Input(shape=(input_shape,), name='input_layer')

    embedding_layer = Embedding(input_dim=vocab_size, output_dim=512, input_length=input_shape, mask_zero=True)(input_layer)

    lstm_layer1 = LSTM(256, return_sequences=True)(embedding_layer)
    lstm_layer2 = LSTM(128)(lstm_layer1)

    dense_layer1 = Dense(128, activation='relu')(lstm_layer2)

    dropout1 = Dropout(0.15)(dense_layer1)

    dense_layer3 = Dense(32, activation='relu')(dropout1)
    output_layer = Dense(num_styles, activation='softmax', name='output_layer')(dense_layer3)

    model = Model(inputs=input_layer, outputs=output_layer)

    return model
```

Рисунок 2.13 Функція build_model

Слід зазначити, що кількість нейронів для LSTM та для Dense шарів було підібрано емпіричним методом. У нашому випадку великий обсяг даних дозволив використовувати велику кількість нейронів, оскільки є більше прикладів для навчання моделі. Проте занадто велика кількість нейронів також призводила до перенавчання, тому оптимальна кількість була підібрана після тривалого тестування результатів навчання. Було запроваджено зменшення кількості нейронів на кожен наступний однаковий шар — це допомогло зменшити складність моделі і знизити ймовірність перенавчання.

2.6. Програмний код створення моделі для тестування

Керуючись тим, що модель для проведення тестування функціонально виконує дії схожі з моделлю для навчання — майже весь код моделі ідентичний (Додаток 4). Єдина її відмінність від моделі для навчання — модель для тестування не зберігає ваги, проте використовує створені минулою моделлю для прогнозування результату.

Як і програми для створення і обробки датасету, модель виконує такі самі дії з текстом, які ми використовуємо для тестування: зчитує текстовий файл, токенізує текст, перетворює слова у індекси, заповнює послідовності. Щодо перетворення слів у індекси, слід зазначити, що модель для тестування використовує готовий словник індексів, створений нами на етапі створення датасету, а не створює новий. Це зумовлено потребою правильного кодування слів. Після заповнення послідовностей, модель, використовуючи ваги та оброблені тестові дані, робить передбачення за допомогою стандартної функції predict, відповідно до рисунку 2.14.

```
text_file_path = '/content/drive/MyDrive/Diploma/DiplomaDataset/input_offc5.txt'
word_index_file_path = '/content/drive/MyDrive/Diploma/DiplomaDataset/word_index.csv'

indexes = text_to_indexes(text_file_path, word_index_file_path)

padded_indexes = np.zeros((1, max_sequence_length), dtype=int)
if len(indexes) <= max_sequence_length:
    padded_indexes[0, :len(indexes)] = indexes
else:
    padded_indexes[0, :] = indexes[:max_sequence_length]

predictions = model.predict(padded_indexes)
```

Рисунок 2.14 Функція для передбачення

За допомогою заздалегідь створеного словника styles_mapping, функція розпізнає стиль, що набрав найбільший відсоток збігів та виводить результат у вигляді назви одного із 3 функціональних стилів, що були вказані у нашому файлі разом із відсотками збігів, а саме: Науковий, Офіційно-Діловий, Художній. Код представлений на рисунку 2.15.

```
predictions = model.predict(padded_indexes)

styles_mapping_df = pd.read_csv('/content/drive/MyDrive/Diploma/DiplomaDataset/styles_mapping.csv')
styles_mapping = dict(zip(styles_mapping_df['Encoded_Style'], styles_mapping_df['Style']))

predicted_styles = []
for prediction in predictions:
    max_index = np.argmax(prediction)
    style_name = styles_mapping[max_index]
    percentage = prediction[max_index] * 100
    predicted_styles.append((style_name, percentage))

for style, percentage in predicted_styles:
    print(f"Style: {style}, Percentage: {percentage:.2f}%")
```

Рисунок 2.15 Код для виведення результату функції передбачення

2.7. Результати роботи нейронної мережі

Нейронна мережа була протестована шістьома текстами. Таке рішення було прийнято з огляду на те, щоб продемонструвати ефективність роботи рекурентної нейронної мережі у специфічних випадках. Їх можна поділити на дві групи:

- 3 тексти, що представляють об’єктивні, базові підстилі та відповідно найчастотніші характеристики кожного трьох стилів;
- 3 тексти, що репрезентують специфічні підстилі або тематику, що суміжна з іншими функціональними стилями.

Перша група представлена текстами української мови, що демонструють найчастотніші характеристики функціональних стилів. Науковий стиль представлений текстом власне наукового підстилю, має жанр дисертації, відноситься до галузі біології [25]. Вибраний текст відображає найпоширеніші ознаки наукового стилю: специфічність термінології, притаманну науці іменниковість та структуру наукових текстів. На рисунку 2.16 продемонстровано результат визначення наукового стилю.

```
text_file_path = '/content/drive/MyDrive/Diploma/DiplomaDataset/diploma_nauk2.txt'
word_index_file_path = '/content/drive/MyDrive/Diploma/DiplomaDataset/word_index.csv'

indexes = text_to_indexes(text_file_path, word_index_file_path)

padded_indexes = np.zeros((1, max_sequence_length), dtype=int)
if len(indexes) <= max_sequence_length:
    padded_indexes[0, :len(indexes)] = indexes
else:
    padded_indexes[0, :] = indexes[:max_sequence_length]

predictions = model.predict(padded_indexes)

styles_mapping_df = pd.read_csv('/content/drive/MyDrive/Diploma/DiplomaDataset/styles_mapping.csv')
styles_mapping = dict(zip(styles_mapping_df['Encoded_Style'], styles_mapping_df['Style']))

predicted_styles = []
for prediction in predictions:
    max_index = np.argmax(prediction)
    style_name = styles_mapping[max_index]
    percentage = prediction[max_index] * 100
    predicted_styles.append((style_name, percentage))

for style, percentage in predicted_styles:
    print(f"Style: {style}, Percentage: {percentage:.2f}%")

for style, percentage in predicted_styles:
    display(HTML(f"<b><span style='font-size: 24px;'>{style.upper()}, {percentage:.2f}%</span></b>"))
```

```
WARNING:absl:Skipping variable loading for optimizer 'Adam', because it has 1 variables whereas the saved optimizer has 28 variables.
Weights loaded successfully!
1/1 [==============================] - 6s 6s/step
Style: Науковий, Percentage: 99.97%
НАУКОВИЙ, 99.97%
```

Рисунок 2.16 Результат визначення наукового стилю тексту 1-ої групи

Для тестування ефективності роботи нейронної мережі з офіційно-діловим стилем, було вибрано текст законодавчого підстилю, представлений таким жанром як закон [15]. Текст оперує статтями, має чітку та строгу структуру написання офіційних документів, містить високу частоту вживання: іменників, що ідентифікують осіб за статтю, громадянством, статусом; числівників, що ідентифікують вік особи та номер статей тощо. На рисунку 2.17 продемонстровано результат роботи нейронної мережі щодо визначення офіційно-ділового стилю.

```
text_file_path = '/content/drive/MyDrive/Diploma/DiplomaDataset/diploma_ofc2.txt'
word_index_file_path = '/content/drive/MyDrive/Diploma/DiplomaDataset/word_index.csv'

indexes = text_to_indexes(text_file_path, word_index_file_path)

padded_indexes = np.zeros((1, max_sequence_length), dtype=int)
if len(indexes) <= max_sequence_length:
    padded_indexes[0, :len(indexes)] = indexes
else:
    padded_indexes[0, :] = indexes[:max_sequence_length]

predictions = model.predict(padded_indexes)

styles_mapping_df = pd.read_csv('/content/drive/MyDrive/Diploma/DiplomaDataset/styles_mapping.csv')
styles_mapping = dict(zip(styles_mapping_df['Encoded_Style'], styles_mapping_df['Style']))

predicted_styles = []
for prediction in predictions:
    max_index = np.argmax(prediction)
    style_name = styles_mapping[max_index]
    percentage = prediction[max_index] * 100
    predicted_styles.append((style_name, percentage))

for style, percentage in predicted_styles:
    print(f"Style: {style}, Percentage: {percentage:.2f}%")

for style, percentage in predicted_styles:
    display(HTML(f"<b><span style='font-size: 24px;'>{style.upper()}, {percentage:.2f}%</span></b>"))
```

```
WARNING:absl:Skipping variable loading for optimizer 'Adam', because it has 1 variables whereas the saved optimizer has 28 variables.
Weights loaded successfully!
1/1 [==============================] - 7s 7s/step
Style: Офіційно-Діловий, Percentage: 99.96%
```

ОФІЦІЙНО-ДІЛОВИЙ, 99.96%

Рисунок 2.17 Результат визначення офіційно-ділового стилю тексту 1-ої групи

Для перевірки рівня визначеності художнього стилю, був вибраний текст, що відноситься до епічного підстилю та за жанром є романом. Роман під назвою «Тигролови» Івана Багряного, вивантажений з української онлайн бібліотеки [19]. Він демонструє найчастотніші ознаки художнього стилю, а саме: відсутність чіткої та строгої структури тексту; наявність емоційно-експресивної лексики та художніх засобів; суб’єктивність викладу тексту; дієслівний та прикметниковий характер тексту, що надає йому характерної динамічності тощо. На рисунку 2.18 можна побачити результат визначення художнього стилю нейронної мережею.

```
text_file_path = '/content/drive/MyDrive/Diploma/DiplomaDataset/diploma_hud.txt'
word_index_file_path = '/content/drive/MyDrive/Diploma/DiplomaDataset/word_index.csv'

indexes = text_to_indexes(text_file_path, word_index_file_path)

padded_indexes = np.zeros((1, max_sequence_length), dtype=int)
if len(indexes) <= max_sequence_length:
    padded_indexes[0, :len(indexes)] = indexes
else:
    padded_indexes[0, :] = indexes[:max_sequence_length]

predictions = model.predict(padded_indexes)

styles_mapping_df = pd.read_csv('/content/drive/MyDrive/Diploma/DiplomaDataset/styles_mapping.csv')
styles_mapping = dict(zip(styles_mapping_df['Encoded_Style'], styles_mapping_df['Style']))

predicted_styles = []
for prediction in predictions:
    max_index = np.argmax(prediction)
    style_name = styles_mapping[max_index]
    percentage = prediction[max_index] * 100
    predicted_styles.append((style_name, percentage))

for style, percentage in predicted_styles:
    print(f"Style: {style}, Percentage: {percentage:.2f}%")

for style, percentage in predicted_styles:
    display(HTML(f"<b><span style='font-size: 24px;'>{style.upper()}, {percentage:.2f}%</span></b>"))
```

```
WARNING:absl:Skipping variable loading for optimizer 'Adam', because it has 1 variables whereas the saved optimizer has 28 variables.
Weights loaded successfully!
1/1 [==============================] - 5s 5s/step
Style: Художній, Percentage: 99.93%
```

ХУДОЖНІЙ, 99.93%

Рисунок 2.18 Результат визначення художнього стилю тексту 1-ої групи

Друга група також включає тексти української мови. Для тестування точності визначення нейронною мережею наукового стилю був вибраний текст, що пов’язаний з темою літературознавства [5], а отже містить емоційно-експресивну та описову лексику. Без врахування семантики нейронною мережею, текст може бути визначений як текст художнього стилю.

```
text_file_path = '/content/drive/MyDrive/Diploma/DiplomaDataset/diploma_nauk.txt'
word_index_file_path = '/content/drive/MyDrive/Diploma/DiplomaDataset/word_index.csv'

indexes = text_to_indexes(text_file_path, word_index_file_path)

padded_indexes = np.zeros((1, max_sequence_length), dtype=int)
if len(indexes) <= max_sequence_length:
    padded_indexes[0, :len(indexes)] = indexes
else:
    padded_indexes[0, :] = indexes[:max_sequence_length]

predictions = model.predict(padded_indexes)

styles_mapping_df = pd.read_csv('/content/drive/MyDrive/Diploma/DiplomaDataset/styles_mapping.csv')
styles_mapping = dict(zip(styles_mapping_df['Encoded_Style'], styles_mapping_df['Style']))

predicted_styles = []
for prediction in predictions:
    max_index = np.argmax(prediction)
    style_name = styles_mapping[max_index]
    percentage = prediction[max_index] * 100
    predicted_styles.append((style_name, percentage))

for style, percentage in predicted_styles:
    print(f"Style: {style}, Percentage: {percentage:.2f}%")

for style, percentage in predicted_styles:
    display(HTML(f"<b><span style='font-size: 24px;'>{style.upper()}, {percentage:.2f}%</span></b>"))
```

```
WARNING:absl:Skipping variable loading for optimizer 'Adam', because it has 1 variables whereas the saved optimizer has 28 variables.
Weights loaded successfully!
1/1 [==============================] - 5s 5s/step
Style: Науковий, Percentage: 84.25%
```

НАУКОВИЙ, 84.25%

Рисунок 2.21 Результат визначення наукового стилю тексту 2-ої групи

Для перевірки правильності класифікації текстів офіційно-ділового стилю, було вибрано текст, що відноситься до дипломатичного підстилю та жанру ноти [20]. Вибрана нота не апелює законами та статтями, на відміну від текстів законодавчого підстилю, має форму офіційного листування між Надзвичайним і Повноважним Послом Японії в Україні та міністром розвитку громад, територій та інфраструктури України з питань європейської інтеграції, а не документу. Відповідно до рисунку 2.22 маємо такий результат:

```
text_file_path = '/content/drive/MyDrive/Diploma/DiplomaDataset/diploma_ofc.txt'
word_index_file_path = '/content/drive/MyDrive/Diploma/DiplomaDataset/word_index.csv'

indexes = text_to_indexes(text_file_path, word_index_file_path)

padded_indexes = np.zeros((1, max_sequence_length), dtype=int)
if len(indexes) <= max_sequence_length:
    padded_indexes[0, :len(indexes)] = indexes
else:
    padded_indexes[0, :] = indexes[:max_sequence_length]

predictions = model.predict(padded_indexes)

styles_mapping_df = pd.read_csv('/content/drive/MyDrive/Diploma/DiplomaDataset/styles_mapping.csv')
styles_mapping = dict(zip(styles_mapping_df['Encoded_Style'], styles_mapping_df['Style']))

predicted_styles = []
for prediction in predictions:
    max_index = np.argmax(prediction)
    style_name = styles_mapping[max_index]
    percentage = prediction[max_index] * 100
    predicted_styles.append((style_name, percentage))

for style, percentage in predicted_styles:
    print(f"Style: {style}, Percentage: {percentage:.2f}%")

for style, percentage in predicted_styles:
    display(HTML(f"<b><span style='font-size: 24px;'>{style.upper()}, {percentage:.2f}%</span></b>"))
```

```
WARNING:absl:Skipping variable loading for optimizer 'Adam', because it has 1 variables whereas the saved optimizer has 28 variables.
Weights loaded successfully!
1/1 [==============================] - 7s 7s/step
Style: Офіційно-Діловий, Percentage: 99.91%
```

ОФІЦІЙНО-ДІЛОВИЙ, 99.91%

Рисунок 2.22 Результат визначення офіційно-ділового стилю тексту 2-ої групи

Для тестування художнього стилю було вибрано науково-фантастичний роман під назвою «2001 — Космічна одіссея» Артура Кларка. Текст роману було вивантажено з української онлайн бібліотеки [1]. Відноситься до епічного підстилю художнього стилю. Має певні ознаки наукового стилю, а саме високу частоту вживання термінології та слів, що стосуються різних наукових тем: антропології, зоології, фізики, астрономії та космосу, штучного інтелекту тощо. Віднесення цього тексту до художнього стилю зумовлено наявністю сюжету, діалогів, головних героїв, характерною описовістю подій та місцевостей за допомогою художніх прийомів.

```
text_file_path = '/content/drive/MyDrive/Diploma/DiplomaDataset/diploma_hud2.txt'
word_index_file_path = '/content/drive/MyDrive/Diploma/DiplomaDataset/word_index.csv'

indexes = text_to_indexes(text_file_path, word_index_file_path)

padded_indexes = np.zeros((1, max_sequence_length), dtype=int)
if len(indexes) <= max_sequence_length:
    padded_indexes[0, :len(indexes)] = indexes
else:
    padded_indexes[0, :] = indexes[:max_sequence_length]

predictions = model.predict(padded_indexes)

styles_mapping_df = pd.read_csv('/content/drive/MyDrive/Diploma/DiplomaDataset/styles_mapping.csv')
styles_mapping = dict(zip(styles_mapping_df['Encoded_Style'], styles_mapping_df['Style']))

predicted_styles = []
for prediction in predictions:
    max_index = np.argmax(prediction)
    style_name = styles_mapping[max_index]
    percentage = prediction[max_index] * 100
    predicted_styles.append((style_name, percentage))

for style, percentage in predicted_styles:
    print(f"Style: {style}, Percentage: {percentage:.2f}%")

for style, percentage in predicted_styles:
    display(HTML(f"<b><span style='font-size: 24px;'>{style.upper()}, {percentage:.2f}%</span></b>"))
```

```
WARNING:absl:Skipping variable loading for optimizer 'Adam', because it has 1 variables whereas the saved optimizer has 28 variables.
Weights loaded successfully!
1/1 [==============================] - 5s 5s/step
Style: Художній, Percentage: 94.00%
```

ХУДОЖНІЙ, 94.00%

Рисунок 2.23 Результат визначення художнього стилю тексту 2-ої групи

2.8. Висновки до розділу 2

Було встановлено, що для ефективного автоматичного визначення стилю тексту необхідно використати такий тип нейронної мережі як рекурентна нейронна мережа. Це зумовлено тим, що принцип роботи рекурентної нейронної мережі полягає в тому, що на кожному кроці послідовності, мережа оновлює свій стан, використовуючи вхідні та попередні дані. Цей принцип дозволяє мережі запам'ятовувати попередні елементи послідовності, що є критично важливим для розуміння контексту.

Розроблена рекурентна нейронна мережа дозволяє використовувати шар LSTM, що має спеціальні комірки пам'яті та вентилі, які дозволяють вибірково зберігати, видаляти або використовувати інформацію. Це робить її здатною зберігати важливу контекстну інформацію протягом довгих послідовностей. Було використано 2 шари LTSM на 256 та 128 нейронів. Великий розмір датасету (345134 унікальних слововживань) дозволив використовувати велику кількість нейронів у кожному шарі. Точна кількість для першого шару була підібрана емпіричним шляхом, після декількох спроб, що виявили недо- та перенавчання. Кількість нейронів другого шару було зменшено, керуючись

принципом зменшення кількості нейронів від шару до шару. Таким чином, перший шар LSTM з 256 нейронами може взяти на себе важку обробку довгих залежностей, в той час як другий шар з 128 нейронами може взяти на себе більш абстрактне представлення даних і допомогти у вирішенні завдань класифікації стилю. Також для запобігання перенавчання було використано 1 шар Dropout з відсотком втрат 15%. Такого невеликого значення цілком достатньо для того, щоб уникнути втрати важливої інформації, водночас забезпечуючи навчання загальним принципам функціонування даних.

Із додаткових шарів було використано також стандартні, для будь-якої нейронної мережі, а саме Dense шари з 128 та 32 нейронами кожен та один шар Embedding (вкладання слів) з 512 векторами. Використання такої невеликої кількості нейронів для Dense шарів було зумовлено другорядною роллю у навчанні. Рішення щодо використання двох шарів Dense було зумовлено відмінністю їхніх задач. Dense шар з 128 нейронами може виділити важливі ознаки у векторі властивостей, а другий шар з 32 нейронами може допомогти у вирішенні завдання класифікації з меншою кількістю параметрів. Водночас такий великий розмір векторів у Embedding шарі (512) дозволяє моделі вчити більш складні та інформативні представлення слів, що допомогло нейронній мережі краще розпізнавати контекст.

Останніми важливими факторами у питанні балансу недо- та перенавчання були: кількість повних циклів навчання на тренувальному датасеті, що пройде нейронна мережа у процесі навчання, тобто кількість епох та швидкість навчання. Три епохи навчання було обрано емпіричним шляхом для досягнення балансу між достатньою тривалістю навчання та уникненням перенавчання. Більша кількість епох з урахуванням попередніх конфігурацій, призводила до перенавчання.

Налаштування низької швидкості навчання (7e-7) було вибрано, зважаючи на те, що низька швидкість навчання дозволяє моделі уважніше вивчати особливості даних, тим самим покращувати як розуміння контексту, так і здатність до узагальнення даних. Крім того низька швидкість забезпечила

стабільний процесу навчання та уникнення проблем адаптації ваг моделі до тренувальних даних, уникаючи стрімких змін, які можуть призвести до перенавчання.

ВИСНОВКИ

На початку дослідження була опрацьована наукова література, що стосується функціональних стилів української мови, а саме: науковий, офіційно-діловий, художній. За допомогою теоретичних джерел, було встановлено, що на граматичних рівнях (морфологічному і синтаксичному) досить важко встановити унікальні характеристики до кожного стилю. Науковий та офіційно-ділові стилі через обмеженість своєї структури майже не мають відмінностей. В той же час для художнього стилю важко встановити унікальні для нього ознаки на граматичному рівні, адже цей стиль не має чітких вимог щодо оформлення текстового матеріалу, тому може приймати різні форми, в залежності від індивідуального авторського стилю, порушеної проблематики творів, тематики тощо.

На лексичному рівні було встановлено, що всі три стилі дійсно відрізняються один від одного. Науковий стиль має специфічну термінологію, причому у кожної наукової галузі є своя термінологія. В текстах переважають слова іншомовного походження, дотримана структура тексту. Деякі підстилі, наприклад, науково-популярний або науково-навчальний, допускають відхилення від строгої структури та однозначності для полегшеного сприйняття тексту. Офіційно-діловий стиль має також свою термінологію, проте дещо відмінну від наукового стилю. Зазвичай, терміни офіційно-ділового стилю можуть бути словами-омонімами, тобто використовуватися в інших галузях або ситуаціях. Проте терміни, що вживані у офіційних текстах, мають одне значення, яке використовується тільки у контексті офіційної документації. Також від цього стилю вимагається чітка структура тексту та однозначність семантики у всіх підстилях. Художній стиль має дуже широкий склад лексики, тож важко відмітити якусь конкретну групу слів. Проте йому єдиному з трьох стилів притаманна емоційно-експресивна лексика та використання художніх засобів, які відображають суб’єктивне сприйняття автора або ліричного героя до явищ у тексті.

У ході виконання роботи я дійшла висновку, що нейронна мережа не буде ефективно працювати з окремими стилістичними параметрами, тому вона враховує всі параметри за допомогою розуміння контексту. Лексичні параметри: мережа визначає частотність слів через Embedding шар, який представляє слова у вигляді векторів у багатовимірному просторі. Embedding шар також враховує морфологічні параметри, частини мови, оскільки векторні представлення слів враховують контекст, в якому вони зустрічаються, що включає інформацію про них. Синтаксичні параметри: LSTM шари враховують порядок слів та їхню послідовність у реченнях. LSTM шари також здатні визначати такі параметри як пасивні конструкції, складнопідрядні та інші типи речень, зберігаючи довготривалі залежності у тексті. Embedding шар і LSTM шари разом вивчають семантику тексту, оскільки Embedding представляє семантичну інформацію (семантичні параметри) про слова, а LSTM зберігає контекстну інформацію, що дозволяє мережі ідентифіковувати зв'язки між лексичними параметрами (словами) в різних темах/підмовах.

Для розробки нейронної мережі, що автоматично визначає стиль тексту, було зібрано тексти наукового, офіційно-ділового та художнього стилів за допомогою лінгвістичного порталу [26]. Обсяг усіх навчальних даних, що представлені текстами трьох стилів, складає 6 828 856 слововживань. Унікальних слововживань, що дозволили нейронній мережі навчитися розпізнавати різноманітні лексичні, морфологічні та синтаксичні характеристики становить 345 134 слововживань для кожного зі стилів.

У ході дослідження, було з’ясовано, що рекурентна нейронна мережа є ефективним інструментом для автоматичної класифікації текстів. Про це свідчать отримані результати. Точність для текстів, що представляють об’єктивні, базові підстилі та відповідно найчастотніші характеристики кожного трьох стилів є дуже високою. Таким чином, для кожного зі стилю вона становила:

- для тексту власне наукового підстилю наукового стилю — 99.97%;
- для тексту законодавчого підстилю офіційно-ділового стилю — 99.96%;

- для тексту епічного підстилю художнього стилю — 99.93%.

Для того, щоб впевнитися у тому, що рекурентна нейронна мережа ефективно працює з семантикою тексту, було проведено ще одне тестування на текстах, що мають ознаки не тільки одного стилю, а й інших:

- для перевірки визначення наукового стилю було вибрано жанр дисертації у галузі літературознавства. Результат становить 84.25% приналежності до наукового стилю;
- для офіційно-ділового було вибрано жанр угоди, у формі офіційного листування (обміну нот). Вибір зумовлений тим, що цей текст не апелює статтями та законами, а вхідні дані для навчання нейронної мережі, складали, в основному, законодавчі тексти. Результат становить 99.91% приналежності до офіційно-ділового стилю;
- для художнього стилю був вибраний роман на наукову тематику. Результат становить 94.00% приналежності до художнього стилю.

Після інтерпретації результатів можна дійти висновку, що мережа працює за принципом врахування контексту та складних зв’язків, а не окремих слів чи граматичних ознак. Про це свідчить висока точність передбачення стилю, що перевищує 99% для стилістично визначених текстів і понад 80% для текстів зі змішаними стилістичними ознаками. Таким чином, впровадження рекурентних нейронних мереж у автоматичний стилістичний аналіз текстів не лише підвищує точність і ефективність подібних задач, але й сприяє розвитку нових методів і підходів у лінгвістичних дослідженнях, дозволяючи більш глибоко та швидко досліджувати й аналізувати стилістичні характеристики текстів різних жанрів і стилів. Це відкриває широкі перспективи для застосування рекурентних нейронних мереж у галузі автоматичного аналізу текстів, включаючи завдання автоматизованої класифікації текстів та аналізу великих текстових даних.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. 2001 – Космічна одіссея – Артур Кларк, повний текст твору. Бібліотека української літератури УкрЛіб. URL: https://www.ukrlib.com.ua/world/printit.php?tid=3306
2. Актуальні питання гуманітарних наук: міжвузівський збірник наукових праць молодих вчених Дрогобицького державного педагогічного університету імені Івана Франка / [редактори-упорядники В. Ільницький, А. Душний, І. Зимомря]. – Дрогобич: Видавничий дім "Гельветика", 2019. – Вип. 23. Том 3. – 188 с.
3. Англістика та американістика : зб. наук. пр. Вип. 10., Дніпро: Ліра, 2014. 270 с.
4. Войтенко К. І. Функціональний стиль художнього мовлення. Наукові записки Національного університету «Острозька академія»: Серія «Філологія». 2012. Т. 1, № 26. С. 53–56.
5. Грищенко О. Моделі урбаністичного простору в сучасній українській прозі: дисертація на здобуття наукового ступеня кандидата філологічних наук : thesis. 2016.
6. Дудик П. С. Стилістика української мови : навч. посіб. Київ: Академія, 2005. 368 с.
7. Капелюшний А. О. Практична стилістика української мови: навч. посіб. 2-ге вид. Львів: ПАЮ, 2007. 400 с.
8. Кузнецова Г. П. Офіційно-діловий стиль в освітній адміністративно-управлінській сфері спілкування. Науковий часопис НПУ імені М.П. Драгоманова. 2014. Т. 8, № 6. С. 125–140.
9. Кузнецова Г. П., Кухарчук І. О., Корчова О. М. Ортологічні основи офіційно-ділового мовлення в галузі педагогічної освіти: навчальний посібник. Глухів, Глухівський НПУ ім. О. Довженка, 2020. 274 с.
10. Культура академічного письма: навч. посіб. для аспірантів / Нар. укр. акад., [каф. українознав.] ; упоряд. О. В. Слюніна. – Харків : Вид-во НУА, 2020. – 56 с.

11. Марцин С. О. Безсполучникові складні речення як елементи художньо-образної системи прозових і поетичних творів. Типологія та функції мовних одиниць. 2014. № 1. С. 149–159.
12. Наука та освіта як основа суспільного розвитку : матеріали Міжнародної науковопрактичної конференції / Міжнародний гуманітарний дослідницький центр (Житомир, 21 лютого 2024 р). Research Europe, 2024. 138 с.
13. Писарська Н. В., Заверющенко М. П., Лухіна М. Ю. Семантико-синтаксична організація наукового стилю. Вчені записки ТНУ імені В. І. Вернадського. Серія: Філологія. Журналістика. 2023. Т. 34 (73), № 5. С. 24–29.
14. Пономарів О. Д. Стилістика сучасної української мови : підручник. 3-тє вид. Тернопіль: Навчальна книга - Богдан, 2000. 248 с.
15. Про внесення змін до деяких законодавчих актів України щодо окремих питань проходження військової служби, мобілізації та військового обліку : Закон України від 11.04.2024 р. № 3633-IX. URL: https://zakon.rada.gov.ua/laws/show/3633-20#Text
16. Просяна А. В. Синтаксичні особливості українських законодавчих текстів. Одеський лінгвістичний вісник. 2017. Спецвипуск. С. 181–184.
17. Селігей П. О. Іменниковість versus дієслівність: у пошуках золотої середини. Мовознавство. 2014. № 4. С. 36–55.
18. Сиклитенко К. О. Сиклитенко Ксенія_курсова робота. URL: https://docs.google.com/document/d/11D_VfAoPYEH6ek-nOwoAYLZeVhsNHqnKdlPYd-t6og8/edit?usp=sharing
19. Тигролови – Іван Багряний, повний текст твору. Бібліотека української літератури УкрЛіб. URL: https://www.ukrlib.com.ua/books/printit.php?tid=30
20. Угода (у формі обміну нотами) між Урядом України та Урядом Японії про надання Уряду України гранту для реалізації Програми екстреного відновлення (Фаза 3) : Угода Каб. Міністрів України від 19.02.2024 р. URL: https://zakon.rada.gov.ua/laws/show/392_003-24#Text

21. Українська мова (за професійним спрямуванням): навч. посібн. (упор. А.В. Лисенко, Т.К. Ісаєнко, С.М. Дорошенко). - Полтава: ПолтНТУ, 2015. - 280 с.
22. Учасники проектів Вікімедіа. Епістолярний стиль мовлення - Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Епістолярний_стиль_мовлення
23. Учасники проектів Вікімедіа. Конфесійний стиль мовлення – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Конфесійний_стиль_мовлення
24. Учасники проектів Вікімедіа. Передавальна функція штучного нейрона – Вікіпедія. Вікіпедія. 2017. URL: https://uk.wikipedia.org/wiki/Передавальна_функція_штучного_нейрона
25. Хома Ю. А. Біотехнологічні підходи дослідження стійкості швидкорослих дерев до абіотичних стресів для сталого виробництва біопалива : дисертація на здобуття наукового ступеня доктора філософії : 091. Київ, 2024.
26. Корпус текстів української мови. Лінгвістичний портал MOVA.info. URL: http://www.mova.info/corpus2.aspx
27. Bansal M., Goyal A., Choudhary A. A comparative analysis of K-Nearest Neighbour, Genetic, Support Vector Machine, Decision Tree, and Long Short Term Memory algorithms in machine learning. Decision Analytics Journal. 2022. P. 100071. C. 21.
28. Gao T., Yao X., Chen D. SimCSE: Simple Contrastive Learning of Sentence Embeddings. Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Online and Punta Cana, Dominican Republic. Stroudsburg, PA, USA, 2021.
29. Mehlig B. Machine Learning with Neural Networks. Cambridge University Press, 2021.
30. Nayebi A., Srivastava S., Ganguli S., Yamins D.L.K. Identifying Learning Rules From Neural Network Observables. 34th Conference on Neural

Information Processing Systems (NeurIPS 2020), Vancouver, Canada. Stanford University, California, USA, 2020. 12 c.

31. Neelakantan, A., Xu, T., Puri, R., Radford, A., Han, J.M., Tworek, J., Yuan, Q., Tezak, N.A., Kim, J.W., Hallacy, C., Heidecke, J., Shyam, P., Power, B., Nekoul, T.E., Sastry, G., Krueger, G., Schnurr, D.P., Such, F.P., Hsu, K.S., Thompson, M., Khan, T., Sherbakov, T., Jang, J., Welinder, P., & Weng, L. Text and Code Embeddings by Contrastive Pre-Training. ArXiv, abs/2201.10005. 2022. C. 13.
32. Noh S.-H. Analysis of Gradient Vanishing of RNNs and Performance Comparison. Information. 2021. Vol. 12, no. 11. P. 442.
33. Schmidt, Robin M. Recurrent Neural Networks (RNNs): A gentle Introduction and Overview. 2019. 16 c.
34. Survey: Tensorflow in Machine Learning / M. Ramchandani et al. Journal of Physics: Conference Series. 2022. Vol. 2273, no. 1. P. 012008.
35. Yasar K. What is a Neural Network? Definition, Types and How It Works. Enterprise AI. URL: https://www.techtarget.com/searchenterpriseai/definition/neural-network
36. Apache MXNet. Apache MXNet. URL: https://mxnet.apache.org/versions/1.9.1/
37. GitHub - skupriienko/Ukrainian-Stopwords: the list of ~2000 ukrainian stopwords (with numbers). GitHub. URL: https://github.com/skupriienko/Ukrainian-Stopwords
38. Google Colab. Google Colab. URL: https://colab.research.google.com/
39. Introduction to Recurrent Neural Network - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/
40. Kaggle: Your Machine Learning and Data Science Community. Kaggle: Your Machine Learning and Data Science Community. URL: https://www.kaggle.com/
41. Keras documentation: Callbacks API. Keras: Deep Learning for humans. URL: https://keras.io/api/callbacks/

42. Keras documentation: Model training APIs. Keras: Deep Learning for humans. URL: https://keras.io/api/models/model_training_apis/
43. Keras documentation: ModelCheckpoint. Keras: Deep Learning for humans. URL: https://keras.io/api/callbacks/model_checkpoint/
44. Keras: Deep Learning for humans. Keras: Deep Learning for humans. URL: https://keras.io/
45. ML | Underfitting and Overfitting - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning
46. PyTorch. PyTorch. URL: https://pytorch.org/
47. TensorFlow. TensorFlow. URL: https://www.tensorflow.org/
48. tf.keras.optimizers.Adam|TensorFlow v2.16.1. TensorFlow. URL: https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam
49. The Concepts of Dense and Sparse in the Context of Neural Networks | Baeldung on Computer Science. Baeldung on Computer Science. URL: https://www.baeldung.com/cs/neural-networks-dense-sparse
50. What is a neural network? - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/neural-networks-a-beginners-guide/

ДОДАТКИ

1. Додаток 1. Код програми DIPLOMA_DatasetCreation для створення датасету.

```
import os
import pandas as pd
import re
import random

def read_stopwords(stopwords_file):
 with open(stopwords_file, 'r', encoding='utf-8') as file:
 stop_words = set(file.read().splitlines())
 return stop_words

def preprocess_text(text, stop_words):
 text = re.sub(r'\r\n', '', text)
 text = re.sub(r'\n\r', '', text)
 text = re.sub(r'\r', '', text)
 text = re.sub(r'\n', '', text)
 text = re.sub(r'\s', ' ', text)
 text = text.lower()
 text = ' '.join(word for word in text.split() if word not in stop_words)
 return text

def read_files_from_folders(root_folder, stop_words):
 data = set()
 for style_folder in os.listdir(root_folder):
 style_path = os.path.join(root_folder, style_folder)
 if os.path.isdir(style_path):
 for filename in os.listdir(style_path):
 file_path = os.path.join(style_path, filename)
 if os.path.isfile(file_path) and filename.endswith('.txt'):
 with open(file_path, 'r', encoding='utf-8') as file:
 lines = file.read()
 lines = preprocess_text(lines, stop_words)
 sentences = re.split(r'(?<!\w\.\w.)(?<![A-Z][a-z]\.)(?<=\.|\?)\s', lines)
 previous_el = 0
 while True:
 if previous_el >= len(sentences):
 break
 num_of_el = random.randint(1,6)
 sentence = ' '.join(sentences[previous_el: min(previous_el+num_of_el, len(sentences)-1)])
 sentence = re.sub(r'[\.,\\/:;?!()\{\}\[\]<>]', '', sentence)
 if sentence is not None and sentence != '' and not sentence.isspace() and len(sentence.split(' ')) > 3:
 data.add((sentence, style_folder))
 previous_el = previous_el+num_of_el
 return data

def save_to_csv(data, output_file):
 df = pd.DataFrame(data, columns=['Text', 'Style'])
 df.to_csv(output_file, index=False)

def create_dataset(root_folder, output_file, stopwords_file):
 stop_words = read_stopwords(stopwords_file)
 data = read_files_from_folders(root_folder, stop_words)




 save_to_csv(data, output_file)
 print(f"Total rows processed: {len(data)}")
```

```
root_folder = 'D:\\DIPLOMA_KSUSHA'
output_file = 'dataset.csv'
stopwords_file = 'stopwordsUkrainian.txt'
create_dataset(root_folder, output_file, stopwords_file)
```

2. Додаток 2. Код програми DIPLOMA_DatasetPreprocessing для підготовки навчального набору.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

MAX_TOKENS = 0

def tokenize_text(df):
 word_index = {}
 tokens_list = []
 global MAX_TOKENS
 for index, text in enumerate(df['Text']):
 tokens = text.split()
 tokens_list.append(tokens)
 for word in tokens:
 if word not in word_index:
 word_index[word] = len(word_index) + 1
 MAX_TOKENS = max(len(tokens), MAX_TOKENS)
 return word_index, tokens_list

def pad_sequences(tokens_list, word_index, max_sequence_length):
 padded_sequences = []
 for tokens in tokens_list:
 sequence = [word_index.get(word, 0) for word in tokens]
 padded_sequences.append(sequence)
 return padded_sequences

def encode_labels(df):
 label_encoder = LabelEncoder()
 labels_encoded = label_encoder.fit_transform(df['Style'])
 return label_encoder, labels_encoded

def save_mapping_styles(label_encoder, output_file):
 styles_mapping_df = pd.DataFrame({'Style': label_encoder.classes_, 'Encoded_Style':
np.arange(len(label_encoder.classes_))})
 styles_mapping_df.to_csv(output_file, index=False)

def save_preprocessed_data(train_df, test_df, train_output_file, test_output_file):
 train_df.to_csv(train_output_file, index=False)
 test_df.to_csv(test_output_file, index=False)

def preprocess_data(dataset_path, train_output_file, test_output_file):
 df = pd.read_csv(dataset_path)

 word_index, tokens_list = tokenize_text(df)
 global MAX_TOKENS
 max_sequence_length = MAX_TOKENS
 padded_sequences = pad_sequences(tokens_list, word_index, max_sequence_length)

 word_index_df = pd.DataFrame(word_index.items(), columns=['Word', 'Index'])
 word_index_df.to_csv('word_index.csv', index=False)

 label_encoder, labels_encoded = encode_labels(df)
```

```
save_mapping_styles(label_encoder, 'styles_mapping.csv')

X_train, X_test, y_train, y_test = train_test_split(padded_sequences, labels_encoded, test_size=0.2, random_state=42)

train_df = pd.DataFrame({'Text': X_train, 'Style': y_train})
test_df = pd.DataFrame({'Text': X_test, 'Style': y_test})
save_preprocessed_data(train_df, test_df, train_output_file, test_output_file)

print("Training set shape:", len(X_train), "x", len(X_train[0]))
print("Testing set shape:", len(X_test), "x", len(X_test[0]))
print("Max tokens:", MAX_TOKENS)

dataset_path = 'dataset.csv'
train_output_file = 'train_dataset.csv'
test_output_file = 'test_dataset.csv'
preprocess_data(dataset_path, train_output_file, test_output_file)
```

3. Додаток 3. Програмний код для навчання нейронної мережі.

```
import os
import pandas as pd
import json
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.utils import to_categorical
from tensorflow.keras import Input, Model
from tensorflow.keras.layers import Embedding, Dense, LSTM, Dropout
train_df = pd.read_csv('/kaggle/input/diploma-text-stuff/train_dataset.csv')
test_df = pd.read_csv('/kaggle/input/diploma-text-stuff/test_dataset.csv')
train_df['Text'] = train_df['Text'].apply(lambda x: json.loads(x))
test_df['Text'] = test_df['Text'].apply(lambda x: json.loads(x))
max_sequence_length = 7538
train_sequences = pad_sequences(train_df['Text'], maxlen=max_sequence_length, padding='post')
test_sequences = pad_sequences(test_df['Text'], maxlen=max_sequence_length, padding='post')
label_encoder = LabelEncoder()
train_labels = label_encoder.fit_transform(train_df['Style'])
test_labels = label_encoder.transform(test_df['Style'])
num_styles = len(label_encoder.classes_)
train_labels = to_categorical(train_labels, num_classes=num_styles)
test_labels = to_categorical(test_labels, num_classes=num_styles)
def build_model(input_shape, vocab_size, num_styles):
    input_layer = Input(shape=(input_shape,), name='input_layer')
        embedding_layer = Embedding(input_dim=vocab_size, output_dim=512, input_length=input_shape,
mask_zero=True)(input_layer)
    lstm_layer1 = LSTM(256, return_sequences=True)(embedding_layer)
    lstm_layer2 = LSTM(128)(lstm_layer1)
    dense_layer1 = Dense(128, activation='relu')(lstm_layer2)

    dropout1 = Dropout(0.15)(dense_layer1)

    dense_layer3 = Dense(32, activation='relu')(dropout1)
```

```
    output_layer = Dense(num_styles, activation='softmax', name='output_layer')(dense_layer3)

    model = Model(inputs=input_layer, outputs=output_layer)

    return model

input_shape = max_sequence_length
vocab_size = 345134
weights_save_location = '/kaggle/working/Diploma/DiplomaWork/weights'

if(not os.path.isdir(weights_save_location)):
    os.makedirs(weights_save_location)

checkpoint_callback                                                                    =
tf.keras.callbacks.ModelCheckpoint(weights_save_location+'/train_file_weights.weights.h5',\
                                   monitor = 'val_loss',\
                                   save_best_only=True,\
                                   save_weights_only=True)



model = build_model(input_shape, vocab_size, num_styles)

if(len(os.listdir(weights_save_location)) != 0):
    model.load_weights(weights_save_location+'/train_file_weights.weights.h5')
    print("Weights loaded successfully!")
model.compile(loss='categorical_crossentropy',
              optimizer= tf.keras.optimizers.Adam(learning_rate = 7e-7),
              metrics=['accuracy'])
history = model.fit(train_sequences, train_labels, epochs=3, batch_size=32, validation_data=(test_sequences,
test_labels), callbacks=[checkpoint_callback])
model.summary()
```

4. Додаток 4. Програмний код створення моделі для тестування.

```
from IPython.display import display, HTML
import os
import pandas as pd
import json
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.utils import to_categorical
from tensorflow.keras import Input, Model
from tensorflow.keras.layers import Embedding, Dense, LSTM, Dropout

max_sequence_length = 7538
label_encoder = LabelEncoder()
num_styles = 3
```

```
def build_model(input_shape, vocab_size, num_styles):
    input_layer = Input(shape=(input_shape,), name='input_layer')
        embedding_layer = Embedding(input_dim=vocab_size, output_dim=512, input_length=input_shape,
mask_zero=True)(input_layer)
    lstm_layer1 = LSTM(256, return_sequences=True)(embedding_layer)
    lstm_layer2 = LSTM(128)(lstm_layer1)
    dense_layer1 = Dense(128, activation='relu')(lstm_layer2)

    dropout1 = Dropout(0.15)(dense_layer1)

    dense_layer3 = Dense(32, activation='relu')(dropout1)

    output_layer = Dense(num_styles, activation='softmax', name='output_layer')(dense_layer3)
    model = Model(inputs=input_layer, outputs=output_layer)

    return model
input_shape = max_sequence_length
vocab_size = 345134
weights_save_location = '/content/drive/MyDrive/Diploma/DiplomaWork/weights'

if(not os.path.isdir(weights_save_location)):
    os.makedirs(weights_save_location)

checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(weights_save_location,\
                                        monitor = 'val_loss',\
                                        save_best_only=True,\
                                        save_weights_only=True)



model = build_model(input_shape, vocab_size, num_styles)
model.compile(loss='categorical_crossentropy',
            optimizer= tf.keras.optimizers.Adam(learning_rate = 7e-7),
            metrics=['accuracy'])

if(len(os.listdir(weights_save_location)) != 0):
    model.load_weights(weights_save_location+'/train_file_weights.weights.h5')
    print("Weights loaded successfully!")

def text_to_indexes(text_file_path, word_index_file_path):
    word_index_df = pd.read_csv(word_index_file_path)
    word_to_index = dict(zip(word_index_df['Word'], word_index_df['Index']))
    with open(text_file_path, 'r', encoding='utf-8') as file:
        text = file.read().lower()
    tokens = text.split()
    indexes = [word_to_index[word] for word in tokens if word in word_to_index]

    return indexes
```

```
text_file_path = '/content/drive/MyDrive/Diploma/DiplomaDataset/diploma_ofc2.txt'
word_index_file_path = '/content/drive/MyDrive/Diploma/DiplomaDataset/word_index.csv'

indexes = text_to_indexes(text_file_path, word_index_file_path)

padded_indexes = np.zeros((1, max_sequence_length), dtype=int)
if len(indexes) <= max_sequence_length:
    padded_indexes[0, :len(indexes)] = indexes
else:
    padded_indexes[0, :] = indexes[:max_sequence_length]

predictions = model.predict(padded_indexes)

styles_mapping_df = pd.read_csv('/content/drive/MyDrive/Diploma/DiplomaDataset/styles_mapping.csv')
styles_mapping = dict(zip(styles_mapping_df['Encoded_Style'], styles_mapping_df['Style']))

predicted_styles = []
for prediction in predictions:
    max_index = np.argmax(prediction)
    style_name = styles_mapping[max_index]
    percentage = prediction[max_index] * 100
    predicted_styles.append((style_name, percentage))

for style, percentage in predicted_styles:
    print(f"Style: {style}, Percentage: {percentage:.2f}%")

for style, percentage in predicted_styles:
    display(HTML(f"<b><span style='font-size: 24px;'>{style.upper()}, {percentage:.2f}%</span></b>"))
```