

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

**Факультет інформатики та обчислювальної техніки Кафедра
обчислювальної техніки**

«На правах рукопису»
УДК 004.8, 004.912

До захисту допущено:

Завідувач кафедри

_____ Сергій СТИРЕНКО

«__» _____ 20__ р.

Магістерська дисертація

на здобуття ступеня магістра

**за освітньо-науковою програмою «Інженерія програмного забезпечення»
зі спеціальності 121 «Інженерія програмного забезпечення комп'ютерних
систем»**

на тему: «Спосіб підвищення ефективності аналізу схожості текстів»

Виконав:

студент VI курсу, групи ІМ-11мн
Мірошник Денис Валерійович _____

Керівник:

Доц., к.т.н., доц.
Волокита Артем Миколайович _____

Консультант:

Проф., д.т.н., проф.,
Жабін Валерій Іванович _____

Рецензент:

Доц., к.т.н., доц.
Шимкович Володимир Миколайович _____

Засвідчую, що у цій магістерській
дисертації немає запозичень з праць
інших авторів без відповідних
посилань.

Студент (-ка) _____

Київ – 2023 року

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМ. ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – другий (магістерський) за освітньо-науковою програмою

Спеціальність: 121 Інженерія програмного забезпечення

Спеціалізація: Інженерія програмного забезпечення комп'ютерних систем

ЗАТВЕРДЖУЮ

Завідувач кафедри

Сергій СТИРЕНКО

(підпис)

“ ”

2023 р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Мірошнику Денису Валерійовичу

1. Тема дисертації «Спосіб підвищення ефективності аналізу схожості текстів.»

Науковий керівник дисертації доц., к.т.н., доц. Волокита Артем Миколайович

2. Строк подання студентом дисертації 15.05.2023

3. Об'єкт дослідження є процес оцінки схожості текстів

4. Предмет дослідження: спосіб оцінки семантичної схожості текстів з використанням нейронних мереж

5. Перелік завдань, які потрібно розробити: Дослідження предметної області, огляд існуючих рішень, визначення вимог і завдань для програмного забезпечення, реалізація продукту, використання обраних інструментів для розробки.

6. Консультанти розділів дисертації:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	проф. Жабін В.І.		

7. Дата видачі завдання _ 1 вересня 2022 року _____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів дипломного проекту (роботи)	Строк виконання етапів проекту(роботи)	Примітки
1.	<i>Затвердження теми роботи</i>	06.02.2023	
2.	<i>Вивчення та аналіз завдання</i>	06.02.2023	
3.	<i>Розробка архітектури системи</i>	01.03.2023	
4.	<i>Розробка підсистем</i>	04.03.2023	
5.	<i>Програмна реалізація системи</i>	09.03.2023	
6.	<i>Оформлення пояснювальної записки</i>	01.04.2023	
7.	<i>Передзахист</i>	15.05.2023	
8.	<i>Захист</i>	22.05.2023	

Студент Денис МІРОШНИК _____ (підпис)

Керівник Артем ВОЛОКИТА _____ (підпис)

РЕФЕРАТ
на магістерську дисертацію
виконану на тему:

Спосіб підвищення ефективності аналізу схожості текстів

Студентом: **Мірошником Денисом Валерійовичем**

Робота складається із вступу та чотирьох розділів. Загальний обсяг роботи: 67 аркушів основного тексту, 61 ілюстрацій. При підготовці використовувалася література з 15 різних джерел.

Актуальність. Оцінка семантичної схожості текстів це задача, рішення якої дуже часто використовується у ряді програмних рішень, з якими людина взаємодіє за допомогою власної людської мови. Такі рішення, як пошукові машини, AI чати, фільтрація текстів по темі у новинних агрегаторах, медіа-сайтах. Машинне розуміння схожості для пошукових машин генерує більшу кількість результатів кращої якості, AI чати краще розуміють контекст запитання, розпізнаючи складні семантичні конструкції запиту користувача. Агрегатори та фільтри контенту віднаходять споріднене медіа та пропонують його своїм користувачам.

Мета і завдання дослідження. Метою роботи є розробка архітектури нейромережі, яка передбачує схожість тексту.

Для досягнення мети було поставлено і розв'язано наступні завдання:

- Розглянути існуючі рішення , обрати інструменти для вирішення такого завдання.
- Створити нейромережу, здатну до оцінки семантичної схожості текстів
- Проаналізувати ефективність створеної нейромережі

Об'єктом дослідження є процес оцінки схожості текстів.

Предметом дослідження є спосіб оцінки семантичної схожості текстів з використанням нейронних мереж.

Особистий внесок здобувача. Магістерське дослідження є самостійно виконаною роботою, у якій відображається особистий авторський підхід та особисто отримані теоретичні та прикладні результати, що відносяться до вирішення задачі оцінки семантичної схожості текстів.

Наукова новизна: У даній магістерській роботі запропоновано новий спосіб, який за рахунок експериментального підбору комбінації моделі, вставки та оптимізатора, дозволяє підвищити результати оцінки семантичної схожості текстів.

Ключові слова. Нейронна мережа, нейромережа, рекурентна нейромережа, RNN, LSTM, MaLSTM, GRU, MaGRU, MaBiLSTM, MaBiGRU, word2vec, fasttext, GloVe.

ABSTRACT

for a master's thesis

performed on the topic: A method for enhancing text similarity estimation

by student: **Miroshnyk Denys Valeriyovych**

The work consists of an introduction and four chapters. Total amount of work: 67 sheets of the main text, 52 illustrations. Literature from 25 different sources was used in the preparation.

Actuality: Estimating the semantic similarity of texts is a task whose solution is very often used in a number of software solutions with which a person interacts using his own human language. Such solutions as search engines, AI chats, text filtering by topic in news aggregators, media sites. Machine understanding of similarity for search engines generates more results of better quality, chat AIs better understand the context of the question by recognizing the complex semantic constructions of the user's query. Content aggregators and filters find related media and offer it to their users.

The purpose and objectives of the study: The aim of the work is to develop a neural network architecture that predicts text similarity. To achieve the goal, the following tasks were set and solved:

- Consider existing solutions, choose tools for solving such a task.
- Create a neural network capable of evaluating the semantic similarity of texts
- Analyze the effectiveness of the created neural network

Object of study is the method of assessing the similarity of texts.

Subject of study is a method of evaluating the semantic similarity of texts using neural networks.

Personal contribution of the applicant. The master's research is an independently completed work, which reflects the author's personal approach and personally obtained theoretical and applied results related to solving the problem of evaluating the semantic similarity of texts.

Scientific novelty. In this master's thesis, a new method is proposed, which, due to the experimental selection of a combination of a model, an insert, and an optimizer, allows to improve the results of evaluating the semantic similarity of texts.

Keywords. Neural network, recurrent neural network, RNN, LSTM, MaLSTM, GRU, MaGRU, MaBiLSTM, MaBiGRU, word2vec, fasttext, GloVe.

**Пояснювальна записка
до магістерської дисертації**

на тему: «Спосіб підвищення ефективності аналізу схожості
текстів»

ЗМІСТ

Вступ	11
Розділ 1 Актуальність теми і сфери застосування	12
Висновок до розділу 1	26
Розділ 2 Аналіз способів глибокої обробки текстів	27
2.1. Рекурентні нейронні мережі	27
2.2. Проблема зникаючих градієнтів	29
2.3. Двонаправлені рекурентні нейронні мережі	32
2.4. LSTM: Long Short-Term Memory	34
2.5. Використані підходи в роботі. Порівняльна характеристика.	39
Висновок до розділу 2	42
Розділ 3 Опис архітектури моделі та наборів даних	43
3.1. Опис архітектури моделі	43
3.2. Word2vec вставка	45
3.3. GloVe вставка	49
3.4. Опис наборів даних використаних в роботі	53
Висновки до розділу 3	57
Розділ 4 Результати досліджень	58
4.1. Інструменти розробки представлені у роботі	58
4.2. Оптимізатори	59
4.3. Опис елементів програми та результати виконання	65
Висновок до розділу 4	73

Висновок.....	74
Список використаної літератури.....	75
Додаток 1. Графіки поширення передбачення схожості.....	78
Додаток 2. Таблиця результатів критеріїв оцінки ефективності моледей	92

ВСТУП

Semantic Text Similarity (STS) – семантична схожість текстів є важливим завданням у програмах обробки природних мов (NLP: Natural Language Processing), таких як пошук інформації, класифікація, вилучення, відповіді на запитання та виявлення плагіату. Завдання STS вимірює ступінь схожості між двома текстами і може бути виражено шкалою від 0 до 1, при цьому 0 вказує на те, що семантика речень повністю незалежна, а 1 це означає, що існує семантична еквівалентність.

STS є складною проблемою, оскільки мови мають численні неоднозначності та синонімічні вирази, тоді як речення можуть мати різну довжину та складні структури. Тому базові моделі, bag-of-words або TF-IDF обмежені своєю специфікою, яка відкидає роль порядку слів і ігнорує синтаксичні, а також семантичні зв'язки. Останні успіхи в схожості речень були досягнуті за допомогою нейронних мереж RNNs: Recurrent Neural Networks і CNN: Convolutional Neural Networks. Нейронні мережі (NN: Neural Networks) використовують глибокий аналіз речень і слів, щоб краще врахувати як семантику, так і структуру речень, щоб передбачити подібність речень.

РОЗДІЛ 1

АКТУАЛЬНІСТЬ ТЕМИ І СФЕРИ ЗАСТОСУВАННЯ

Обробка природної мови (Natural Language Processing, NLP) – це галузь комп'ютерних наук, а точніше штучного інтелекту, що займається наданням комп'ютерам можливості розуміти тексти та розмовну мову так само, як і люди. Обробка природної мови поєднує обчислювальну лінгвістику (моделювання людської мови на основі правил) зі статистичним моделюванням, машинним навчанням і моделями глибокого навчання. Разом ці методи дозволяють комп'ютерам обробляти людську мову у вигляді тексту та мовних даних і "розуміти" її повне значення, а також наміри та емоції мовця або автора. Обробка природної мови лежить в основі комп'ютерних програм, що перекладають тексти між людськими мовами, відповідають на голосові команди і швидко узагальнюють великі обсяги тексту в режимі реального часу, таких як голосові системи GPS, цифрові асистенти, програмне забезпечення для перетворення голосу в текст, чат-боти для обслуговування клієнтів та інші форми споживчої зручності – багато хто з нас стикався з NLP. NLP починає мати важливе місце рішеннях, які допомагають впорядкувати діяльність, підвищити продуктивність праці співробітників і спростити критичні бізнес-процеси.

Людська мова сповнена неоднозначностей, що робить надзвичайно складним написання програм, що визначають значення тексту та усних даних. Синоніми, омофони, іронія, ідіоми, метафори, граматичні винятки, відмінності у структурі речень – це лише деякі з нерівностей людської мови, які люди вивчили протягом багатьох років. Однак програмісти повинні бути навчені розпізнавати і правильно розуміти з самого початку, чи корисні програми на природній мові, чи ні. Деякі завдання обробки природної мови розбивають

людські тексти та мовні дані на частини таким чином, щоб допомогти комп'ютеру зрозуміти, що він отримує.

Деякі з цих завдань включають наступне:

- **Розпізнавання мовлення**, також відоме як Speech-to-Text, - це процес надійного перетворення усних даних у текст. Розпізнавання мовлення необхідне для додатків, які виконують голосові команди або відповідають на запитання. Розпізнавання мовлення є особливо складним для людського мовлення, наприклад, швидкого і нечіткого мовлення, різних наголосів та інтонацій, різних акцентів і часто неправильної граматики.
- **Позначення частин мови**, також відоме як граматичне позначення, – це процес визначення частини мови певного слова або фрагмента тексту на основі його вживання та контексту. Частина мови визначає слово «make» як дієслово у вислові «I can make a paper plane» і як іменник у вислові «What make of car do you own?»
- **Розділення слів** – це процес вибору значень багатозначних слів за допомогою семантичного аналізу для визначення слова, яке має найбільше сенсу в даному контексті. Наприклад, усунення неоднозначності слів допомагає розрізнити значення дієслова «make» у «make the grade» (досягти) від «make a bet» (поставити).
- **Розпізнавання іменованих сутностей**, визначає слова або фрази як корисні сутності. Розпізнавання іменованих об'єктів визначає «Кентуккі» як місцезнаходження або «Фред» як ім'я людини.
- **Розділення співпосилання** – це завдання визначення того, чи і коли два слова відносяться до однієї сутності. Найпоширенішим прикладом є визначення особи чи предмета, до якого відноситься

певний займенник (наприклад, «вона» = «Мері»), але це також може включати ідентифікацію метафори чи ідіоми в тексті.

- **Аналіз настроїв** намагається витягнути з тексту суб'єктивні якості – ставлення, емоції, сарказм, збентеження, підозру.

Генерація природної мови, яку іноді описують як протилежність розпізнавання мови або Speech-to-Text, є завданням перекладу структурованої інформації на людську мову.

Обробка природної мови є рушієм глибокого навчання в багатьох сучасних застосуваннях. Ось кілька прикладів:

- **Виявлення спаму.** Системи виявлення спаму використовують глибоке розпізнавання схожості тексту для виявлення та фільтрації небажаного або шкідливого вмісту в електронних листах, повідомленнях та інших формах цифрового зв'язку. Виявлення спаму за допомогою глибокого розпізнавання схожості тексту аналізує текстовий вміст вхідних повідомлень і порівнює його з наявними наборами даних про відомі спам-повідомлення. Для цього використовуються різні лінгвістичні ознаки, такі як вибір слів, структура речень і контекст. Алгоритми, що використовуються для глибокого розпізнавання схожості тексту при виявленні спаму, зазвичай є моделями машинного навчання, навченими на великих маркованих наборах даних. Ці навчальні дані складаються як з легальних, так і зі спам-повідомлень і використовуються для навчання моделей розпізнаванню шаблонів і ознак, характерних для спам-повідомлень. Після навчання модель можна використовувати для аналізу нових вхідних повідомлень у режимі реального часу. Модель обчислює оцінку, яка відображає схожість між новим повідомленням і наявним набором даних спам-повідомлень. Якщо оцінка перевищує певний поріг, повідомлення позначається як спам і або фільтрується в

окрему папку спаму, або негайно видаляється. Системи виявлення спаму можуть також використовувати додаткові методи, такі як аналіз репутації відправника або поведінковий аналіз, щоб підвищити точність алгоритму виявлення спаму. Загалом, глибоке розпізнавання схожості тексту є важливим компонентом систем виявлення спаму, що дозволяє їм швидко і точно ідентифікувати небажаний або шкідливий вміст і захищати користувачів від потенційної шкоди.

- Машинний переклад: використовує глибоке розпізнавання схожості тексту для розуміння вхідного тексту однією мовою та створення еквівалентного вихідного тексту іншою мовою. Глибинне розпізнавання схожості тексту дозволяє системі аналізувати вхідний текст і визначати оптимальний еквівалент для вихідного тексту на основі лінгвістичних особливостей, таких як вибір слів, структура речень і контекст.

Машинний переклад із застосуванням глибокого розпізнавання схожості тексту зазвичай використовує нейромережеві моделі, такі як архітектури кодер-декодер. Ці моделі навчаються на великих паралельних текстових базах даних, що складаються з пар текстів вихідної мови та перекладів цільової мови. Під час навчання моделі вчаться розпізнавати шаблони та особливості у вхідному тексті, які вказують на певний переклад цільовою мовою. Це відбувається за допомогою методів глибокого розпізнавання схожості текстів для виявлення схожих шаблонів між вихідним і цільовим текстами, беручи до уваги контекст і значення слів і речень. Після того, як модель навчена, її можна використовувати для перекладу щойно введеного тексту в режимі реального часу. Вхідний текст спочатку кодується в багатовимірне представлення за допомогою нейронної мережі, а потім мережа-декодер створює вихідний текст цільовою мовою. Вихідний текст створюється по одному слову або реченню за раз, і кожен крок визначається контекстом і значенням попереднього слова.

Глибоке розпізнавання схожості тексту має вирішальне значення для успішного машинного перекладу. Це пов'язано з тим, що воно дає змогу системі точно вловити значення й контекст вхідного тексту та створити лінгвістично й семантично послідовний та еквівалентний вихідний текст. Хоча машинний переклад залишається складною проблемою, останні досягнення в галузі глибокого розпізнавання схожості текстів і архітектури нейронних мереж значно покращили якість систем машинного перекладу.

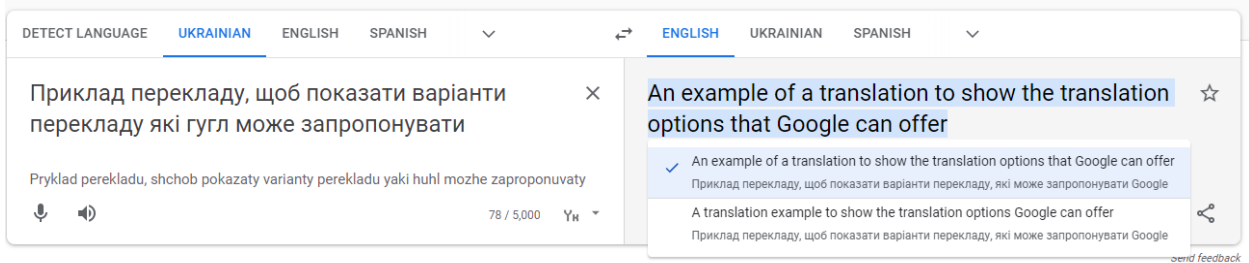


Рис. 1.1. Google Translate пропонує декілька перекладів.

На рисунку показано два варіанти перекладу речення англійською мовою. Обидва варіанти є підходящими, адже система розуміє важливість порядку слів та оцінює ці переклади як подібні, або ідентичні.

- Віртуальні агенти та чат-боти: такі віртуальні агенти, як Siri від Apple і Alexa від Amazon, розпізнають шаблони у голосових командах та генерують голос, щоб відповідати відповідними діями чи корисними коментарями.

Наступним удосконаленням для цих програм є можливість відповідати на запитання доречними та корисними відповідями своїми словами. Одна із найновіших систем чат-ботів є ChatGPT.

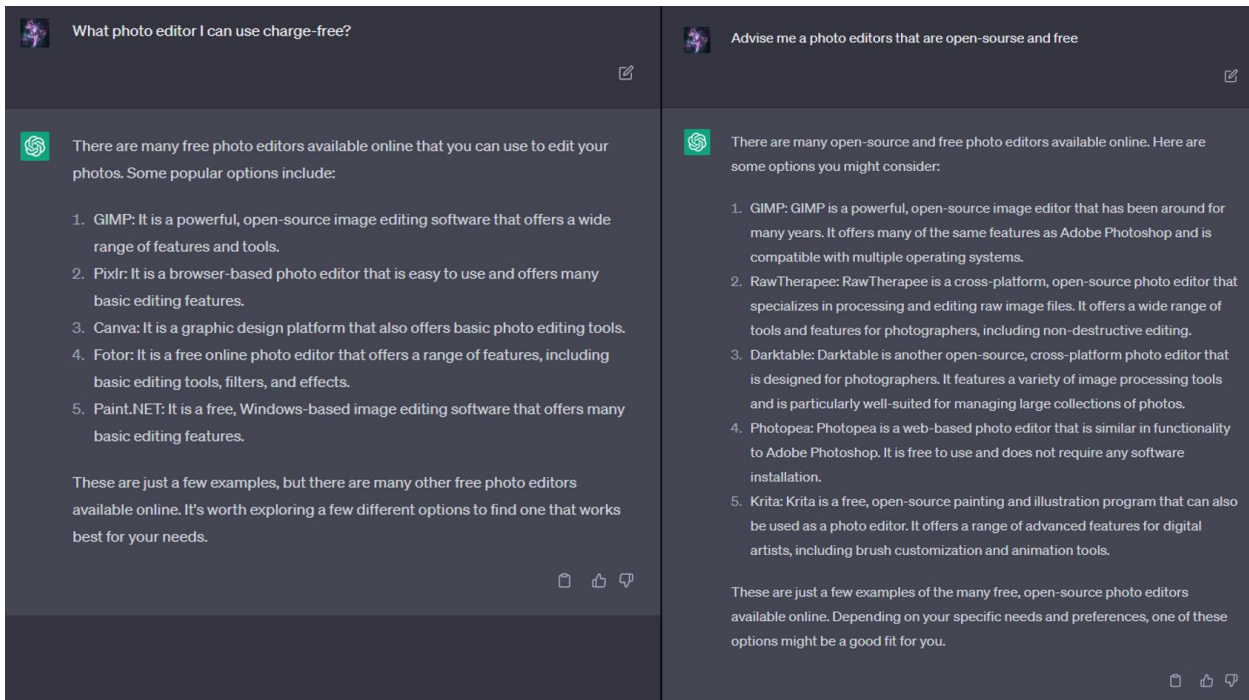


Рис. 1.2. Відповіді ChatGPT на запити.

Важливою складовою є розпізнавання семантичних особливостей запиту. На рисунку 1.2 чат-боту було задано два запити: “What photo editor I can use charge-free”, “Advise me a photo editors that are open source and free”. Питання були задані в окремі сесії чату, то ж питання не були зв’язані між собою однією розмовою. Однак, система розпізнала такі питання, як “free” та “charge-free” як розмову про вартість ліцензії програми фото-редактора, також формати запитів є різними: перше – це питання, а друге – ні. Результати при цьому дуже подібні, адже і питання подібні: користувач шукає варіанти фото редактора, які безкоштовні.

ChatGPT використовує глибоке розпізнавання схожості тексту, щоб розуміти введення користувача і генерувати відповідні відповіді. Глибоке розпізнавання схожості тексту аналізує вхідний текст і порівнює його з великим набором текстових даних, на яких навчається модель. Це порівняння базується на різних лінгвістичних особливостях, таких як вибір слів, структура речень і контекст.

Для реалізації глибокого розпізнавання схожості тексту ChatGPT використовує архітектуру нейронної мережі, спеціально розроблену для задач обробки природної мови (NLP). Ця архітектура називається архітектурою-трансформером і лежить в основі моделі GPT-3.5.

Трансформаторна архітектура використовує механізм самоконтролю для зважування важливості різних частин вхідного тексту при створенні вихідних даних. Це дозволяє моделі ідентифікувати шаблони та особливості, пов'язані з вхідним текстом, навіть якщо вони не очевидні одразу.

Після того, як відповідні особливості вхідного тексту визначено, ChatGPT порівнює їх з текстовим корпусом за допомогою метрики схожості. Метрика схожості обчислює оцінку, яка показує, наскільки вхідний текст схожий на кожен ознаку в корпусі. Потім ця оцінка використовується для вибору найбільш відповідної відповіді на введення користувача.

В цілому, глибоке розпізнавання схожості тексту є важливим компонентом здатності ChatGPT розуміти природну мову і генерувати відповідні відповіді. Використовуючи можливості нейромережових архітектур, таких як Transformer, ChatGPT здатний аналізувати і розуміти складні нюанси людської мови і генерувати контекстуально і лінгвістично відповідні відповіді.

- Аналіз настроїв у соціальних мережах: Платформи соціальних мереж використовують глибоке розпізнавання схожості текстів різними способами, зокрема для покращення взаємодії з користувачами, посилення модерації контенту та підтримки таргетованої реклами.

Одне з основних застосувань глибокого розпізнавання схожості текстів у соціальних мережах – це аналіз створеного користувачами контенту та визначення релевантних тем. Це дозволяє соціальним мережам рекомендувати користувачам релевантний контент на основі їхніх інтересів і поведінки. Наприклад, якщо користувач часто публікує пости

про фітнес і харчування, платформа може запропонувати відповідний контент, наприклад, тренування, корисні рецепти і харчові добавки.

Глибоке розпізнавання схожості тексту також використовується для управління контентом на платформах соціальних мереж. Платформи можуть використовувати цю технологію для автоматичного позначення потенційно проблемного контенту, такого як мова ворожнечі, переслідування і погрози насильства. Модель навчається на великому наборі даних позначеного контенту, щоб виявити закономірності та характеристики, пов'язані з проблемним контентом. Після цього платформа може видалити такий контент або вжити інших відповідних заходів, наприклад, видати попередження або призупинити дію облікових записів користувачів.

Іншим важливим випадком використання глибокого розпізнавання схожості текстів у соціальних мережах є таргетована реклама. Аналізуючи контент, створений користувачами, соціальні медіа-платформи можуть визначати відповідні інтереси та поведінку і відповідно таргетувати рекламу. Наприклад, якщо користувач часто публікує пости про подорожі, платформа може показувати рекламу товарів і послуг, пов'язаних з подорожами, таких як авіаквитки, бронювання готелів і путівки на відпочинок.

Загалом, глибоке розпізнавання схожості текстів є важливим інструментом для платформ соціальних мереж, який дозволяє аналізувати контент, створений користувачами, розуміти їхню поведінку та надавати їм персоналізований досвід. Використовуючи цю технологію, платформи можуть покращити модерацію контенту, підвищити залученість користувачів і збільшити доходи від таргетованої реклами.

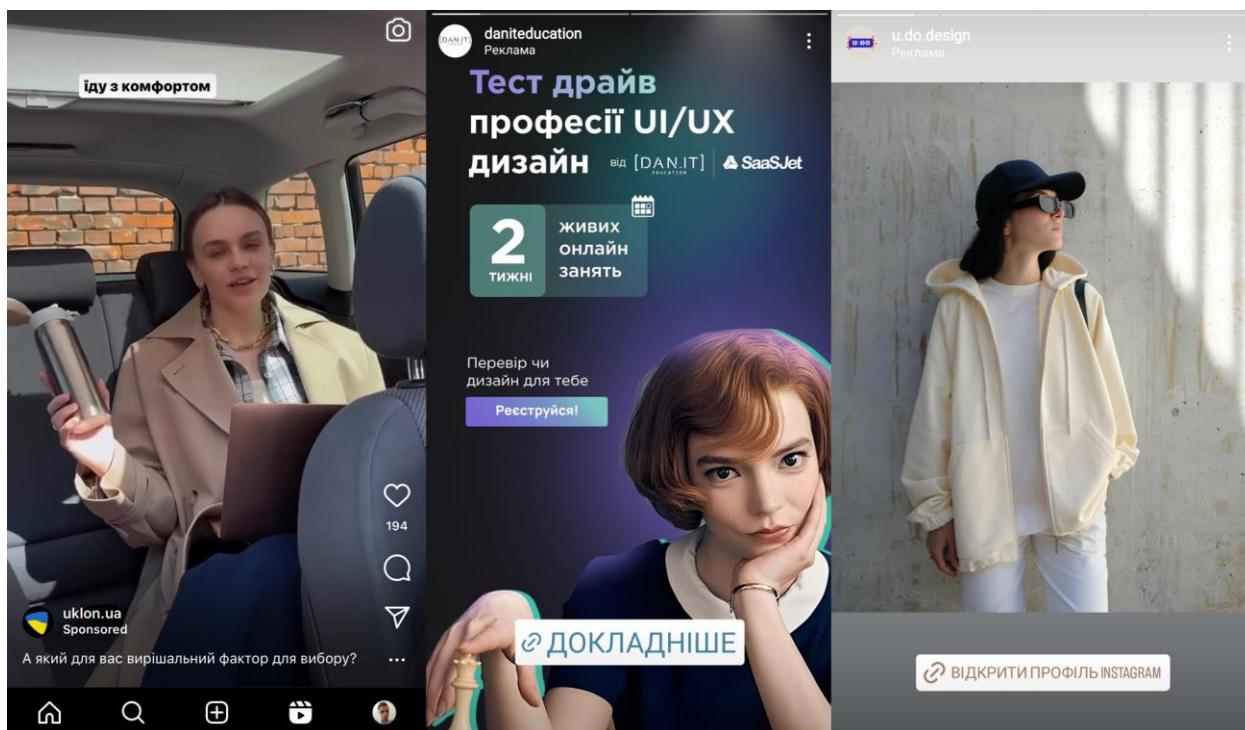


Рис. 1.3. Персоналізована реклама для користувачів в Інстаграм

Така соціальна мережа, як Інстаграм пропонує таргетовану, персоналізовану рекламу для кожного користувача, спираючись на відомості, такі як: з якими пропозиціями користувач взаємодіяв раніше, які товари шукав, якими послугами користувався, а також хобі, спорт, відпочинок, здоров'я, тощо.

На рисунку 1.3. проілюстровано три різних реклами: таксі, курси з навчання, одяг. Така система пропозиції реклами оцінює подібність контенту та контексту кожної реклами. Наприклад реклама Uber, Uklon та інших сервісів таксі як подібні, якщо раніше користувач часто користувався сервісом таксі Bolt. Аналогічні зацікавленості користувача як одяг до таксі не мають схожості і реклама Uklon для такого користувача – не релевантна.

- Резюмування тексту: використовує методи обробки природної мови для перероблення величезних обсягів цифрового тексту та створення резюме та конспектів для показчиків, баз даних досліджень або для

зайнятих читачів, які не мають часу читати повний текст. Найкращі програми для резюмування тексту використовують семантичне міркування та створення природної мови, щоб додати корисний контекст і висновки до резюме.

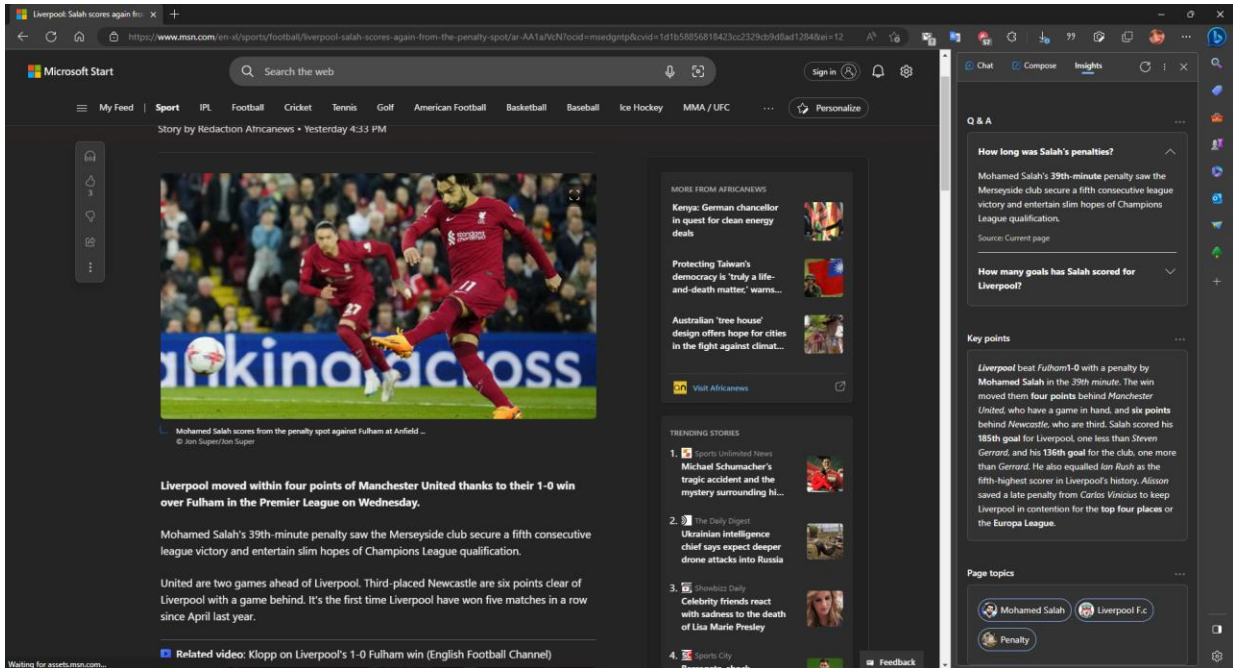


Рис. 1.4. Microsoft Bing Edge представляє резюмування тексту

Один із прикладів є функція браузера Microsoft Edge. При відкритті будь-якої сторінки з текстом: новини, наукові статті, тощо, Bing Discover Insights представить резюмування написаного тексту на сторінці у вигляді Q&A та ключових тез.

- **Перевірка на плагіат:** процес порівняння тексту з іншими джерелами для виявлення випадків скопійованого або неоригінального контенту. Хоча це завдання може здатися простим, насправді воно є досить складним і може зайняти багато часу, особливо коли йдеться про великі обсяги тексту.

Поширений підхід до перевірки плагіату за допомогою NLP полягає у використанні алгоритмів машинного навчання для виявлення шаблонів у

тексті, які вказують на плагіат. Наприклад, алгоритми можуть шукати певні фрази або структури речень, які часто зустрічаються в плагіатному контенті. Крім того, для навчання алгоритму можна використовувати великий масив тексту, щоб краще зрозуміти, яким є оригінальний контент. Інший підхід до перевірки плагіату за допомогою NLP полягає у використанні метрик схожості тексту для порівняння певного тексту з іншими джерелами. Алгоритми можуть виявляти випадки скопійованого контенту, обчислюючи схожість між двома текстами за допомогою показника схожості текстів. Окрім виявлення плагіату, методи NLP також можна використовувати для створення звітів, які виділяють певні частини тексту, що не є оригінальними. Це особливо корисно для викладачів і дослідників, які хочуть переконатися, що їхня робота є оригінальною і не містить плагіату. Загалом, перевірка на плагіат – це завдання, яке добре підходить для технології NLP. Завдяки здатності швидко і точно аналізувати великі обсяги тексту, NLP допомагає забезпечити оригінальність і високу якість письмових робіт.

UNICHECK

63 pages left | Buy More Pages

Active until: September 10, 2019

My Library > Diploma > Papers & Projects > Cultivate back-end methodologies.docx

Last checked Jul 12, 2018 | 10 621 Words | 102 Paid pages

Plagiarism Citation Reference Character replacement

violations. Therefore, we should constantly improve methods of plagiarism spotting to remain competitive and to meet the challenges of the times.

"The potential for the disastrous rise of misplaced power exists and will persist." - Dwight D. Eisenhower, January 17, 1961; Washington, D.C.

When we started our company we realized that there wasn't a real competition in the market and all alternatives were relatively conceptually similar to each other, so to breathe a new life into this market someone had to create a brand new concept of plagiarism checker.

"There results a general attitude either of cynical belief in and indifference to public corruption or else of a distrustful inability to discriminate between the good and the bad." - Theodore Roosevelt, April 14, 1906; Washington, D.C.

And it's absolutely insignificant whether the person plagiarized someone's intellectual work intentionally or simply didn't take the copyright seriously enough, in both cases it leads to establishing of irresponsible attitude towards the intellectual property.

There's nothing wrong in using parts of someone else's work, knowledge or experience as long as it is properly indicated and helps to create something new. The whole concept of science based on evolution and continuity. So it's undoubtedly that you've got to always go back in time if you want to move forward. But in order not to get stuck in the past you need to reconsider it not replicate

SIMILARITIES CITATIONS REFERENCES

9% of Citations

Sources 1,460
Similarities 76%
References 2%

DETECT CITATIONS

- 1 It is not what you do for your children, but what you
- 2 First, have a definite, clear practical ideal; a goal, an

EXCLUDE CITATION

First, have a definite, clear practical ideal; a goal, an objective. Second, have the necessary means to achieve your ends; wisdom, money, materials, and methods. Third, adjust all your means to that end. -Aristotle

- 3 Twenty years from now you will be more
- 4 I've missed more than 9000 shots in my career. I've

Рис. 1.5. Робота сервісу Unicheck

Unicheck – це хмарне програмне забезпечення для виявлення плагіату, яке може перевірити оригінальність академічних робіт, дослідницьких робіт або будь-якого письмового контенту. Програма перевіряє текст на наявність плагіату за кількома онлайн-джерелами та базами даних. Завдяки таким функціям, як зручний дизайн, інтеграція з системами управління навчанням (LMS), інструменти оцінювання та підтримка різних форматів файлів, Unicheck широко використовується навчальними закладами, видавництвами та бізнесом для забезпечення оригінальності контенту.

The screenshot shows the Turnitin interface. At the top, it says 'feedback studio' and 'Joe Bloggs | The Goliath of the Sea'. The document title is '20 /100' and '1 of 30'. The main content area displays three paragraphs of text about blue whales, with several lines highlighted in pink to indicate matches. On the right, a 'Match Overview' panel shows a total similarity score of 45% and a list of four sources:

Rank	Source	Similarity
1	animals.nationalgeogr... Internet Source	14%
2	agaunews.com Internet Source	12%
3	pro-solutions.texthelp... Internet Source	12%
4	animals-partner.blogsp... Internet Source	7%

Рис. 1.7. Робота сервісу Turnitin

Turnitin – це хмарне програмне забезпечення для виявлення плагіату, яке дозволяє користувачам перевіряти оригінальність свого письмового контенту. Програмне забезпечення широко використовується навчальними закладами для забезпечення оригінальності студентських робіт, а Turnitin має інтеграцію з системами управління навчанням (LMS), інструментами оцінювання та підтримує різні формати файлів.

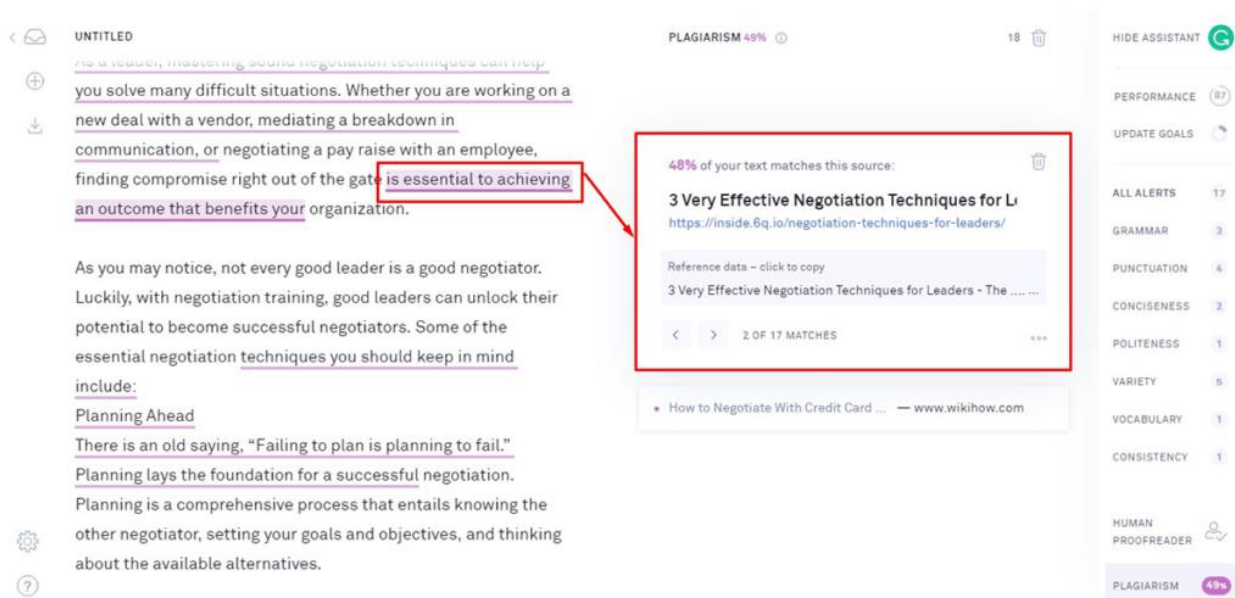


Рис. 1.8. Робота сервісу Grammarly

Grammarly – це інструмент-помічник для написання текстів, який перевіряє граматичні, орфографічні, пунктуаційні та інші помилки. Він також пропонує перевірку на плагіат, щоб перевірити оригінальність контенту, написаного користувачем. Grammarly широко використовується студентами, письменниками та бізнесом для забезпечення якості та оригінальності контенту.

ВИСНОВОК ДО РОЗДІЛУ 1

Обробка природної мови це передова у цифровому світі технологія, яка має велику кількість застосувань:

- Розпізнавання мовлення
- Позначення частини мови
- Усунення неоднозначності слова
- Розпізнавання іменованих сутностей
- Розділення співпосилання
- Аналіз настроїв
- Генерація природної мови

Одним із практичних застосувань є перевірка на плагіат. Такі сервіси як Unicheck, Turnitin та Grammarly дозволяють оцінити фрагменти поданого тексту на плагіаризм, повторюваність та подібність до інших текстів.

Семантична подібність відіграє велику роль у оцінці схожості тексту на інші, таким чином актуальною є задача розробки нового способу оцінки семантичної схожості текстів з використанням глибокого навчання.

РОЗДІЛ 2

АНАЛІЗ СПОСОБІВ ГЛИБОКОЇ ОБРОБКИ ТЕКСТІВ

2.1. Рекурентні нейронні мережі

RNN (Recurrent Neural Networks – рекурентні нейронні мережі) – це тип архітектури нейронної мережі, призначений для обробки безперервних даних, таких як текст або мова. На відміну від CNN, які мають справу з вхідними даними фіксованого розміру, RNN можуть обробляти вхідні послідовності змінної довжини і зберігати внутрішній стан або пам'ять про попередні вхідні дані, таким чином фіксуючи довгострокові залежності в даних.

У контексті завдань NLP RNN особливо корисні для обробки текстових даних, дозволяючи моделювати і вивчати зв'язки між словами або символами в реченні або документі. Зберігаючи внутрішню пам'ять і стан, RNN можуть отримувати інформацію з попередніх епізодів послідовності і використовувати її для прогнозування наступних епізодів послідовності.

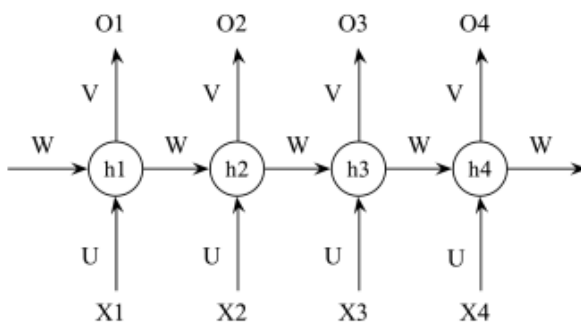


Рис. 2.1. Типова RNN

Порівняно з CNN, RNN краще пристосовані для обробки послідовних даних з довгостроковими залежностями, що дозволяє їм ефективніше виконувати завдання NLP. Це пов'язано з тим, що внутрішня пам'ять RNN дозволяє їй отримувати інформацію з

попередніх епізодів послідовності і використовувати її для прогнозування наступних епізодів послідовності. З іншого боку, CNN зазвичай обробляють

локальні фрагменти вхідних даних і не мають пам'яті про попередні вхідні дані.

Відмінністю рекурентних мереж є спільне використання параметрів на всіх шарах мережі. У той час як мережі прямого зв'язку мають різні параметри у кожному вузлі, рекурентні нейронні мережі мають однаковий параметр ваги на кожному шарі мережі.

Ваги коригуються за допомогою процесів зворотного поширення та градієнтного спуску. Рекурентні мережі користуються алгоритмом зворотного поширення в часі (BPTT – backpropagation through time) для визначення градієнтів. Алгоритм є спеціальним для роботи з даними, що містять послідовності: коли модель навчається, обчислюючи помилки від вихідного до вхідного рівня, що дозволяє корегувати параметри моделі. BPTT підсумовує помилки на кожному кроці часу, оскільки параметри є спільними на кожному рівні.

Рекурентні нейромережі стикаються з двома проблемами: вибухові градієнти та зникаючі градієнти. Коли градієнт занадто малий, зміни вагів параметрів наближаються до нуля. Коли градієнт занадто великий ваги моделі виростуть занадто великими. В обох випадках алгоритм більше не коректно впливає на навчання. Одним із рішень цих проблем є зменшення кількості прихованих шарів у нейронній мережі, усунення частини складності в моделі RNN.

На рисунку 2.1 показано, що RNN розгортається в повну мережу. Наприклад, якщо вхідна послідовність являє собою речення з чотирьох слів, мережа буде розгорнута в 4-шарову нейронну мережу, один шар для кожного слова. Формули, які керують обчисленнями в RNN, є такими:

- x_t є вхідним сигналом на етапі часу t . Наприклад, x_t може бути вектором або вбудовуванням слів, що відповідає i -му слову речення.
- h_t – це прихований стан і «пам'ять» мережі на етапі часу, який

$$h_t = \sigma(Ux_t + Wh_{t-1})$$

розраховується відповідно до попереднього значення прихованого стану та значенням введеним на поточному кроці: Тут h_0 зазвичай ініціалізується нульовим вектором, щоб обчислити перший прихований стан.

- o_t – вихід на етапі часу t . Для класифікації настроїв коротко текстів, це був би вектор ймовірностей для всіх настроїв категорії. o_t розраховується так:

$$o_t = \text{softmax}(Vh_t)$$

2.2. Проблема зникаючих градієнтів

Градієнт – це вектор, який представляє напрямок і величину найкрутішого підйому функції в даній точці. У контексті нейронних мереж градієнт функції втрат щодо параметрів моделі використовується для оновлення параметрів під час навчання, щоб мінімізувати функцію втрат і покращити продуктивність моделі.

Проблема зникнення градієнта, яка може виникнути під час навчання глибоких нейронних мереж, включаючи RNN, відноситься до явища, коли градієнт функції втрат відносно параметрів моделі стає дуже малим, що ускладнює ефективне оновлення параметрів у мережі та призводить до у повільному або неефективному тренуванні.

Після того, як рекурентна нейромережа виведе вектор передбачення $h(k)$, ми обчислюємо помилку передбачення $E(k)$ і використовуємо алгоритм зворотного поширення через час для обчислення градієнта.

$$\frac{\partial E}{\partial W} = \sum_{t=1}^T \frac{\partial E_t}{\partial W}$$

Градієнт використовується для оновлення параметрів моделі:

$$W \leftarrow W - \alpha \frac{\partial E}{\partial W}$$

І продовжуємо процес навчання за допомогою градієнтного спуску, навчальне завдання, яке включає T часових кроків, задано градієнт помилки на k часовому кроці.

$$\begin{aligned} \frac{\partial E_k}{\partial W} &= \frac{\partial E_k}{\partial h_k} \frac{\partial h_k}{\partial c_k} \dots \frac{\partial c_2}{\partial c_1} \frac{\partial c_1}{\partial W} \\ &= \frac{\partial E_k}{\partial h_k} \frac{\partial h_k}{\partial c_k} \left(\prod_{t=2}^k \frac{\partial c_t}{\partial c_{t-1}} \right) \frac{\partial c_1}{\partial W} \end{aligned}$$

Оскільки $W=[W_{rec}, W_{in}]$, $c(t)$ можна записати як

$$c_t = \sigma (W_{rec} \cdot c_{t-1} + W_{in} \cdot x_t)$$

Похідна від $c(t)$:

$$\begin{aligned}\frac{\partial C_t}{\partial C_{t-1}} &= \sigma'(W_{rec} \cdot c_{t-1} + W_{in} \cdot x_t) \cdot \frac{\partial}{\partial C_{t-1}} [W_{rec} \cdot c_{t-1} + W_{in} \cdot x_t] \\ &= \sigma'(W_{rec} \cdot c_{t-1} + W_{in} \cdot x_t) \cdot W_{rec}\end{aligned}\quad (2)$$

Маємо з (1) і (2) градієнт зворотного поширення:

$$\frac{\partial E_k}{\partial W} = \frac{\partial E_k}{\partial h_k} \frac{\partial h_k}{\partial c_k} \left(\prod_{t=2}^k \sigma'(W_{rec} \cdot c_{t-1} + W_{in} \cdot x_t) \cdot W_{rec} \right) \frac{\partial c_1}{\partial W}$$

Останній вираз має прямує до нуля, коли k велике, це пов'язано з похідною функції активації \tanh , яка менша за 1. Добуток похідних також може вибухнути, якщо вагові коефіцієнти W_{rec} достатньо великі, щоб подолати меншу похідну \tanh , це відомо як проблема вибухового градієнта.

Існує декілька методів, які можуть бути використані для вирішення проблеми зникнення градієнта в RNN:

- Відсікання градієнта: Цей метод передбачає масштабування градієнта так, щоб він не перевищував певний поріг. Обмежуючи розмір градієнта, він запобігає тому, щоб градієнт став занадто малим або занадто великим, таким чином покращуючи стабільність процесу навчання.
- Методи ініціалізації: за допомогою правильної ініціалізації параметрів мережі можна уникнути проблеми зникнення градієнта. Наприклад, ініціалізація ваг мережі малими випадковими значеннями та використання відповідних розподілів може запобігти надто малому градієнту.

- Використання альтернативних функцій активації: Проблему зникаючого градієнта можна зменшити, використовуючи альтернативні функції активації, такі як ReLU. Ці функції активації, як правило, створюють більші градієнти, ніж традиційні сигмоїдні або тангенціальні функції активації.
- Використання альтернативних архітектур RNN: Існує кілька альтернативних архітектур RNN, таких як LSTM та GRU, призначених для полегшення проблеми затухання градієнта. Ці архітектури використовують механізм воріт, який дозволяє мережі вибірково зберігати та оновлювати інформацію з часом, що може покращити стабільність та ефективність навчання.

2.3. Двонаправлені рекурентні нейронні мережі

Двонаправлені рекурентні нейронні мережі (BRNN) [9] – це тип нейромережевої архітектури, який може вловлювати часові залежності в послідовних даних, таких як природна мова, мовлення та дані часових рядів. На відміну від традиційних рекурентних нейронних мереж (RNN), які обробляють послідовні дані в одному напрямку, BRNN обробляють дані як в прямому, так і в зворотному напрямку, що робить їх ефективними для моделювання складних послідовних даних.

Основною перевагою BRNN над традиційними RNN є їхня здатність фіксувати залежності, що виникають як у минулому, так і в майбутньому контексті, що робить їх придатними для моделювання складних послідовностей з довготривалими залежностями. Наприклад, при обробці природної мови BRNN можуть краще вловлювати значення слів, враховуючи слова до і після них у реченні.

Ще одна перевага BRNN полягає в тому, що вони можуть краще обробляти вхідні послідовності змінної довжини. Це пов'язано з тим, що

BRNN можуть обробляти вхідні послідовності в обох напрямках і об'єднувати виходи прямого і зворотного шляхів для формування векторного представлення фіксованого розміру, яке може бути використано як вхідні дані для наступного рівня мережі.

Існують різні способи реалізації BRNN, але найпоширенішим є використання двох окремих RNN, один для обробки вхідної послідовності в прямому напрямку, а інший для обробки вхідної послідовності в зворотному напрямку. Потім виходи двох RNN об'єднуються, щоб отримати вихід кінцевої BRNN. Такий підхід іноді називають "двопрохідною" або "дзеркальною" архітектурою.

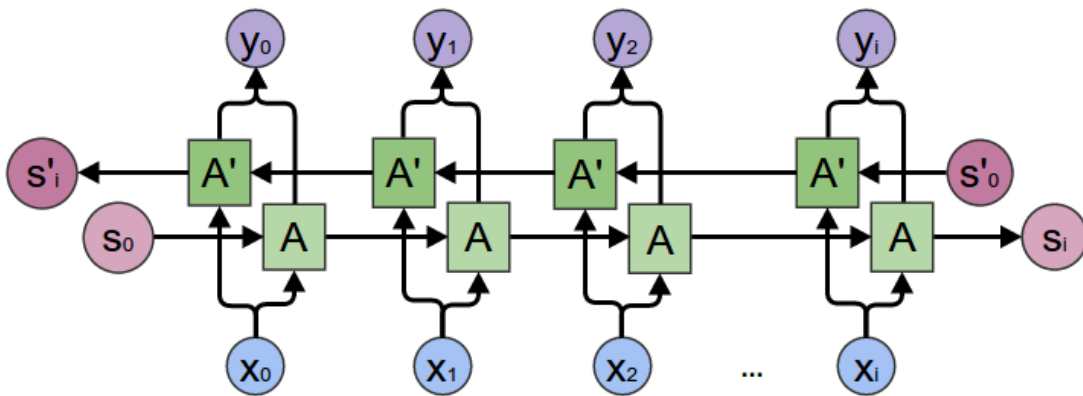


Рис. 2.2. Двонаправлена нейромережа.

На рисунку вище шар S – шар прямого проходження, а S' – шар зворотного проходження.

На додаток до вищезазначених переваг, було доведено, що BRNN ефективні в різних додатках, таких як розпізнавання мови, аналіз настрою та машинний переклад. Однак вони є більш трудомісткими, ніж традиційні RNN, через необхідність обробляти вхідні послідовності в обох напрямках.

Таким чином, BRNN – це потужний тип нейромережевої архітектури, який може фіксувати залежності як у минулому, так і в майбутньому контекстах і підходить для моделювання складних послідовних даних.

Порівняно з традиційними RNN, BRNN можуть обробляти вхідні послідовності змінної довжини. Вони мають низку переваг, зокрема кращу продуктивність у завданнях, де потрібно моделювати довгострокові залежності. Це робить їх цінним інструментом у багатьох додатках глибокого навчання.

2.4. LSTM: Long Short-Term Memory

Мережі довгострокової короткострокової пам'яті [5] – скорочено LSTM – це тип рекурентної нейромережі, створений спеціально для роботи з довгими послідовностями з залежностями, які довгострокові, та уникають проблем при роботі з ними. Запам'ятовування у тривалому часі є їхньою поведінкою за замовчуванням. Усі рекурентні нейронні мережі є зв'язаними між собою повторюваними шарами. У типових рекурентних нейромереж повторюваний шар містить дуже просту структуру – \tanh .

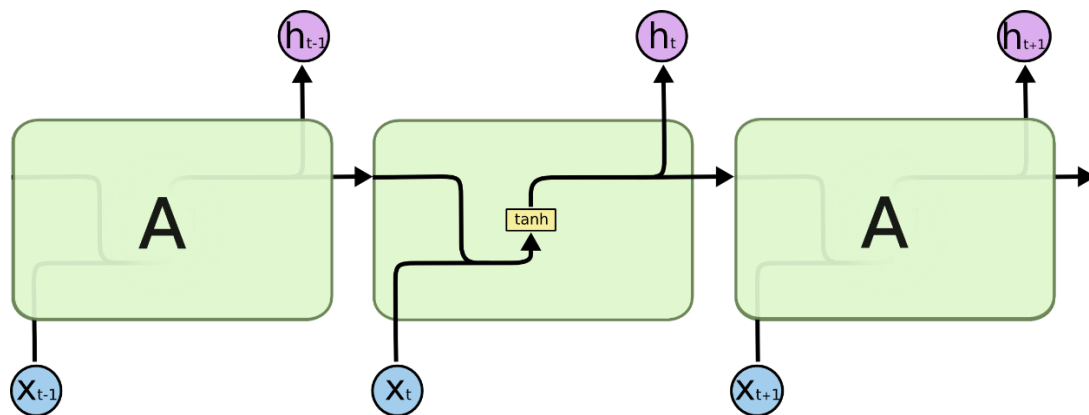


Рис. 2.3. Повторюваний модуль у стандартній RNN, що містить один шар[17]

LSTM також мають таку структуру, проте кожен шар має значно складнішу будову. Замість одної складової нейронної мережі існує чотири, які взаємодіють особливим чином.

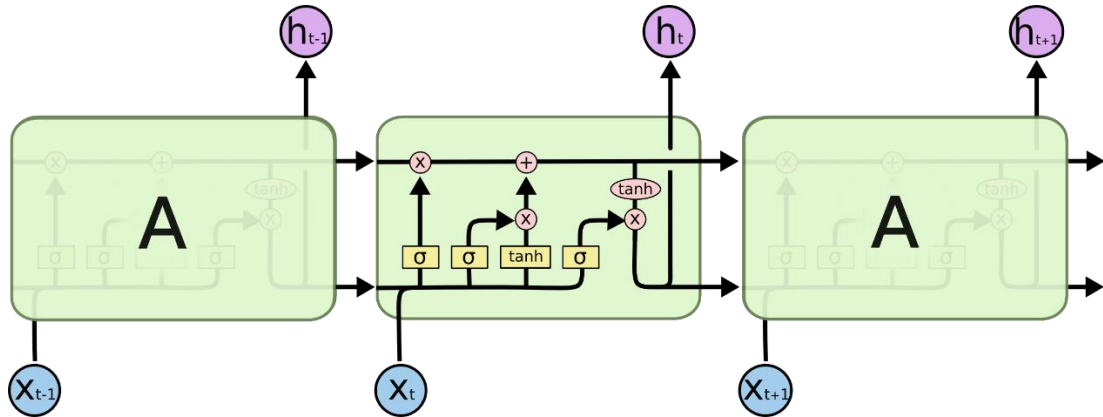


Рис. 2.4. Повторюваний модуль у LSTM містить чотири взаємодіючі рівні[17]

На наведеному вище рисунку кожен рядок містить цілий вектор, від виходу одного вузла до входів інших. Рожеві кружечки представляють поточкові операції, як-от додавання векторів, а жовті прямокутники – вивчені шари нейронної мережі. Рядки, що об'єднуються, означають конкатенацію, тоді як розгалуження рядків означає, що його вміст копіюється, а копії спрямовуються в різні місця.

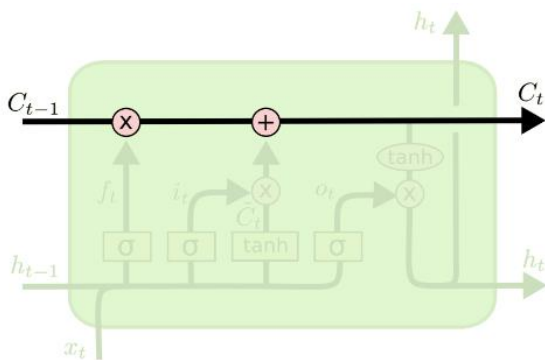


Рис. 2.5. Стан комірки [17]

Важливою частиною LSTM є стан комірки, який являє собою горизонтальну лінію у верхній частині діаграми. Стан комірки схожий на конвеєрну стрічку, що проходить прямо через весь ланцюжок з незначними лінійними взаємодіями. Інформацію передається без змін.

LSTM дійсно має здатність видаляти або додавати інформацію про стан комірки, ретельно регулюючи структури, які називаються воротами. Ворота – це необов'язковий спосіб передачі інформації. Він складається з сигмоїдального шару нейронної мережі та операції потокового множення.

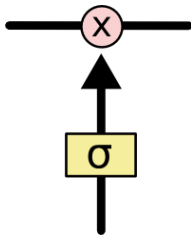
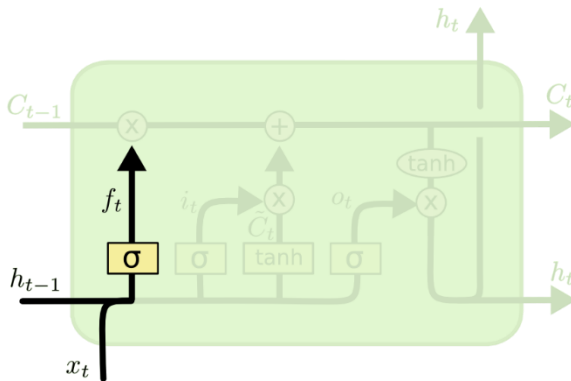


Рис. 2.6.
Сигмоподібний
шар RNN [17]

Сигмоїдальний шар виводить число від 0 до 1, що описує кількість кожного переданого компонента. Значення нуль означає, що нічого не пропускається, тоді як значення один означає, що пропускається все. LSTM має трое таких воріт для захисту та контролю стану комірки.

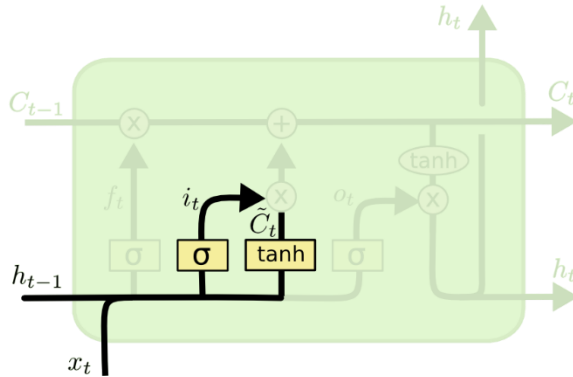
Розглянемо покрокову роботу LSTM. Перший крок – вирішення, якої інформації позбутися зі стану комірки. Це рішення приймається сигмоїдним шаром, або воротами очистки або забування. Ворота, залежно від значення h_{t-1} та x_t дають значення між 0 та 1 для кожного числа в стані комірки C_{t-1} . 1 означає повністю зберегти стан, тоді як 0 означає повністю стан скинути.



$$f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f)$$

Рис. 2.7. Ворота забування [17]

Наступний крок – вирішити, яку нову інформацію зберігати в стані комірки. Це має дві частини. По-перше, сигмоїдний шар, який називається вхідними воротами, вирішує, які значення ми будемо оновлювати. Далі шар \tanh створює вектор нових значень-кандидатів \hat{C}_t , які можна додати до стану. На наступному кроці обидва комбінуються, щоб створити оновлення стану.

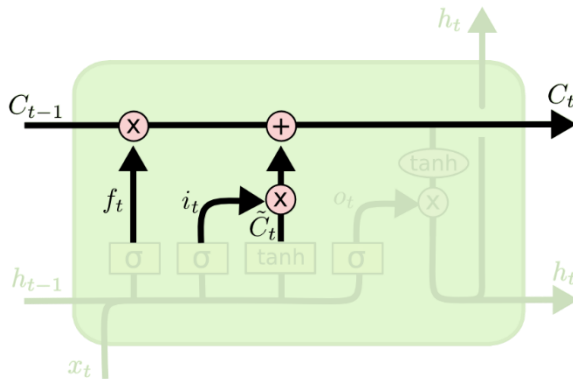


$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Рис. 2.8. Вхідний шар та шар \tanh [17]

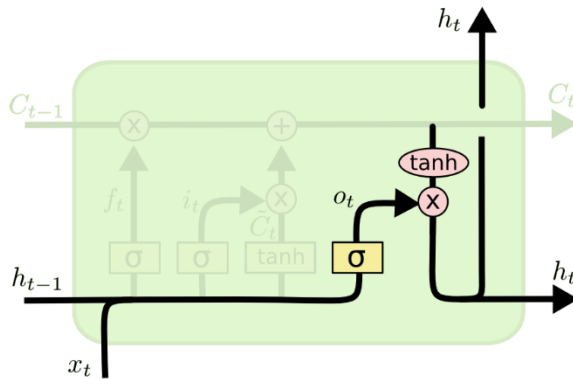
Далі, потрібно оновити попередній стан C_{t-1} новим станом C_t . Попередніми кроками було описано що потрібно зробити. Помножити старий стан на f_t , забувши те, що ми вирішили забути раніше. Потім ми додаємо $i_t * \tilde{C}_t$. Це нові значення кандидатів, масштабовані за тим, наскільки оновити кожне значення стану.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Рис. 2.9. Оновлення старого стану на новий [17]

Нарешті, потрібно вирішити, що виводити. Цей результат базуватиметься на стані клітинки, але буде відфільтрованою версією. Спочатку запускається сигмоподібний шар, який вирішує, які частини стану комірки вивести. Потім ми передаємо стан комірки через \tanh (щоб підштовхнути значення між -1 і 1) і помножимо його на вихід сигмоїдних воріт, щоб вивести лише ті частини, які потрібно.

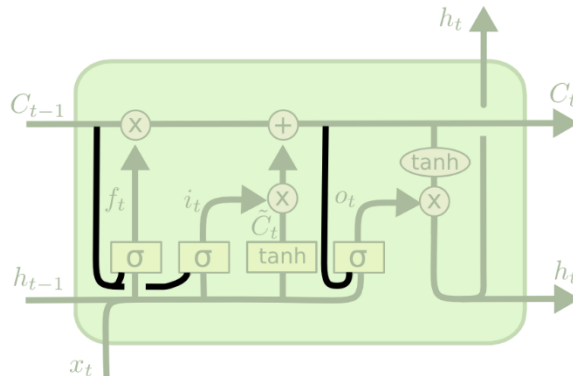


$$o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

Рис. 2.10. Формування результату роботи мережі [17]

Але не всі LSTM однакові. Фактично, здається, що майже кожен проект із використанням LSTM використовує дещо іншу версію. Відмінності незначні, але згадати деякі з них варто. Одним із популярних варіантів LSTM, представлених Gers & Schmidhuber [7], є додавання «з'єднань через вічко». Це означає, що шари воріт можуть дивитися на стан комірки.



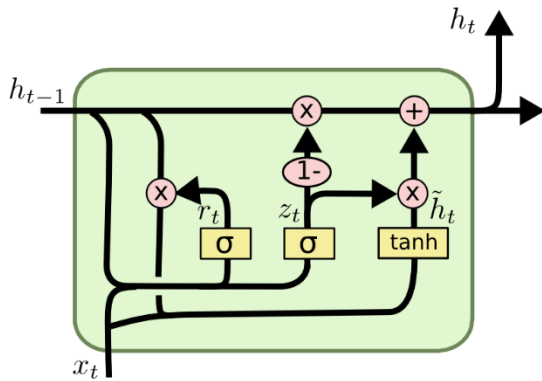
$$f_t = \sigma(W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [C_{t-1}, h_{t-1}, x_t] + b_i)$$

$$o_t = \sigma(W_o \cdot [C_t, h_{t-1}, x_t] + b_o)$$

Рис. 2.11. З'єднання через вічко [17]

Наведена вище схема додає вічка до всіх воріт, але в багатьох проектах одні вічка будуть, а інші ні. Іншим варіантом є використання пов'язаних вентилів забуття та введення. Замість того, щоб окремо вирішувати, що забути, а що додати нову інформацію, ми приймаємо ці рішення разом. Ми забуваємо лише тоді, коли збираємося ввести щось на своє місце. Ми вводимо нові значення в стан лише тоді, коли забуваємо щось старе.



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Рис. 2.12. Gate Recurrent Unit [17]

Деяко відмінною варіацією LSTM є Gated Recurrent Unit, або GRU, представлений Чо та інші [8]. Він поєднує ворота забуття та введення в єдині ворота оновлення. Він також об'єднує стан комірки та прихований стан і вносить деякі інші зміни. Отримана модель простіша, ніж стандартні моделі LSTM, і набуває все більшої популярності.

2.5. Використані підходи в роботі. Порівняльна характеристика.

У роботі використано чотири архітектури рекурентних нейронних мереж: LSTM, GRU, BiLSTM, BiGRU.

LSTM вважається традиційним підходом до вирішення задачі обробки природніх мов, однак пропонується застосувати три інших методи та оцінити їх ефективність у порівнянні з традиційним.

Таблиця 2.1.

Зіставлення кожного підходу

Архітектура	Особливості	Переваги	Недоліки	Кількість параметрів
LSTM	Запам'ятовування, механізм воріт	Краща довготривала пам'ять, може обробляти більш тривалі послідовності	Складніші, повільніші в навчанні	$4(n^2 + n)$

Продовження таблиці 2.1.

1	2	3	4	5
GRU	Спрощений механізм воріт, менше параметрів	Швидше тренування, менш піддаються перенавчанню	Може не працювати так добре на довших послідовностях	$3(n^2 + n)$
BiLSTM	Двонаправлена обробка з використанням LSTM	Краще розуміння контексту, краща ефективність виконання завдань, які потребують розуміння контексту	Більша складність, повільніше тренується	$8(n^2 + n)$
BiBRU	Двонаправлена обробка з використанням GRU	Навчається швидше, ніж BiLSTM, хороша продуктивність на коротших послідовностях	Може не працювати так добре на довших послідовностях	$6(n^2 + n)$

Коротко про визначні особливості кожного підходу:

- LSTM: має комірки пам'яті та механізми воріт для вибіркового запам'ятовування або забування інформації протягом тривалих періодів часу.
- GRU: має менше параметрів і використовує простіший механізм воріт порівняно з LSTM.

- BiLSTM: дозволяє двонаправлену обробку послідовних даних за допомогою комірок LSTM.
- BiGRU: дозволяє двонаправлену обробку послідовних даних за допомогою комірок GRU.

ВИСНОВОК ДО РОЗДІЛУ 2

У розділі 2 було розглянуто поняття глибокого навчання, наведено опис типів нейронних мереж: згорткові та рекурентні.

Рекурентна нейромережа була розглянута у протиположності до згорткової, як така, що в середовищі задач обробки природних мов має переваги. Описана проблема зникаючого градієнта, яка означає що при обробці послідовностей даних з залежністю, градієнт зворотного поширення у часі прямує до нуля, або проблема вибухових градієнтів, коли градієнт набуває великого значення. Ця проблема означає некоректність, або, відсутність навчання нейромережі.

Розглянуто поняття двонаправленої нейромережі та її переваги, які необхідні для вирішення задачі оцінки схожості текстів.

Як рішення, було оглянуто LSTM та похідні від нього варіанти шару нейронної мережі, які, однаково, за мету мають збереження стану такого шару для покращення обробки текстів, які і є пов'язаними послідовностями даних.

РОЗДІЛ 3

ОПИС АРХІТЕКТУРИ МОДЕЛІ ТА НАБОРІВ ДАННИХ

3.1. Опис архітектури моделі

Досить простий підхід до поширеної проблеми схожості речень, названий MaLSTM його архітектура зображена на рисунку 3.1Рис. (діаграма виключає частину попередньої обробки речень). Варто звернути увагу на те, що оскільки це сіамська мережа, її легше тренувати, оскільки вона розподіляє ваги з обох сторін.

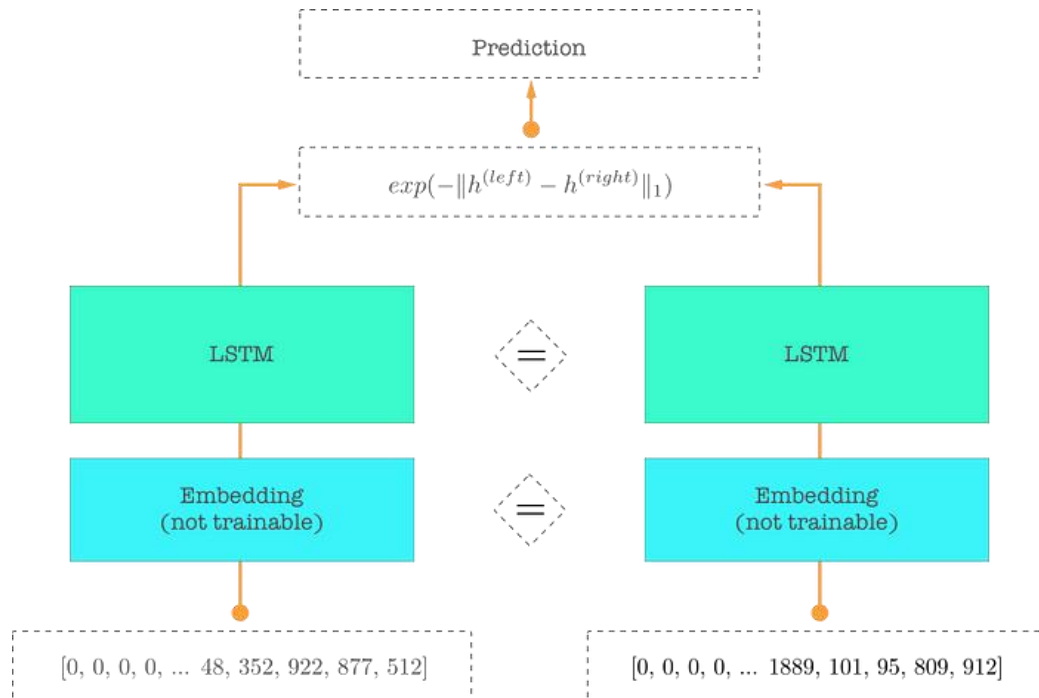


Рис. 3.1. Архітектура моделі MaLSTM

Сіамські мережі – це мережі, які мають дві або більше однакових підмереж. Схоже, що сіамські мережі добре справляються із завданнями на схожість і використовувалися для таких завдань, як семантична подібність речень, розпізнавання підроблених підписів тощо.

Сіамські мережі – це мережі, які мають дві або більше однакових підмереж. Схоже, що сіамські мережі добре справляються із завданнями на

схожість і використовувалися для таких завдань, як семантична подібність речень, розпізнавання підроблених підписів тощо.

У MaLSTM ідентична підмережа знаходиться на всьому шляху від вбудовування до останнього прихованого стану LSTM. Вбудовування слів – це сучасний спосіб представлення слів у моделях глибокого навчання. По суті, це метод надання словам семантичного значення у векторному представленні.

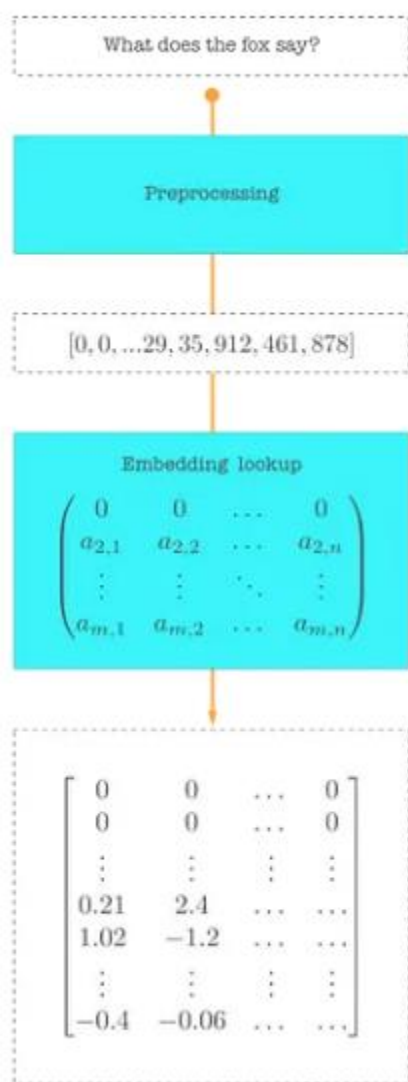


Рис. 3.2. Розгорнута ілюстрація вставлення

Вхідними даними в мережу є послідовності індексів слів із доповненням нуля. Ці вхідні дані є векторами фіксованої довжини, де перші нулі ігноруються, а ненульові є індексами, які однозначно ідентифікують слова. Потім ці вектори подаються на шар вбудовування. Цей шар шукає відповідне вбудовування для кожного слова та інкапсулює їх усі в матрицю. Ця матриця представляє даний текст як серію вставок. У цій роботі використана вставка word2vec від Google, GloVe: Global Vectors for Word Representation та Fasttext.

Маємо дві вбудовані матриці, які представляють кандидата двох подібних запитань. Потім передається у LSTM, і кінцевим станом LSTM для кожного запитання є 50-вимірний вектор. Навчається вловлювати смисловий зміст питання. На Рис. цей вектор позначено літерою h . Два вектори, які містять семантичне значення кожного питання пропускаємо через визначену функцію схожості і оскільки – показник від’ємного значення, результат буде між 0 і 1.

3.2. Word2vec вставка

Word2vec – це двохарова нейронна мережа, яка обробляє текст шляхом «векторизації» слів. Його вхід – це текст, а вихід – набір векторів, які представляють слова в цьому тексті. Хоча Word2vec не є глибокою нейронною мережею, він перетворює текст у числову форму, яку розуміють глибокі нейронні мережі [14].

Слова – це просто дискретні стани, як і інші дані, і ми просто шукаємо перехідні ймовірності між цими станами: ймовірність того, що вони відбуватимуться одночасно.

Метою та корисністю Word2vec є групування векторів подібних слів у векторному просторі. Тобто він виявляє подібність математично. Word2vec створює вектори, які є розподіленими числовими представленнями характеристик слів, таких як контекст окремих слів. Він робить це без втручання людини.

За наявності достатньої кількості даних, використання та контекстів Word2vec може зробити дуже точні припущення щодо значення слова на основі минулих появ. Ці припущення можна використати, щоб встановити зв'язок слова з іншими словами (наприклад, «чоловік» означає «хлопець», а «жінка» – «дівчина») або згрупувати документи та класифікувати їх за темами. Ці кластери можуть стати основою пошуку, аналізу настроїв і рекомендацій у таких різноманітних сферах, як наукові дослідження, юридичні відкриття, електронна комерція та управління взаємовідносинами з клієнтами [14].

Результатом роботи нейронної мережі Word2vec є словник, у якому до кожного елемента прикріплено вектор, який можна передати в мережу глибокого навчання або просто запитати, щоб виявити зв'язки між словами.

Вимірюючи косинусну подібність, відсутність схожості виражається як кут 90 градусів, тоді як загальна подібність 1 є кутом 0 градусів, повне перекриття.

Таблиця 3.1.

Слова, пов'язані зі словом «Швеція» за допомогою Word2vec [14]

Слово	Відстань за косинусом
Norway	0.760124
Denmark	0.715460
Finland	0.620022
Switzerland	0.588132
Belgium	0.585835
Netherlands	0.574631
Iceland	0.562368
Estonia	0.547621
Slovenia	0.531408

Країни Скандинавії та кілька заможних країн Північної Європи та Німеччини входять до першої дев'ятки.

Вектори, які використовуються для представлення слів, називаються нейронними вбудованими словами. Word2vec «векторизує» слова, і таким чином робить природну мову читабельною для комп'ютера – ми можемо почати виконувати потужні математичні операції зі словами, щоб виявити їхню схожість.

Отже, нейронне вбудовування слів представляє слово з числами. Це простий, але малоймовірний переклад.

Word2vec схожий на автокодувальник, який кодує кожне слово у векторі, але замість навчання вхідних слів через реконструкцію, як це робить

обмежена машина Больцмана, word2vec навчає слова іншим словам, які знаходяться поруч із ними у вхідному корпусі.

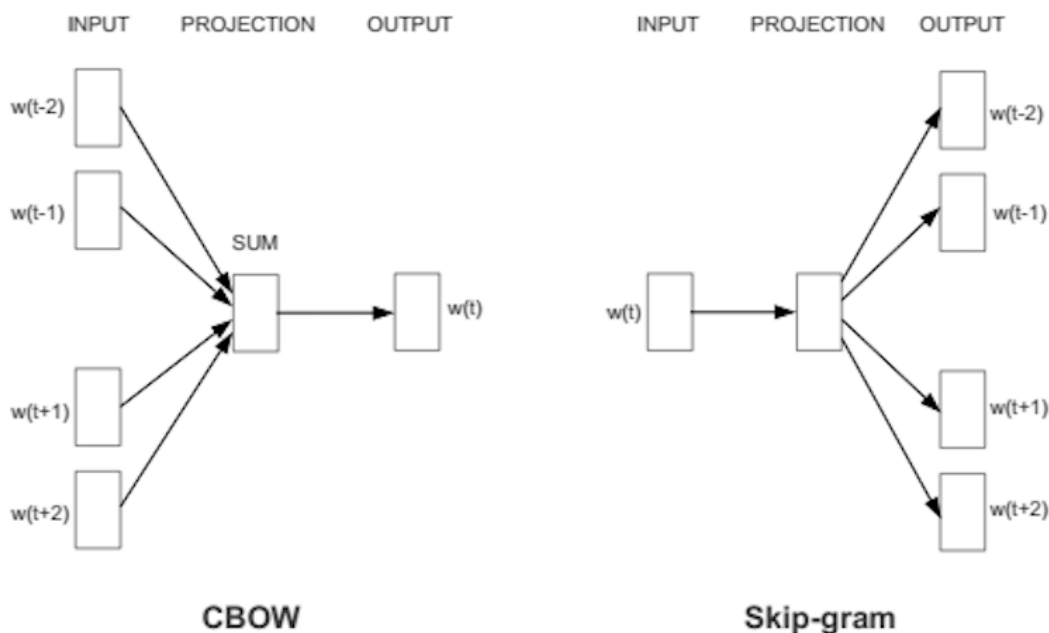


Рис. 3.3. CBOW та skip-gram

Це робиться одним із двох способів: або за допомогою контексту для передбачення цільового слова (метод, відомий як безперервний пакет слів, або CBOW), або за допомогою слова для передбачення цільового контексту, що називається skip-gram. Використано останній метод, оскільки він дає точніші результати на великих наборах даних.

Якщо вектор ознак, призначений слову, не можна використовувати для точного прогнозування контексту цього слова, компоненти вектора коригуються. Контекст кожного слова в корпусі – це вчитель, який надсилає сигнали про помилки для коригування вектора ознак. Вектори слів, які вважаються схожими за контекстом, зближуються один з одним шляхом коригування чисел у векторі.

Ці числа розташовують кожне слово як точку в 500-вимірному векторному просторі. Простори з більш ніж трьома вимірами важко уявити.

Добре навчений набір векторів слів розмістить подібні слова близько одне до одного в цьому просторі. Подібні речі та ідеї показуються «близькими».

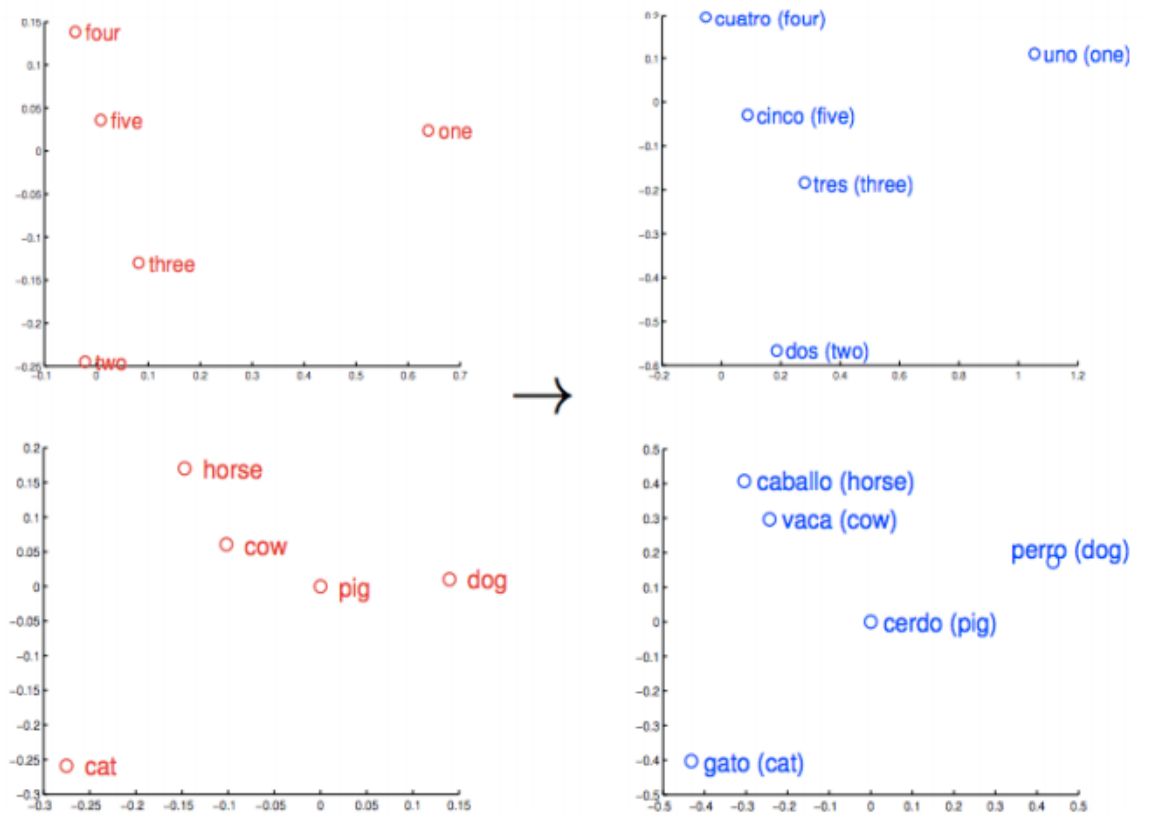


Рис. 3.4. Зв'язки між словами різних мов

Їх відносне значення було перекладено на вимірювані відстані. Якості стають величинами, а алгоритми можуть виконувати свою роботу. Але подібність – це лише основа багатьох асоціацій, які може вивчати Word2vec. Наприклад, він може оцінювати зв'язки між словами однієї мови та відображати їх на іншій.

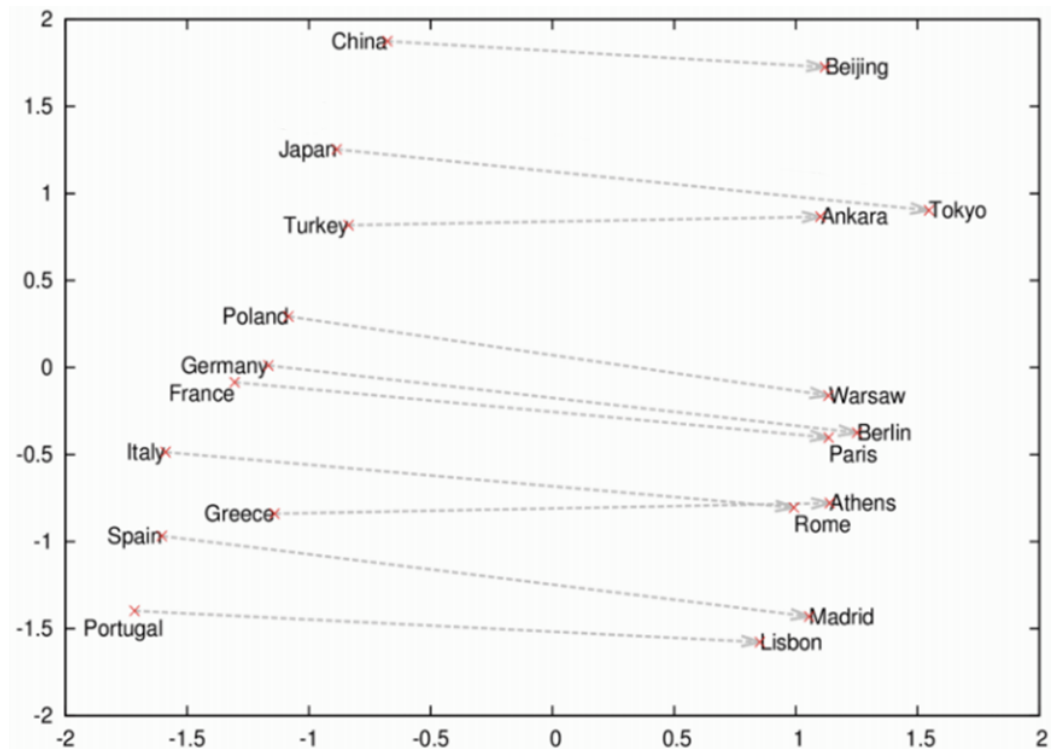


Рис. 3.5. Вектори країна-столиця

Ці вектори є основою більш повної геометрії слів. Рим, Париж, Берлін і Пекін не тільки згрупуються поруч один з одним, вони матимуть однакову відстань у векторному просторі до країн, столицями яких вони є; тобто Рим - Італія = Пекін - Китай. І якщо відомо, що Рим є столицею Італії, а питання яке місто є столицею Китаю, то рівняння Рим-Італія + Китай повернуло б Пекін.

3.3. GloVe вставка

GloVe – це алгоритм неконтрольованого навчання для отримання векторних представлень для слів [11]. Навчання виконується на агрегованій глобальній статистиці спільного використання слів із корпусу, а отримані представлення демонструють цікаві лінійні підструктури векторного простору слів.

Статистика появи слів у корпусі є основним джерелом інформації, доступною для всіх неконтрольованих методів вивчення репрезентації слів, і хоча зараз існує багато таких методів, все ще залишається питання про те, як

значення генерується з цієї статистики та як результат вектори слів можуть представляти це значення. Знання використовуються, щоб побудувати нову модель представлення слів GloVe. Спочатку встановлюються деякі позначення. Нехай матриця підрахунків сумісних появ слово-слово буде позначена X , записи якої X_{ij} зводять у таблицю кількість разів, коли слово j зустрічається в контексті слова i . Нехай $X_i = \sum_k (X_{ik})$ буде кількістю разів, коли будь-яке слово з'являється в контексті слова i . Нарешті, нехай $P_{ij} = P(j|i) = X_{ij}/X_i$ – ймовірність того, що слово j з'явиться в контексті слова i .

Таблиця 3.2.

Ймовірності спільної появи цільових слів *ice* і *steam* із вибраними контекстними словами з корпусу токенів із 6 мільярдів.

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k ice)/P(k steam)$	8.9	8.5×10^{-2}	1.36	0.96

Розглянемо два слова i та j , які виявляють особливий аспект інтересу; для конкретності припустімо, що нас цікавить поняття термодинамічної фази, для якої ми можемо взяти $i = \text{лід}$ і $j = \text{пара}$. Взаємозв'язок цих слів можна перевірити, вивчивши співвідношення ймовірностей їх спільної появи з різними пробними словами, k . Для слів k , які стосуються льоду, але не пари, скажімо, $k = \text{твердий}$, ми очікуємо, що співвідношення P_{ik}/P_{jk} буде великим. Подібним чином, для слів k , пов'язаних із парою, але не льодом, скажімо, $k = \text{газ}$, співвідношення має бути невеликим. Для таких слів k , як вода або мода, які або пов'язані з льодом і парою, або не пов'язані ні з тим, ні з іншим, співвідношення має бути близьким до одиниці. Порівняно з необробленими ймовірностями, це співвідношення краще розрізняє релевантні слова (тверда

речовина та газ) від нерелевантних слів (вода та мода), а також краще розрізняє два релевантних слова. Наведений вище аргумент припускає, що відповідна відправна точка для вивчення вектора слів має бути зі співвідношеннями ймовірностей спільного появи, а не самих ймовірностей.

Досліди проводяться за завданням на аналогію слова Mikolov et al. (2013a), різноманітні завдання на подібність слів, як описано в (Luong et al., 2013), а також у спільному наборі даних CoNLL-2003 для порівняння NER (Tjong Kim Sang and De Meulder, 2003).

Таблиця 3.3.

Результати виконання завдання на аналогію слова, подано [11]

Model	Dim.	Size	Sem.	Syn.	Tot.
ivLBL	100	1.5B	55.9	50.1	53.2
HPCA	100	1.6B	4.2	16.4	10.8
GloVe	100	1.6B	<u>67.5</u>	<u>54.3</u>	<u>60.3</u>
SG	300	1B	61	61	61
CBOW	300	1.6B	16.1	52.6	36.1
vLBL	300	1.5B	54.2	<u>64.8</u>	60
ivLBL	300	1.5B	65.2	63	64
GloVe	300	1.6B	<u>80.8</u>	61.5	<u>70.3</u>
SVD	300	6B	6.3	8.1	7.3
SVD-S	300	6B	36.7	46.6	42.1
SVD-L	300	6B	56.6	63	60.1
CBOW	300	6B	63.6	<u>67.4</u>	65.7
SG+	300	6B	73	66	69.1
GloVe	300	6B	<u>77.4</u>	67	<u>71.7</u>
CBOW	1000	6B	57.3	68.9	63.7
SG	1000	6B	66.1	65.1	65.6
SVD-L	300	42B	38.4	58.2	49.2
GloVe	300	42B	<u>81.9</u>	<u>69.3</u>	<u>75</u>

Підкреслені результати є найкращими в групах моделей однакового розміру; Жирні оцінки є найкращими в цілому. Вектори HPCA є загальнодоступними; результати vLBL взяті з (Mnih et al., 2013); результати

скіп-грами (SG) і BOW взяті з (Mikolov et al., 2013a,b); модель навчена SG і CBOW за допомогою інструмента word2vec.

Кількісне порівняння GloVe з word2vec ускладнюється наявністю багатьох параметрів, які сильно впливають на продуктивність. Найважливішою змінною, яку потрібно контролювати, є час навчання. Для GloVe відповідним параметром є кількість ітерацій навчання. Для word2vec очевидним вибором буде кількість епох навчання. Наразі код розроблено лише для однієї епохи: він визначає розклад навчання, специфічний для одного проходу даних, що робить модифікацію для кількох проходів нетривіальним завданням. Іншим варіантом є зміна кількості негативних зразків. Додавання негативних зразків ефективно збільшує кількість навчальних слів, які бачить модель, тому певним чином це аналогічно додатковим епохам. Ми встановлюємо для будь-яких невизначених параметрів значення за замовчуванням, припускаючи, що вони близькі до оптимальних, хоча ми визнаємо, що це спрощення слід послабити під час більш ретельного аналізу.

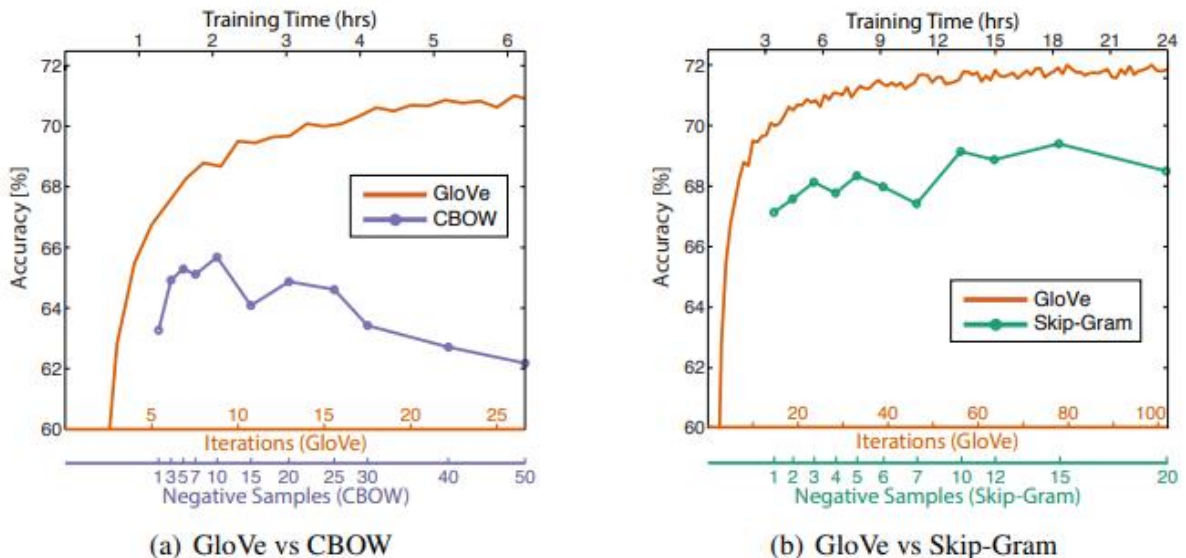


Рис. 3.6. Загальна точність завдання на аналогію слова як функція часу навчання, який регулюється кількістю ітерацій для GloVe та кількістю негативних зразків для CBOW і skip-gram.

На рисунку 3.6. показано загальну ефективність виконання аналогічного завдання як функцію часу навчання. Загальна точність завдання на аналогію слова як функція часу навчання, який регулюється кількістю ітерацій для GloVe та кількістю негативних зразків для CBOW і skip-gram. У всіх випадках ми навчаємо 300-вимірні вектори на тому самому корпусі токенів 6B (Wikipedia 2014 + Gigaword 5) з однаковим словниковим запасом із 400 000 слів і використовуємо симетричне контекстне вікно розміром 10. Дві осі x внизу вказують відповідну кількість ітерацій навчання для GloVe та негативних зразків для word2vec. Зауважимо, що продуктивність word2vec фактично знижується, якщо кількість негативних зразків перевищує приблизно 10. Імовірно, це тому, що метод негативної вибірки погано наближає цільовий розподіл ймовірностей. За однакового корпусу, словникового запасу, розміру вікна та часу навчання GloVe стабільно перевершує word2vec і досягає кращих результатів швидше, а також отримує найкращі результати незалежно від швидкості.

3.4. Опис наборів даних використаних в роботі

GoogleNews-vectors-negative300 – це попередньо навчена модель word2vec, яка була навчена на корпусі Новин Google із 3 мільярдів постійних слів. Модель містить 3 мільйони 300-вимірних векторів англійських слів.

Вставка glove.840B.300d – модель на даних Common Crawl із 840 мільярдами токенів, 2,2 мільйонами словників, регістром і 300-вимірними векторами

Crawl-300d-2M-subword – це попередньо навчена векторна модель від fastText. Вона містить 2 мільйони векторів слів, навчених інформацією про підслова на даних Common Crawl із 600 мільярдами токенів

Набір даних Quora Question Pairs (QQP) – це набір даних, який складається з понад 400 000 пар запитань. Кожна пара запитань анотована

двійковим значенням, яке вказує, чи є ці два запитання перефразами одне одного. Набір даних базується на фактичних даних із Quora та може використовуватися для навчання та тестування моделей семантичної еквівалентності. Основні ярлики істинності за своєю суттю є суб'єктивними, оскільки справжнє значення речень ніколи не може бути відоме з упевненістю. Навішування ярликів на людину також є «шумним» процесом, і розумні люди з цим не погодяться. Як наслідок, базові мітки істинності в цьому наборі даних слід сприймати як «інформовані», але не на 100% точні та можуть містити неправильні мітки.

Таблиця 3.4.

Вибірка перших 5 пар речень і значення їх ярликів, де 1 означає що речення є перефразом, 0 – ні [13]

Питання 1	Питання 2	Ярлик істинності
How can I be a good geologist?	What should I do to be a great geologist?	1
When do you use シ instead of ㇿ?	When do you use "&" instead of "and"?	0
Motorola (company): Can I hack my Charter Motorolla DCX3400?	How do I hack Motorola DCX3400 for free internet?	0
Method to find separation of slits using fresnel biprism?	What are some of the things technicians can tell about the durability and reliability of Laptops and its components?	0
How do I read and find my YouTube comments?	How can I see all my Youtube comments?	1

В такому вигляді набір даних не підходить до навчання та використання глибоким навчанням у роботі. Необхідно провести вставку, замінити слова та вирази відповідними цифровими векторними значення з моделі word2vec, про яку йшлося раніше. На прикладі тих самих обраних п'яти пар речень представимо результат вставки:

Таблиця 3.5.

Результат векторизації [13]

Питання 1	Питання 2	Ярлик істинності	Закодоване питання 1	Закодоване питання 2
How can I be a good geologist?	What should I do to be a great geologist?	1	69,70	71,70
When do you use シ instead of し?	When do you use "&" instead of "and"?	0	72,73	72,73
Motorola (company): Can I hack my Charter Motorolla DCX3400?	How do I hack Motorola DCX3400 for free internet?	0	74,75,51,76,77,78,79	76,74,79,80,21
Method to find separation of slits using fresnel biprism?	What are some of the things technicians can tell about the durability and reliability of Laptops and its components?	0	81,31,82,83,23,84,85	86,87,88,89,90,91,92
How do I read and find my YouTube comments?	How can I see all my Youtube comments?	1	93,31,94,95	96,94,95

З наведеної таблиці необхідними і достатніми даними є стовпчики «Закодоване питання 1», «Закодоване питання 2» та «Ярлик істинності».

STS Benchmark містить вибірку наборів даних англійською мовою, які використовуються в завданнях STS. Вибір наборів даних включає текст із підписів до зображень, заголовків новин і форумів користувачів.

Таблиця 3.6.

Статистика набору даних

	Тренування	Розробка	Тестування	Всього
Новини	3299	500	500	4299
Підписи	2000	625	625	3250
Форуми	450	375	254	1079
Всього	5749	1500	1379	8628

Щоб забезпечити стандартний еталон для порівняння між системами представлення значень у наступні роки, його організовано за навчанням, розробкою та тестуванням. Розробку частину можна використовувати для розробки та налаштування гіперпараметрів систем, а тестову частину слід використовувати лише один раз для остаточної системи. STS benchmark забезпечує стандартне налаштування для навчання, розробки та тестування трьох вибраних жанрів (новини, підписи, форуми).

ВИСНОВКИ ДО РОЗДІЛУ 3

Для завдання магістерської роботи найдоцільніше використовувати рекурентну сіамську нейронну мережу. Було вирішено побудувати такі нейронні мережі з двох ідентичних прихованих частин: LSTM-похідних моделей та вставки word2vec, GloVe, Fasttext, а також вхідних та вихідних шарів для даних та результатів. Далі розглянуто та описано окремі частини архітектури нейромережі.

У цьому розділі також було розглянуто набори даних, на якому планується робота, тобто його зміст, структуру та наповнення. Було проаналізовано структуру та визначено, які елементи даних були обрані для подальшої роботи.

РОЗДІЛ 4

РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ

4.1. Інструменти розробки представлені у роботі

Проведемо короткий огляд інструментів, за допомогою яких виконано тренування моделей.

TensorFlow – це програмна бібліотека з відкритим вихідним кодом для чисельних обчислень і машинного навчання, розроблена командою Google Brain. Випущена в 2015 році, вона стала однією з найпопулярніших і широко використовуваних бібліотек машинного навчання. TensorFlow, призначений для того, щоб розробники могли створювати та навчати моделі машинного навчання для різноманітних застосувань, включаючи комп'ютерний зір, обробку природної мови та розпізнавання мови.

TensorFlow також надає високорівневий API під назвою Keras, який спрощує процес створення і навчання нейронних мереж; Keras надає простий у використанні інтерфейс для визначення моделей і вказівки гіперпараметрів, що дозволяє розробникам зосередитися на поставленому завданні, а не на деталях реалізації. Keras надає простий у використанні інтерфейс для визначення моделей та гіперпараметрів, що дозволяє розробникам зосередитися на поставленому завданні.

Загалом, TensorFlow став популярним вибором для проектів машинного навчання та глибокого навчання завдяки простоті використання, гнучкості та розширюваності.

Ця бібліотека дозволяє реалізувати продукти на декількох мовах програмування. В цій роботі використана мова Python.

4.2. Оптимізатори

Оптимізатори є важливим компонентом навчання нейронних мереж і відіграють ключову роль у досягненні кращої точності при мінімізації часу навчання. TensorFlow має Adadelta, SGD, RMSprop, Adagrad, Adam, Adamax, Nadam та багато інших алгоритмів оптимізації. Нижче наведено короткий опис кожного оптимізатора, значення параметрів за замовчуванням, приклади використання та вплив на навчання RNN [15]:

- Adadelta – це метод адаптивної швидкості навчання, який допомагає зменшити чутливість до початкової швидкості навчання. Він адаптується до мінливих темпів навчання на основі інформації про історичний градієнт. Він підходить для великих наборів даних і може сходитися швидше, ніж інші алгоритми оптимізації.

За замовчуванням Adadelta має такі параметри:

```
tf.keras.optimizers.experimental.Adadelta(
    learning_rate=0.001,
    rho=0.95,
    epsilon=1e-07,
    weight_decay=None,
    clipnorm=None,
    clipvalue=None,
    global_clipnorm=None,
    use_ema=False,
    ema_momentum=0.99,
    ema_overwrite_frequency=None,
    jit_compile=True,
    name='Adadelta',
    **kwargs
)
```

Adadelta можна використовувати для навчання RNN, особливо для моделей послідовності, де важливо мати хороший механізм адаптації швидкості навчання.

- Стохастичний градієнтний спуск (SGD) є одним із найпопулярніших алгоритмів оптимізації, який використовується в глибокому навчанні. Він оновлює параметри моделі на основі від'ємного градієнта функції втрат щодо параметрів. Він простий, обчислювально ефективний і може добре працювати з шумними градієнтами.

Значення параметрів за замовчуванням для SGD:

```
tf.keras.optimizers.experimental.SGD(
    learning_rate=0.01,
    momentum=0.0,
    nesterov=False,
    weight_decay=None,
    clipnorm=None,
    clipvalue=None,
    global_clipnorm=None,
    use_ema=False,
    ema_momentum=0.99,
    ema_overwrite_frequency=None,
    jit_compile=True,
    name='SGD',
    **kwargs
)
```

SGD можна використовувати для навчання RNN, але він може повільно сходитися або застрягати в локальних мінімумах, якщо швидкість навчання не встановлена належним чином.

- RMSprop – це адаптивний алгоритм оптимізації швидкості навчання, який допомагає пом'якшити коливання в оновленнях

градієнтного спуску. Він використовує експоненціально зважену ковзну середню квадратів градієнтів для регулювання швидкості навчання. Він підходить для нестационарних задач, де градієнти можуть варіюватися в широких межах.

Значення параметрів за замовчуванням для RMSprop:

```
tf.keras.optimizers.experimental.RMSprop(
    learning_rate=0.001,
    rho=0.9,
    momentum=0.0,
    epsilon=1e-07,
    centered=False,
    weight_decay=None,
    clipnorm=None,
    clipvalue=None,
    global_clipnorm=None,
    use_ema=False,
    ema_momentum=0.99,
    ema_overwrite_frequency=100,
    jit_compile=True,
    name='RMSprop',
    **kwargs
)
```

RMSprop можна використовувати для навчання RNN, особливо для довготривалих залежностей у послідовних даних, де важливо контролювати адаптацію швидкості навчання.

- Adagrad – це адаптивний алгоритм оптимізації швидкості навчання, який регулює швидкість навчання на основі суми історичних градієнтів. Він підходить для розріджених даних і може обробляти різні швидкості навчання для різних параметрів.

Значення параметрів за замовчуванням для Adagrad такі:

```
tf.keras.optimizers.experimental.Adagrad(  
    learning_rate=0.001,  
    initial_accumulator_value=0.1,  
    epsilon=1e-07,  
    weight_decay=None,  
    clipnorm=None,  
    clipvalue=None,  
    global_clipnorm=None,  
    use_ema=False,  
    ema_momentum=0.99,  
    ema_overwrite_frequency=None,  
    jit_compile=True,  
    name='Adagrad',  
    **kwargs  
)
```

Adagrad можна використовувати для навчання RNN, але він може повільно сходитися або застрягати в локальних мінімумах, якщо швидкість навчання не встановлена належним чином.

- Adam – це адаптивний алгоритм оптимізації швидкості навчання, який поєднує в собі переваги RMSprop і оновлення імпульсу. Він адаптує швидкість навчання на основі величин градієнта та імпульсу градієнтів. Він підходить для широкого кола проблем і може сходитися швидше, ніж інші алгоритми оптимізації.

Значення параметрів за замовчуванням для Адама:

```
tf.keras.optimizers.Adam(  
    learning_rate=0.001,  
    beta_1=0.9,  
    beta_2=0.999,  
    epsilon=1e-07,  
    amsgrad=False,  
    weight_decay=None,  
    clipnorm=None,  
    clipvalue=None,  
    global_clipnorm=None,  
    use_ema=False,  
    ema_momentum=0.99,  
    ema_overwrite_frequency=None,  
    jit_compile=True,  
    name='Adam',  
    **kwargs  
)
```

- Adamax – це варіант оптимізатора Adam, який використовує максимальний із минулих градієнтів замість експоненціального середнього. Він підходить для проблем із великомірними даними, такими як текст або дані зображення.

Значення параметрів за замовчуванням для Adamax:

```
tf.keras.optimizers.experimental.Adamax(  
    learning_rate=0.001,  
    beta_1=0.9,  
    beta_2=0.999,  
    epsilon=1e-07,  
    weight_decay=None,  
    clipnorm=None,  
    clipvalue=None,  
    global_clipnorm=None,  
    use_ema=False,  
    ema_momentum=0.99,  
    ema_overwrite_frequency=None,  
    jit_compile=True,  
    name='Adamax',  
    **kwargs  
)
```

Adamax можна використовувати для навчання RNN, особливо для великомасштабних наборів даних, де він може ефективно обробляти різні швидкості навчання для різних параметрів.

- Nadam – це варіант Адама, який використовує прискорений градієнтний спуск Нестерова. Він поєднує в собі переваги адаптивної швидкості навчання та оновлення імпульсу та може збігатися швидше, ніж інші алгоритми оптимізації.

Значення параметрів за замовчуванням для Nadam:


```
tf.keras.optimizers.experimental.Nadam(
    learning_rate=0.001,
    beta_1=0.9,
    beta_2=0.999,
    epsilon=1e-07,
    weight_decay=None,
    clipnorm=None,
    clipvalue=None,
    global_clipnorm=None,
    use_ema=False,
    ema_momentum=0.99,
    ema_overwrite_frequency=None,
    jit_compile=True,
    name='Nadam',
    **kwargs
)
```

Nadam можна використовувати для навчання RNN, і він особливо підходить для завдань, які вимагають швидшої конвергенції та хорошої продуктивності узагальнення.

Підсумовуючи, кожен оптимізатор має свої переваги та недоліки, а вибір оптимізатора залежить від конкретної проблеми та наявних ресурсів. Загалом, адаптивним алгоритмам оптимізації швидкості навчання, таким як Adadelta, RMSprop, Adagrad, Adam, Adamax і Nadam, часто віддають перевагу перед стохастичним градієнтним спуском (SGD), оскільки вони можуть збігатися швидше та мають кращу продуктивність узагальнення. Однак завжди корисно експериментувати з різними алгоритмами оптимізації та гіперпараметрами, щоб знайти найкращу комбінацію для конкретного завдання.

4.3. Опис елементів програми та результати виконання

Робота виконувалася у форматі файлу Jupyter Notebook. Jupyter Notebook надає інтерактивне середовище для аналізу даних, наукових обчислень, машинного навчання та інших обчислювальних завдань. Блокнот складається

з комірок, які можуть містити код або текст розмітки. Користувачі можуть виконувати комірки коду в інтерактивному режимі та негайно бачити результати, що полегшує експерименти з різними фрагментами коду та даними. Перш за все робота програми починається з компоновки моделі:

1. Завантажується вставка Word2Vec, GloVe або Fasttext. Створено окремі функції для зручності.

```
import gensim
from gensim.scripts.glove2word2vec import glove2word2vec

def load_word2vec(path):
    word2vec = gensim.models.KeyedVectors.load_word2vec_format(path, binary=True)
    return word2vec

def load_glove(path):
    tmp_file = "/tmp/glove.840B.300d.w2v.txt"
    glove2word2vec(path, tmp_file)
    glove = gensim.models.KeyedVectors.load_word2vec_format(tmp_file)
    return glove

def load_fasttext(path):
    fasttext = gensim.models.KeyedVectors.load_word2vec_format(path,
binary=False)
    return fasttext
```

Функції застосовуються для завантаження вставок:

```
word2vec = load_word2vec("GoogleNews-vectors-negative300.bin.gz")
glove = load_glove("glove.840B.300d.txt")
fasttext = load_fasttext("crawl-300d-2M-subword.vec")
```

2. Ініціалізуються оптимізатори.

```

from keras.optimizers import Adadelta, SGD, RMSprop, Adagrad, Adam, Adamax, Nadam

gradient_clipping_norm = 1.25

adadelta = Adadelta(clipnorm=gradient_clipping_norm)
sgd = SGD(clipnorm=gradient_clipping_norm)
rmsprop = RMSprop(clipnorm=gradient_clipping_norm)
adagrad = Adagrad(clipnorm=gradient_clipping_norm)
adam = Adam(clipnorm=gradient_clipping_norm)
adamax = Adamax(clipnorm=gradient_clipping_norm)
nadam = Nadam(clipnorm=gradient_clipping_norm)

```

Параметр оптимізатора `clipnorm` (Float): якщо встановлено, градієнт кожної ваги обрізається окремо, щоб його норма не перевищувала це значення. Це один із способів вирішення проблеми зникаючого/вибухового градієнта. Всі інші параметри залишені за замовчуванням.

3. Завантажуються всі необхідні набори даних: STS benchmark та Quora question pairs. Quora question pairs використовуються для тренування моделі, тоді як інша – для оцінки роботи.

4. Компоновка моделі. Архітектура моделі зображена на рисунку 3.1. У роботі шари LSTM у лівій та правій половині варіювалися: для отримання результатів традиційного підходу у першому тестуванні, звісно, використана архітектура LSTM. У подальшому, підставлялися архітектури GRU, BiLSTM, BiGRU. Оптимізатори також варіювалися. Наприклад, код компоновання традиційної моделі з оптимізатором Adadelta:

```

import itertools
import keras
from keras import Input, Model
from keras.layers import (
    Input,
    Embedding,
    LSTM,
)
from keras.models import Model, Sequential

n_hidden = 50

x = Sequential()
x.add(
    Embedding(
        len(embeddings),
        embedding_dim,
        weights=[embeddings],
        input_shape=(max_seq_length,),
        trainable=False,
    )
)

x.add(LSTM(n_hidden))

shared_model = x

left_input = Input(shape=(max_seq_length,), dtype="int32")
right_input = Input(shape=(max_seq_length,), dtype="int32")

malstm_distance = Lambda(function=lambda x: exponent_neg_manhattan_distance(x[0],
x[1]),
                        output_shape=lambda x: (x[0][0], 1))([left_input,
right_input])
model = Model(inputs=[left_input, right_input], outputs=[malstm_distance])

model.compile(
    loss="mean_squared_error",
    optimizer=adadelta,
    metrics=["accuracy", keras.metrics.AUC(), keras.metrics.Precision()],
)

```

5. Наступним кроком є тренування моделі та отримання статистики процесу тренування. Загалом статистика у всіх варіаціях однакова.

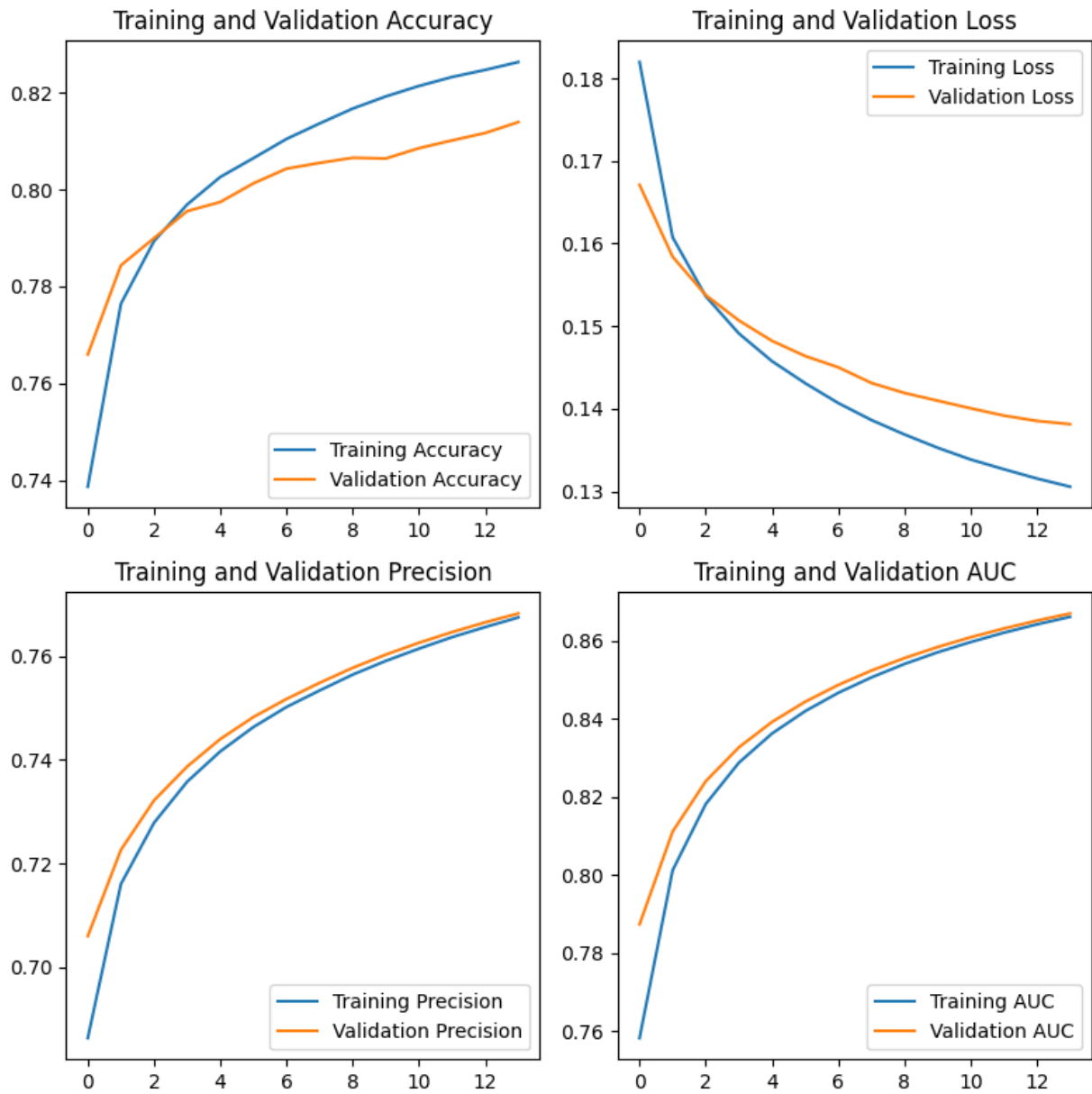


Рис. 4.1. Графіки динаміки тренування моделей

6. Останнім етапом є отримання результатів оцінки ефективності всіх моделей на основі набору даних STS benchmark.

```

batch_size=64
sims = model.predict([STS_benchmark['left'], STS_benchmark['right']],
batch_size=batch_size)
formatted_sims = []

for sim in sims:
    formatted_sims.append(sim[0])

for i in range(0, 3):
    test['predicted_sim'] = pd.Series(formatted_sims).values
    pearson_correlation = scipy.stats.pearsonr(formatted_sims, test['sim'])[0]
    spearman_correlation = scipy.stats.spearmanr(formatted_sims, test['sim'])[0]
    rmse = sqrt(mean_squared_error(test['predicted_sim'], test['sim']))
    textstr = 'RMSE=%.3f\n$Pearson Correlation=%.3f\n$Spearman
Correlation=%.3f$' % (
    rmse, pearson_correlation, spearman_correlation)
    sick_test_sorted = test.sort_values('sim')
    id = list(range(0, len(test.index)))
    sick_test_sorted['id'] = pd.Series(id).values

    if (i < 3):
        row = 0
        column = i

    sick_test_sorted.plot(kind='scatter', x='id', y='sim', color='DarkBlue',
label='Similarity', title=f"LSTM-{embedding_name}-{adadelta.name}",
ax=axes[column]);
    sick_test_sorted.plot(kind='scatter', x='id', y='predicted_sim',
color='DarkGreen', label='Predicted Similarity',
ax=axes[column]);
    axes[column].text(1500, 0.05, textstr, fontsize=12)

```

У роботі результати представляються графіками поширення оцінки схожості тестового набору STS. Критеріями оцінки результатів тестування архітектур є середньоквадратичне відхилення, кореляція Пірсона та Спірмена.

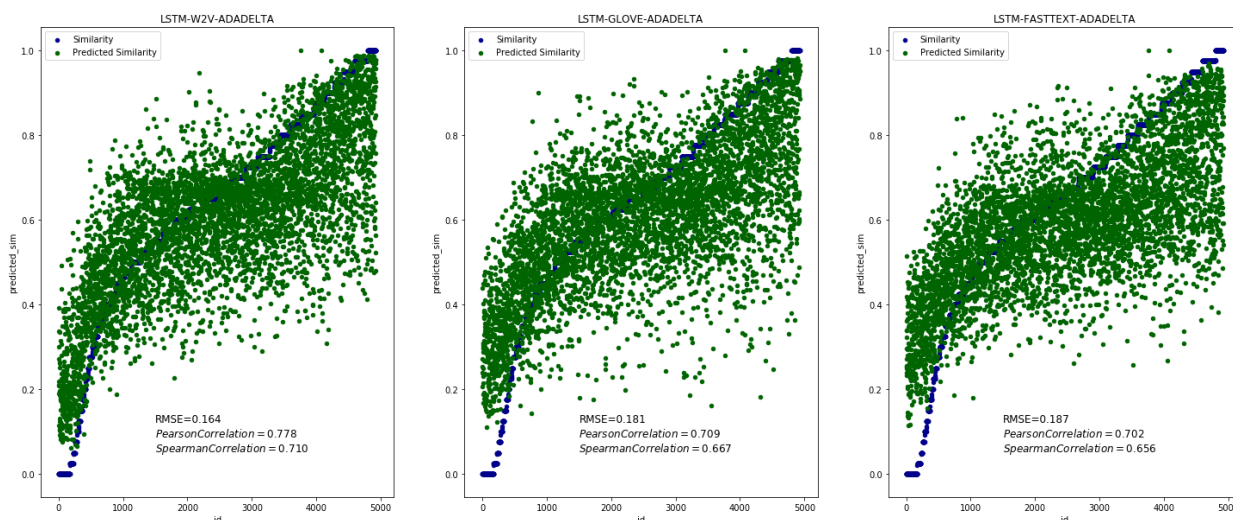


Рис. 4.2. Результати поширення передбачень неймереж з LSTM з оптимізатором Adadelta

Окремо можна виділити результати тренування моделей з оптимізатором SGD.

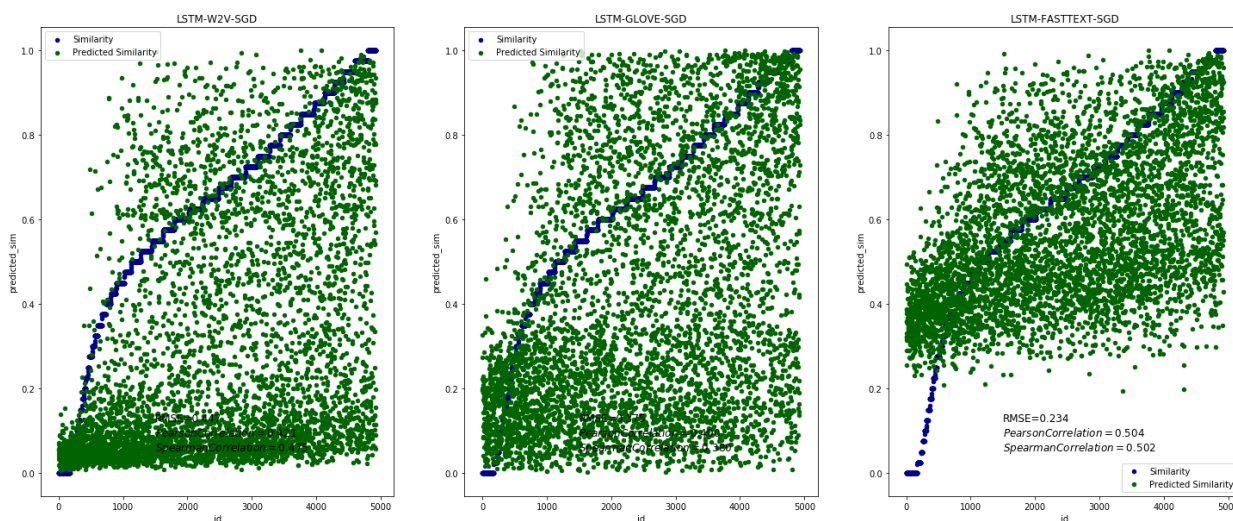


Рис. 4.3. Результати поширення передбачень з оптимізатором SGD.

При тому, що градієнт був обмеженим оптимізатор SGD погано підійшов до тренування моделей. Подібна тенденція поширення з цим оптимізатором була в усіх випадках моделей. Скупчення результатів передбачення моделі означає, що параметри моделі не належним чином були

натреновані, передбачення дають оцінку у екстремумі: від 0 до 0.1, тобто вхідні речення не подібні зовсім.

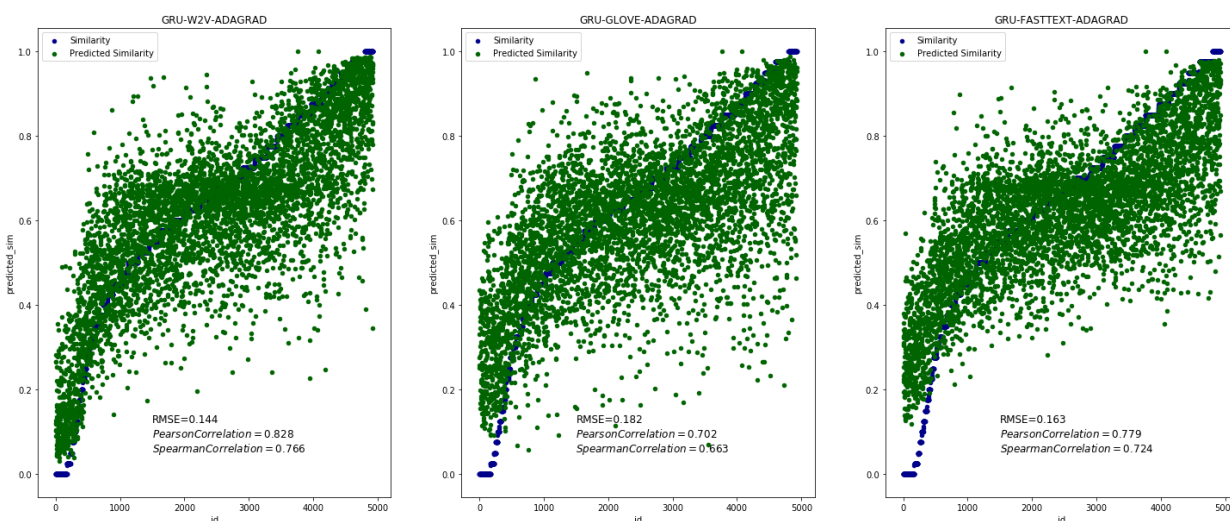


Рис. 4.4. Поширення передбачення MAGRU з оптимізатором ADAGRAD

Таблиця 4.1.

Значення критеріїв MAGRU з оптимізатором Adadelta

Оптимізатор	Вставка	Середньоквадратичне відхилення	Коефіцієнт кореляції Пірсона	Коефіцієнт кореляції Спірмена
ADAGRAD	Word2vec	0.144	0.828	0.766
ADAGRAD	GloVe	0.182	0.702	0.663
ADAGRAD	Fasttext	0.163	0.779	0.724

Найкращим результатом з усіх експериментів є модель з архітектурою GRU, вставкою word2vec та оптимізатором ADAGRAD. Показники відхилення та кореляції у цьому випадку найкращі: відхилення найменше, кореляція – найбільша. Візуально, графік показує рівномірне поширення результатів передбачення, позначених зеленим кольором, вздовж кривої істинних оцінок схожості, позначеної блакитним кольором. Найбільші помилки при цьому виглядають поодинокими випадками, які чітко видно окремими точками на графіку, далеко від всіх інших.

ВИСНОВОК ДО РОЗДІЛУ 4

Нові запропоновані методи в ході досліджень перевершують результати роботи традиційної моделі з використанням LSTM. Окремим, найкращим результатом є модель GRU з оптимізатором ADAGRAD і вставкою word2vec. В однакових умовах проведення експериментів ця модель не тільки показала найкращі результати по критеріям вимірювання ефективності, а й мала найменший час тренування. Факторами, які впливали на швидкість навчання та проходження епох є кількість параметрів та оптимізатор. У архітектурі GRU параметрів є найменша кількість з усіх чотирьох архітектур, адже вона менше за LSTM, а також вдвічі менша за двонаправлену архітектуру BiGRU, не кажучи про BiLSTM, яка є архітектурою з найбільшою кількістю параметрів та мала найдовшу тривалість навчання.

Оптимізатор ADAGRAD, як і призначений, адаптував швидкість навчання найкраще за інші оптимізатори.

ВИСНОВОК

У роботі представлена актуальність теми: семантична схожість текстів є одною із завдань цифрової обробки природніх мов.

Було розглянуто теоретичну частину підходу глибинного навчання для розпізнавання природніх мов, наведено опис деяких видів нейромереж, їх недоліків і переваг. Описано проблему зникаючого/вибухового градієнта та LSTM з похідними від нього нейромережі, як рішення проблеми градієнта.

Огляд методів аналізу та наборів даних, для досліджень: вставки векторизації слів, загальну архітектуру моделі нейромережі.

У результаті проведення дослідів на основі цієї роботи було створено об'єктивну порівняльну характеристику тридцяти двох архітектур моделей глибокого навчання. Результатом передбачення моделей (нейромереж) є оцінка схожості двох вхідних текстів. Всі представлені результати у роботі були зроблені на основі тренуваних у однакових умовах, на однакових наборах даних та протестовані однаковим набором пар текстів.

Результати досліджень були представлені візуально – графіком поширення результатів передбачення відносно очікуваного передбачення. Оцінкою роботи нейромереж є кореляція Пірсона та Спірмена, середньоквадратичне відхилення. Оцінки рахувалися на основі результатів передбачення моделей та очікуваних результатів набору порівняльних даних STS.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. IBM: What is natural language processing? [Електронний ресурс]. Режим доступу: <https://www.ibm.com/topics/natural-language-processing> — Дата доступу : Травень 2023.
2. IBM: What is deep learning [Електронний ресурс]. Режим доступу: <https://www.ibm.com/topics/deep-learning> — Дата доступу : Травень 2023.
3. IBM: What are convolutional neural networks [Електронний ресурс]. Режим доступу: <https://www.ibm.com/topics/convolutional-neural-networks> — Дата доступу : Травень 2023.
4. IBM: What are recurrent neural networks [Електронний ресурс]. Режим доступу: <https://www.ibm.com/topics/recurrent-neural-networks> — Дата доступу : Травень 2023.
5. Long short-term memory, S Hochreiter, J Schmidhuber 1997 [Електронний ресурс]. Режим доступу: <https://papers.baulab.info/Hochreiter-1997.pdf> — Дата доступу : Травень 2023.
6. Siamese Recurrent Architectures for Learning Sentence Similarity J Mueller, A Thyagarajan 2016 [Електронний ресурс]. Режим доступу: <https://ojs.aaai.org/index.php/AAAI/article/view/10350/10209> — Дата доступу : Травень 2023.
7. Learning to forget: Continual prediction with LSTM [Електронний ресурс]. Режим доступу: <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=11540131ea85b2e11d53df7f1360eeb6476e7f4> — Дата доступу : Травень 2023.

8. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation [Электронный ресурс]. Режим доступа: <https://arxiv.org/pdf/1406.1078v3.pdf> — Дата доступа : Травень 2023.
9. Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing, 45(11), 2673-2681. [Электронный ресурс]. Режим доступа: <https://deeplearning.cs.cmu.edu/S23/document/readings/Bidirectional%20Recurrent%20Neural%20Networks.pdf> — Дата доступа : Травень 2023.
10. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling J Chung, C Gulcehre, KH Cho, Y Bengio 2014 [Электронный ресурс]. Режим доступа: <https://arxiv.org/pdf/1412.3555> — Дата доступа : Травень 2023.
11. GloVe: Global Vectors for Word Representation J Pennington, R Socher 2014 [Электронный ресурс]. Режим доступа: <https://aclanthology.org/D14-1162.pdf> — Дата доступа : Травень 2023.
12. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Cross-lingual Focused Evaluation D Cer, M Diab, E Agirre, I Lopez-Gazpio 2017 [Электронный ресурс]. Режим доступа: <https://arxiv.org/pdf/1708.00055> — Дата доступа : Травень 2023.
13. How to predict Quora Question Pairs using Siamese Manhattan LSTM Elinor Cohen [Электронный ресурс]. Режим доступа: <https://blog.mlreview.com/implementing-malstm-on-kaggles-quora-question-pairs-competition-8b31b0b16a07> — Дата доступа : Травень 2023.
14. word2vec KW Church - Natural Language Engineering, 2017 [Электронный ресурс]. Режим доступа: [https://www.cambridge.org/core/services/aop-](https://www.cambridge.org/core/services/aop-cambridge-)
[cambridge-](https://www.cambridge.org/core/services/aop-cambridge-)

- [core/content/view/B84AE4446BD47F48847B4904F0B36E0B/S1351324916000334a.pdf/div-class-title-word2vec-div.pdf](https://arxiv.org/abs/1808.06452v1) — Дата доступа : Травень 2023.
15. Module: tf.keras.optimizers | TensorFlow [Электронный ресурс]. Режим доступа: https://www.tensorflow.org/api_docs/python/tf/keras/optimizers — Дата доступа : Травень 2023.
16. STS Benchmark Dataset | Papers With Code [Электронный ресурс]. Режим доступа: <https://paperswithcode.com/dataset/sts-benchmark> — Дата доступа : Травень 2023.
17. Understanding LSTM Networks [Электронный ресурс]. Режим доступа: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/> — Дата доступа : Травень 2023.

ДОДАТОК 1. ГРАФІКИ ПОШИРЕННЯ ПЕРЕДБАЧЕННЯ СХОЖОСТІ

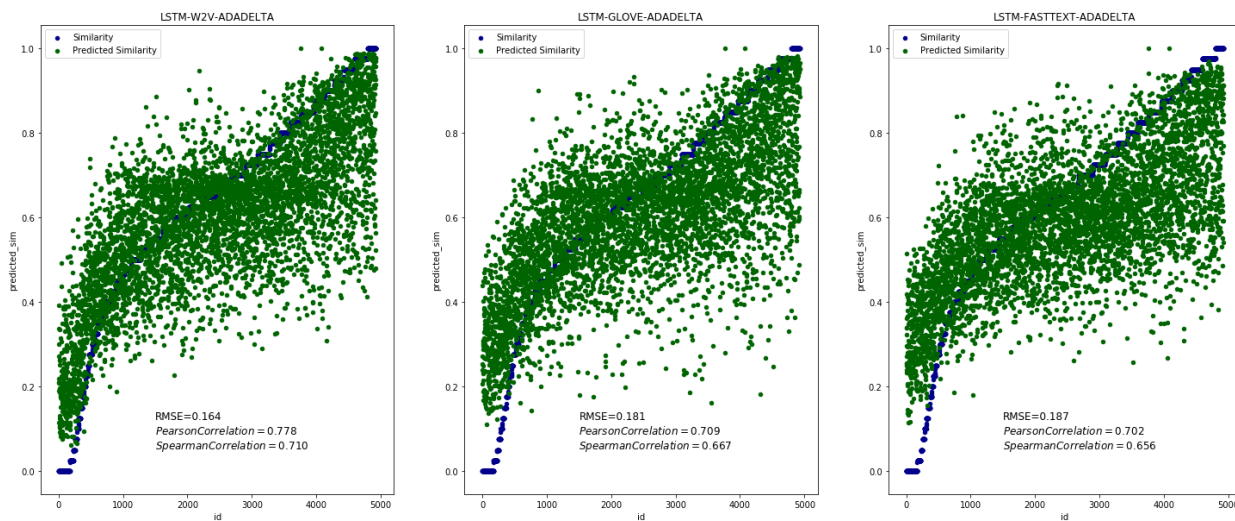


Рис. Д1.1. Поширення передбачення MALSTM з оптимізатором Adadelta

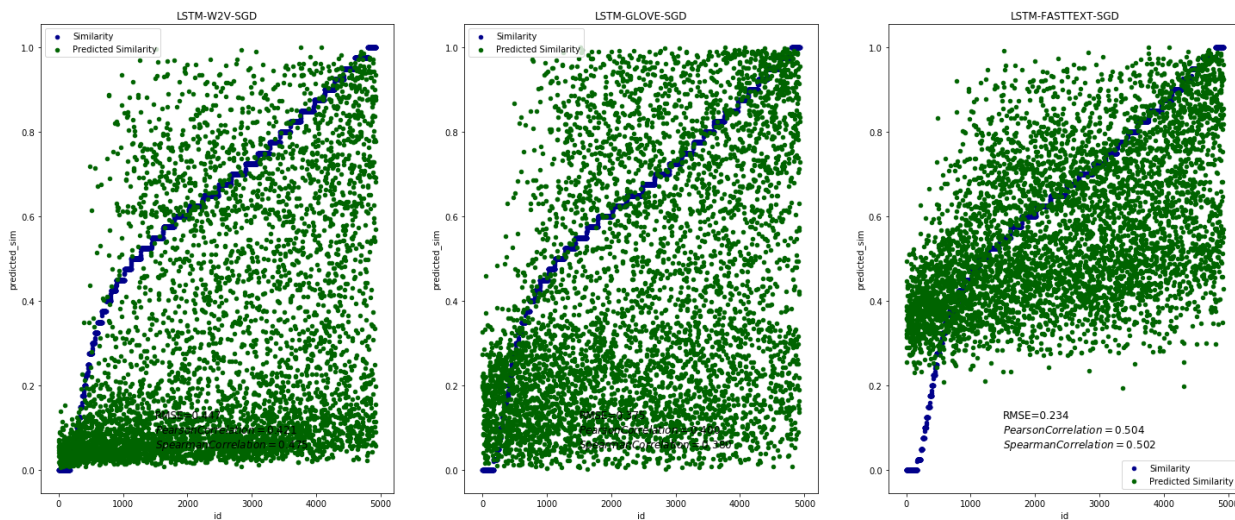


Рис. Д1.2. Поширення передбачення MALSTM з оптимізатором SGD

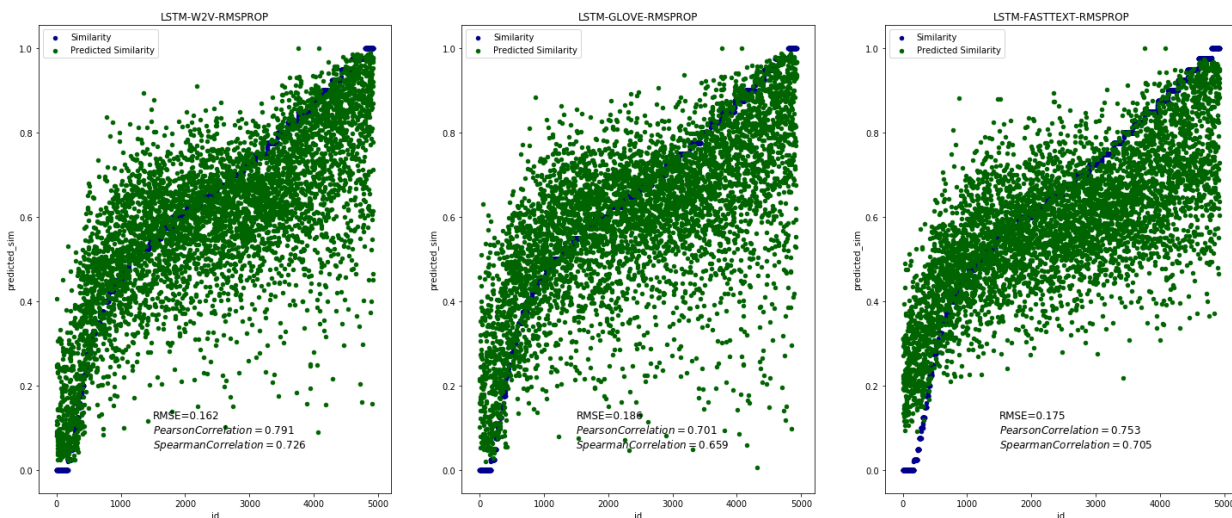


Рис. Д1.3. Поширення передбачення MALSTM з оптимізатором RMSPROP

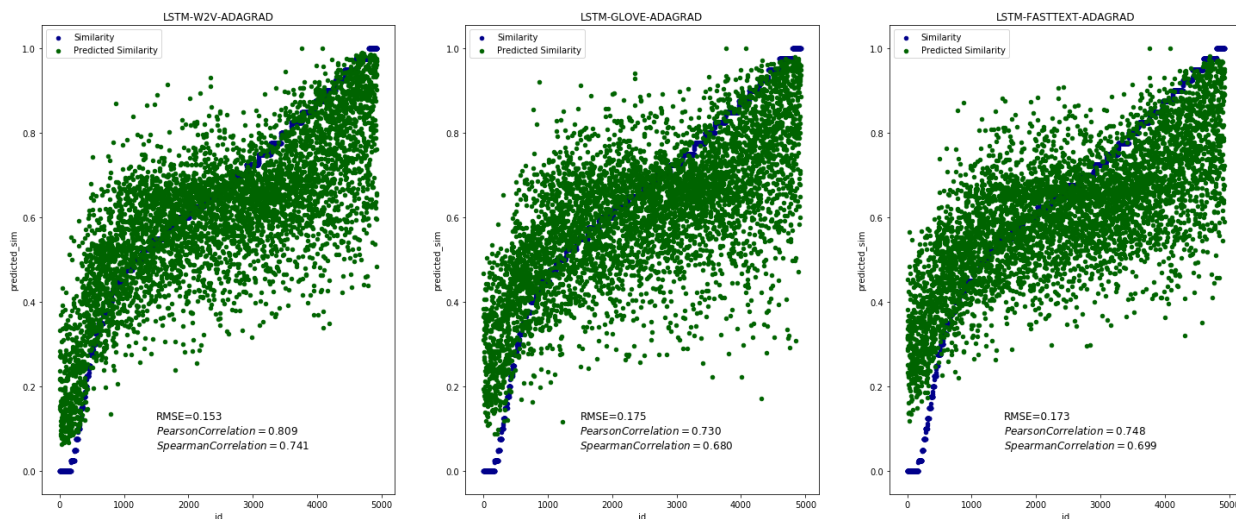


Рис. Д1.4. Поширення передбачення MALSTM з оптимізатором ADAGRAD

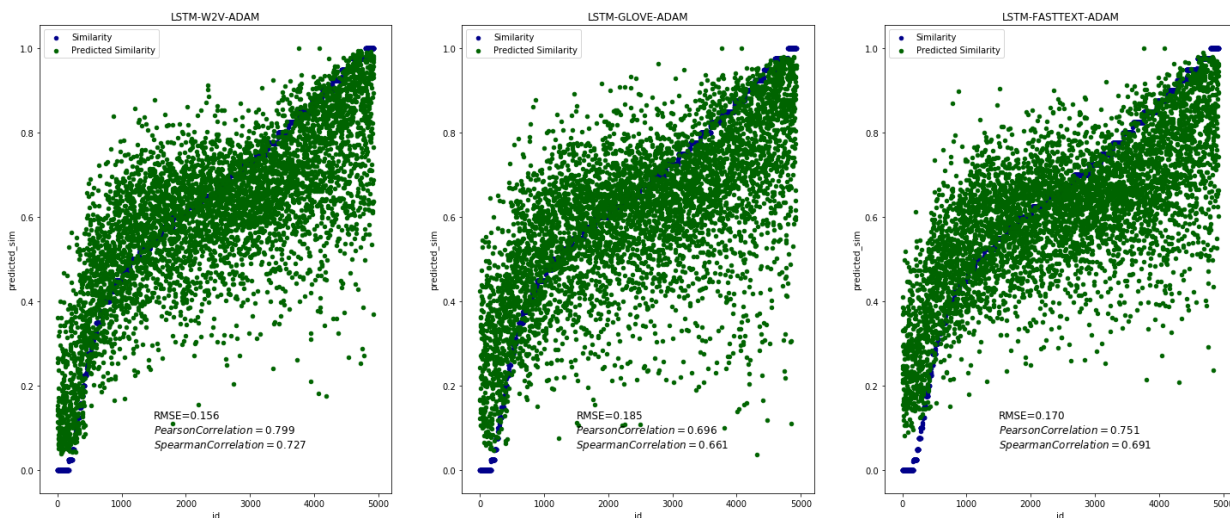


Рис. Д1.5. Поширення передбачення MALSTM з оптимізатором ADAM

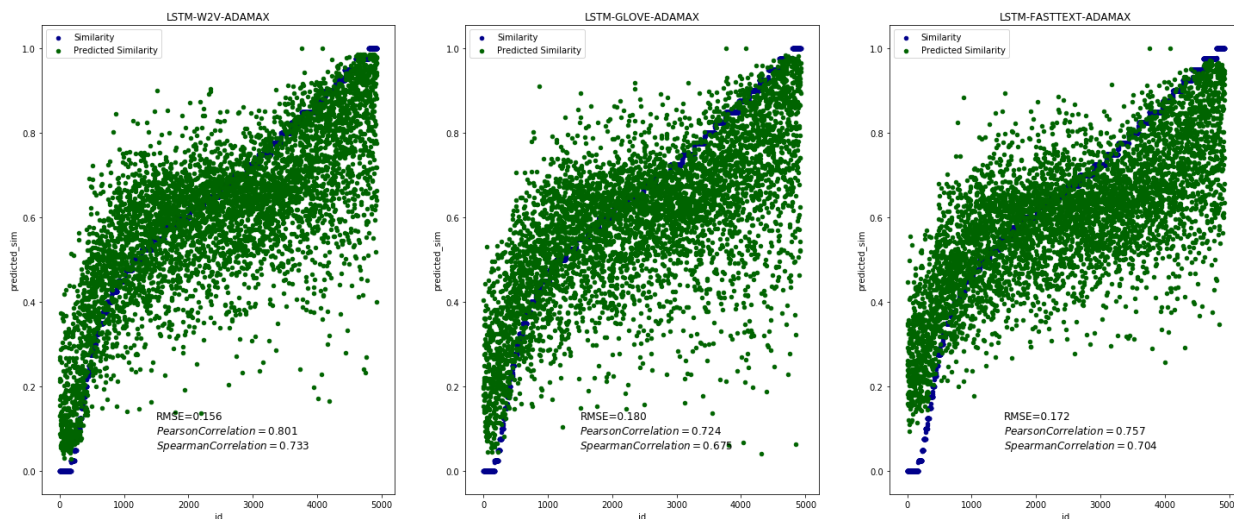


Рис. Д1.6. Поширення передбачення MALSTM з оптимізатором ADAMAX

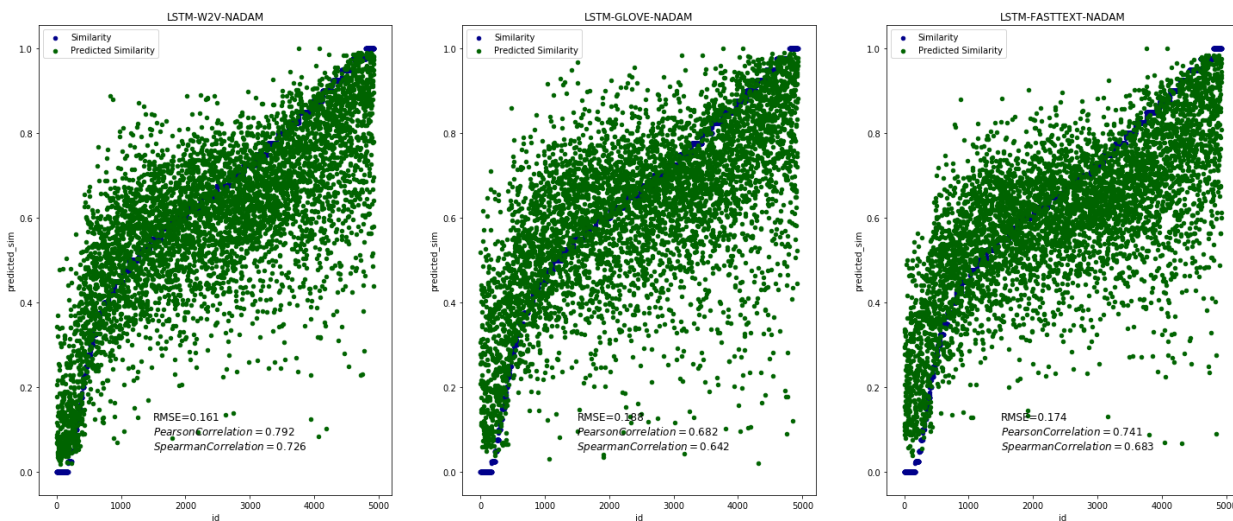


Рис. Д1.7. Поширення передбачення MALSTM з оптимізатором NADAM

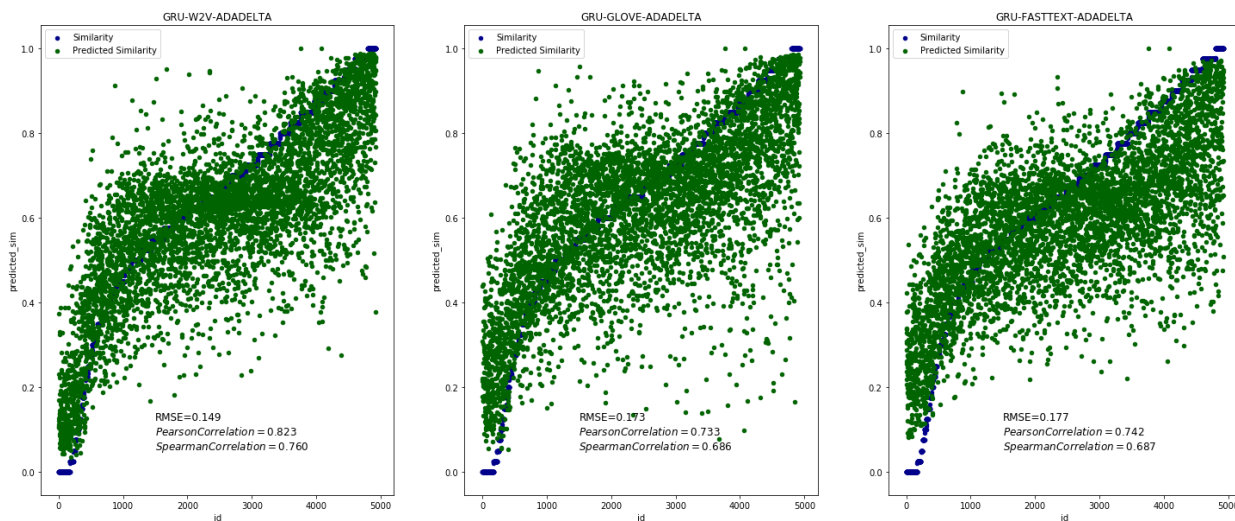


Рис. Д1.8. Поширення передбачення MAGRU з оптимізатором Adadelat

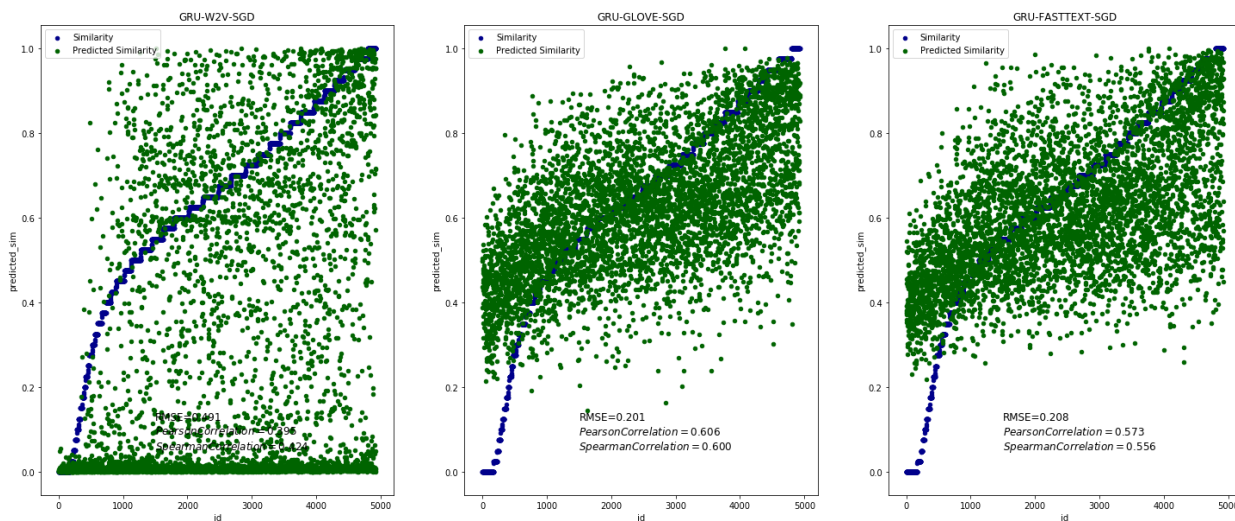


Рис. Д1.9. Поширення передбачення MAGRU з оптимізатором SGD

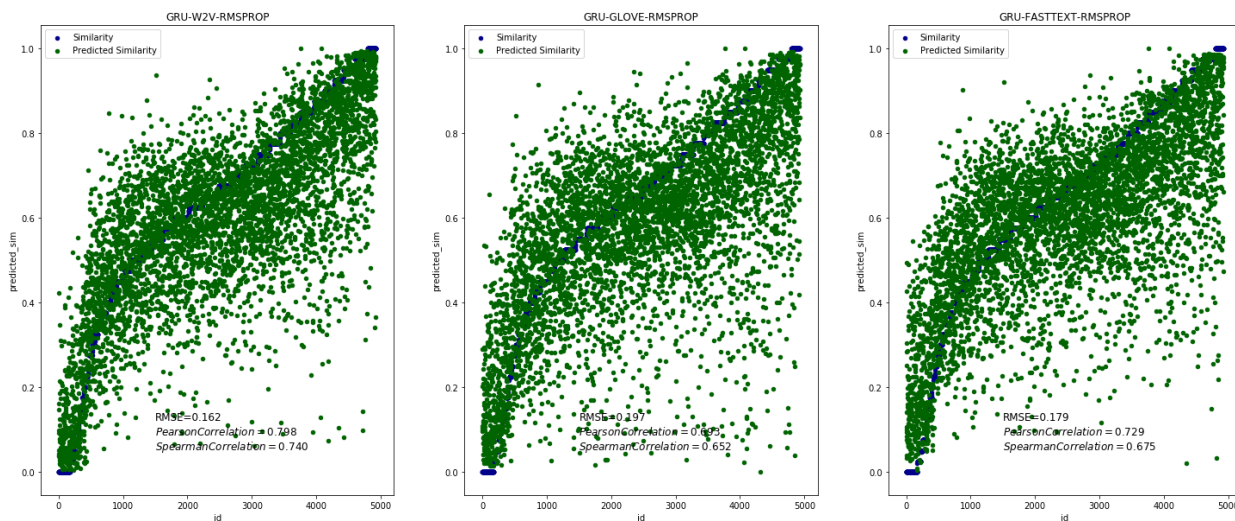


Рис. Д1.10. Поширення передбачення MAGRU з оптимізатором RMSPROP

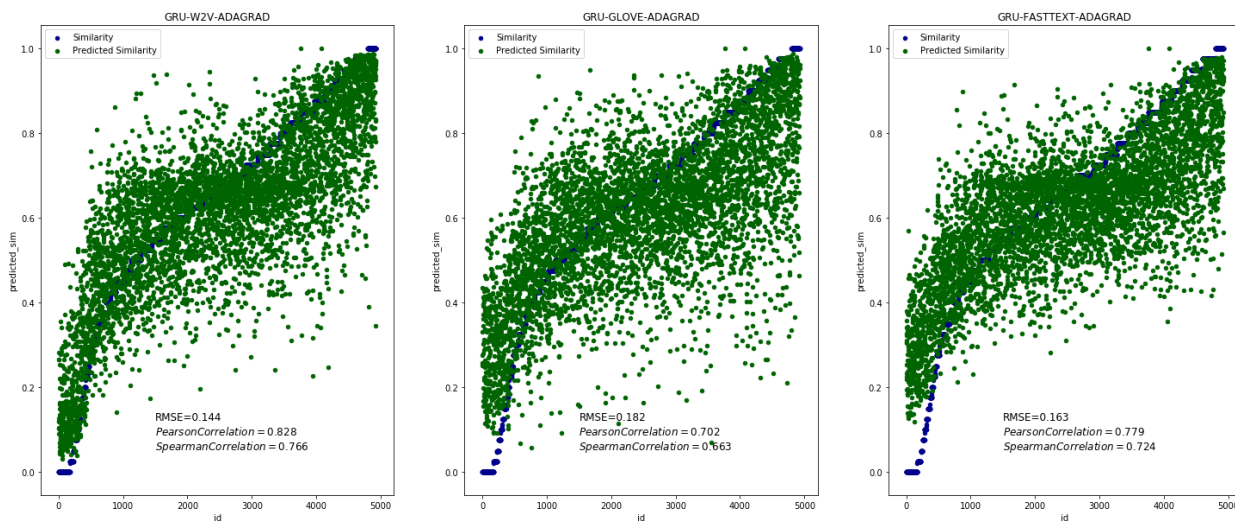


Рис. Д1.11. Поширення передбачення MAGRU з оптимізатором ADAGRAD

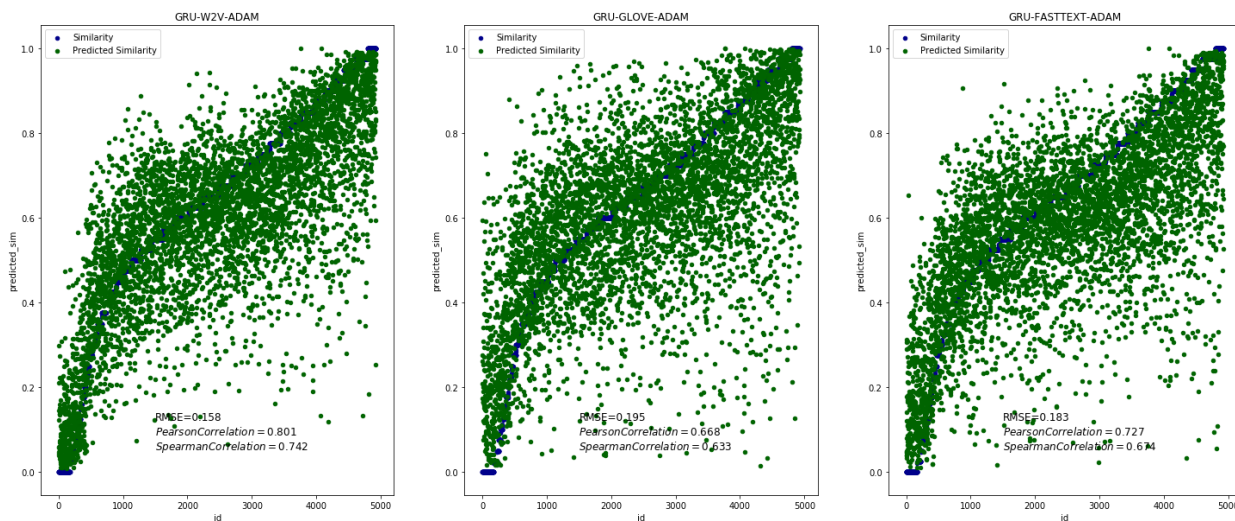


Рис. Д1.12. Поширення передбачення MAGRU з оптимізатором ADAM

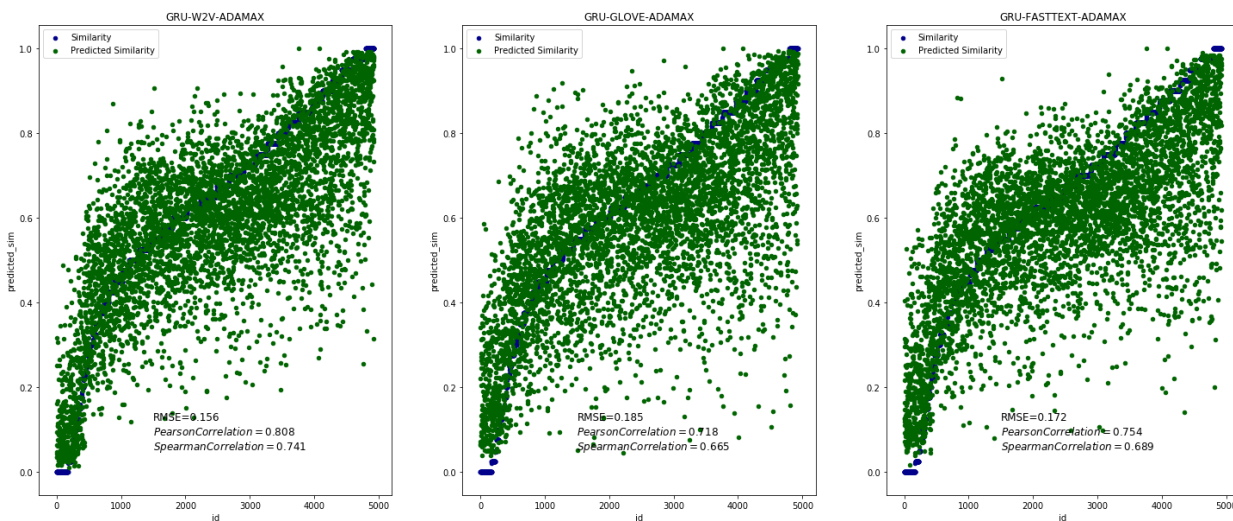


Рис. Д1.13. Поширення передбачення MAGRU з оптимізатором ADAMAX

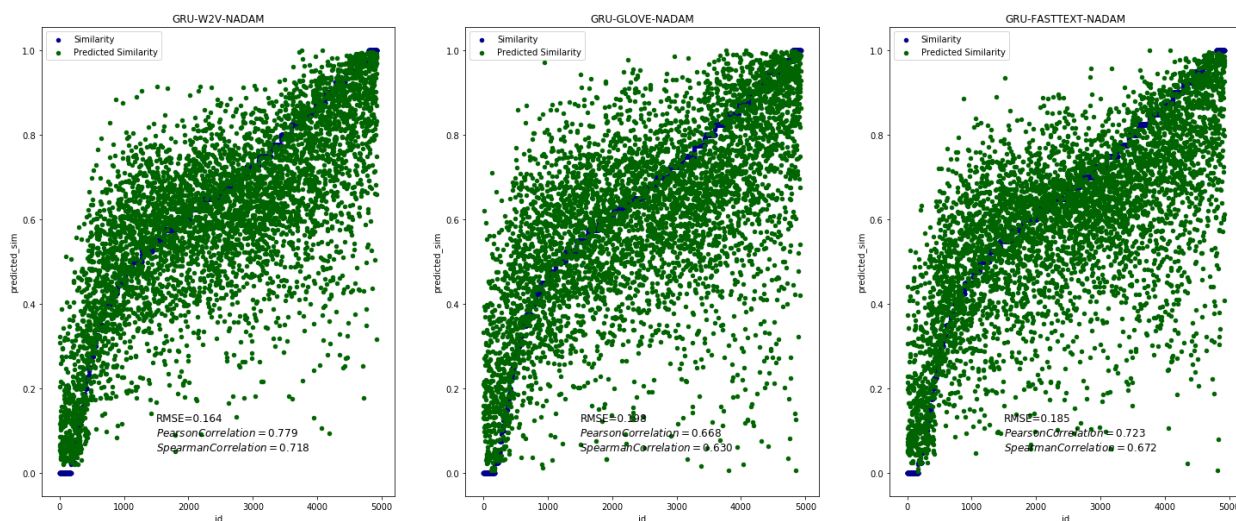


Рис. Д1.14. Поширення передбачення MAGRU з оптимізатором NADAM

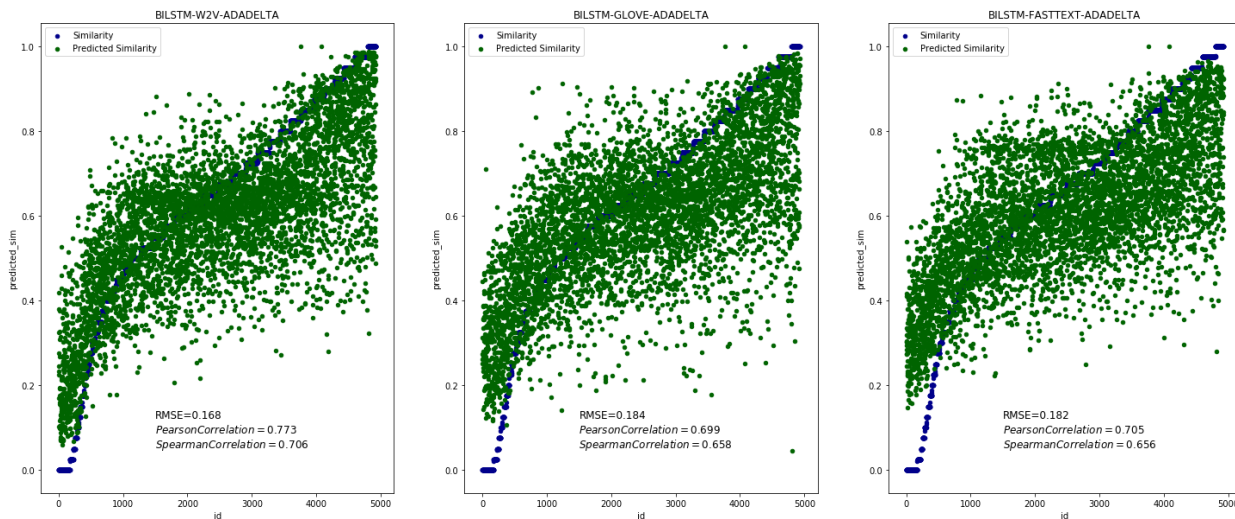


Рис. Д1.15. Поширення передбачення MABILSTM з оптимізатором Adadelta

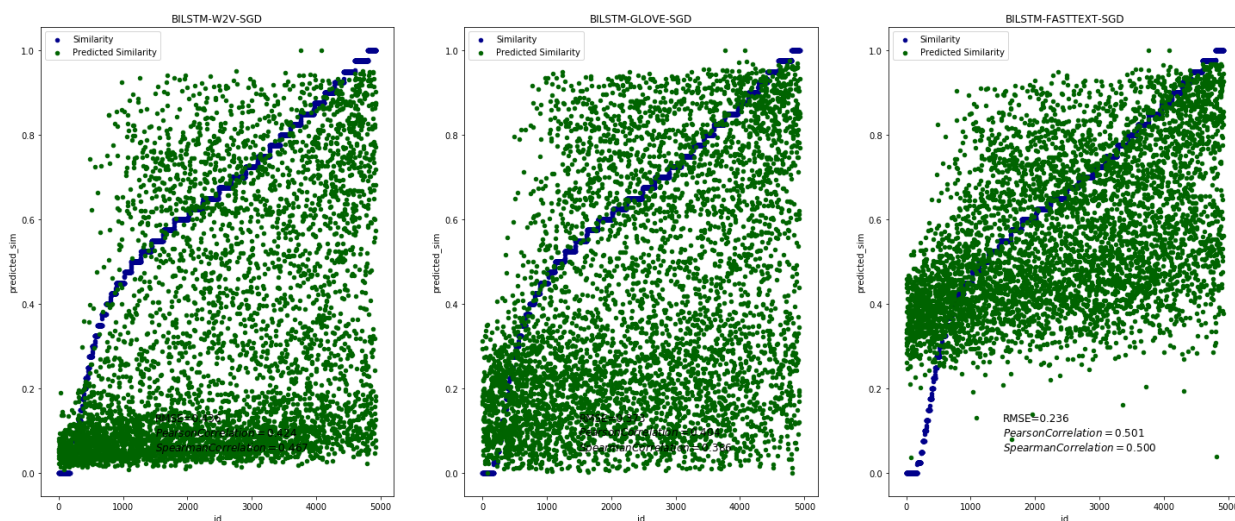


Рис. Д1.16. Поширення передбачення MABILSTM з оптимізатором SGD

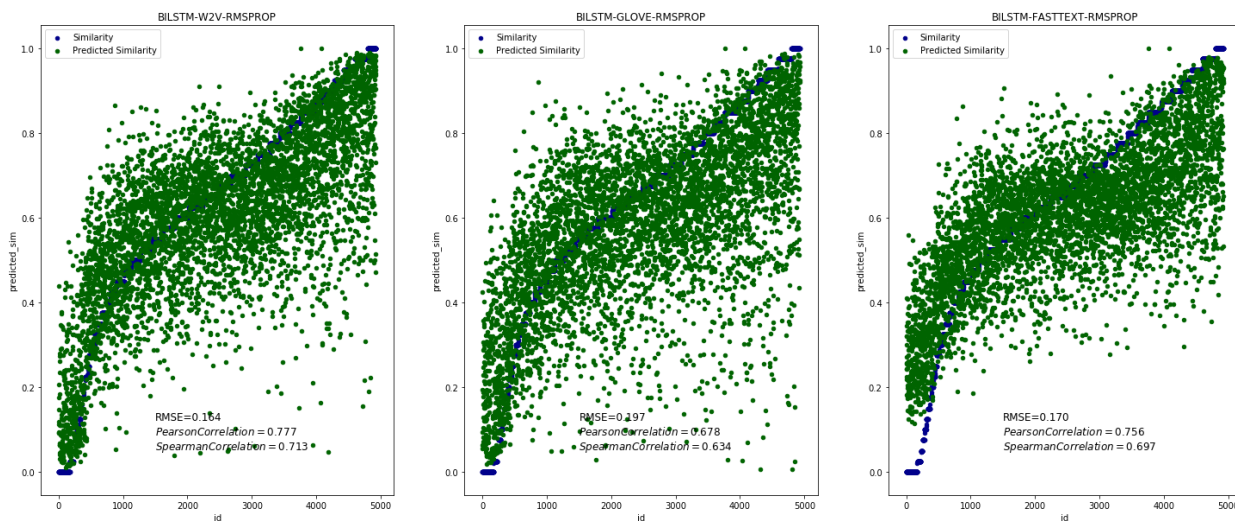


Рис. Д1.17. Поширення передбачення MABiLSTM з оптимізатором RMSPROP

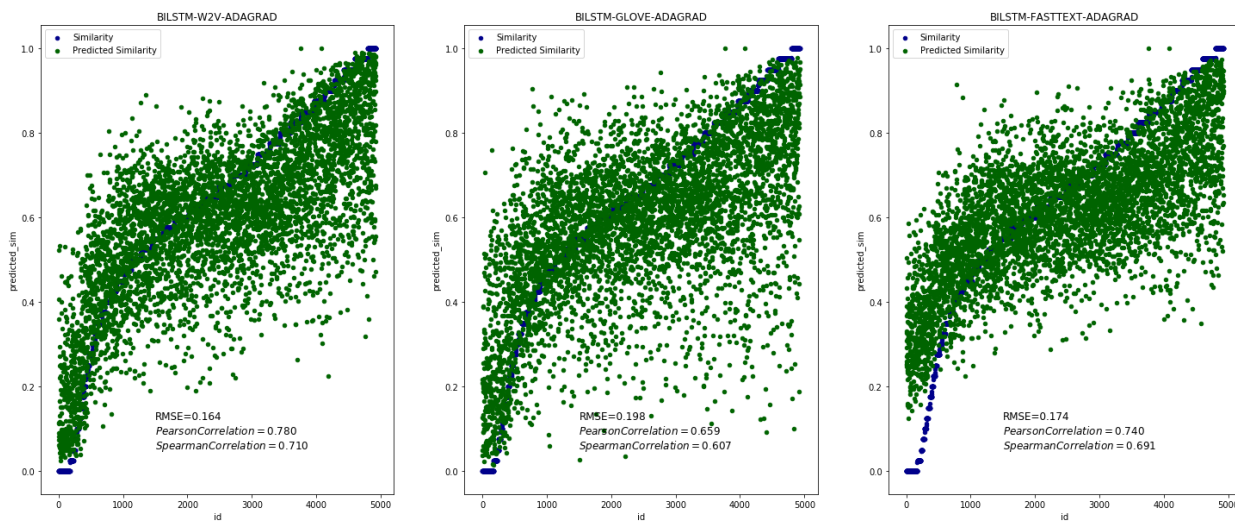


Рис. Д1.18. Поширення передбачення MABiLSTM з оптимізатором ADAGRAD

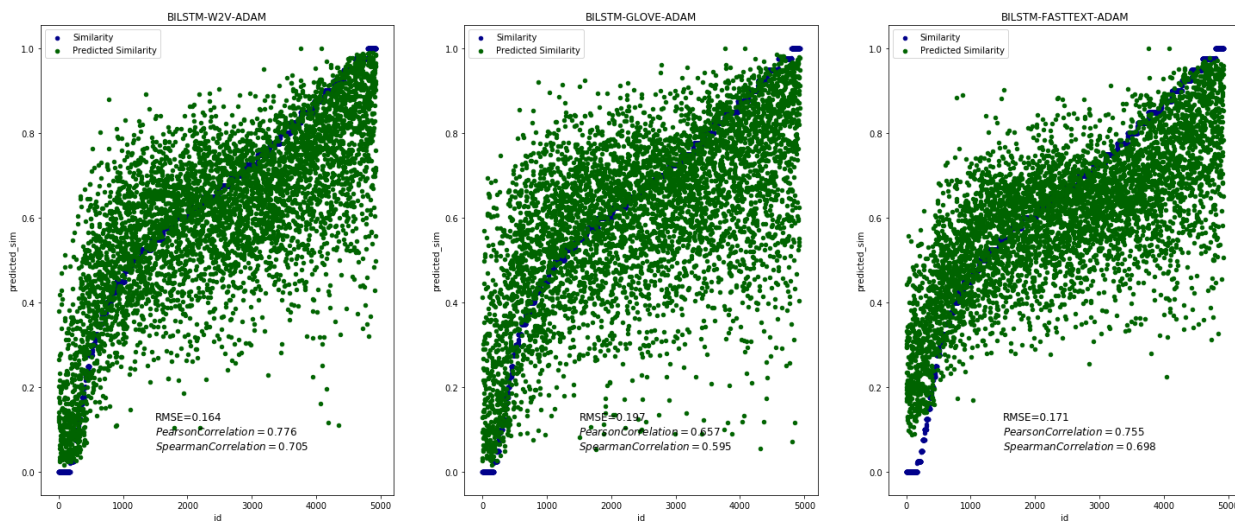


Рис. Д1.19. Поширення передбачення MABILSTM з оптимізатором ADAM

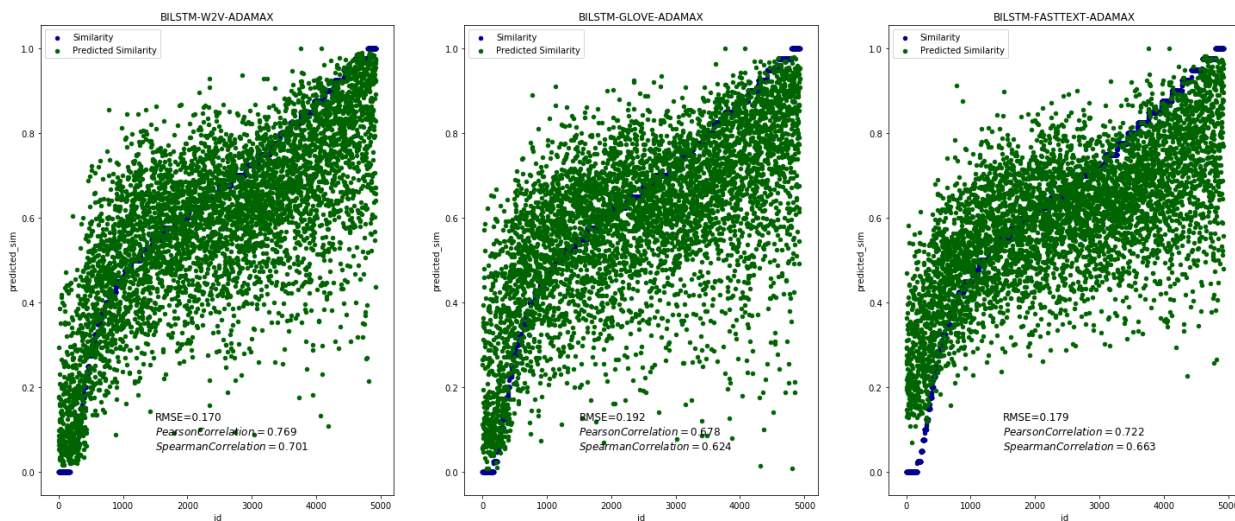


Рис. Д1.20. Поширення передбачення MABILSTM з оптимізатором ADAMAX

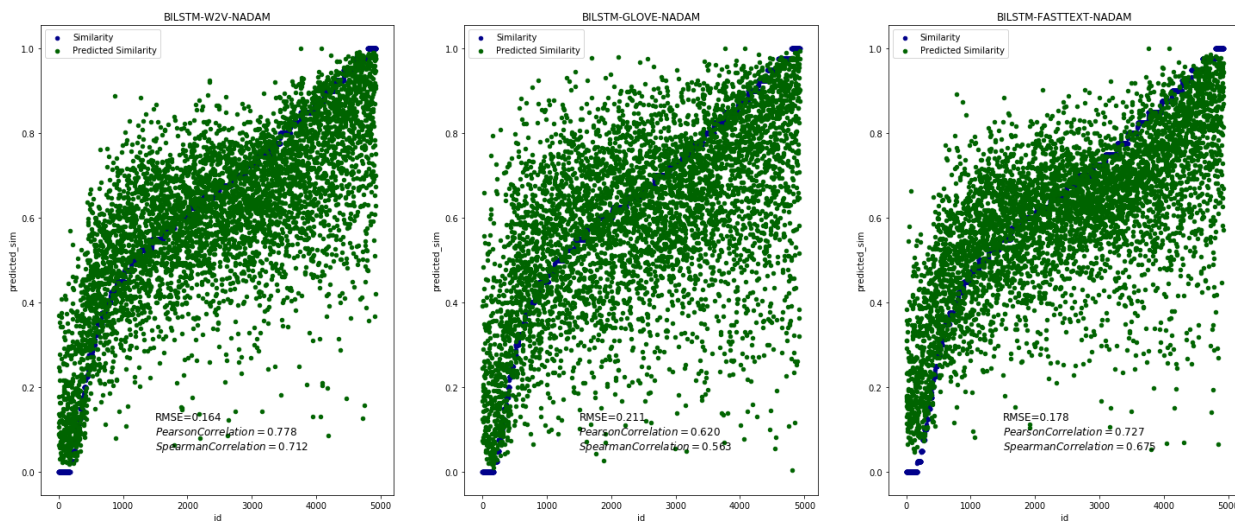


Рис. Д1.21. Поширення передбачення MABILSTM з оптимізатором NADAM

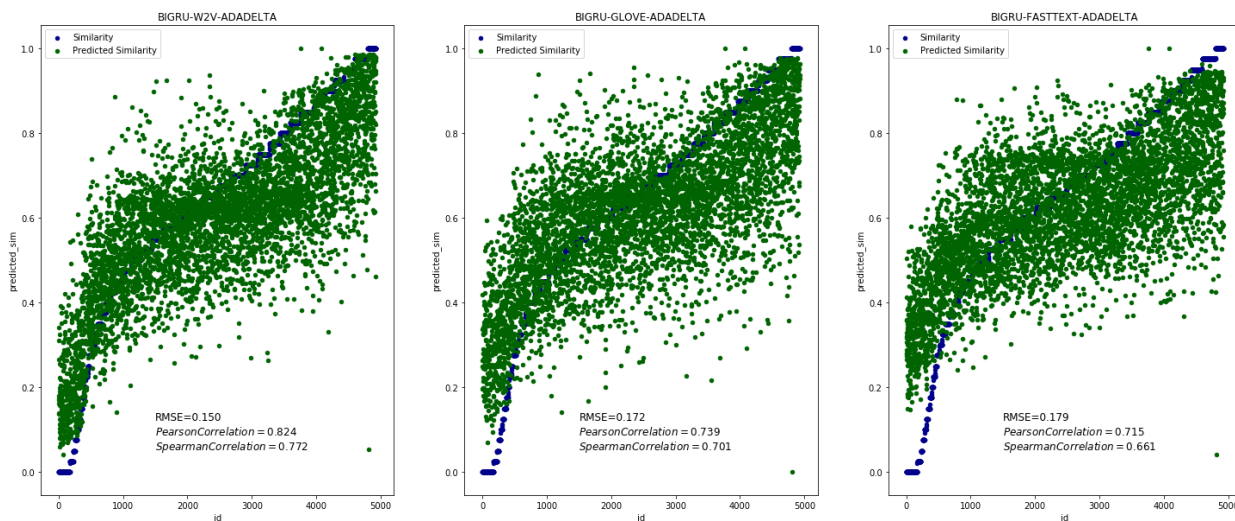


Рис. Д1.22. Поширення передбачення MABIGRU з оптимізатором Adadelta

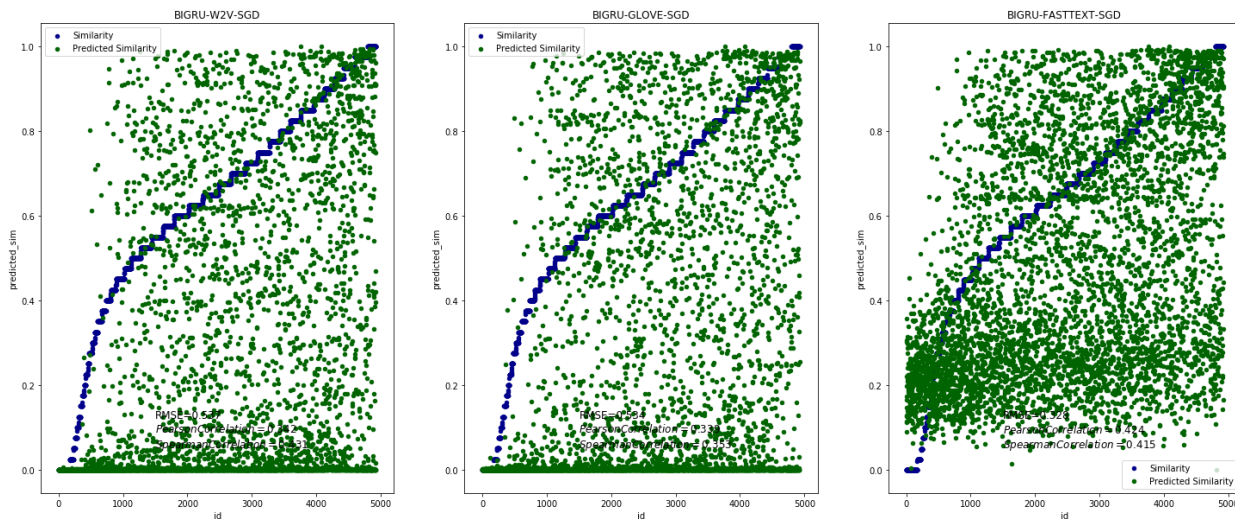


Рис. Д1.23. Поширення передбачення MABIGRU з оптимізатором SGD

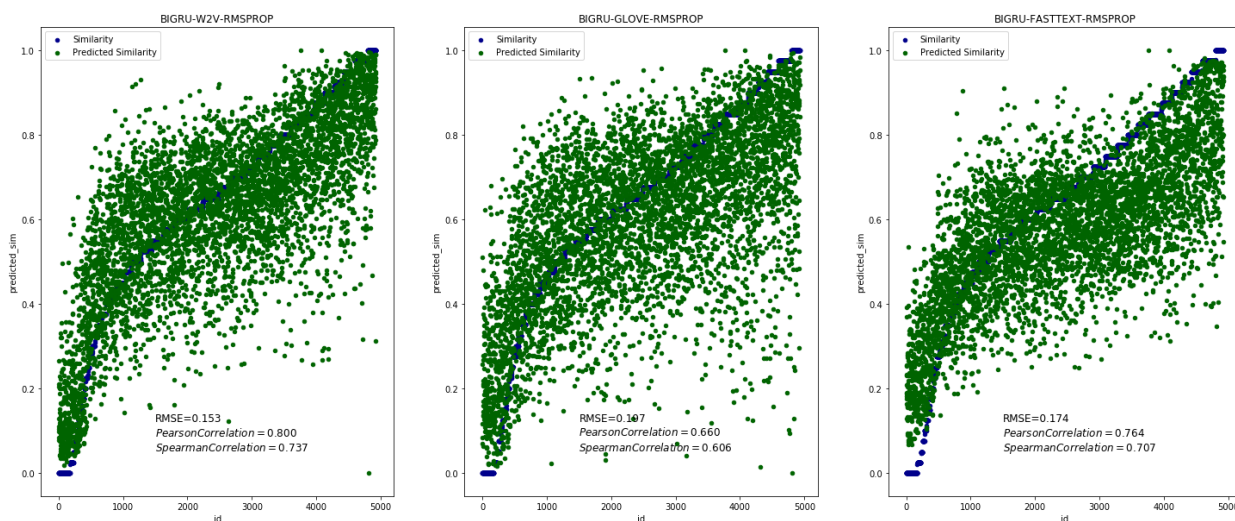


Рис. Д1.24. Поширення передбачення MABIGRU з оптимізатором RMSPROP

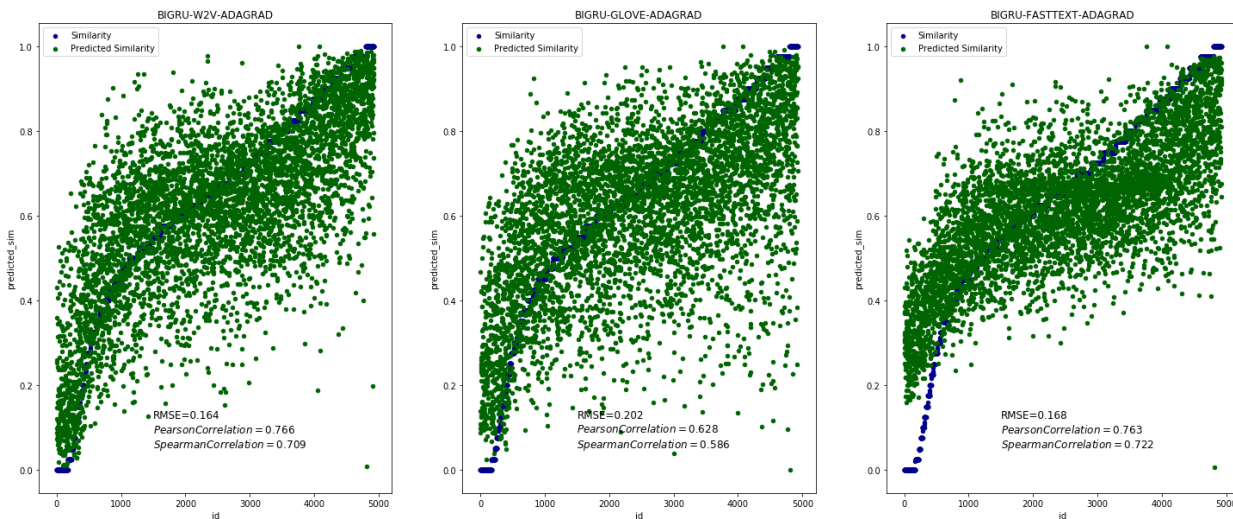


Рис. Д1.25. Поширення передбачення MABIGRU з оптимізатором ADAGRAD

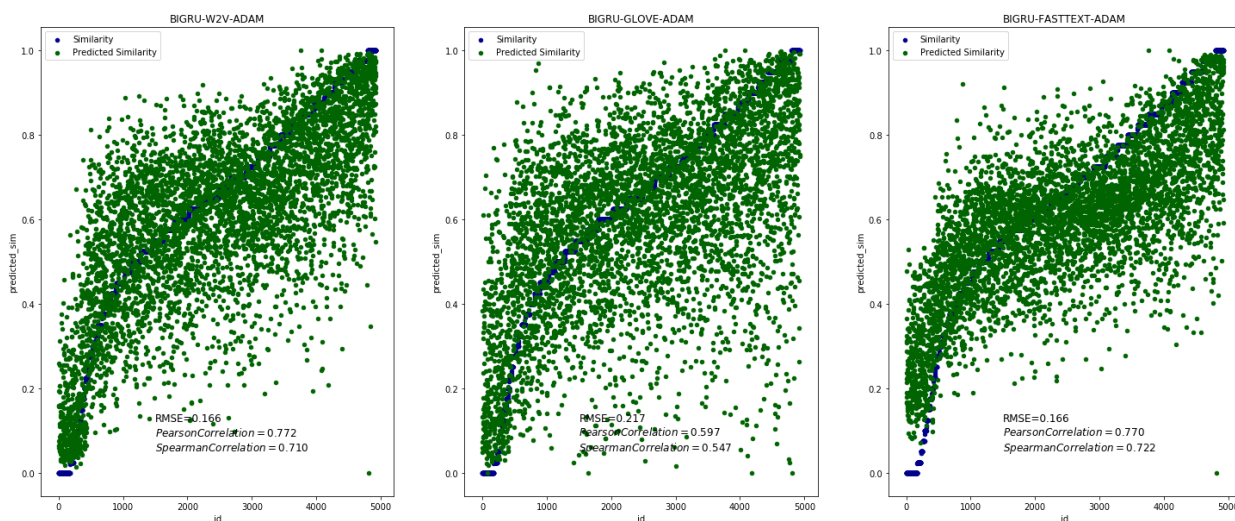


Рис. Д1.26. Поширення передбачення MABIGRU з оптимізатором ADAM

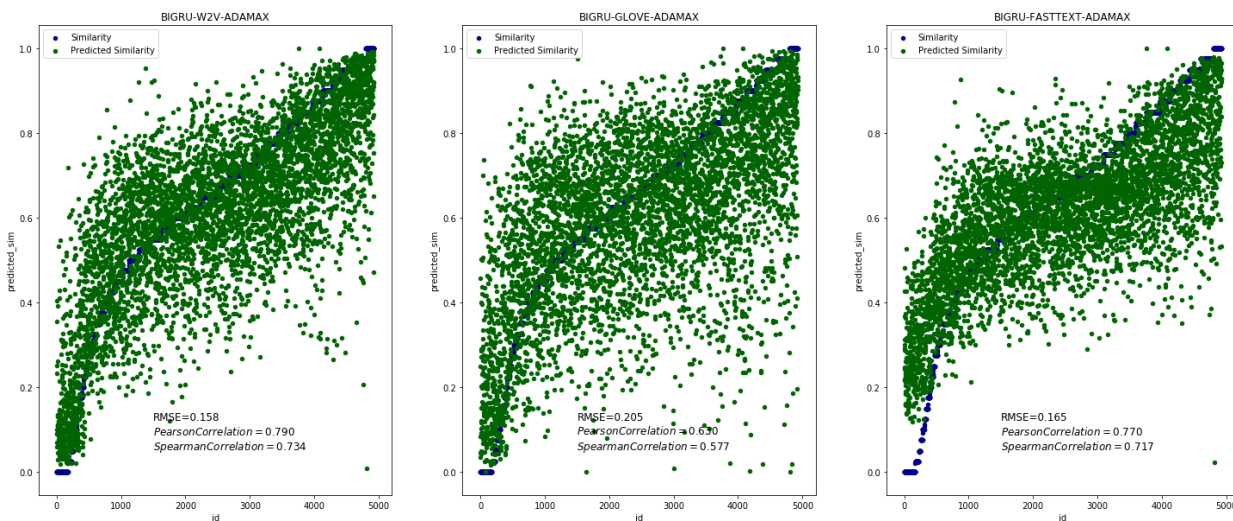


Рис. Д1.27. Поширення передбачення МАВІGRU з оптимізатором ADAMAX

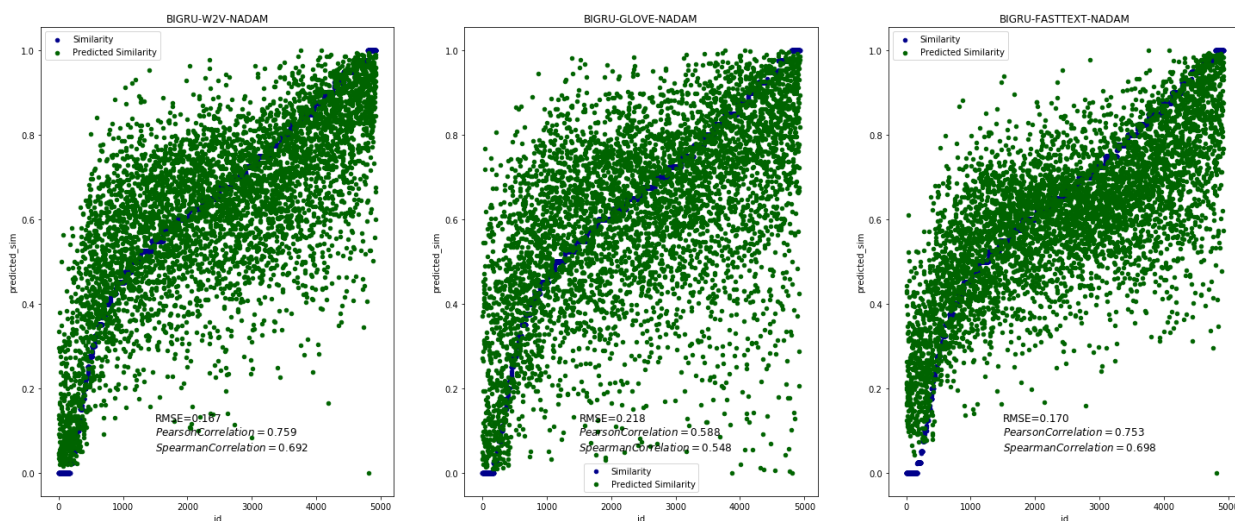


Рис. Д1.28. Поширення передбачення МАВІGRU з оптимізатором NADAM

ДОДАТОК 2. ТАБЛИЦЯ РЕЗУЛЬТАТІВ КРИТЕРІЇВ ОЦІНКИ ЕФЕКТИВНОСТІ МОДЕДЕЙ

Таблиця Д2.1.

Результати дослідження моделей

Модель	Оптимізатор	Вставка	Середньоквадратичне відхилення	Коефіцієнт кореляції Пірсона	Коефіцієнт кореляції Спірмена
LSTM	Adadelta	Word2vec	0.164	0.778	0.710
LSTM	Adadelta	GloVe	0.181	0.709	0.667
LSTM	Adadelta	Fasttext	0.187	0.702	0.656
LSTM	SGD	Word2vec	0.447	0.421	0.475
LSTM	SGD	GloVe	0.375	0.400	0.380
LSTM	SGD	Fasttext	0.234	0.504	0.502
LSTM	RMSPROP	Word2vec	0.162	0.791	0.726
LSTM	RMSPROP	GloVe	0.186	0.701	0.659
LSTM	RMSPROP	Fasttext	0.175	0.753	0.705
LSTM	ADAGRAD	Word2vec	0.153	0.809	0.741
LSTM	ADAGRAD	GloVe	0.175	0.730	0.680
LSTM	ADAGRAD	Fasttext	0.173	0.748	0.699
LSTM	ADAM	Word2vec	0.156	0.799	0.727
LSTM	ADAM	GloVe	0.185	0.696	0.661
LSTM	ADAM	Fasttext	0.170	0.751	0.691
LSTM	ADAMAX	Word2vec	0.156	0.801	0.733
LSTM	ADAMAX	GloVe	0.180	0.724	0.675
LSTM	ADAMAX	Fasttext	0.172	0.757	0.704
LSTM	NADAM	Word2vec	0.161	0.792	0.726
LSTM	NADAM	GloVe	0.188	0.682	0.642
LSTM	NADAM	Fasttext	0.174	0.741	0.683
GRU	Adadelta	Word2vec	0.149	0.823	0.760
GRU	Adadelta	GloVe	0.173	0.733	0.686
GRU	Adadelta	Fasttext	0.177	0.742	0.687
GRU	SGD	Word2vec	0.491	0.395	0.424
GRU	SGD	GloVe	0.201	0.606	0.600

1	2	3	4	5	6
GRU	SGD	Fasttext	0.208	0.573	0.556
GRU	RMSPROP	Word2vec	0.162	0.798	0.740
GRU	RMSPROP	GloVe	0.197	0.693	0.652
GRU	RMSPROP	Fasttext	0.179	0.729	0.675
GRU	ADAGRAD	Word2vec	0.144	0.828	0.766
GRU	ADAGRAD	GloVe	0.182	0.702	0.663
GRU	ADAGRAD	Fasttext	0.163	0.779	0.724
GRU	ADAM	Word2vec	0.158	0.801	0.742
GRU	ADAM	GloVe	0.195	0.668	0.633
GRU	ADAM	Fasttext	0.183	0.727	0.674
GRU	ADAMAX	Word2vec	0.156	0.808	0.741
GRU	ADAMAX	GloVe	0.185	0.718	0.665
GRU	ADAMAX	Fasttext	0.172	0.754	0.689
GRU	NADAM	Word2vec	0.164	0.779	0.718
GRU	NADAM	GloVe	0.198	0.668	0.630
GRU	NADAM	Fasttext	0.185	0.723	0.672
BiLSTM	Adadelta	Word2vec	0.168	0.773	0.706
BiLSTM	Adadelta	GloVe	0.184	0.699	0.658
BiLSTM	Adadelta	Fasttext	0.182	0.705	0.656
BiLSTM	SGD	Word2vec	0.436	0.424	0.467
BiLSTM	SGD	GloVe	0.373	0.404	0.386
BiLSTM	SGD	Fasttext	0.236	0.501	0.500
BiLSTM	RMSPROP	Word2vec	0.164	0.777	0.713
BiLSTM	RMSPROP	GloVe	0.197	0.678	0.634
BiLSTM	RMSPROP	Fasttext	0.170	0.756	0.697
BiLSTM	ADAGRAD	Word2vec	0.164	0.780	0.710
BiLSTM	ADAGRAD	GloVe	0.198	0.659	0.607
BiLSTM	ADAGRAD	Fasttext	0.174	0.740	0.691
BiLSTM	ADAM	Word2vec	0.164	0.776	0.705
BiLSTM	ADAM	GloVe	0.197	0.657	0.595
BiLSTM	ADAM	Fasttext	0.171	0.755	0.698
BiLSTM	ADAMAX	Word2vec	0.170	0.769	0.701
BiLSTM	ADAMAX	GloVe	0.192	0.678	0.624

1	2	3	4	5	6
BiLSTM	ADAMAX	Fasttext	0.179	0.722	0.663
BiLSTM	NADAM	Word2vec	0.164	0.778	0.712
BiLSTM	NADAM	GloVe	0.211	0.620	0.563
BiLSTM	NADAM	Fasttext	0.178	0.727	0.675
BiGRU	Adadelta	Word2vec	0.164	0.784	0.708
BiGRU	Adadelta	GloVe	0.192	0.692	0.638
BiGRU	Adadelta	Fasttext	0.177	0.734	0.681
BiGRU	SGD	Word2vec	0.537	0.342	0.431
BiGRU	SGD	GloVe	0.534	0.339	0.353
BiGRU	SGD	Fasttext	0.328	0.424	0.415
BiGRU	RMSPROP	Word2vec	0.153	0.800	0.737
BiGRU	RMSPROP	GloVe	0.197	0.660	0.606
BiGRU	RMSPROP	Fasttext	0.174	0.764	0.707
BiGRU	ADAGRAD	Word2vec	0.164	0.766	0.709
BiGRU	ADAGRAD	GloVe	0.202	0.628	0.586
BiGRU	ADAGRAD	Fasttext	0.168	0.763	0.722
BiGRU	ADAM	Word2vec	0.166	0.772	0.710
BiGRU	ADAM	GloVe	0.217	0.597	0.547
BiGRU	ADAM	Fasttext	0.166	0.770	0.722
BiGRU	ADAMAX	Word2vec	0.158	0.790	0.734
BiGRU	ADAMAX	GloVe	0.205	0.630	0.577
BiGRU	ADAMAX	Fasttext	0.165	0.770	0.717
BiGRU	NADAM	Word2vec	0.167	0.759	0.692
BiGRU	NADAM	GloVe	0.218	0.588	0.548
BiGRU	NADAM	Fasttext	0.170	0.753	0.698