УДК 004.75 DOi 10.30837/ bi.2021.2(97).01

М.П. Дудник[1], С.Г. Удовенко[2], Л.Е. Чала [3], М.М. Соколовська [4]

[1] студент групи ДСм-21-1,
Харківський національний університет радіоелектроніки,
mykola.dudnyk@nure.ua, ORCID iD: 0000-0002-8259-0221,

[2] доктор технічних наук, професор, завідувач кафедри інформатики та комп'ютерної техніки,
Харківський національний економічний університет ім. С. Кузнеця,
serhiy.udovenko@hneu.net, ORCID iD: 0000-0001-5945-8647

[3] кандидат технічних наук, доцент, доцент кафедри штучного інтелекту,
Харківський національний університет радіоелектроніки,
larysa.chala@nure.ua, ORCID iD: 0000-0002-9890-4790

[4] студентка групи ІТШІ-19-3,
Харківський національний університет радіоелектроніки,
mariia.sokolovska@nure.ua, ORCID iD: 0000-0002-9789-6928

НЕЙРОМЕРЕЖЕВА ТЕХНОЛОГІЯ БАГАТОМОВНОЇ КЛАСИФІКАЦІЇ ЕЛЕКТРОННИХ ТЕКСТІВ

Статтю присвячено розробці технології побудови багатомовних класифікаторів, яка основана на нейромережевій обробці векторного подання текстів, згенерованого за допомогою моделі XLM-RoBerta. Розглянуто переваги використання для векторизації текстів рекуррентної нейронної мережі на основі трансформеру моделі XLM-RoBerta. Наведено схему взаємодії розробленого класифікатору на основі мережі LSTM з моделлю векторизації текстів. Запропоноване архітектурне рішення обумовлено необхідністю оптимізації витрат ресурсів та їх економії під час використання моделі у релізному середовищі за допомогою розробленого веб-сервісу. Здійснено програмну реалізацію запропонованої технології класифікації. Програмний додаток реалізовано засобами мови програмування Python за допомогою бібліотеки для машинного навчання TensorFlow та комплексної платформи Tensorflow Extended. Серверну частину реалізовано з використанням фреймворку aiohttp. Експериментальне дослідження розробленого класифікатору текстів здійснено з використанням News Category Dataset, що представляє собою багатомовні заголовки текстових новин. Застосування запропонованої технології класифікації характеризується незначним погіршенням показників якості під час зміни мови, що дозволяє розробляти багатомовні моделі без втрати їх продуктивності при зміні мови вхідних даних. Результати тестування підтверджують ефективність наведеного підходу.

ВЕКТОРИЗАЦІЯ БАГАТОМОВНИХ ТЕКСТІВ, МОДЕЛЬ XLM-RoBerta, НЕЙРОМЕРЕЖЕВИЙ КЛАСИФІКАТОР БАГАТОМОВНИХ ТЕКСТІВ, МЕРЕЖА LSTM, СЕРВЕР ОБРОБКИ ЗАПИТІВ

Дудник М.П., Удовенко С.Г., Чалая Л.Э., Соколовская М.М. Нейросетевая технология многоязычной классификации электронных текстов. Статья посвящена разработке технологии построения многоязычных классификаторов, основанной на нейросетевой обработке векторного представления текстов, сгенерированного с помощью модели XLM-RoBerta. Рассмотрены преимущества использования векторизации текстов рекуррентной нейронной сети на основе трансформера модели XLM-RoBerta. Приведена схема взаимодействия разработанного классификатора на основе сети LSTM с моделью векторизации текстов. Предложенное архитектурное решение обусловлено необходимостью оптимизации затрат ресурсов и их экономии при использовании модели в релизной среде с помощью разработанного веб-сервиса. Осуществлена программная реализация предлагаемой технологии классификации. Программное приложение реализовано на языке программирования Python с помощью библиотеки для машинного обучения TensorFlow и комплексной платформы Tensorflow Extended. Серверная часть реализована с использованием фреймворка aiohttp. Экспериментальное исследование разработанного классификатора текстов проведено с использованием News Category Dataset, представляющего собой многоязычные заголовки текстовых новостей. Применение предлагаемой технологии классификации характеризуется незначительным ухудшением показателей качества при изменении языка, что позволяет разрабатывать многоязычные модели без потери производительности при изменении языка входных данных. Результаты тестирования подтверждают эффективность приведенного подхода.

ВЕКТОРИЗАЦИЯ МНОГОЯЗЫЧНЫХ ТЕКСТОВ, МОДЕЛЬ XLM-RoBerta, НЕЙРОСЕТЕВЫЙ КЛАССИФИКАТОР МНОГОЯЗЫЧНЫХ ТЕКСТОВ, СЕТЬ LSTM, СЕРВЕР ОБРАБОТКИ ЗАПРОСОВ

Dudnyk M., Udovenko S., Chala L., Sokolovska M. Neural network technology for multilingual classification of electronic texts. The article is devoted to the development of a technology for building multilingual classifiers based on neural network processing of a vector representation of texts generated using the XLM-RoBerta model. The advantages of using the text vectorization of a recurrent neural network based on the transformer of the XLM-RoBerta model are considered. The interaction scheme of the developed classifier based on the LSTM network with the text vectorization model is presented. The proposed architectural solution is due to the need to optimize the cost of resources and save them when using the model in the release environment using the developed web service. The software implementation

of the proposed classification technology has been implemented. The software application is implemented in the Python programming language using the TensorFlow machine learning library and the Tensorflow Extended integrated platform. The server part is implemented using the aiohttp framework. An experimental study of the developed text classifier was carried out using the News Category Dataset, which is a multilingual heading of text news. The use of the proposed classification technology is characterized by a slight deterioration in quality indicators when changing the language, which allows developing multilingual models without loss of performance when changing the language of the input data. The test results confirm the effectiveness of the above approach.

VECTORIZATION OF MULTILINGUAL TEXTS, XLM-RoBerta MODEL, NEURAL NETWORK CLASSIFIER OF MULTILINGUAL TEXTS, LSTM NETWORK, QUERY PROCESSING SERVER

Вступ

В області автоматичної обробки електронних текстів набув розвитку напрямок інтелектуального аналізу даних Text Mining, в рамках якого здійснюється пошук нових корисних знань з неструктурованої текстової інформації. Під неструктурованими текстовими даними розуміють електронні документи будь-якого типу: Web-сторінки, електронну пошту, нормативні документи тощо [1, 2]. Завдання організації ефективного доступу до неструктурованої тематичної інформації безпосередньо пов'язано із завданням класифікації електронних текстів, які надходять з ресурсів мережі Інтернет. Для його вирішення розроблено чимало ефективних методів, деякі з яких характеризуються якістю класифікації, що можна порівняти з результатами класифікації текстів, що виконується кваліфікованими експертами. Під час вирішення задачі автоматичної класифікації текстів використовується їх представлення у векторному вигляді. Існує чимало методів векторизації тексту (зокрема, Word2Vec, TF-IDF, Skip-gram, Continuous Bag of Words, MUSE тощо). Але всі ці методи не дозволяють вирішити задачу багатомовної векторизації текстів, адже їх використання передбачає необхідність попереднього визначення мови, якою написано текст, перед опрацюванням тексту з застосуванням відповідної мови моделі. Однак такий спосіб потребує написання значної кількості коду для підтримки цієї логіки, а також наявності моделі для векторизації та для обробки векторів під кожну мову, з якою потрібно працювати. Розглянемо докладніше модель MUSE (Universal Multilingual Sentence Encoding), що використовується для вирішення проблеми багатомовної класифікації текстів, підтримує 16 мов, та працює на рівні векторизації речення [3]. Основними компонентами цієї моделі є токенізатор тексту «Sentencepiece» та енкодер, який векторизує речення за допомогою архітектури «Transformer». Процес токенізації полягає в перетворенні речення на набір токенів (слів та окремих символів). Особливістю роботи токенізатора, що використовується в моделі MUSE, є можливість опрацьовувати слова, які не присутні в його словнику. Якщо слово не знайдено в словнику токенізатора, то він починає розділяти слово на частини, поки не отримає слова зі словника або окремі букви. Після цього токени заміняються на індекси елементів словнику моделі, що відповідають цим токенам, та подаються на вхід енкодеру, який застосовує двонаправлений Self-attention механізм для обчислення контексту використання токена у реченні. Після цього контекст використання кожного токену у реченні підсумовується та усереднюється для подальшої векторної репрезентації речення. Зазначимо, що розмір вихідного вектору буде обмежений (кількість токенів які можна подати на вхід до блоку «Transformer» дорівнює 512). Таким чином отримується контекстуально зважена векторна репрезентація тексту, що дозволяє опрацювувати тексти на одній з шістнадцяти мов. В той же час, побудувати систему, яка буде одночасно підтримувати, наприклад, як англійську так і українську мову, за допомогою даного методу неможливо. Більш прийнятними для багатомовної реалізації класифікаторів тексту є моделі mBERT та XLM-RoBerta, що передбачають використання замаскованих токенів [4]. На вхід цих моделей подається текст, в якому деякі токени замінені на спеціальний токен-маску, після чого необхідно, користуючись контекстом незамаскованих токенів, визначити які токени знаходяться під маскою. Для багатомовних реалізацій таких моделей можна використовувати навчальний набір даних, де присутні фрагменти текстів кількома мовами. Втім архітектурно багатомовні варіанти моделей mBERT та XLM-RoBerta не відрізняються від їх одномовної реалізації. Для багатомовної моделі необхідно мати датасет, який враховує велику кількість мов. При цьому виникають суттєві проблеми з мовами, що недостатньо представлені в навчальному корпусі. Акуальною є проблема комбінованого використання можливостей моделей типу XLM-RoBerta та рекурентних нейронних мереж для побудови ефективних багатомовних класифікаторів текстів.

У даній роботі пропонується технологія побудови таких класифікаторів, яка основана на нейромережевій обробці векторного подання текстів, згенерованого за допомогою XLM-RoBERTa.

1. Технологія векторизації багатомовних текстів на основі моделі XLM-RoBerta

Для попередньої векторизації текстів в класифікаторах традиційно використовується метод, запропонований в [5], сутність якого полягає в тому що вже для навчених векторних представлень тексту з FastText здійснюється проекція всіх мов у векторний

простір англійської мови (наприклад, вектор одного слова з української мови відображається в аналог цього слова в англійській мові).

Матриця-проектор при цьому обирається так, щоб мінімізувати відстань між словом x_i і його еквівалентною проекцією y_i. Тобто, якщо словник складається з пар (x_i, y_i), то матриця-проектор M формується таким чином:

$$M = argmin_w \sum_i \left\| x_i - Wy_i \right\|^2 , \quad (1)$$

де $\|^2$ – позначення норми.

Крім того, має бути обмежена матриця W проектора, щоб збереглися вихідні відстані між векторами вбудовування слів.

Недоліком такого підходу є те, що на виході отримується векторне подання речення без врахування контексту використання слів та без урахування взаємодії слів між собою в реченні.

Розглянемо переваги використання для векторизації текстів рекуррентної нейронної мережі на основі трансформеру моделі XLM-RoBerta, що підтримує 100 мов. На відміну від традиційних моделей ця модель має вищу точність на великому спектрі задач обробки природно мовних текстів (NLP – Natural-language processing).

Слід відзначити, що моделі mBERT та MUSE також здійснюють векторизацію на рівні речення, як і XLM-RoBerta, але XLM-RoBerta навчена на потужному датасеті CommonCrawl, що складається з 7.8 терабайт текстів англійською мовою та має спільний словник для усіх мов цієї моделі. Це дозволяє збільшити можливість використання проекції у векторному просторі одного й того ж слова різними мовами, що, в свою чергу, дає можливість навчати нейронні мережі векторизації та класифікації однією мовою, а підтримувати всі 100 мов моделі XLM-RoBerta.

Однією з особливостей трансформеру моделі XLM-RoBerta є використання в механізмі внутрішньої уваги (self-attention) технології multi-head attention (багатоголової уваги) [6]. Ця технологія дозволяє замість використання однієї голови уваги з 512 розмірними векторами ключів, запитів та значень лінійно проеціювати ключі, значення та запити h разів з різними вивченими проекціями на відповідні вектори. Якщо значення h дорівнює 8, то кожна голова уваги має розмірність вихідної послідовності 64, що в сумі дозволяє обробляти послідовності, що не перевищують розмірність в 512 токенів. На кожну з цих проекцій ключів, запитів та значень паралельно застосовується функція внутрішньої уваги та отримуються відповідні вектори. Після цього вектори об'єднуються та знову проеціюються, в результаті чого формуються остаточні значення (рис. 1).

Багатоголова увага дозволяє моделі XLM-RoBerta здійснювати доступ паралельно до різних частин вхідного речення. Формула для обчислення багатоголової уваги має наступний вигляд:

$$MultiHead(Q,K,V) = Concat(head_1, \ldots, head_h)W^O \ , (2)$$

де $head_1 = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right)$; W_i^Q – матриця ваг для матриці запитів i; W_i^K – матриця ваг для матриці ключів i; W_i^V – матриця ваг для матриці значень i.

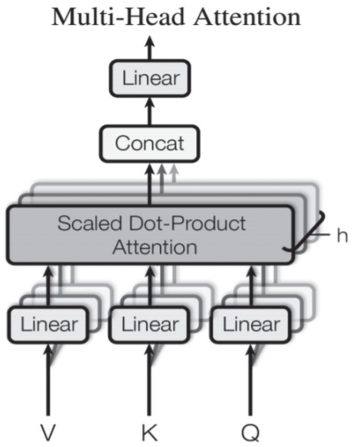

Рис. 1. Ілюстрація механізму багатоголової уваги моделі XLM-RoBerta

Результати порівняння складності операцій в різних шарах нейронної мережі з механізмом внутрішньої уваги, що передбачається далі використовувати у багатомовному класифікаторі, наведено в табл. 1.

Таблиця 1

Результати порівняння складності операцій в різних шарах нейронної мережі

Тип шару	Складність шару	Послідовність операцій	Максимальна довжина
Внутрішня увага	$O(n^2 * d)$	$O(1)$	O(1)
Рекурентний	$O(n * d^2)$	$O(n)$	O(n)
Згортковий	$O(k * n * d^2)$	$O(1)$	$O(log_k(n))$
Обмежена внутрішня увага	$O(r * n * d)$	$O(1)$	$O(n/r)$

У табл. 1 прийнято такі позначення: n – кількість елементів у вхідній послідовності; d – розмір простору репрезентації вхідних даних (розмірність словнику моделі); k – розмір вікна операції згортки, r – кількість сусідів в шарі обмеженої складності внутрішньої уваги (Self-Attention restricted).

Результати порівняння показують, що шар внутрішньої уваги з'єднує усі позиції з постійною кількістю послідовно виконаних операцій, в той час рекурентна мережа потребує $O(n)$ послідовних операцій. Шар власної уваги працює швидше, ніж рекурентна мережа, коли вхідна послідовність n менша ніж розмір простору репрезентації d, що є досить поширеною ситуацією.

Якщо замість рекурентних мереж використовувати згорткову мережу з розміром вікна $k < n$, то виникає проблема з'єднання вхідних та вихідних позицій, адже декодер буде втрачати частину інформації, що надійшла з енкодеру. Щоб вирішити цю проблему, потрібно використання $O(n/k)$ згорткових мереж. При цьому необхідним є врахування позиції кожного слова у вхідних даних моделі. Позиційне кодування додається до вхідних ембедінгів слів у нижніх частинах стеків енкодерів та декодерів. Так як позиційні кодування мають розмірність, яка співпадає з розмірністю ембедінгу, то позиційне кодування додають до ембедінгів. Доцільним є використання синусоїдальної функції позиційного кодування:

$$PE_{(pos,2i)} = sin\left(pos / 10000^{2i/d_{model}}\right), \qquad (3)$$

де pos – позиція елементу у вхідній послідовності; i – розмірність.

Кожний вимір позиційного кодування відповідає синусоїді. Довжини хвиль утворюють геометричну прогресію від 2π до $10000 \cdot 2\pi$. Така функція дозволяє моделі обробляти послідовності, що мають більшу довжину відносно послідовностей в тренувальних даних, так як для будь якого фіксованого зсуву k функція $P_{\text{pos}} + k$ може бути представлена як лінійна функція від P_{pos}.

В трансформері моделі XLM-RoBerta, кожна компонента енкодера має залишковий зв'язок навколо себе, за котрим слідує етап нормалізації шару (layer-normalization step). Візуалізацію операції нормалізації шару внутрішньої уваги наведено на рис. 2 [7].

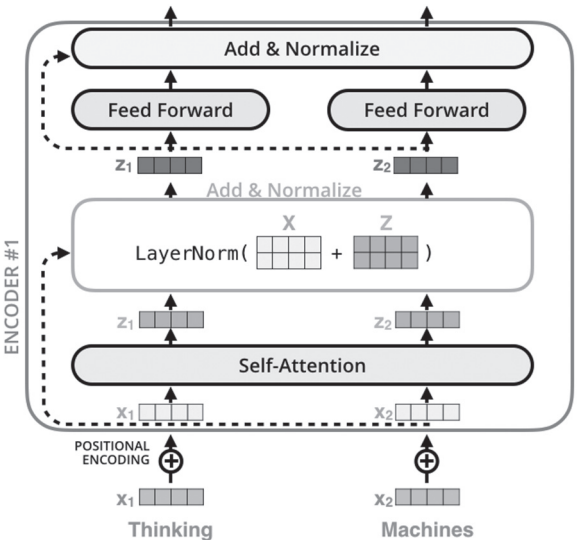

Рис. 2. Візуалізація операції нормалізації шару внутрішньої уваги моделі XLM-RoBerta

На етапі регуляризації в трансформері моделі XLM-RoBerta до вихідних даних кожного елементу шару енкодера та декодера застосовується техніка дропаут (Dropout). Згідно з цією технікою під час навчання на різних ітераціях відбувається вилучення певного проценту нейронів (як у прихованих шарах так і у видимих). Внаслідок цього найбільш натреновані нейрони отримують найбільші ваги. Крім того, дропаут використовується для суми ембедінгів з позиційним кодуванням (як в енкодері так і в декодері) з коефіцієнтом 0.1, тобто 10 відсотків цієї суми відкидається. Блок декодеру трансформеру моделі XLM-RoBerta подібний за архітектурою з блоком енкодеру, але має додатковий шар – encoder-decoder attention. Внутрішньо робота декодеру не відрізняється від роботи енкодеру, за винятком нюансів маскування усіх токенів, що йдуть після поточного токена [8].

Після того, як вхідна послідовність пройде через усі блоки енкодерів, з вихідних даних останнього енкодеру формуються матриці ключів та значень, які використовуються шаром encoder-decoder attention. При цьому encoder-decoder attention працює як механізм багатоголової уваги, за винятком того, що він створює матрицю запиту з попереднього шару, який знаходиться нижче, та отримує матриці значень та ключів з вихідних даних стеку енкодерів.

Внутрішня увага у шарі декодеру може фокусуватися лише на попередніх позиціях відносно поточної. Це потрібно для збереження можливості авторегресії та досягається за допомогою маскування всіх наступних токенів відносно поточного. Значення для замаскованих токенів встановлюються перед етапом софтмакс в обчисленні внутрішньої уваги.

Після того як декодер завершує свою роботу, його вихід подається до повнозв'язної мережі, яка формує репрезентацію вихідних даних моделей. Тобто її вихід буде складатися з вектору, який матиме таку ж саму розмірність, як і словник моделі, а кожний елемент вектору репрезентує кожне слово, що є в словнику відносно вхідних даних. Далі цей вектор подається на функцію софтмакс, яка перетворює репрезентацію кожного слова на його вірогідність, після чого формується вектор з вірогідністю кожного слова. Відібравши індекси елементів з найбільшою вірогідністю, отримаємо індекси слів у словнику, з яких і складається вихідне речення.

2. Класифікатор багатомовних текстів на основі моделі XLM-RoBERTa та мережі LSTM

Для вирішення задачі багатомовної класифікації тексту було розроблено нейромережевий класифікатор, що обробляє векторне подання, згенероване за допомогою моделі XLM-RoBerta. Схему взаємодії цього класифікатору з моделлю XLM-RoBerta наведено на рис. 3.

XLM-RoBerta в даному класифікаторі знаходиться зовні архітектури та є окремою моделлю, яка не приймає участь в навчанні. Запропоноване архітектурне рішення обумовлено необхідністю оптимізації витрат ресурсів та їх економії під час використання моделі у релізному середовищі за допомогою розробленого

веб- сервісу. Після того, як сервер отримує запит від клієнта на обробку тексту, він посилає запит до Tensorflow Serving Server, вже підготовлений для моделі XLM-RoBerta вхідний текст та отримує тензор як результат виконання запиту, після чого подає цей тензор знову на сервер з моделями, але вже в якості вхідних даних для моделі класифікації. Після обробки класифікатором даного тензору отримується відповідь у вигляді розподілу вірогідностей кожного класу, що надалі трансформується в текст та надається клієнту, як остаточна відповідь [9].

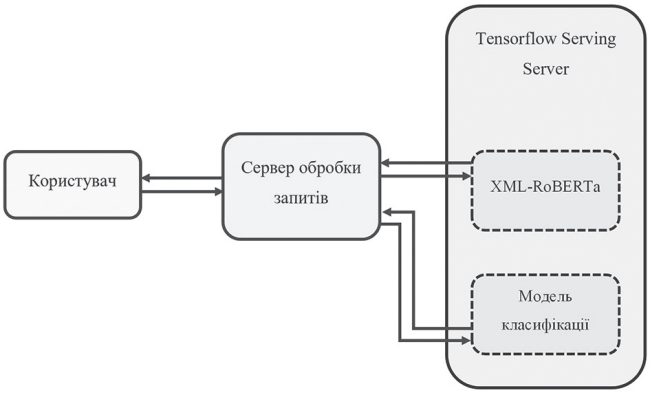

Рис. 3. Схема взаємодії класифікатору з моделлю XLM-RoBerta

Даний підхід дозволяє інваріантно до кількості запитів, що надходять для обробки одного тексту, оптимізувати ресурси та середній час обробки одного запиту клієнта. Завдяки тому, що моделі знаходяться на окремому сервері, при клонуванні серверу обробки запитів потрібно менше ресурсів, тому що моделі не клонуються разом з ним. Клонування може бути потрібно, якщо працює балансер навантаження та підіймає додаткові віртуальні машини з веб-сервісом при зростанні навантаження.

Модель класифікатору складається з 4 шарів. Перший шар (InputLayer) є стандартним шаром, що отримує дані та передає їх далі у модель. Шар LSTM – основний шар моделі, що аналізує вектор, який надходить від XLM-RoBerta. Шари Dense – останні два шари мережі, перший з яких виконує функцію зниження розмірності та має функцію активації ReLU, а другий є софтмакс шаром, який відповідає за класифікацію тексту.

Схематичне зображення моделі нейромережевого класифікатору наведено на рис. 4.

Встановлення платформи Tensorflow Serving на сервер та її компілювання з підтримкою інструкцій CPU (таких як AVX, SSE та інші) дозволяють отримати сервер хостингу моделей, оптимізований конкретно під характеристики серверу, що використовується. Використання платформою Tensorflow Serving заморожених моделей, які потребують незначних ресурсів оперативної пам'яті, сприяє підвищенню продуктивності роботи серверу та значній економії ресурсів.

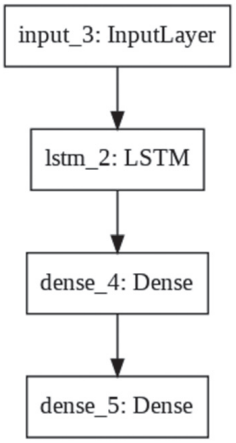

Рис. 4. Узагальнена архітектура пропонованого класифікатору

Розглянемо детальніше особливості архітектури рекуррентної мережі LSTM (Long Short Term Memory), що реалізає завдання класифікації [10]. Ця мережа є модифікацією архітектури рекурентних нейронних мереж RNN, що були розроблені для вирішення задач, в яких вхідна послідовність представлена у вигляді ряду. Але якщо вхідні послідовності мають велику довжину, то архітектура RNN працює незадовільно через проблему затухаючого градієнту: градієнт під час проходження від кінця мережі до її початку становиться занадто малим, внаслідок чого не змінюються ваги шарів в мережі. Вирішенню даної проблеми сприяє архітектура LSTM (рис. 5).

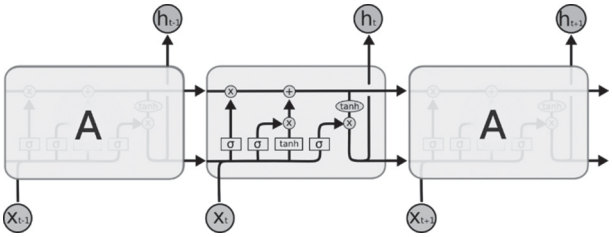

Рис. 5. Узагальнена архітектура мережі LSTM

В LSTM використовуються у кожній комірці вхідний фільтр, вихідний фільтр та фільтр-відсіювач.

Фільтр-відсіювач необхідний для відсіювання нерелевантних даних з попередньої комірки, що допомагає використовувати лише релевантну для нейронної мережі інформацію в поточний час. Вихід даного фільтра формується таким чином:

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f, \quad (4)$$

де W_f – ваги для шару фільтру; h_{t-1} – вихід попереднього шару; x – вхідні дані; b_f – зсув

Фільтр вхідних даних потрібен для того, щоб визначати, яку інформацію з вхідної послідовності потрібно зберегти у стані даної комірки. Вихід даного фільтра формується таким чином:

$$i_t = \sigma\left(W_i * [h_{t-1}, x] + b_i\right), \quad (5)$$

де W_i – ваги для шару фільтру; h_{t-1} – вихід попереднього шару; x – вхідні дані; b_i – зсув.

Фільтр вихідних даних потрібен для коригування інформації, яку ми хочемо передати у наступну комірку. Вихід даного фільтра формується таким чином:

$$o_t = \sigma\left(W_o * \left[h_{t-1}, x\right] + b_o\right), \quad (6)$$

де W_o – ваги для шару фільтру; h_{t-1} – вихід попереднього шару; x – вхідні дані; b_o – зсув.

LSTM здатна обробляти великі послідовності текстових даних без виникнення проблем з вагами мережі, які не змінюються.

Згідно зі схемою взаємодії кінцевого клієнту з додатком, що наведена на рис. 3, необхідно розробити модуль для звернення до серверу з моделями Tensorflow Serving та методи обробки запиту від клієнту. Платформа Tensorflow Serving характеризується наявністю високорівневого API Keras, що дозволяє зменшити кількість часу, необхідного для створення прототипів моделей, та зменшує складність коду, необхідного для створення програмного продукту.

3. Експериментальні дослідження

Розглянемо результати експериментальних досліджень розробленого нейромережевого класифікатору багатомовних текстів, що обробляє векторне подання, згенероване за допомогою моделі XLM-RoBerta.

Для тренування класифікатору використовувався датасет, що був завантажений з відкритої платформи kaggle.com, призначеної для здійснення машинного навчання, яка має відносно великий репозиторій публічних датасетів для різних завдань.

Датасет, що було обрано для експериментального дослідження розробленого класифікатору текстів, має назву News Category Dataset та представляє собою багатомовні заголовки текстових новин [11]. Вибір лише заголовків для тестування обумовлюється тим, що архітектура трансформеру моделі XLM-RoBerta має обмеження на довжину вхідної послідовності в 512 токенів, а текст новин зазвичай перевищує задану максимальну довжину послідовності.

Структура датасету відповідає формату JSON та містить наступні реквізити: Category (визначає категорію, до якої відноситься текст); Headline (заголовок статті); authors (автори статті); link (посилання на статтю), short_description (короткий описа теми статті), date (дата публікації). Оригінальний датасет містить 40 різноманітних категорій, зібраних з HuffPost.

Для виконання експериментального моделювання та навчання моделі для подальшого використання в системі датасет було оброблено та скорочено до 5 категорій, кожна з яких містить 500 записів. Таким чином датасет було збалансовано.

Фрагмент обробленого фінального датасету наведено на рис. 6.

text	label
Dean Foods Ex-Chairman, Pro Gambler Charged Wi...	SPORTS
Royal Rumble Winner Rumors! Will Kenny Omega M...	SPORTS
LeBron James Flopped So Hard That 'LMAO LeBron...	SPORTS
We Need These Old-School Baseball Cards Featur...	SPORTS
D.C. United 2 - 1 Philadelphia Union...but the...	SPORTS
...	...
Monday Matters: A Walk Home To Remember, An Un...	GOOD NEWS
Refuge in Americal remember his brown leather ...	GOOD NEWS
Dog Discovers The Many Wonders Of A Hula HoopF...	GOOD NEWS
Teacher And His Students Recreate 'Uptown Funk...	GOOD NEWS
Man Has Adorable Conversation With Dozens Of B...	GOOD NEWS

Рис. 6. Фрагмент обробленого датасету

Поле text представляє собою сумму компонент headline та short_description, а поле label – категорію, до якої відноситься даний текст. Усі перетворення були виконані за допомогою можливостей, що надає бібліотека Pandas. Даний датасет представляє собою навчальну вибірку для дослідження розробленого класифікатору.

Для валідаційного датасету було відібрано 10% від навчального та переведено за допомогою translate-api на іспанську мову.

Для проведення експериментів з навчання моделей було використано середовище Google Collaboratory, що надає 8 гб GPU TESLA K80. Під час моделювання досліджувалися моделі XLM-RoBerta base та mBERT base.

В моделях класифікації використовувалася векторна репрезентація тексту, згенерована за допомогою трансформерів mBERT та XLM-RoBerta, що надається відкритим хабом моделей від huggingface. Завантаження моделі в середовище здійснювалося наступним чином:

```
tokenizer=BertTokenizer.from_pretrained('bert-base-multilingual-cased')
transformer_mode=TFBertModel.from_pretrained('bert-base-multilingual-cased').
```

Так як класифікатор використовує вже готову векторну репрезентацію, наступним кроком була генерація ембедінгів, що здійснювалася згідно з наступним кодом:

```
def generate_embeddings(texts, model, tokenizer):
list_of_embeddings = []
for text in tqdm(texts):
tokenizer_output = [to_tokens(text, tokenizer)]
input_ids_train = np.array(select_field(tokenizer_output, 'input_ids'))
attention_masks_train = np.array(select_field(tokenizer_output, 'attention_mask'))
inputs = [input_ids_train,
```

```
attention_masks_train]
embeddings = model.predict(inputs)
# print(np.shape(embeddings[0]))
list_of_embeddings.append(embeddings[0][0])
return np.array(list_of_embeddings).
```

Після підготовки даних здійснювалися генерація моделі класифікатору та запуск навчання згідно з наступним кодом:

```
def create_model():
embeddings = tf.keras.layers.Input((125,768), dtype=
tf.float32)
x=tf.keras.layers.LSTM(units=512,dropout=0.2,recur
rent_dropout=0.2)(embeddings)
x = tf.keras.layers.Dense(units=256, activation='relu')
(x)
outputs = tf.keras.layers.Dense(5, activation=
'softmax')(x)
model = tf.keras.models.Model(inputs=embeddings,
outputs=outputs)
return model
model = create_model()
model.summary()
model.compile(loss='categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])
history = model.fit(embeddings, y_train, batch_size=
12, epochs=10).
```

В якості функції похибки було обрано стандартну для завдань багатокласової класифікації категорійну кросс-ентропію.

Після завершення навчання класифікатору з трансформером mBERT на тренувальному наборі даних, яке відбувалось 10 епох, були отримали наступні результати: метрика точності (accuracy) дорівнювала 0.9711, а значення функції похибки дорівнювало 0.0767. Зміну значень функції похибки та accuracy для моделі з трансформером mBERT base наведено на рис. 7 та рис. 8.

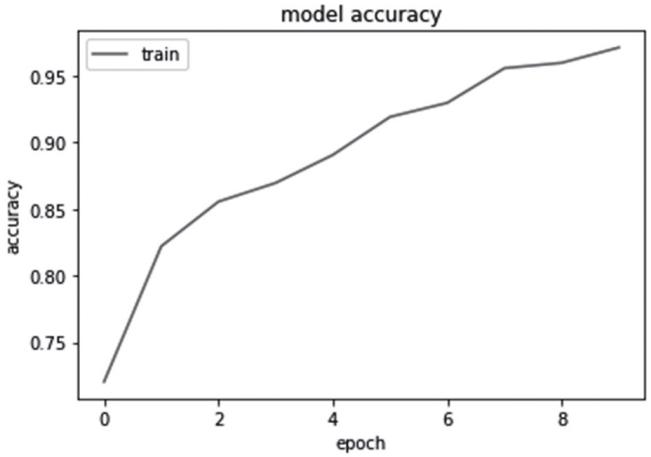

Рис. 7. Зміна метрики accuracy для моделі з трансформером mBERT base

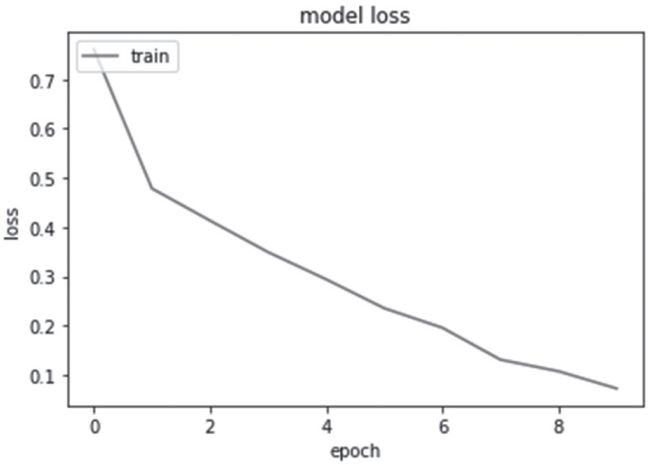

Рис. 8. Зміна функції похибки для моделі з трансформером mBERT base

Наступним кроком була генерація векторного подання для валідаційного набору даних та підрахунок функції похибки та метрики accuracy для валідаційного набору даних згідно з таким кодом:

```
eval_history=model.evaluate(eval_embeddings, y_test,
batch_size=2).
```

Значення функції похибки після закінчення підрахунку дорівнювало 1.6751, а метрики accuracy 0.7400. З цього можна зробити наступний висновок: моделі на базі mBERT показують прийнятний результат для англійської мови, але мають спад точності в разі зміни мови тексту.

Після завершення навчання класифікатору з трансформером XLM-RoBerta на тренувальному наборі даних, яке відбувалось 10 епох, були отримані наступні результати: метрика точності (accuracy) дорівнювала 0.8732, а значення функції похибки дорівнювало 0.4020. Зміну значень функції похибки та accuracy для моделі з трансформером XLM-RoBerta наведено на рис. 9 та рис. 10.

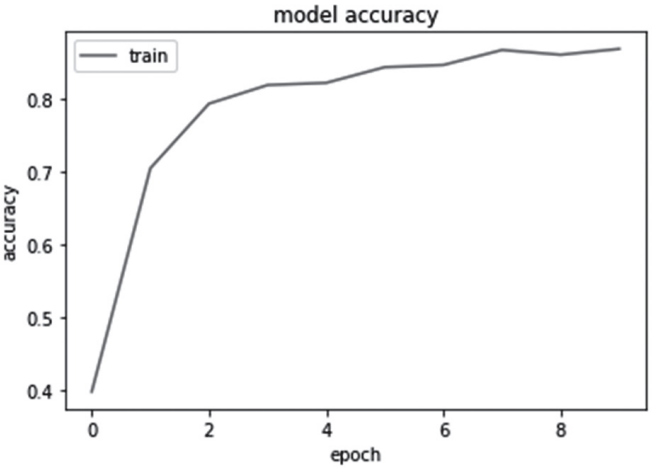

Рис. 9. Зміна метрики accuracy для моделі з трансформером XLM-RoBerta

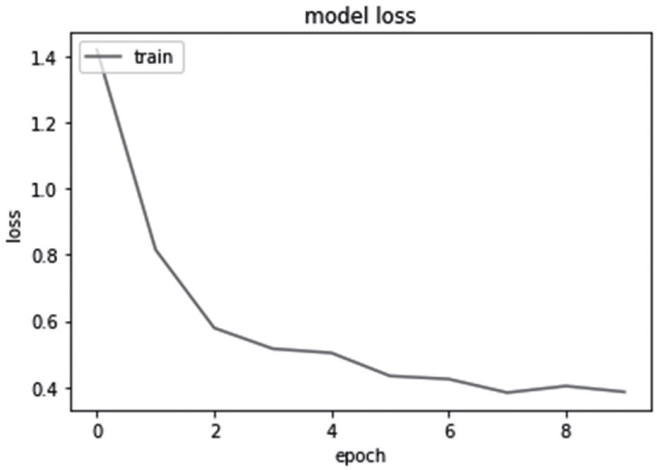

Рис. 10. Зміна функції похибки для моделі з трансформером XLM-RoBerta

Після підрахунків відповідних показників для валідаційного набору даних метрика accuracy дорівнювала 0.81, а значення функції похибки дорівнювало 0.5030. З цього можна зробити висновок, що XLM-RoBerta на відміну від mBERT має незначне погіршення метрик під час зміни мови, що дозволяє розробляти багатомовні моделі без втрати продуктивності моделі при зміні мови вхідних даних.

За результатами експериментального моделювання було обрано модель XLM-RoBerta для подальшої розробки програмного додатку, що реалізує процедури звернення користувача до серверу класифікатору та обробку відповідних запитів.

4. Розробка програмного додатку

Згідно зі схемою взаємодії запропонованого багатомовного класифікатору з моделлю XLM-RoBerta (рис. 3) були розроблені програмні модулі для звернення користувача до серверу з моделями Tensorflow Serving та методи обробки відповідних запитів. Для модуля з посиланням запитів до Tensorflow Serving та їх подальшою обробки були розроблені класи CustomEmbGenerator та TextClassifier.

CustomEmbGenerator відповідає за звернення до моделі XLM-RoBerta для генерації ембедингів та реалізує методи для токенізації тексту, його форматування для вхідного формату моделі та посилання запиту до моделі через проток grpc. Відправка запитів до моделі XLM-RoBerta відбувається згідно з наступним кодом:

```
def _send_request(self, body: List[‘np.ndarray’]) -> ‘np.ndarray’:
    Send gRPC request to model.
    :param body: Request body.
    :return: Embeddings.
    ``````
    logger = logging.getLogger(‘generator.serving’)
    serving = ServingGRPCClient(self.address, None)
    try:
    logger.info(‘Sending gRPC request to XLM-R model ...’)
    result = serving.predict(name=»xlm-roberta-base»,
    inputs={‘input_ids’: tf.compat.v1.make_tensor_proto (body[0]),
    ‘attention_mask’: tf.compat.v1.make_tensor_proto (body[1])})
    embeddings = np.squeeze(tf.make_ndarray(result. outputs[‘output_0’]))
    except Exception as exception: # pylint: disable=broad-except
    embeddings = []
    logger.exception(exception)
    return embeddings.
```

Як можна бачити, відправлення запиту до віддаленого серверу з використанням grpc дещо відрізняється від класичного протоколу HTTP. Наступним кроком була розробка класу TextClassifier для звернення та обробки результатів роботи моделі класифікації. Даний клас реалізує два методи: send_grpc_request та analyze. Перший метод має вихідний код, що є подібний до методу з класу CustomEmbGenerator. Як вхідні дані цей метод приймає результат, отриманий від моделі XLM-RoBerta та посилає його до моделі класифікації для подальшої обробки, а у відповідь отримує тензор розподілу вірогідностей кожного класу, згідно з яким натренована модель класифікації. Метод analyze визиває методи класу CustomEmbGenerator для отримання ембедінгів та передає їх у метод send_grpc_request для отримання відповіді від моделі класифікації, а потім перетворює вірогідності на класи, відповідно словнику. Метод працює згідно з наступним кодом:

```
def analyze(self, input_data: Any) -> Tuple:
    ``````
    Analyzing sentiments.
    If  input data is str that contains a text returns list that contain a np.array with sentiments score.
    :param input_data: string or list of strings.
    :return: ``np.ndarray`` with sentiments score if single text or
    ``List[np.ndarray]`` with sentiments score if multiple.
    ``````
    logger = logging.getLogger(‘classifier.analyze’)
    emb_gen = CustomEmbGenerator(max_pad_length=125, crop=False)
    # print(input_data)
    logger.debug(«IN ANALYZE»)
    if isinstance(input_data, list):
    count_of_sentences = []
    sentence_list = []
    all_sentence_list = []
    logger.debug(input_data)
    # print(input_data)
```
```

```
for i, text in enumerate(input_data):
sentences = sent_tokenizer(text)
sentence_list.append(sentences)
 count_of_sentences.append(len(sentences))
all_sentence_list.append(list(chain(*sentence_
list)) logger.debug(f'All sentences in input data array: {all_
sentence_list}')
logger.debug('Creating embeddings for texts')
embeddings = emb_gen.create_embedding(all_
sentence_list[0], batching=True, return_full=True)
prediction = self._send_grpc_request(embeddings)
predictions = emb_gen.chunk_data(prediction,
count_of_sentences)
labels = [self.__labels[p.argmax()] if float(p[p.
argmax()]) > 0 else 0 for p in predictions]
# probability for label class
probabilities = [p[p.argmax()] for p in predictions]
else:
sentence_list = sent_tokenizer(input_data)
logger.debug(f»Sentence list: {sentence_list}»)
if len(sentence_list) == 1:
embeddings = [emb_gen.create_embedding(sentence_
list[0], return_full=True)]
else:
embeddings = emb_gen.create_embedding(sentence_
list, return_full=True)
prediction = self._send_grpc_request(embeddings)
prediction = prediction.astype(float)
if prediction.ndim == 2:
labels = [self.__labels[p.argmax()] if float(p[p.
argmax()]) > 0 else 0 for p in prediction]
probabilities = [p[p.argmax()] for p in prediction]
else:
labels = [self.__labels[prediction.argmax()] if float
(prediction.argmax()) > 0.2 else 0]
# probability for label class
probabilities = [prediction[prediction.argmax()]]
return labels, probabilities.
```

Для обробки запиту від клієнту було розроблено хендлер (спеціальний метод, який дозволяє обробляти HTTP запити), що використовує aiohttp. Розроблений хендлер реалізує метод з назвою text_analayzer та відповідає на запит, що здійснено за адресою app_url/text_analyze. Для його коректної роботи необхідно, щоб у заголовку запита від клієнту містився ключ доступу. Дана опція дозволяє перевірити, чи має клієнт права доступу до системи.

Для перевірки можливостей додатку в якості клієнту буде використано програмний засіб Postman, що дозволяє тестувати веб-сервіси з можливостю посилання запитів до серверу.

Програмний додаток було реалізовано засобами високорівневої мови програмування Python за допомогою відкритої програмної бібліотеки для машинного навчання TensorFlow та комплексної платформи для розгортання виробничих конвеєрів машинного навчання Tensorflow Extended, серверну частину якого було реалізовано за допомогою фреймворку aiohttp, та за допомогою програмної бібліотеки мовою Python для обробки і аналізу даних Pandas.

Розроблений веб-сервіс здатен вирішувати завдання багатомовної класифікації з прийнятною точністю (зокрема, класифікувати описи новин на 100 мовах).

Перспективним продовженням проведених досліджень можуть бути: інтеграція багатомовного класифікатору з інструментами великих даних; розширення функціональних можливостей сервісу (розробка модулів для емоціонального аналізу тексту, пошуку в текстах подібного контенту тощо).

Висновки

Запропонована технологія сприяє вирішенню завдання багатомовної класифікації природномовних електронних текстів, що надходять з ресурсів мережі Інтернет або електронних бібліотек. Основою цієї технології є комбіноване використання можливостей моделей типу XLM-RoBerta та рекуррентних нейронних мереж типу LSTM для побудови ефективних багатомовних класифікаторів текстів.

Проведений аналіз особливостей векторного подання текстів свідчить, що найбільш прийнятною для багатомовної реалізації класифікаторів тексту є модель XLM-RoBerta, яка передбачає використання замаскованих токенів. Для багатомовних реалізацій цієї моделі можна використовувати навчальний набір даних, де присутні фрагменти текстів кількома мовами. XLM-RoBerta в даному класифікаторі знаходиться зовні архітектури та є окремою моделлю, яка не приймає участь в навчанні. Запропоноване архітектурне рішення обумовлено необхідністю оптимізації витрат ресурсів та їх економії під час використання моделі у релізному середовищі за допомогою розробленого веб-сервісу.

Модель класифікатору реалізовано з використанням чотирьохшарової нейронної мережі, основним компонентом якої є шар LSTM, що аналізує вектор, який надходить від трансформеру моделі XLM-RoBerta. Після того, як сервер класифікатору отримує запит від клієнта на обробку тексту, він посилає запит до Tensorflow Serving Server та отримує тензор як результат виконання запиту, після чого подає цей тензор знову на сервер з моделями, але вже в якості вхідних даних для моделі класифікації. Після обробки класифікатором даного тензору отримується відповідь у вигляді розподілу вірогідностей кожного класу, що надалі трансформується в текст та надається клієнту, як остаточна відповідь.

Експериментальне дослідження розробленого класифікатору текстів здійснено з використанням

News Category Dataset, що представляє собою багатомовні заголовки текстових новин.

Застосування запропонованої технології класифікації характеризується незначним погіршенням показників якості під час зміни мови, що дозволяє розробляти багатомовні моделі без втрати продуктивності моделі при зміні мови вхідних даних.

Можна вважати доцільним продовження досліджень щодо удосконалення розробленого багатомовного класифікатора з метою підвищення його функціональних можливостей та поліпшення якісних характеристик.

Список літератури:

[1] Чалая Л.Э. Метод двухэтапной классификации электронных текстов // Л.Э. Чалая, С.Г. Удовенко, Е.С. Кушвид // Біоніка интелекту. – 2016. – № 2 (87). – С.16 – 23.

[2] Інформаційні технології та системи: монографія / Удовенко С.Г. Розділ 8. Класифікація електронних науково-технічних текстів в інформаційно-пошукових системах // С.Г. Удовенко, Л.Е. Чала. – Х.: ФОП Бровін О.В., 2019. – С.108 – 123.

[3] Yang Y., Cer D., Amin A., Guo M., Law J., Constant N., Hernandez Abrego G., Yuan S., Tar C., Yun-Hsuan S., Strope B., Kurzweil R. Multilingual Universal Sentence Encoder for Semantic Retrieval, 9 Jul, 2019, URL: https://arxiv.org/pdf/1907.04307.pdf (Last accessed: 15.05.2021).

[4] Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 11 Oct, 2018, URL: https://arxiv.org/pdf/1810.04805.pdf (Last accessed: 15.05.2021).

[5] Stoyanov V., Necip F. A., Under the hood: Multilingual embeddings, January 24, 2018, URL: https://ai.facebook.com/blog/under-the-hood-multilingual-embeddings/ (Last accessed: 15.05.2021).

[6] Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L., Polosukhin I., Attention Is All You Need, 12 Jun, 2017, URL: https://arxiv.org/pdf/1706.03762.pdf (Last accessed: 15.05.2021).

[7] Alammar J. The Illustrated Transformer, 27 Jun, 2018, URL: http://jalammar.github.io/illustrated-transformer/ (Last accessed: 15.05.2021).

[8] Conneau A., Khandelwal K., Goyal N., Chaudhary V., Wenzek G., Guzmán F., Grave E., Ott M., Zettlemoyer L., Stoyanov V. Unsupervised Cross-lingual Representation Learning at Scale, 8 Apr, 2020, URL: https://arxiv.org/pdf/1911.02116.pdf (Last accessed: 15.05.2021).

[9] Tay Y., Dehghani M., Gupta J., Bahri D., Aribandi V., Qin Z., Metzler D. Are Pre-trained Convolutions Better than Pre-trained Transformers? 7 May, 2021, URL: https://arxiv.org/pdf/2105.03322.pdf (Last accessed: 15.05.2021).

[10] Olah C. LSTM - мережі довгої короткострокової пам'яті, 21 червень, 2017, URL: https://habr.com/ru/company/wunderfund/blog/331310/ (Last accessed: 15.05.2021).

[11] Misra R. News Category Dataset, 2 Dec, 2018, URL: https://www.kaggle.com/rmisra/news-category-dataset (Last accessed: 15.05.2021).

Надійшла до редколегії 17.09.2021