Київський національний університет імені Тараса Шевченка
Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка
Міністерство освіти і науки України

Кваліфікаційна наукова
праця на правах рукопису

СКУРЖАНСЬКИЙ ОЛЕКСАНДР ГРИГОРОВИЧ

УДК 004.8

ДИСЕРТАЦІЯ

РОЗРОБКА ПИСЬМОВОГО АСИСТЕНТА ЗА ДОПОМОГОЮ СУЧАСНИХ НЕЙРОМЕРЕЖЕВИХ ПІДХОДІВ ДЛЯ УМОВНОЇ ГЕНЕРАЦІЇ ТЕКСТУ

Спеціальність 122 Комп’ютерні науки
Галузь знань 12 Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело
_____ Скуржанський Олександр Григорович

Науковий керівник:
доктор фізико-математичних наук, професор
Марченко Олександр Олександрович

Київ - 2024

АНОТАЦІЯ

Скуржанський О.Г. Розробка письмового асистента за допомогою сучасних нейромережевих підходів для умовної генерації тексту. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 112 «Комп'ютерні науки». – Київський національний університет імені Тараса Шевченка, Київ, 2024.

Робота присвячена дослідженню побудові нейронних мереж для умовної генерації тексту у розробці письмового асистента.

У сучасному світі, де інформаційний потік набуває експоненційних обертів, виникає нагальна потреба в інноваційних технологіях для ефективної обробки та інтерпретації цих масивів даних. Вони включають у себе великі та гетерогенні набори інформації: тексти, відео, аудіо, зображення, табличні дані. Застосування традиційних методів для аналізу такої широкої та неоднорідної інформації стає неефективним, що зумовлює необхідність створення інновацій у галузі штучного інтелекту для ефективного вирішення цих викликів.

У цьому контексті, однією з найбільш перспективних та впливових галузей штучного інтелекту є обробка природної мови (NLP). Значення цього напрямку сьогодні виходить далеко за рамки академічних досліджень, проникаючи у більшість сфер сучасного цифрового життя людини: від щоденної комунікації до автоматизації складних бізнес-процесів. Поява великих мовних моделей таких як ChatGPT, які не дивлячись на те що працюють виключно з текстовими даними, здійснила прорив у наближенні можливостей штучного інтелекту до людського рівня. Завдяки їхнім вражаючим здібностям до генерації контекстно-релевантного тексту відчинились шляхи для більш комплексних застосувань.

Розвиток письмових асистентів у сфері NLP набуває особливої актуальності, адже ці системи відіграють важливу роль у підвищенні продуктивності та якості щоденної комунікації. У суспільстві, зануреному в безперервний потік інформації через численні платформи, здатність до швидкого і ясного вираження думок є

ключовою. Ринок письмових асистентів, з його значним економічним потенціалом, продемонстрував вражаюче зростання і за прогнозами тенденція до зростання буде зберігатися у наступні роки. Останні оцінки капіталізації напряму сягають десятків мільярдів доларів.

У світлі цих викликів та можливостей, розробка власного письмового асистента в рамках дисертації стає актуальним та перспективним напрямом у галузі NLP. З розширенням сфери цифрової комунікації виникає потреба в автоматизованих системах, які б не тільки пришвидшували, але й підвищували точність комунікаційних процесів. Такий асистент має потенціал стати невід'ємним інструментом в повсякденному житті, сприяючи ефективності професійної діяльності та особистісному розвитку.

Ця робота ставить за мету розробку універсального письмового асистента, що втілює в собі передові рішення у галузі нейронних мереж для вирішення комплексу задач з автоматичного виправлення граматичних та орфографічних помилок, перефразування тексту та його спрощення. Завдяки інтеграції передових технологій штучного інтелекту та машинного навчання, розроблений інструмент забезпечить більш ефективний та зручний метод обробки повідомлень для користувачів, що суттєво підвищить продуктивність та якість міжособистісної комунікації.

В роботі здійснено аналіз існуючих рішень для задач письмового асистента. Детально розглянуто принцип роботи провідних практик для вирішення вищезгаданих задача умовної генерації тексту. Проаналізовано спільні риси підходів з метою побудови універсального рішення для вирішення комбінації завдань. Основним недоліком існуючих методів є їх спеціалізація на конкретних задачах, що ускладнює перенесення набутих практик з однієї задачі на іншу. Наведено аналіз методів редагування тексту на основі використання різнотипних нейронних мереж для семантичного представлення елементів тексту на рівні речень. Зазначено доцільність архітектури трансформер для проектування моделі нейронної мережі водночас розглянути згорткові та рекурентні нейронні мережі як альтернативні рішення. На основі проведеного порівняльного аналізу методів

показано ефективність застосування нейронних мереж для вирішення поставлених задач письмового асистента.

Згідно зі сформованими метою та задачами дисертаційної роботи побудовано архітектуру та натреновано нейронну мережу здатну виконувати задачі письмового асистента з високою якістю та ефективністю. Нижче наведені **основні результати та наукова новизна**.

У результаті порівняльного аналізу існуючих провідних підходів до вирішення задач виправлення граматичних та орфографічних помилок, спрощення тексту та перефразування показано доцільність:

- використання контекстно-орієнтованої архітектури типу трансформер у конфігурації кодування-декодування (encoder-decoder) з великою кількістю шарів уваги, попередньо натренованих на універсальних задачах відновлення зашумленого тексту або генерації;
- впровадження багатоетапного тренування від найбільш частотних й найменш якісних даних до найбільш якісних; фільтрація тренувальних даних завдяки моделям винагороди.

Вперше запропоновано метод комбінованої оцінки якості роботи письмового асистента, який узагальнює ключові показники ефективності та адекватності текстових виправлень. Виконано дослідження існуючих провідних робіт на найбільш репрезентативних тестових вибірках для відповідних задач. Аналіз результатів свідчить що не всі метрики корелюють з людською оцінкою якості роботи моделей і нагальне застосування нормалізуючих коефіцієнтів для збалансованої комбінації задач. Крім того, завдяки аналізу літератури виявлено, що деякі задачі більш важливі для письмового асистента ніж інші. Так задача виправлення граматичних та орфографічних помилок є найважливішою згідно з очікуваннями користувачів.

Створено новий тренувальний набір даних з нуля для задач виправлення граматичних та орфографічних помилок, спрощення тексту та перефразування.

Застосовано сучасний підхід з використанням великих мовних моделей. За допомогою ретельно підібраних текстових запитів (prompts), згенеровано переписування передовою моделлю ChatGPT, використовуючи існуючі набори даних як вхідні речення; крім того утворені переписування автоматично оцінено з точки зору рівня переписування іншою великою мовною моделлю LLaMA-2 задля покращення якості фінального набору даних. Далі показано, що згенеровані дані суттєво покращують якість індивідуальних та універсальних моделей на тестових вибірках.

Проведено велику кількість експериментів з метою побудови оптимальної конфігурації для тренування нейронних мереж здатних виконувати окремі задачі письмового асистента:

1. натреновано індивідуальну baseline нейронну мережу для кожної із згаданих вище задач;
2. проведено порівняльний аналіз отриманих моделей відносно існуючих провідних підходів;
3. оцінено вплив параметрів тренування (темп навчання, кількість кроків, оптимізація тренування, розклад, розмір батча);
4. розглянуто різні типи нейронних мереж й архітектури та їхній вплив на фінальну якість моделі;
5. використано різні існуючі набори даний та їх комбінування під час тренування, багатоетапність підходу;
6. проаналізовано різні попередньо натреновані моделі та проведено власні спроби попереднього тренування;
7. оцінено вплив використання набору даних згенерованого великими мовними моделями.

Проведено експерименти з комбінування завдань письмового асистента для єдиної універсальної мета-моделі. Застосовано різноманітні підходи: тренування спеціалізованих токенів векторних представлень, поділ шарів нейронної мережі на спільні (перевикористані між задачами) та індивідуальні (для окремих задач),

використання архітектури адаптерів. Проаналізовано отримані універсальні системи на запропонованій комбінованій метриці. Вираховано ефективність підходів з точки зору кількості параметрів на задачу та одиницю якості.

Оптимізовано фінальну універсальну модель з метою пришвидшення її роботи, з уникненням суттєвої втрати якості на задачах письмового асистента та комбінованої метрики. Розглянуто різноманітні стратегії такі, як квантизація вагів у формати нижчої точності, розподілене виведення, ефективне управління пам'яттю, дистиляція знань. Також проведено аналіз програмних бібліотеки для пришвидшення генерації тексту як ONNX.

Ключові слова: обробка природної мови, умовна генерація тексту, нейронна мережа, згорткова нейронна мережа, рекурентна нейронна мережа, архітектура трасформер, виправлення граматичних та орфографічних помилок, спрощення тексту, перефразування, велика мовна модель, мета-модель, адаптер.

SUMMARY

Skurzhanskyi O.H. Development of efficient writing assistant using modern neural network approaches for conditional text generation. – Qualification scientific work on the rights of the manuscript.

The PhD thesis on competition of a scientific degree of the doctor of philosophy on a specialty 122 “Computer Science”. – Taras Shevchenko National University of Kyiv, Kyiv, 2024.

The work is devoted to the study of building neural networks for conditional text generation for the development of efficient writing assistant.

In today's world, where the flow of information is gaining exponential momentum, there is an urgent need for innovative technologies to effectively process and interpret these data sets. They include large and heterogeneous sets of information: texts, videos, audio, images, and tabular data. The use of traditional methods to analyze such broad and heterogeneous information is becoming inefficient, which necessitates the creation of innovations in the field of artificial intelligence to effectively address these challenges.

In this context, one of the most promising and influential areas of artificial intelligence is natural language processing (NLP). The importance of this field today goes far beyond academic research, penetrating most areas of modern digital life: from daily communication to automation of complex business processes. The emergence of large language models such as ChatGPT, which, despite working exclusively with textual data, has made a breakthrough in bringing artificial intelligence capabilities closer to the human level. Their impressive ability to generate contextually relevant text has opened the door to more complex applications.

The development of NLP-based writing assistants is particularly relevant, as these systems play an important role in improving the productivity and quality of daily communication. In a society immersed in a continuous flow of information through numerous platforms, the ability to express thoughts quickly and clearly is key. The market for writing assistants, with its significant economic potential, has demonstrated

impressive growth, which is projected to continue its upward trend in the coming years. Recent estimates of the market capitalization reach tens of billions of dollars.

Beyond the scope of textual data, the realm of natural language processing is rapidly expanding into multimodal AI, which integrates text, image, and audio processing to create more comprehensive and intuitive systems. This integration reflects the way humans communicate and process information, involving various sensory inputs. With the advancement of multimodal AI, systems can now interpret visual cues alongside textual information, enhancing understanding and interaction. This capability is particularly transformative for industries such as healthcare, where AI can analyze medical images in conjunction with clinical notes to aid in diagnostics, and in customer service, where chatbots can understand and respond to visual and verbal cues, creating a more human-like and satisfying user experience. The integration of these complex functionalities is not only a technological triumph but also a step towards more natural and efficient human-computer interaction, with applications that were previously unattainable in the field of artificial intelligence.

In light of these challenges and opportunities, developing your own writing assistant as part of your dissertation is becoming a relevant and promising area in the field of NLP. With the expansion of digital communication, there is a growing need for automated systems that would not only speed up but also increase the accuracy of communication processes. Such an assistant has the potential to become an indispensable tool in everyday life, contributing to professional efficiency and personal growth.

This work aims to develop a universal writing assistant that embodies advanced solutions in the field of neural networks to solve a set of tasks for automatically correcting grammatical and spelling errors, paraphrasing text, and simplifying it. By integrating advanced artificial intelligence and machine learning technologies, the tool aims to provide a more efficient and convenient way to improve user messages, which significantly increases the productivity and quality of interpersonal communication.

The paper analyzes existing solutions for writing assistant tasks. The principle of operation of the leading practices for solving the above-mentioned tasks of conditional text generation is considered in detail. The common features of the approaches with the

method of building a universal solution for solving a combination of tasks are analyzed. The main disadvantage of existing methods is their specialization in specific tasks, which makes it difficult to transfer the acquired practices from one task to another. The article analyzes the methods of text editing based on the use of different types of neural networks for the semantic representation of text elements at the sentence level. The expediency of the transformer architecture for designing a neural network model is noted, while convolutional and recurrent neural networks are considered as alternative solutions. Based on the comparative analysis of methods, the effectiveness of using neural networks to solve the tasks of a writing assistant is shown.

In accordance with the goal and objectives of the thesis, the architecture was built and the neural network was trained to perform the tasks of a writing assistant with high quality and efficiency. Below are **the main results and scientific novelty**.

As a result of a comparative analysis of existing leading approaches to solving the tasks of correcting grammatical and spelling errors, simplifying text and paraphrasing, the expediency of

- use of a context-aware transformer-type architecture in an encoder-decoder configuration with a large number of attention layers pre-trained on universal tasks of noisy text recovery or generation;
- implementation of multi-stage training from the most frequent and lowest quality data to the highest quality data; filtering of training data using reward models.

For the first time, a method of combined evaluation of the quality of a writing assistant's work is proposed, which summarizes the key indicators of efficiency and adequacy of text corrections. A study of existing leading works on the most representative test samples for the relevant tasks is carried out. The analysis of the results shows that not all metrics correlate with human evaluation of the quality of the models' performance and that it is necessary to apply normalization factors for a balanced combination of tasks. In addition, the literature analysis revealed that some tasks are more important for a writing

assistant than others. For example, the task of correcting grammatical and spelling errors is the most important according to users' expectations.

A new training dataset was created from scratch for the tasks of grammatical and spelling error correction, text simplification, and paraphrasing. A state-of-the-art approach using large-scale language models was applied. Using carefully selected textual prompts, rewrites are generated by the advanced ChatGPT model using existing datasets as input sentences; in addition, the generated rewrites are automatically evaluated in terms of rewrite quality by another large language model LLaMA-2 to improve the quality of the final dataset. Furthermore, it is shown that the generated data significantly improves the quality of individual and universal models on the test sets.

A large number of experiments have been conducted to build an optimal configuration for training neural networks capable of performing individual tasks of a writing assistant:

1. an individual baseline neural network was trained for each of the above tasks;
2. a comparative analysis of the obtained models in relation to the existing leading approaches;
3. the influence of training parameters (learning rate, number of steps, training optimization, schedule, and batch size) was evaluated;

4. different types of neural networks and architectures and their impact on the final quality of the model are considered;

5. different existing datasets and their combination during training, multi-stage approach are used;

6. analyzed various pre-trained models and conducted our own pre-training attempts;

7. the impact of using a dataset generated by large language models was evaluated.

Experiments were conducted to combine tasks of a writing assistant for a single universal meta-model. Various approaches were used: training specialized tokens of vector representations, dividing the layers of the neural network into common (reused between tasks) and individual (for individual tasks), and using the architecture of

adapters. The resulting universal systems are analyzed using the proposed combined metric. The effectiveness of the approaches in terms of the number of parameters per task and quality unit is calculated.

Finally, the resulting model is optimized to speed up its operation, avoiding a significant loss of quality on the tasks of a writing assistant and the combined metric. Various strategies are considered, such as quantizing weights into lower-precision formats, distributed inference, efficient memory management, and knowledge distillation. We also analyze software libraries for accelerating text generation such as ONNX.

Keywords: natural language processing, conditional text generation, neural network, convolutional neural network, recurrent neural network, transformer architecture, grammar and spelling correction, text simplification, paraphrasing, large language model, meta-model, adapter.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Статті у наукових фахових виданнях України:

1. Skurzhanskyi O., & Marchenko A. (2021). Review of neural approaches for conditional text generation. *Bulletin of Taras Shevchenko National University of Kyiv. Series: Physics and Mathematics*, *1*, 102–107. https://doi.org/10.17721/1812-5409.2021/1.13
2. Скуржанський О., Марченко О., & Анісімов, А. (2024). Спеціалізоване попереднє навчання нейромережевих моделей на синтетичних даних для покращення генерації перефразувань. *Кібернетика та системний аналіз, том 60*, 3–12. https://doi.org/10.34229/KCA2522-9664.24.2.1
3. Skurzhanskyi O., & Marchenko A. (2023). Neural approaches for writing assistant tasks. *Bulletin of Taras Shevchenko National University of Kyiv. Series: Physics and Mathematics*, *2*, 232–238. https://doi.org/10.17721/1812-5409.2023/2.40

Праці, які засвідчують апробацію матеріалів дисертації:

4. Omelianchuk, K., Atrasevych, V., Chernodub, A., & Skurzhanskyi, O. (2020). GECToR–Grammatical Error Correction: Tag, Not Rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, 163–170. https://doi.org/10.18653/v1/2020.bea-1.16
5. Omelianchuk, K., Raheja, V., & Skurzhanskyi, O. (2021). Text Simplification by Tagging. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications,* 11–25. https://doi.org/10.48550/arXiv.2103.05070
6. Skurzhanskyi, O., & Marchenko, O. (2022). Task-specific pre-training improves models for paraphrase generation. In *Proceedings of the 2022 6th International Conference on Natural Language Processing and Information Retrieval*, 44–48. https://doi.org/10.1145/3582768.3582791
7. Skurzhanskyi, O. (2023). Distillation of Large Language Models for Text Simplification. *Modeling, control and information technologies: Proceedings of VI International scientific and practical conference*, 230–231. https://doi.org/10.31713/MCIT.2023.071

Праці, які додатково відображають наукові результати дисертації:

8. Скуржанський О. (2021). Збагачення семантичного представлення тексту в нейронних мережах морфологічною інформацією для задачі виправлення граматичних та орфографічних помилок. *Systems and measures of artificial intelligence AIIS'2021*, 104–110.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AI	штучний інтелект (artificial intelligence)
BERT	двосторонні представлення трансформер-енкодеру (bidirectional encoder representations from transformers)
BiLSTM	двостороння довга короткочасна пам'ять (bidirectional long short-term memory)
CNN	згорткова нейронна мережа (convolutional neural network)
CPU	центральний процес (central processing unit)
GEC	задача виправлення граматичних помилок (grammatical error correction)
GPU	графічний процесор (graphics processing unit)
GRU	вентильний рекурентний вузол (gated recurrent unit)
LCS	найдовша спільна підпослідовність (longest common subsequence)
LLM	велика мовна модель (large language model)
LSTM	довга короткочасна пам'ять (long short-term memory)
NLP	обробка природної мови (natural language processing)
PCA	метод головних компонент (principal component analysis)
RLHF	навчання з підкріпленням на основі людського зворотного зв'язку (reinforcement learning from human feedback)
PPO	оптимізація проміжної політики (proximal policy optimization)
RNN	рекурентна нейронна мережа (recurrent neural network)
TPU	тензорний процесор (tensor processing unit)
UMAP	uniform manifold approximation and projection

ВСТУП

Обґрунтування вибору теми дослідження. В сучасному світі, де обсяги інформації зростають експоненційно, ми стикаємося з викликами, пов'язаними з обробкою та інтерпретацією цих масивів даних. Особливу увагу привертає категорія Big Data [1], яка включає великі та гетерогенні набори інформації: текстові дані, відео, аудіо та зображення. Застосування традиційних методів для аналізу такої широкої та неоднорідної інформації є неефективним, що зумовлює необхідність вдосконалення та розвитку технологій штучного інтелекту (AI) для ефективного вирішення цих викликів.

В цьому контексті, однією з найбільш перспективних та впливових галузей штучного інтелекту є обробка природної мови (NLP). Ця дисципліна має на меті зробити взаємодію між людьми та комп'ютерами такою ж природною та ефективною, як взаємодія між самими людьми. Важливість NLP виходить далеко за межі академічних досліджень і знаходить практичне застосування у різних сферах: від семантичного аналізу до суммаризації тексту.

Саме реліз великої мовної моделі (LLM) ChatGPT [2], здатної працювати на рівні тексту, спричинив величезний стрибок у задачі досягнення штучного інтелекту людського рівня. Завдяки здатності моделі генерувати текст, що відображає глибоке розуміння контексту та семантики, вона відкрила двері до більш комплексних застосувань: від спрощення повсякденної комунікації до автоматизації складних бізнес-процесів.

Зокрема, розробка письмових асистентів стає особливо актуальною в сфері NLP. Вони відіграють ключову роль у сучасній щоденній комунікації, оскільки вони сприяють зростанню ефективності та якості взаємодії між людьми. У світі, де ми знаходимося в неперервному інформаційному потоці через електронну пошту, соціальні мережі, форуми та інші платформи, здатність швидко та чітко висловлювати свої думки є пріоритетним. Ці системи виконують широкий спектр завдань, починаючи від граматичної та орфографічної корекції тексту до автоматичного генерування відповідей на запитання користувачів. Економічний потенціал цього ринку є величезним [3]: за оцінками, ринок програмного

забезпечення для письмових асистентів складав 818 мільйонів доларів в 2021 році і прогнозується зростання до 6.4 мільярдів доларів до 2030 року з темпом росту в 26,94% на рік.

Враховуючи вищеописані виклики і перспективи, актуальність дослідження розробки власного письмового асистента в контексті цієї дисертації стає ще більш очевидною. Перш за все, сфера письмової комунікації продовжує розширюватися, і з цим приходить підвищена потреба в автоматизованих системах, які можуть зробити цей процес не тільки швидшим, але й більш точним. Розробка власного письмового асистента може стати ключовим елементом в оптимізації таких процесів.

Використання однієї конкретної моделі для підтримки цього асистента надає можливість для глибокого спеціалізованого налаштування та оптимізації. Це не тільки забезпечує вищу ефективність, але і дозволяє нашій системі стати більш адаптивною до конкретних потреб користувача або організації. У рамках цієї дисертації, розробка власного письмового асистента на основі однієї моделі відкриває нові напрямки дослідження, демонструючи практичність і виправданість такого підходу.

Мета і завдання дослідження. Метою дисертаційного дослідження є побудова системи, здатної виконувати основні задачі письмового асистента: виправлення граматичних та орфографічних помилок, спрощення та перефразування тексту.

Для досягнення поставленої мети сформовано наступні завдання:

1. Проаналізувати існуючі передові практики для тренування нейронних мереж для кожної з вибраних задач.
2. Створити метод оцінки комбінованої якості моделі, здатної виконувати задачі умовної генерації тексту.
3. Побудувати набір даних, здатний враховувати особливості кожної задачі та останні здобутки у напрямку NLP.

4. Підібрати архітектуру машинного навчання та розробити ефективний алгоритм для тренування універсальної нейронної мережі для виконання задач письмового асистента.
5. Оптимізувати швидкість та якість роботи моделі на запропонованих даних для оцінки якості.

Об’єктом дослідження є системи комп'ютерної лінгвістики та механізми обробки природної мови — алгоритми та моделі, які можуть виконувати функції письмового асистента.

Предметом дослідження є процес проектування нейронних мереж різної архітектури для виконання задач письмового асистента; аналіз згаданих вище задач, наявних даних та оцінок виміру якості роботи.

Методи дослідження. У дисертаційній роботі застосовуються методи машинного навчання, багатошарові нейронні мережі різної архітектури. Для програмної реалізації методів та відповідних моделей використовується мова програмування Python 3.9.

Наукова новизна отриманих результатів. Протягом вирішення сформованих завдань дослідження вперше отримано наступні результати:

1. Удосконалено метод оцінки для якості роботи на задачах умовної генерації тексту за рахунок використання нормалізації метрик та оцінювання вкладів задач. Запропонований комбінований метод дозволяє оцінювати, як добре модель виконує задачі письмового асистенту.
2. Зібрано новий набір даних для тренування асистентів, використовуючи останні здобутки у напрямку великих мовних моделей. Набір даних включає декілька задач, проанотованих на різних доменах.

3. Вперше побудовано єдину універсальну нейронну мережу, задану виконувати набір задач письмового асистента. Оцінено якість робити моделі, проведено порівняння з іншими існуючими системами.
4. Оптимізовано швидкість та ефективність роботи моделі. Створено рішення, здатне працювати у режимі реального часу.

Структура та обсяг дисертаційної роботи. Дисертаційна робота складається зі вступу, трьох розділів, висновків, а також списку використаних джерел. Загальний обсяг роботи складає 143 сторінки, з яких основний зміст викладено на 141 сторінці. Дисертація містить 26 рисунків, 16 таблиць та список використаних джерел зі 63 найменувань.

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ НЕЙРОМЕРЕЖЕВИХ МЕТОДІВ ДЛЯ ЗАДАЧ УМОВНОЇ ГЕНЕРАЦІЇ ТЕКСТУ

Генерація тексту є чи не найбільш провідним напрямком обробки природної мови та штучного інтелекту, що найактивніше розвивається на сьогодні. Суттєвою частиною цієї області є умовна генерація, коли на вхід подається певна інформація (умова) і текст (вихід) генерується з її врахуванням.

Задачі письмового асистента розглянуті у даній роботі включають виправлення граматичних та орфографічних помилок, спрощення та перефразування тексту. Усі вони підпадають під визначення задач умовної генерації тексту, де переписаний (виправлений) текст генерується з умовою на вхідний текст користувача.

1.1 Постановка задачі умовної генерації тексту

Постановка задачі умовної генерації тексту може бути сформульована математично наступним чином.

Маємо вхідний текст $X = \{x_1, x_2, \dots, x_n\}$ довжиною n, де кожний x_i є словом або символом в тексті. Завдання полягає в генерації вихідного тексту $Y = \{y_1, y_2, \dots, y_m\}$ довжиною m, на основі даного вхідного тексту X. Існує невідома цільова залежність – відображення

$$y^*: X \to Y,$$

значення якої відомо лише на об’єктах скінченної навчальної вибірки

$$D = \{(x^1, y^1), (x^2, y^2), \dots, (x^l, y^l)\}.$$

Необхідно побудувати модель $f: X \to Y$ приймає на вхід X і параметри θ, а видає ймовірностний розподіл вихідного тексту Y:

$$f(Y|X;\ \theta) = \prod_{t=1}^{m} P(y_t|y_{<t}, X;\ \theta)$$

де $y_{<t}$ позначає послідовність словоформ, що передують y_t, і $P(y_t|y_{<t}, X;\ \theta)$ є умовною ймовірністю наступного слова y_t при заданому контексті $y_{<t}$ та вхідному тексту X.

Завдання полягає в оптимізації параметрів θ таким чином, щоб максимізувати логарифмічну правдоподібність вихідного тексту Y для даного вхідного тексту X вздовж всього набору даних:

$$\theta^* = \arg\max_{\theta} \sum_{(X,Y)\in D} \log f(Y|X;\ \theta)$$

де D є тренувальним набором даних.

Така постановка задачі може бути використана для різних завдань, включаючи, але не обмежуючись, синтезом тексту, генерацією відповідей в діалогових системах, спрощенням тексту, тощо.

1.2 Метрики оцінювання якості

Найбільш використовуваними метриками для задач умовної генерації тексту в залежності від задачі є BLEU, ROUGE, METEOR, SARI, FKGL, MaxMatch та TER. Усі вони використовують n-грамами для визначення різниці між згенерованим (передбаченим моделлю) та анотованим (написаним людиною, зазвичай у тестовому наборі даних) текстом.

BLEU (BiLingual Evaluation Understudy) [4] — це одна з найпоширеніших метрик для оцінки якості машинного перекладу та інших завдань генерації тексту. Вона була введена у 2002 році і з того часу стала стандартом в оцінці машинного перекладу. Основна ідея BLEU полягає в порівнянні n-грамів у вихідному тексті (згенерованому тексті) з n-грамами в розміченому тексті (оригінальний текст або "золотий стандарт") з точки зору точності співпадінь.

Метрика BLEU для одного тексту (речення) обчислюється наступним чином: обчислюється наступним чином:

$$BLEU\ =\ BP\ \cdot \exp(\sum_{n=1}^{N} w_n \log p_n\ )$$

де:

- BP — фактор скорочення (Brevity Penalty), який штрафує занадто короткі вихідні тексти.
- N — максимальна довжина n-грами.
- w_n — ваги для кожної n-грами (зазвичай, для 4-грами ваги встановлюються як $w_n = 0.25$ для $n = 1,2,3,4$).
- p_n — точність n-грами, обчислена як відношення кількості співпадаючих n-грамів в згенерованому та розміченому текстах до загальної кількості n-грамів в згенерованому тексті.

Фактор скорочення BP обчислюється так:

$$B = \begin{cases} exp(1 - \frac{|ref|}{|hyp|}, & \text{якщо } |ref| > |hyp| \\ 1, & \text{інакше} \end{cases}$$

де:

- $|hyp|$ — довжина згенерованого тексту.
- $|ref|$ — довжина розміченого тексту найближчого за довжиною до згенерованого тексту.

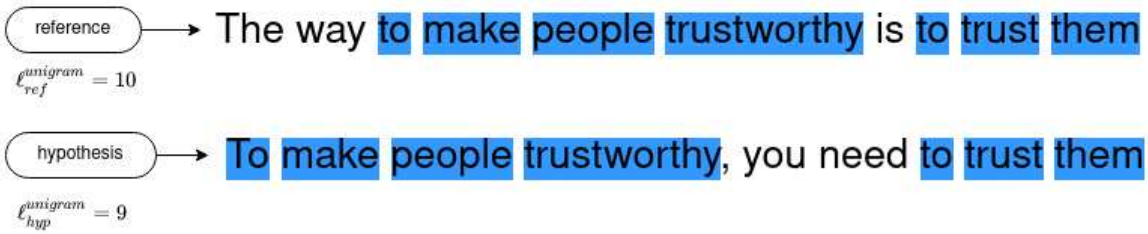


Рис. 1.1 – Приклад пошуку співпадіння n-грам

Метрика BLEU є однією з найбільш розповсюджених у використанні, оскільки вона дозволяє швидко і відносно об'єктивно оцінювати якість задач умовної генерації тексту, хоча її також критикують за недостатню увагу до семантики та структури речень.

Метрика ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [5] є іншою популярною метрикою для оцінки якості генерації тексту, зокрема, у задачах текстового стиснення та анотування. Вона фокусується на вимірюванні відтворюваності (Recall) між згенерованим текстом і анотованим (золотим стандартом).

Основні форми ROUGE включають у себе три компоненти для підрахунку метрики:

- **ROUGE-N**: Співставлення n-грамів у генерованому та посилальному текстах.
- **ROUGE-L**: Найдовша спільна підпослідовність (LCS) між двома текстами.
- **ROUGE-S**: Відстань між послідовними словами в тексті.

ROUGE для n-грам (ROUGE-N) можна виразити наступною формулою:

$$ROUGE \cdot N = \frac{\sum_{s \in Ref} \sum_{ngram \in s} Count_{match}(ngram)}{\sum_{s \in Ref} \sum_{ngram \in s} Count(ngram)}$$

де $Count_{match}(ngram)$ — кількість разів, коли n-грама зустрічається як у згенерованому, так і у анотованому тексті, а $Count(ngram)$ — загальна кількість зустрічей n-грами в анотованому тексті.

Метрика ROUGE-L використовує концепцію найдовшої спільної підпослідовності (LCS). Формула виглядає таким чином:

$$ROUGE \cdot L = \frac{LCS(X, Y)}{len(X)}$$

де $LCS(X, Y)$ — довжина найдовшої спільної підпослідовності між X (згенерованим текстом) та Y (анотованим текстом), а $len(X)$ — довжина згенерованого тексту.

Метрика ROUGE-S вимірює відношення між кількістю послідовних пар слів (skip-bigrams), що збігаються в анотованому і згенерованому текстах. Skip-bigram — це пара слів у тексті, між якими може бути будь-яка кількість інших слів.

$$ROUGE \cdot N = \frac{\sum_{s \in Ref} \sum_{ngram \in s} Count_{match}(skipgram)}{\sum_{s \in Ref} \sum_{ngram \in s} Count(skipgram)}$$

де $Count_{match}(skipgram)$ — кількість разів, коли skip-bigram збігається у згенерованому та анотованому текстах, а $Count(skipgram)$ — загальна кількість skip-bigram в анотованому тексті.

ROUGE-S важлива для випадків, коли порядок слів важливий, але де можливі невеликі варіації в порядку. Ця метрика дозволяє моделі бути гнучкою щодо порядку слів, але при цьому ще залишається здатність оцінювати якість генерації тексту.

Як BLEU, так і ROUGE мають свої переваги та недоліки. BLEU зазвичай використовується для оцінки точності перекладу, тоді як ROUGE зазвичай застосовується для оцінки відтворюваності і часто використовується таких задачах, як текстове стиснення. Обидві метрики можуть бути використані разом для отримання більш об'єктивної оцінки якості генерованого тексту.

METEOR (Metric for Evaluation of Translation with Explicit ORdering) [6] — це метрика, яка була розроблена з метою вдосконалення оцінки якості машинного перекладу, додаванням різноманітних факторів, які BLEU ігнорує. METEOR враховує не тільки точні збіги слів, але і синоніми, стеми та інші форми слова. Вона також уважно ставиться до порядку слів і може надати покарання за неявність порядку.

Формально, METEOR вираховується як середнє гармонійне Precision (P) та Recall (R):

$$METEOR = F_{mean} \cdot (1 - p) = \frac{(1 - \alpha) \cdot P \cdot R}{\alpha \cdot P + R} \cdot (1 - p)$$

де P та R розраховуються на рівні уніграм для згенерованого і анотованого речень відповідно, а p є штрафом за порушення порядку слів та інших факторів, таких як синонімія, стемінг тощо.

Для того, щоб обчислити цей штраф, уніграми групуються в найменші можливі групи, де група визначається як набір уніграм, які є суміжними в гіпотезі (згенерованому тексті) та в еталоні (анотованому). Чим довші суміжні відображення між кандидатом і еталоном, тим менша кількість частин. Гіпотеза,

ідентична еталону, надає лише одну групу. Штраф p обчислюється наступним чином:

$$p = 0.5 \cdot \left(\frac{c}{u_m}\right)^3$$

де c — це кількість груп, а u_m — кількість уніграм, які було відображено.

METEOR грає важливу роль у оцінці згенерованого тексту тому, що він намагається взяти до уваги багато аспектів якості переписування, які можуть бути пропущені при використанні менш складних метрик, таких як BLEU чи ROUGE. Ця метрика надає більш глибоке та метологічне розуміння якості згенерованого тексту.

Метрика SARI (System output Against Reference and Input) [7] розроблена спеціально для оцінки якості текстових перетворень, таких як спрощення тексту. Вона враховує як аспекти точності, так і повноти, та оцінює три ключові компоненти: видалені слова (delete), додані слова (add) і збережені слова (keep). Формально, SARI розраховується як середнє арифметичне цих трьох компонентів:

$$SARI = \frac{F_{add} + F_{keep} + P_{del}}{3}$$

де компоненти кожної операції $ope \in [add, keep, del]$ вираховуються на фіксованому рівні n-грам: від уніграм до k-gram. У оригінальній імплементації використовувалося $k = 4$.

$$f_{ope}(n) = \frac{2 \cdot p_{ope}(n) \cdot r_{ope}(n)}{p_{ope}(n) + r_{ope}(n)}$$

$$F_{ope} = \frac{1}{k} \sum_{n=[1,\ \ldots,\ k]} f_{ope}(n)$$

Особливість SARI полягає в тому, що вона оцінює не лише відношення між вихідним текстом та декількома референсними спрощеннями (як це зазвичай роблять інші метрики), але і порівнює вихідний текст із вхідним текстом. Це дозволяє краще зрозуміти, наскільки ефективно система зберегла зміст при спрощенні, та в якому обсязі вона здійснила зміни.

FKGL [8] – метрика призначена для оцінки читабельності тексту. Вона базується на обчисленні кількості слів, складів та речень в тексті, та оцінює, для якого класу освітньої системи цей текст буде найбільш зрозумілим.

Математично FKGL розраховується за наступною формулою:

$$FKGL = 0.39\frac{\#\text{ слів}}{\#\text{ речень}} + 11.8\frac{\#\text{ голосних}}{\#\text{ слів}} - 15.59$$

Метрику запропонував J. Peter Kincaid ще у 1975 для тесту на читабельність. Вона не використовує правильні (анотовані) речення, а тому має застосовуватися у комбінації з SARI.

MaxMatch (M2) scorer [9] використовується найчастіше для оцінки якості задач виправлення граматичних помилок (GEC), куди часто включають і орфографічні. Оскільки існує консенсус як має виглядати граматично правильне речення англійської, а виправлені речення не сильно відрізняються від оригінальних, оцінювання переписувань таких систем є більш простим та зрозумілим у порівнянні з іншими задачами природної мови.

		Our	baseline	system	feeds	a	word	into	PB-SMT	pipeline	.
	0	1	2	3	4	5	6	7	8	9	10
Our	1	0	1	2	3	4	5	6	7	8	9
baseline	2	1	0	1	2	3	4	5	6	7	8
system	3	2	1	0	1	2	3	4	5	6	7
feeds	4	3	2	1	0	1	2	3	4	5	6
word	5	4	3	2	1	1	1	2	3	4	5
into	6	5	4	3	2	2	2	1	2	3	4
PB-SMT	7	6	5	4	3	3	3	2	1	2	3
pipeline	8	7	6	5	4	4	4	3	2	1	2
.	9	8	7	6	5	5	5	4	3	2	1

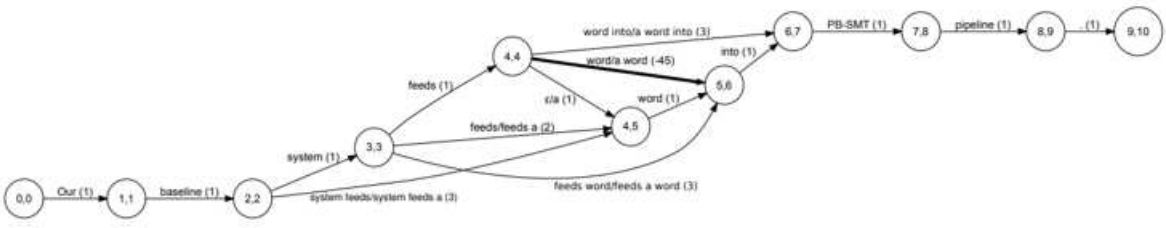


Рис. 1.2 – Використання матриці Левенштейна для побудова графа змін MaxMatch

M2 розраховується на основі знаходження найбільшої послідовності співпадаючих токенів між вихідним (згенерованим) текстом системи та декількома допустимими варіантами, які вважаються "правильними" (зазвичай це текст, оцінений людськими експертами). Таким чином рахується точність (precision) та повнота (recall), а далі застосовується $F_{0.5}$, який пріорітизує перше над другим приблизно у 2 рази.

$$F_{0.5} = \frac{1.25 \cdot precision \cdot recall}{0.25 \cdot precision + recall}$$

Таким чином, MaxMatch (M2) є важливою метрикою, що допомагає розробникам оцінити і поліпшити ефективність систем GEC, спрямованих на створення граматично правильних і зрозумілих текстів.

Також широкого розповсюдження набула метрика ERRANT (ERRor ANnotation Toolkit) [10], який є спеціалізованим інструментом для порівняння та анотації мовних помилок в англійському тексті. Відмінністю ERRANT від M2 є те, що вона фокусується на типах помилок та їх класифікації, замість просто врахування кількості. Це робить її корисною для більш змістовного оцінювання та аналізу систем автоматичного виправлення помилок.

Основна ідея ERRANT полягає в тому, щоб розбити зміни тексту на атомарні операції, такі як вставка, видалення або заміна слова, і потім класифікувати ці зміни за типами помилок, наприклад, орфографічні, граматичні або стилістичні. Це дозволяє отримати докладний аналіз роботи системи та виявити, на яких типах помилок вона працює найкраще чи найгірше

TER (Translation Edit Rate) [11] є ще однією метрикою, яка широко використовується для оцінювання якості генерованих текстів, зокрема в задачах машинного перекладу. Вона вимірює кількість редагувань, необхідних для того, щоб перетворити системний вивід на референсний текст, нормалізований на кількість слів у референсному тексті.

Математично, TER може бути виразена як:

$$TER = \frac{Levenstein(hyp, ref)}{\#\ of\ ref\ tokens}$$

де $Levenstein(hyp, ref)$ – відстань Левенштейна між згенерованим та анотованим текстами.

Так, TER є тісно пов'язаною з розмірною відстанню Левенштейна (Levenshtein distance), яка також вимірює мінімальну кількість одиночних символьних редагувань (вставок, видалень, замін) для перетворення одного рядка в інший. Однак, відстань Левенштейна застосовується на рівні символів, тоді як TER фокусується на рівні слів.

Все вищезазначене робить TER важливою метрикою для оцінки якості згенерованого тексту, дозволяючи не лише порівняти різні системи, але й діагностувати специфічні області тексту для поліпшення.

BERT-score [12] є сучасною метрикою, яка використовує представлення на основі моделі BERT (Bidirectional Encoder Representations from Transformers) для оцінювання подібності між згенерованим та референсним текстами. Ця метрика враховує не лише поверхневу структуру тексту, як у випадку з BLEU або TER, але й глибокі семантичні характеристики, які може вловити BERT.

Математично BERT-score використовує косинусну подібність між векторними представленнями слова в згенерованому тексті та відповідному слові в референсному тексті. Для кожного слова в згенерованому тексті обчислюється подібність з усіма словами в референсному тексті, і вибирається найбільший коефіцієнт подібності.

BERT-score зазвичай використовує F1-метрику для кінцевої оцінки. F1-метрика є гармонійним середнім між точністю (Precision) та повнотою (Recall), і вона допомагає збалансувати ці дві характеристики:

$$F_1 = \frac{2 \cdot Precision_{ER} \cdot Recall_{ER}}{Precision_{ER} + Recall_{ER}}$$

Де

$$Precision = \frac{1}{N} \sum_{n=1}^{N} \max_m CosineSimilarity(E_n, R_m)$$

$$Recall = \frac{1}{M} \sum_{m=1}^{M} \max_{n} CosineSimilarity(E_n, R_m)$$

де E_n — векторне представлення n-го слова в згенерованому тексті, R_m — векторне представлення m-го слова в референсному тексті, N і M — кількість слів у згенерованому і референсному текстах відповідно.

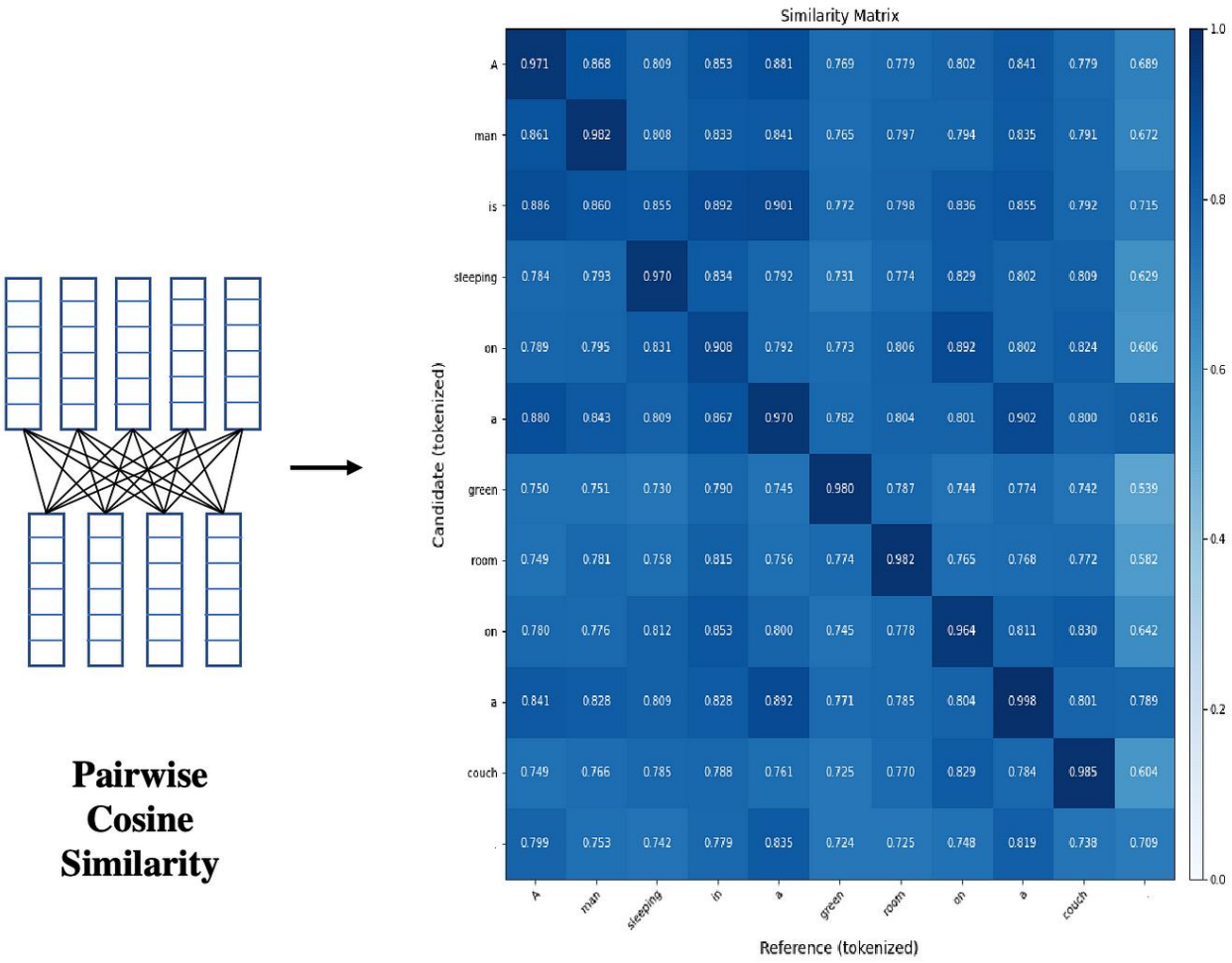


Рис. 1.3 – Попарна косинусна подібність між векторними представленнями BERT згенерованого і анотованого текстів

BERT-score призначена для оцінки якість згенерованого тексту на більш глибокому рівні, що робить її корисною для задач, де необхідна висока семантична точність.

1.3 Провідні практики для задачі перефразування тексту

Перефразування – переказ тексту іншими словами зі збереженням змісту. Наприклад, питання “How do I talk English fluently?” та “How can I improve my English speaking?” можна вважати парафразами. Особливістю даного напрямку у порівнянні з іншими задачами обробки природної мови є велика кількість робіт, які не використовують розмічені дані, а оперують лише звичайними моно корпусами тексту. Справа в тому, що умова і вихід для цієї задачі є взаємозамінними: якщо з речення $x_1, x_2, \dots, x_n$ ми можемо отримати речення $y_1, y_2, \dots, y_m$ з великою ймовірністю, то й логічно що при умові $y_1, y_2, \dots, y_m$ вихід $x_1, x_2, \dots, x_n$ має мати велику ймовірність. Більше того, кожне речення не повинне мати строго 1 парафраз, а може бути переписано різними способами, що підкреслює імовірнісну природу задачі.

Для перефразування існує відносно велика кількість джерел даних різної якості та різного рівня (словосполучення, речення, абзаци). Серед них слід виділити:

- Quora Question Pairs (QQP) [13] складається з 404,290 пар питань з сайту Quora, які були помічені модераторами як дублікати і тому є гарантовано парафразами;
- ParaNMT [14] містить більше 50 млн пар англійською, згенерованих нейромережевим перекладом паралельних корпусів для задач машинного перекладу. Крім самих пар набір даних включає у себе оцінки якості парафразів;
- LanguageNet [15] – набір парафразів отриманий “вирівнюванням” речень твітів, які посилались на один й той самий URL. Він налічує 51,524 пар речень розмічених анотаторами;
- MSCOCO [16] був розроблений для задачі створення субтитрів до зображень. Кожна картинка мала 5 підписів, які з великою ймовірністю є парафразами, що у сумі налічує 117 тисяч пар.

Таблиця 1.1 – Параметри наборів даних для перефразування

Назва	Створення	К-ть наборів	Прикладів у наборі	К-ть пар
ParaNMT	згенерований	50 000 000	2	100 000 000
QQP	проанотований	400 000	2	800 000
MSCOCO	проанотований	123 000	5	2 500 000

Проаналізувавши літературу по перефразуван-ню можна прийти до наступного висновку: найкращою системою для поточної задачі є LBOW-Topk з роботи Paraphrase Generation with Latent Bag of Words [17]. Модель – енкодер-декодер LSTM з модифікаціями для врахування специфіки задачі перефразування.

У класичному seq2seq підході модель спочатку кодує вхідну послідовність $x = x_1, x_2, \dots, x_n$ у послідовність h латентного простору, а потім декодує її у цільову послідовність y.

Енкодер enc_ϕ та декодер dec_θ – нейронні мережі, а функція втрат рахується наступним чином:

$$h = enc_\phi(x)$$

$$p(y|x) = dec_\theta(h)$$

$$L_{S2S} = E_{(x^*,y^*)\sim \mathrm{P}}[-\log p_\theta(y^*|x^*)]$$

де P^* – справжній розподіл даних.

Запропонована ж модель використовує слова з вхідного речення для передбачення їх “сусідів” в мішку слів з цільового речення. З передбачених “сусідів” береться випадковим чином підмножина слів, яка й використовується для передбачення цільового речення. Робиться це наступним чином.

Нехай маємо словник розміру V, тоді мішок слів z розміру k - це вектор з R^V, де k – кількість одиниць, а всі інші значення нулі. Припускається, що z отримується шляхом семплінгу з базового категоріально розподілу $p(\tilde{z}|x)$ k разів без замін. Для кожного слова xi з вхідної послідовності моделюється “сусід” $z_{ij} \in \mathrm{R}^{\mathrm{V}}$:

$$p(z_{ij}|x_i) = Categorical\left(\phi_{ij}(x)\right),$$

де ϕ_{ij} – нейронна мережа. На практиці застосовується лінійний шар з softmax на виході h енкодеру enc_{φ}. Припускається, що кожен вихідний токен x_i має максимальну кількість сусідів l. Далі ймовірності всіх “сусідів” перемішуються усередненням:

$$\tilde{z}{\sim}p_{\varphi}(\tilde{z}|x) = \frac{1}{ml}\sum_{i,j} p\left(z_{ij}|x\right).$$

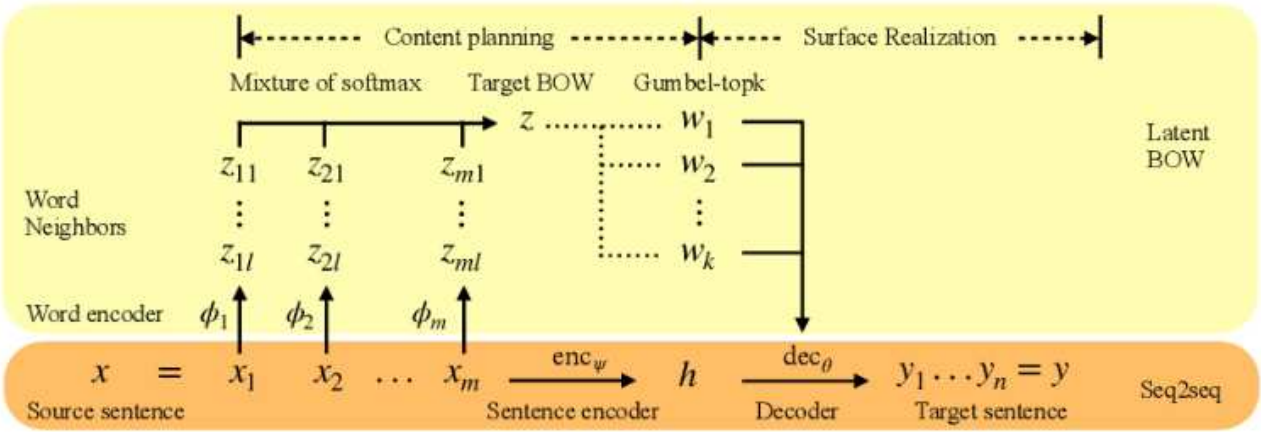


Рис. 1.4 – Схематичне зображення архітектури LBOW-TopK

Як було згадано вище, мішок слів z семплиться k разів без замін з розподілу $p(\tilde{z}|x)$. Тоді z використовується для контрольованого декодування вихідної послідовності y:

$$z{\sim}p_{\varphi}(\tilde{z}|x)$$

$$y{\sim}p_{\theta}(y|x,z) = dec_{\theta}(x,z)$$

Тоді фінальна функція втрат враховує як коректність передбачення мішку слів, так і декодування:

$$L_{S2S'} = E_{(x^*,y^*){\sim}\mathrm{P},\ z{\sim}p_{\varphi}(\tilde{z}|x)}[-log\ \ p_{\theta}(y^*|x^*,z)]$$

$$L_{BOW} = E_{z^*{\sim}P^*}[-log\ \ p_{\varphi}(z^*|x)]$$

$$L = L_{S2S'} + L_{BOW},$$

де z^* представляє мішок слів цільової послідовності.

Сам по собі семплінг z є недиференційованим. Щоб мати змогу рахувати градієнти зворотного поширення помилки використову-ється параметрична оцінка

градієнтів через gumbel-softmax. Нехай ймовірність $\tilde{z}$ дорівнює $p(\tilde{z} = i|x) = \pi_i,\ \ i \in 1, \dots, V$. Збурені ваги і ймовірності отримуються наступним чином:

$$a_i = log\ \ \pi_i + g_i$$

$$g_i \sim Gumbel(0,\ 1)$$

Вибір k найбільших вагів з $x = x_1, x_2, \dots, x_V$ дасть той самий результат, що й семплінг k разів без замін.

Хорошим прикладом побудови системи перефразування без використання відповідних тренувальних даних є робота Zero-Shot Paraphrase Generation with Multilingual Language Models [18]. Автори стверджують, що стандартний підхід генерації парафразів через подвійний переклад є не надійним та часто може змінювати зміст речень. Натомість пропонують використовувати набори даних машинного перекладу для цієї задачі на мовних моделях архітектури трансформер (де тільки декодер).

Спочатку тренується стандартна мовна модель одразу на декількох мовах, для цього на вхід подаються службові токени – початковий і кінцевий, токен, що відповідає мові, та сам текст. Наприклад, *<bos> <en> cat sat on the mat <eos>* або *<bos> <fr> j'aime le football <eos>*. Таким чином, модель вивчає як будуються речення різними мовами, а мовний токен дає можливість контролювати якою мовою ми генеруємо текст.

Далі використовується набори даних з машинного навчання. На вхід подаються пари речень, які є перекладами один одного. Знову застосовуються мовні токени, а також токен розділення речень. Наприклад, *<bos> <en> cat sat on the mat <dim> <delim> <fr> chat assis sur le tapis <eos>*.

В результаті, щоб отримати парафраз як префікс другого речення подається мовний токен тієї ж самої мови. Наприклад, на вхід *<bos> <en> cat sat on the mat <dim> <delim> <en>* , а на вихід *<bos> <en> cat sat on the mat <dim> <delim> <en> the cat sitting on the carpet <eos>*.

Також для покращення результату використовують додатково представлення мов, аби остаточно змусити модель лишатися на одній мові, зашумлення вхідних речень, аби модель лишалась інваріантною до помилок.

Іншою цікавою роботою для задачі перефразування є Learning Semantic Sentence Embeddings using Pair-wise Discriminator [19]. У ній автори тренують не тільки енкодер та декодер, а й дискримінатор.

Енкодер та декодер навчаються зі стандартною функцією втрат – кросс ентропією. Далі дискримінатор, який має спільні ваги з енкодером, будує векторні представлення для згенерованого екодером-декодером парафразу та для цільового речення. Автори пропонують максимізувати скалярний добуток відповідних представлень передбачених й розмічених речень та мінімізувати його для всіх інших речень у батчі.

В результаті ми отримуємо поєднання локальної функції втрат (для кожного токену згенерованого декодером) та глобальної (схожість векторних представлень). Таким чином ми намагаємося також врахувати загальний зміст речення. Після тренування дискримінатор більше не потрібний, достатньо використовувати енкодер-декодер мережу стандартним чином.

1.4 Провідні практики для задачі виправлення граматичних та орфографічних помилок

У задачі виправлення граматичних помилок (GEC) на вхід (як умова) подається речення, в якому потенційно можуть бути помилки, на вихід система має видати виправлене речення, тобто без помилок. Характерною особливістю напрямку є те, що вихідне речення не сильно відрізняється від вхідного – відносно мала відстань Левенштейна по словам. Через це багато робіт у даній області не переписують речення повністю, а лише пропонують виправлення. Таким чином, підходи з маркування послідовностей працюють на рівні з seq2seq.

Для задачі виправлення граматичних та орфографічних помилок існує доволі велика кількість розмічених даних, більшість з яких отримані з курсів вивчення англійської. Серед них:

- National University of Singapore Corpus of Learner English (NUCLE) [20] містить 1,400 есе написаних студентами та проанотованих професійними викладачами англійської. Всього налічує 56,958 речень, кожне з яких проанотоване 2 спеціалістами;
- Lang-8 Learner Corpora [21] є найбільшим серед відомих відкритих корпусів. Містить майже мільйон пар речень англійською з сайту Lang-8, де користувачі виправляють грамматику один одного;
- First Certificate in English (FCE) [22] corpus складається 1,244 виправлених відповідей до екзаменаційних питань FCE;
- W&I+LOCNESS [23] містить 34,304 пар речень, які були написані студентами та розмічені професійними анотаторами з Write & Improve. Крім самих пар міститься інформація про те, який рівень англійської здавав студент.

Найкращою системою для виправлення граматичних і орфографічних помилок є GECToR з роботи GECToR – Grammatical Error Correction: Tag, Not Rewrite [24]. Модель, замість того щоб передбачати правильне речення, передбачає виправлення, які потрібно зробити. З точки зору архітектури – це трансформер XLNet [25], дотренований на передбаченні граматичних та орфографічних виправлень. Нейронна мережа намагається вирішити одразу 2 задачі: передбачити чи потрібно робити виправлення для поточного токена x_i послідовності $x = x_1, x_2, \dots, x_n$ та яке саме виправлення потрібно зробити. Для цього отримане після енкодера представлення h_i подається одразу на 2 лінійні шари:

$$h = enc_\phi(x)$$

$$p_l = W_l h + b_l$$

$$p_d = W_d h + b_d.$$

Далі для обох передбачень рахується функція втрат та сумується:

$$L = L_{labeling} + L_{detection}$$

Існують різні типи виправлень, які передбачає модель:

- KEEP – не чіпати поточний токен;
- DELETE – видалити поточний токен;
- APPEND_w_i – вставити слово w_i після поточ-ного токену;
- REPLACE_w_i – замінити поточний токен на слово w_i;
- CASE – змінити регістр поточного токену;
- MERGE – "з'єднати" поточний токен з наступ-ним;
- SPLIT – розбити поточний токен на два;
- NOUN NUMBER – змінити множину поточно-го іменника;
- VERB FORM – змінити форму поточного дієслова.

Оскільки дана модель є неавторегресивною та має лише енкодер (декодер відсутній), вона є відносно швидкою (приблизно в 2-3 рази за авторегресивні моделі).

Інша відома архітектура для виправлення граматичних та орфографічних помилок була представлена у роботі Improving Grammatical Error Correction via Pre-Training a Copy-Augmented Architecture with Unlabeled Data [26]. Це модифікація трансформеру з використанням так званого механізму копіювання.

На кожному кроці генерації декодеру, окрім вибору токенів з фіксованого словника, нейронна мережа «збагачена» механізмом копіювання має можливість вибирати токени з вхідного речення.

Таким чином фінальний розподіл P_t це комбінація розподілу генерації P_t^{gen} та розподілу копіювання P_t^{copy}. В результаті маємо розширення вхідного словника токенами з вхідного тексту. Баланс між копіюванням та генерацією досягається завдяки змінній $\alpha_t^{copy} \in [0,1]$ на кожному кроці часу t:

$$p_t(w) = \left(1 - \alpha_t^{copy}\right) * p_t^{gen}(w) + \alpha_t^{copy} * p_t^{copy}(w)$$

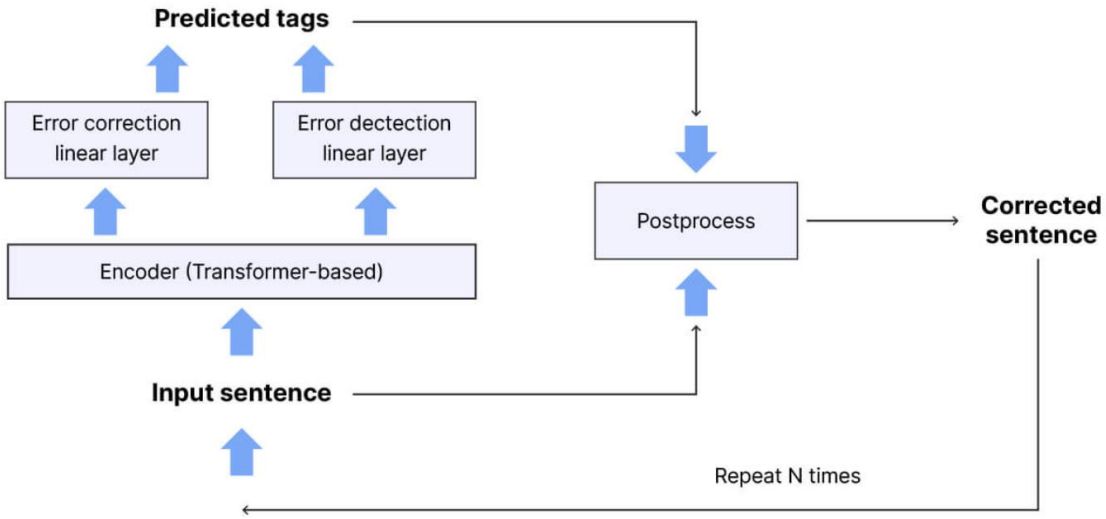


Рис. 1.5 – Схематичне зображення архітектури GECToR

Якщо обрахунок P_t^{gen} відбувається стандартним чином, то P_t^{copy} рахується за допомогою нового розподілу уваги між поточним прихованим станом декодеру h^{trg} та прихованими станами енкодеру H^{src}. Увага копіювання рахується схожим чином, як увага між екодером та декодером:

$$q_t, K, V = h_t^{trg} W_q^T, H^{src} W_k^T, H^{src} W_v^T$$

$$A_t = q_t^T K$$

$$P_t^{copy}(w) = softmax(A_t)$$

Далі використовується нормалізований розподіл уваги та приховані стани копіювання для обрахунку балансуючою змінної α_t^{copy}:

$$\alpha_t^{copy} = \sigma\left(W^T \sum(\mathrm{A}_\mathrm{t}^T V)\right).$$

У роботі Synthetic Data Generation for Grammatical Error Correction with Tagged Corruption Models [27] розглядається інший напрямок для покращення систем виправлення граматичних та орфографічних помилок — генерація синтетичних даних для подальшого претренування.

У традиційних методах внесення помилок до текстів часто використовуються правила, які застосовуються випадковим чином. Проте, новий підхід передбачає

використання спеціалізованої нейронної мережі для створення помилок, який відрізняється своєю цілеспрямованістю та ефективністю.

На першому етапі використовується стандартний набір даних для тренування моделей, призначених для виправлення помилок у тексті. Ці дані попередньо обробляються за допомогою інструменту ERRANT, який дозволяє розмітити текст на конкретні типи помилок. Це дає моделі можливість більш точно зрозуміти, які саме помилки часто зустрічаються в текстах та як вони розподілені.

Наступним кроком є тренування моделі на генерації тексту з помилками на основі вхідного правильного тексту. У цьому випадку, тип помилки подається на вхід моделі у вигляді текстового префікса, що дозволяє моделі цілеспрямовано вносити конкретні помилки у визначені місця тексту. Це суттєво відрізняється від звичайних методів, де помилки генеруються випадково, і дозволяє більш точно моделювати реальні помилки, які зустрічаються в текстах.

Такий підхід може бути особливо корисним для створення більш реалістичних тренувальних наборів даних для систем виправлення текстових помилок, дозволяючи моделям краще адаптуватися до різноманітності помилок, які зустрічаються у природних мовних даних.

Таким чином ми краще контролюємо нашу модель для внесення помилок та можемо зробити розподіл помилок відповідним, наприклад, тестовій частині датасету. Саме це і роблять автори повторюючи розподіл валідаційної частини W&I+LOCNESS. Згенеровані дані обіцяють викласти у відкритий доступ.

Автори пропонують наступні кроки для отримання моделей найкращої якості:

1. Претренування моделі на отриманому синтетичному датасеті разом з помилками з Вікіпедії
2. Тренування на датасеті Lang-8
3. Остаточне дотренування на комбінації FCE та W&I+LOCNESS

Так ми рухаємося від найбільш зашумлених (менш якісних) джерел даних до найбільш якісних, але менших по розміру.

1.5 Провідні практики для задачі спрощення тексту

Спрощення тексту — переписування тексту таким чином, що граматика та структура значно спрощуються, а основне значення та інформація залишаються незмінними. Наприклад, для речення “Both men and women when attending a mosque must adhere to these guidelines.” спрощеним буде “Both men and women when going a mosque must follow these rules.”

В порівнянні з попередніми задачами джерел даних для задачі спрощення тексту помітно менше. Серед них:

- WikiLarge [28] складається з 296 тисяч пар речень (складних і відповідних простих), створених напівавтоматичним чином зі змін у Вікіпедії;
- Newsela [29] містить 1,130 статей новин, переписаних аннотаторами для дітей різних класів з різним “рівнем” спрощення. Налічує 95 тисяч пар речень та є більш якісним у порівнянні з WikiLarge.

Згідно метрик якості SARI та FKGL, найкращою системою на сьогоднішній день є нейрона мережа з роботи MUSS: Multilingual Unsupervised Sentence Simplification [30]. Це seq2seq модель архітектури transformer - BART [31].

У роботі значна увага приділяється контрольованій генерації тексту. Для цього під час навчання на вхід, крім оригінальних (складних) речень подаються також спеціальні токени, які відповідають певним атрибутам спрощення:

- NbChars – співвідношення літер у оригінально-му і спрощеному реченні. Відповідає за рівень “компресії” переписаного;
- LevSim – нормалізована схожість Левенштейна між реченнями на рівні літер. Тобто на скільки відрізняється переписане речення від оригінального;
- WordRank вимірює те, на скільки слова з отриманого речення є вживаними у порівнянні з вхідним;
- DepTreeDepth відповідає глибині Dependency tree для речень. Порівнює синтаксичну складність.

Далі під час генерації тексту подаються ці 4 спеціальні токени в залежності від того, яким ми хочемо бачити переписане речення. Наприклад, щоб отримати максимально коротке речення можна подати токен, який відповідає NbChars = 0.1.

У роботі Integrating Transformer and Paraphrase Rules for Sentence Simplification [32] автори намагаються додати зовнішні знання до нейронної мережи архітектури трансформер. Джерелом у даному випадку виступає Paraphrase Database for Simplification (PPDB) [33] — відкрита база даних для спрощення слів/словосполучень. Наприклад, пара «ключ» → «значення»: recipient → winner.

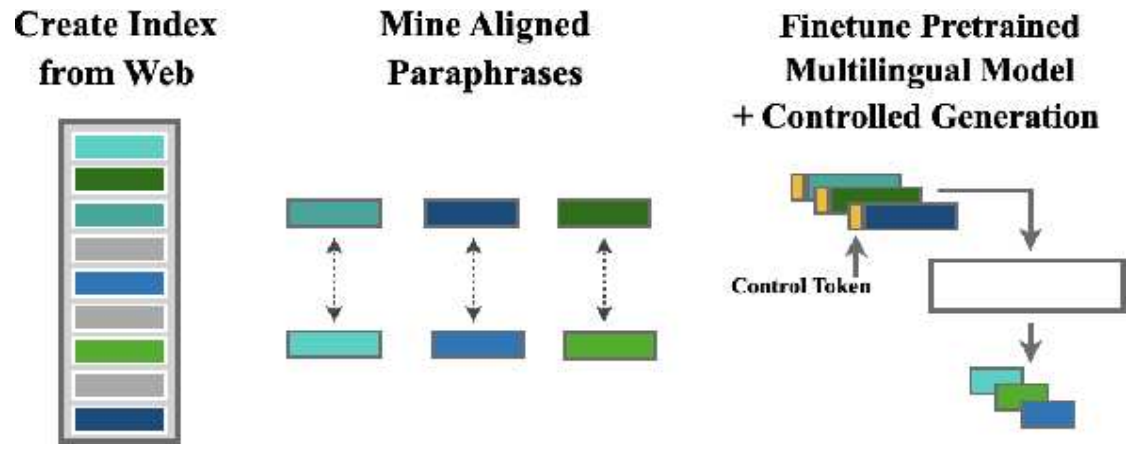


Рис. 1.6 – Контрольована генерація MUSS

Перш за все, використовується спеціальна функція втрат, яка зменшує ймовірність складних слів у реченні (тих, що потрапили як «ключі» до бази даних) та збільшує ймовірність простих слів (тих, що потрапили як «значення»).

По-друге, правила з PPDB враховуються на кожному кроці генерації. Спочатку рахується вектор контексту як середнє зважене виходів енкодеру, де ваги це скалярний добуток з поточним прихованим вектором декодеру. Далі цей вектор контексту проходиться по всім представленням правил з PPDB для слова, яке генерується на поточному кроці. Представлення слів, запропонованими правилами зважуються і додаються до поточного виходу енкодеру. Таким чином нейронна мережа враховує правила з бази даних.

Ще однією знаковою роботою для задачі спрощення тексту є Text Simplification by Tagging [34]. У ній автори використовують вже знайому нам архітектуру GECToR, але для цієї задачі.

Сама по собі авторегресивна модель показує недостатні результати, тому автори намагаються використовувати різній прийоми для її покращення. Розглянемо їх далі.

BackTranslation: для збільшення розміру тренувальних даних вхідні речення з датасету проганяються через моделі машинного перекладу туди назад. Наприклад, спочатку з англійської на німецьку, а далі назад з німецької на англійську. Таким чином отримується парафраз для вхідного речення, який можна використати при тренуванні основної моделі.

Ensemble Generation: на основі поточних найкращих моделей будується ансамбль, який генерує більш якісні дані, але значно повільніше. Далі ансамблем з вхідних речень тренувального набору даних генеруються спрощення. Такі пари теж використовуються під час тренування фінальної моделі.

Ці кроки дозволяю помітно покращити якість моделі. Фінальна модель є другою найкращою на поточний момент згідно метрики SARI. При цьому підіхід демонструє кращу швидкість у порівнянні з першим місцем.

1.6 Порівняльний аналіз підходів до задач

Слід зазначити, що вищезгадані задачі мають спільну рису – в усіх задачах переписування тексту відбувається зі збереженням змісту. Тобто якщо “форма” змінюється, то сенс залишається тим самим. Підходи до вирішення задач теж мають ряд спільних ознак.

Насамперед це архітектура трансформер. З моменту її представлення у роботі "Attention is all you need" [35], трансформери стали де-факто стандартом для state of the art підходів у області обробки природної мови. Суттєву роль у цьому зіграло так зване попереднє тренування моделей на більш загальних текстових задачах. Зазвичай для таких задач не потрібні розмічені дані – достатньо великих моно

корпусів без анотацій для тренування. Наприклад, у найпопулярнішому варіанті Masked Language Modelling, 15% випадкових токенів маскуються і моделі слід відгадати справжні.

Другим спільним моментом усіх підходів є те, що створюються архітектурні особливості під кожну задачу. Так у задачі перефразування для врахування стохастичності виходу (для речення не існує строго одного парафразу) використовується Latent Bag of Words, який допомагає випадковим чином вибрати токени майбутнього речення та, застосовуючи їх, згенерувати текст. Для врахування того факту, що речення суттєво не змінюється у виправленні граматичних помилок передбачаються виправлення замість генерації всього тексту. У спрощенні тексту в модель закладаються її атрибути: наскільки текст має бути коротший, простіший (з лексичної та синтаксичної точки зору) та як сильно відрізнятися від вхідного.

Третя риса – архітектура seq2seq, що є універсальною на сьогоднішній день. Покращений cross-attention з архітектури трансформер дуже добре передає контекст отриманий з енкодеру у декодер. При цьому слід враховувати, що це призводить до авторегресивної генерації, що у свою чергу сповільнює модель, але одночасно дає хорошу якість. У той же час, бачимо, що неавторегресивні моделі у деяких задачах досягають порівняно хороших результатів. Характерно це для задач, де вхід (умова) не сильно відрізняється від виходу (одна мова, схожа довжина тексту).

Також одним з найбільших викликів в області переписування тексту є збереження семантичної цілісності при зміні форми. Іншими словами, модель має не просто переформулювати текст або виправити граматичні помилки, але і зробити це так, щоб основний зміст залишився незмінним. Це особливо важливо у задачах, де необхідна висока точність та довіра до моделі, наприклад, в медичних, юридичних або наукових текстах. Тут можуть застосовуватися такі техніки, як контроль згенерованого тексту через механізми уваги або використання додаткових вхідних даних для моделі, як-от метадані чи контекстуальна інформація.

У той же час визначення state-of-the-art моделей часто формується на базі автоматичних метрик, які не виглядають дуже надійним. Існує багато питань стосовно того на скільки добре *n*-грамні метрики допомагають оцінити якість згенерованого переписування, особливо у випадках коли задача така, що вхідний текст має багато виправлених версій, які можуть вважатися правильними. Тому варто ставитися до таких результатів обережно, оскільки моделі могли просто бути добре підлаштовуваними спеціально під тестові дані.

РОЗДІЛ 2. АРХІТЕКТУРА НЕЙРОННИХ МЕРЕЖ ДЛЯ ЗАДАЧ ПИСЬМОВОГО АСИСТЕНТУ

У даній роботі було вирішено зосередитися на архітектурах глибокого навчання. Вони дозволяють вирішувати конкретні задачі з високою точністю та є найбільш популярним і ефективним інструментом у задачах NLP на сьогодні. Також прийняте рішення використання стохастичного градієнтного спуску для оптимізації, що є стандартним підходом на сьогодні.

Значущість вибору архітектури штучної нейронної мережі не можна недооцінювати. Важливо зазначити, що сфера прикладного дослідження в NLP зазнає впливу різноманітних архітектур, від згорткових (CNN) та рекурентних нейронних мереж (RNN) до трансформерів. Кожна з цих архітектур має свої переваги та недоліки, що робить їх оптимальними для різних сценаріїв застосування.

Для задачі побудови письмового асистента була використано усі загальновідомі типи мереж. Далі наведено опис їх архітектур та компонент, наведено теоретичні деталі.

2.1 Методи представлення токенів для нейронних мереж

Векторні представлення слів, відомі як word embeddings, відіграють суттєву роль у розвитку нейронних мереж, призначених для обробки природної мови. Ці представлення кодують семантичну інформацію слова у вигляді n-вимірного вектора, де n визначає кількість параметрів у векторі. Застосування таких векторів як вхідних даних істотно покращує здатність моделі розуміти контекст і семантику тексту.

Тренування цих представлень відбувається зазвичай у рамках неконтрольованого навчання, на основі обширних текстових корпусів. Для визначення семантичних відносин між словами часто використовується аналіз спільних згадок слів (word co-occurrences). Це має значення, адже взаємодія слів у тексті характеризується нелінійністю та змінюється в залежності від контексту.

Таким чином, векторні простори, де розміщені ці представлення, зазвичай є багатовимірними та складними.

Word embeddings також надають можливість моделям автоматично визначати синоніми, антоніми, а також різноманітні зв'язки між словами і фразами. Їх застосування не обмежується лише задачами класифікації тексту, але й включає більш складні сценарії, такі як машинний переклад, генерація тексту та аналіз емоційного забарвлення.

Важливо зазначити, що існує певна нестабільність цих векторних просторів. Слова з різними семантичними характеристиками можуть знаходитися поруч у цих просторах, що інколи веде до непередбачуваних результатів під час тренування та застосування моделей. Тому збалансований підхід до вибору параметрів і методології тренування є ключовим для досягнення високої точності в задачах обробки мови.

Ці представлення можуть слугувати фундаментом для розробки складніших механізмів у моделях нейронних мереж, які враховують порядок слів, структурні взаємозв'язки між реченнями та інші аспекти мови, що важко виразити через ізольовані слова. Хоча сьогодні попередньо натреновані моделі стають все більш популярні, word embeddings залишаються важливою частиною систем обробки природної мови.

2.1.1 Представлення Word2Vec

Модель Word2Vec [36], розроблена командою дослідників Google під керівництвом Томаса Міколова у 2013 році, швидко стала стандартним інструментом для представлення слів у багатьох дослідженнях, пов'язаних з обробкою тексту. Ця модель використовує векторне представлення слів і має багато спільного з моделлю NNLM (Neural Network Language Model), особливо з її варіантом CBOW (Continuous Bag-of-Words Model). Основна відмінність CBOW від NNLM полягає в відсутності нелінійних прихованих ваг і використанні єдиного шару проектування для всіх слів, де вектори слова усереднюються, щоб зменшити

вплив послідовності слів на результати. Така архітектура забезпечує використання інформації про слова, які можуть зустрічатися в контексті з даним словом.

Інша значуща модель, використовувана у тренуванні ембедингів, - це Continuous Skip-gram Model. Хоча вона має схожості з CBOW, її ключова відмінність полягає в тому, що замість прогнозування поточного слова на основі контексту, вона прагне максимізувати ймовірності слів, які зустрічаються в навколишньому контексті. Continuous Skip-gram Model використовує поточне слово як вхідні дані для лог-лінійного класифікатора з неперервним шаром проекції та намагається прогнозувати слова, розташовані перед і після поточного слова.

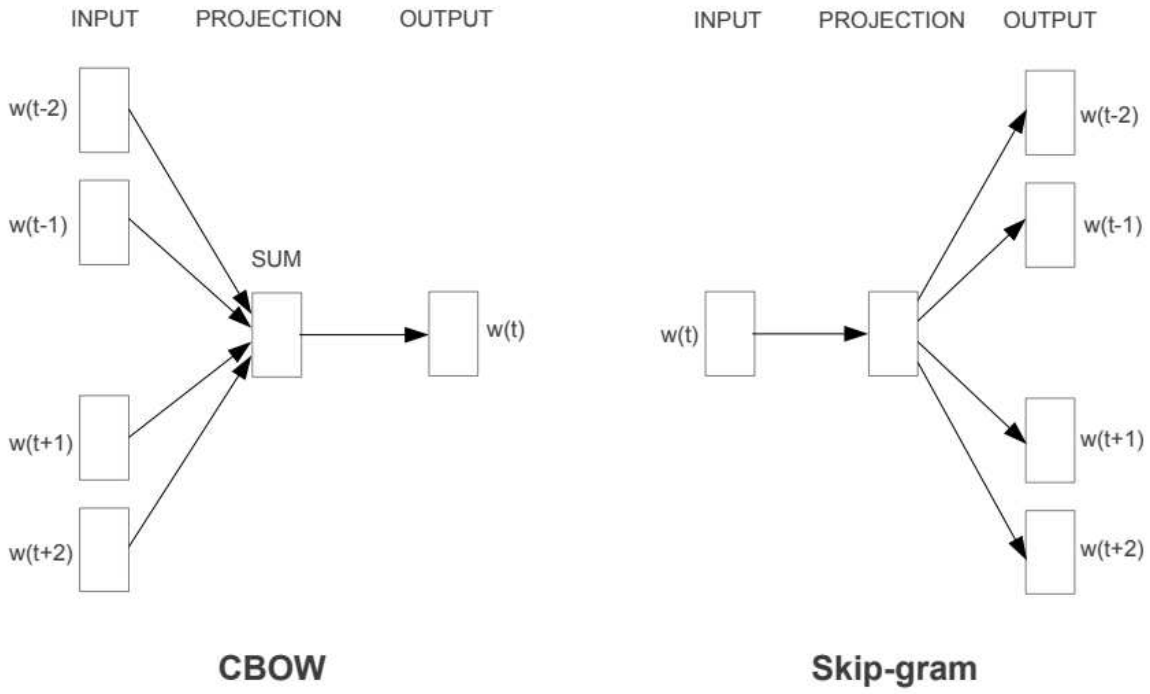


Рис. 2.1 – Схематичне представлення 2-х підходів Word2Vec

Модель Word2Vec, здійснюючи тренування, використовує методи ієрархічного softmax та/або негативного семплінгу. Це здійснюється з метою апроксимації умовної ймовірності, яку модель прагне максимізувати. Ієрархічний softmax впроваджує дерево Хаффмана для ефективного зменшення обчислювальних витрат. В контрасті, метод негативного семплінгу підходить до задачі максимізації ймовірності як до задачі мінімізації ймовірності вибору

некоректних пар слів. Автори моделі зазначають, що ієрархічний softmax ефективніший для рідкісних слів, тоді як негативний семплінг показує кращі результати для частих слів і векторів з меншою кількістю вимірів. Проте, зі зростанням кількості епох тренування, переваги ієрархічного softmax зменшуються.

Тренування моделі на великих текстових корпусах з некоректними словами може призвести до ситуації, де слова з орфографічними помилками стають близькими до правильних слів у векторному просторі. Отже, в деяких випадках, можливо ігнорувати слова з помилками при аналізі тексту.

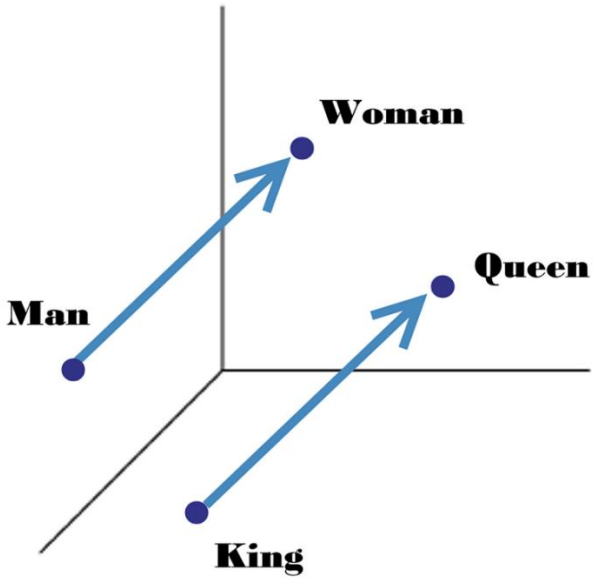


Рис. 2.2 – Візуалізація аналогії слів у Word2Vec

Аналогія слів може бути просто описана за допомогою лінійної алгебри - нехай потрібно підібрати аналогію до слів $man \rightarrow woman \rightarrow king$. Позначимо невідоме слово як w. Тоді слово w, таке що $v_{king} - v_w$ буде мати таку ж саму відстань що й $v_{man} - v_{woman}$, тобто буде виконуватись наступне:

$$||v_{man} - v_{woman} + v_{king} - v_w|| \approx 0.$$

2.1.2 Метод представлень GloVe

Наступним відомим представленням слів є Global Vectors (GloVe) [37]. Статистика по словам в корпусі для тренування є головним джерелом наявної інформації для всіх методів векторних представлень слів, відмінним є лиш те, яким чином використовується ця інформація й як представляються слова в кожному з методів векторного представлення слів.

Позначимо X як матрицю статистики входжень слово-слово в корпусі, тобто X_{ij} – це число, яке характеризує кількість входження слова j в контексті слова i. Позначимо $X_i = \sum_k X_{ik}$ - число входжень будь якого слова в контексті зі словом i. Нехай

$$P_{ij} = P(j|i) = \frac{X_{ij}}{X_i}$$

ймовірність того, що слово j зустрічається в контексті слова i.

За початкову точку для навчання моделі береться функція

$$F(w_i, w_j, \hat{w}_k) = \frac{P_{ik}}{P_{jk}},$$

де $w \in R^d$ – вектори слів, а $\widehat{w} \in R^d$ – окреме контекстне слово. Оскільки простір векторів, за побудовою, це лінійні структури, то найбільш природнім способом кодування інформації P_{ik} та P_{jk} буде їх векторна різниця. Отже, попереднє рівняння можна переписати:

$$F(w_i - w_j, \hat{w}_k) = \frac{P_{ik}}{P_{jk}}$$

Наступне що можна помітити – $w_i - w_j$ є вектором, а w_k скалярною величиною. Оскільки F може бути складною функцією (наприклад нейронною мережею), то краще використовувати скалярний добуток:

$$F\left((w_i - w_j)^T \hat{w}_k\right) = \frac{P_{ik}}{P_{jk}}$$

таким чином, функція F не буде змінювати розмір вектору небажаними способами. Тоді в матриці входжень слово-слово можна задавати різницю між

словом і контекстним словом. Для того щоб консистентно оновити слово не достатньо змінити $w \leftrightarrow \hat{w}$, потрібно ще й оновлювати $X \leftrightarrow X^T$. Оскільки попереднє рівняння не є інваріантним до подібних змін, то потрібно задати обмеження на функцію F - необхідно дотримуватися гомоморфізму між групами $(R, +)$ та $(R_{>0}, \times)$, тобто необхідно, щоб виконувалось:

$$F\left((w_i - w_j)^T \hat{w}\right) = \frac{F(w_i^T \hat{w}_k)}{F(w_j^T \hat{w}_k)}$$

отже, маємо

$$F\left(w_i^T \hat{w}_k\right) = P_{ik} = \frac{X_{ik}}{X_i}.$$

Розв'язком рівняння буде $F = exp$, тобто:

$$w_i^T w_k = \log(P_{ik}) = \log(X_{ik}) - \log(X_i).$$

Оскільки $log(X_i)$ не залежить від k, то його можна позначити як b_i – зміщення для w_i. Додавши зміщення bw для wk можна відновити симетрію рівняння:

$$w_i^T \hat{w}_k + b_i + \hat{b}_k = \log(X_{ik}).$$

Останнє рівняння є сильним скороченням першого рівняння з певними умовами – коли аргумент логарифму збігається до 0, то сам логарифм починає розходитись. Щоб вирішити цю проблему можна використати зсув логарифму – замість $log(X_{ik})$ використати $log(1 + X_{ik})$. Таким чином, буде збережено розрідженість X, уникаючи розбіжності в логарифмі.

Особливістю GloVe є те, що автори розв'язують рівняння як задачу найменших квадратів, додавши функцію зважування $f(X_{ij})$ до функції втрат. Отже, маємо модель

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \hat{w}_j + b_i + \hat{b}_j - \log X_{ij}\right)^2,$$

де V – розмір словника.

Функція зважування має задовольняти наступним властивостям:

1. $f(0) = 0$. Якщо розглядати функцію f як неперервну функцію, то потрібно щоб вона збігалась при $x \to 0$ так, щоб ліміт $\lim_{x \to 0} \frac{f(x)}{\log^2 x}$ був збіжним.
2. $f(x)$ має бути не спадною для невеликих значень x.
3. $f(x)$ має бути достатньо маленькою для достатньо великих значень x.

Зрозуміло, що багато функцій задовольняють подібним умовам, проте існує один доволі зручний клас параметризованих функцій:

$$f(x) = \begin{cases} (\frac{x}{x_{\max}})^{\alpha}, & \text{якщо } x < x_{\max} \\ 1, & \text{інакше} \end{cases}$$

Якість моделі слабо залежить від відсічення, що використовується у формулі. В оригінальній статті використовується $x_{max} = 100$, а також, зазначено, що коли $\alpha = 3/4$ модель показує себе краще, ніж при $\alpha = 1$.

2.1.3 Векторні представлення FastText

FastText [38] має ключову відмінність від Word2Vec: вона враховує статистику слів під час тренування моделі.

Розглянемо словник з розміром W, в якому кожному слову відповідає певний індекс $(w \in 1,2,\dots,W)$. Основною метою моделі є навчання векторного представлення для кожного слова w. Фундаментальна ідея, яка спонукала до створення FastText, базується на гіпотезі про розподіл слів: репрезентація слів повинна ефективно прогнозувати ті слова, які зустрічаються в контексті. Формальніше, ця концепція виражається через максимізацію логарифму правдоподібності на основі великого корпусу послідовностей слів $w_1, \dots, w_T$ максимізувати логарифм правдоподібності:

$$\sum_{t=1}^{T} \sum_{c \in C_t} \log p\left(w_c | w_t\right),$$

де контекст C_t – множина індексів слів, які зустрічаються зі словом w_t.

Нехай є функція оцінки s, яка є відображенням пари слово-контекст в R. Одним з найпопулярніших способів задання ймовірності слова є софтмакс функція:

$$p\left(w_c|w_t\right) = \frac{e^{s(w_t,w_c)}}{\sum_{j=1}^{W} e^{s(w_t,j)}},$$

але софтмакс функція працює добре, коли потрібно визначити для слова w_t ймовірність наступного слова w_c з контексту.

Для того щоб мати можливість прогнозувати декілька слів, виходячи з контексту, автори FastText розв'язують декілька незалежних задач бінарної класифікації. Головною метою є прогнозування наявності або відсутності контекстних слів. Для слова з позицією t всі слова, що зустрічаються з ним разом помічаються як позитивні записи, а довільні слова зі словнику, які не зустрічаються з ним в контексті – негативні записи. Далі, для обраної контекстної позиції c, використовуючи бінарну логістичну функцію втрат, формула негативної лог-правдоподібності має вигляд:

$$\log\left(1 + e^{-s(w_t,w_c)}\right) + \sum_{n \in N_{t,c}} \left(1 + e^{s(w_t,n)}\right),$$

де $N_{t,c}$ – множина негативних записів, які довільним чином обираються зі словаря. Якщо позначити логістичну функцію втрат як

$$l : x \mapsto \log\left(1 + e^{-x}\right),$$

то можна переписати цільову функцію наступним чином

$$\sum_{t=1}^{T} \left[\sum_{c \in C_t} l\left(s(w_t, w_c)\right) + \sum_{n \in N_{t,c}} \left(-s(w_t, n)\right) \right].$$

Запропонована модель є не що інше як SkipGram - модель з негативним семплінгом.

Відмінністю FastText від інших моделей ембедингів є модель підслів. Використовуючи окреме векторне представлення для кожного слова модель SkipGram ігнорує внутрішню структуру слів. Для того щоб виправити це, модель підслів, для кожного слова w, використовує його представлення як "мішок" n-грам,

що побудовано на літерах слова. Для позначення початку й кінця слів використовуються спеціальні символи: <,>. Тобто, для слова *where* його представлення, як послідовність триграм, буде: <*wh*, *whe, her, ere, re*>, включно включно зі спеціальною послідовністю <*where*>.

Варто зазначити, що елемент послідовності *her* не є еквівалентним до слова <*her*>. Зазвичай на практиці використовують *n*-грами слів в діапазоні від 3 до 6.

Нехай є словник *n*-грам розміру G і нехай для слова w маємо $G_w \subset \{1, \dots, G\}$ – множину *n*-грам. Нехай вектор z_g кодує представлення *n*-грами g, тоді кожне слово буде преставлятись як сума його векторних представлень *n*-грам. Тоді скоринг функція приймає вигляд:

$$s(w, c) = \sum_{g \in G_w} z_g^T v_c.$$

2.2 Рекурентні нейронні мережі

Людський розум, не втрачаючи контексту мислення, завжди пам’ятає попередні події та дії. Саме цей принцип лежить в основі рекурентних нейронних мереж (RNN). Відмінною особливістю RNN від традиційних повнозв’язних нейронних мереж є здатність зберігати інформацію про попередні стани, тобто в кожний момент часу існує зв'язок з попередніми станами, що схематично представлено на рисунку 2.3.

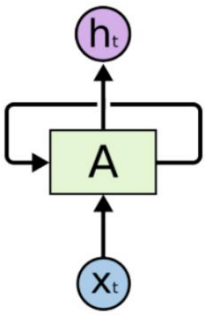


Рис. 2.3 – Схематичне зображення рекурентної нейронної мережі

Саме з задачами, в яких потрібно працювати з послідовностями гарно справляються рекурентні нейронні мережі – акумулюють інформацію, ітеруючись по часу.

Тут x_t вхідні дані, які залежать від часу t, а h_t – це стан системи в даний момент часу. Цикл, зображений на рисунку 2.3, зображує передачу інформації до наступного кроку нейронної мережі.

Даний цикл можна розгорнути для того щоб зрозуміти, що відбувається всередині. Ілюстрація того як передаються дані зображена на рисунку 2.4.

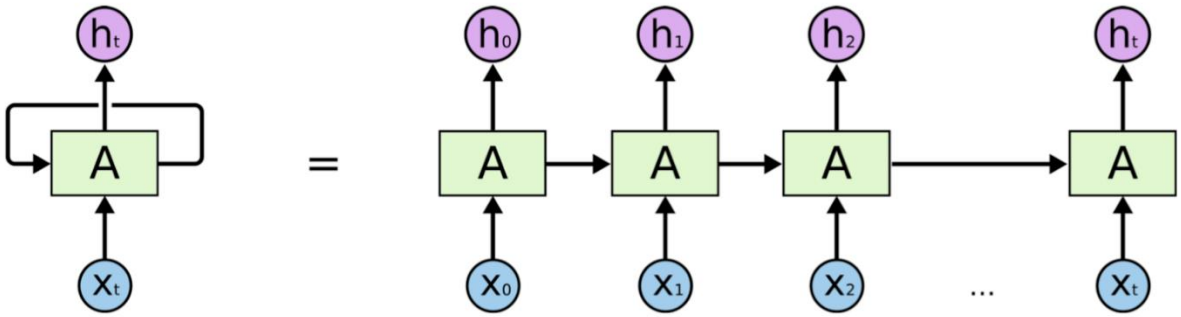


Рис. 2.4 – "Розгорнута" рекурентна нейронна мережа

Через простоту й гнучкість подібного рішення рекурентні нейронні мережі використовують в багатьох задачах – розпізнавання та моделювання природної мови, переклад тексту, генерація звуку з тексту, спостереження за датчиками тощо.

Концептуально, звичайні рекурентні нейронні мережі мають недолік - проблему довгострокових залежностей. Для певних задач це може бути критичним. Для прикладу, нехай є задача розпізнавати слова з потоку звуків, проте в певний момент часу повз мікрофон проходять інші люди й створюють шум, розмовляючи про щось своє, а це означає що в певні моменти часу відбуватиметься збурення датчику й оскільки слова які датчик може почути в шумі будуть без контексту, то це буде призводити до погіршення результатів. Іншим прикладом слугує задача машинного перекладу – в певних мовах, в одне речення може передавати декілька окремих сутностей. Звичайно логічніше було б розділити ці сутності на окремі речення й тренувати модель, перекладаючи їх окремо, проте, для випадку таких

довгих речень, модель акумулюватиме інформацію зі всіх сутностей в одну "узагальнену" сутність, яка, вочевидь, буде не зовсім коректною.

Для вирішення проблеми довгострокових залежностей, важливо зосереджуватися на релевантній інформації та враховувати свіжі дані. Звичайні RNN при обробці довгих послідовностей схильні до "розмивання" контексту. Модифікації RNN, такі як LSTM (Long Short-Term Memory) та GRU (Gated Recurrent Unit), розв'язують цю проблему.

Двонаправлені рекурентні нейронні мережі (BiRNN) [39] складаються з двох незалежних RNN, які об'єднані в одну структуру. Вони обробляють вхідну послідовність одночасно в прямому та зворотному порядках часу. Результати роботи обох мереж в кожний момент часу зазвичай конкатенуються, хоча існують і альтернативні підходи обробки цих даних.

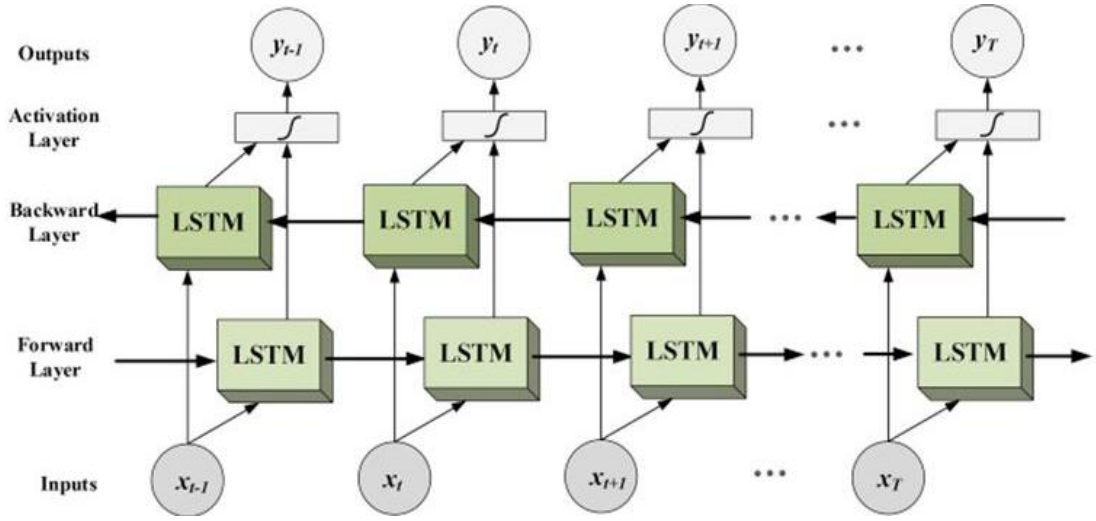


Рис. 2.5 – Схематичне зображення двонаправленої RNN

2.2.1 Архітектура довгої короткочасної пам'яті

Нейронні мережі Long Short-Term Memory (LSTM) [40], представлені Hochreiter & Schmidhuber у 1997 році, є варіацією рекурентних нейронних мереж, спроектованих для вивчення довготривалих залежностей. Хоча цей тип мереж був розроблений ще у 1990-х роках, їхня широка популярність настала в 2010-х, паралельно з ростом обчислювальних можливостей, і з того часу вони знайшли застосування в багатьох сферах.

Особливість LSTM полягає у спеціалізованій архітектурі, яка дозволяє уникнути проблеми з довготривалими залежностями. Завдяки своїй конструкції, ці мережі мають здатність зберігати інформацію на тривалі періоди часу за замовчуванням, а не як результат вивчення.

Як і стандартні рекурентні нейронні мережі, LSTM обробляє дані ітеративно, проте має відмінний механізм взаємодії. Вони містять спеціальні структури, відомі як "ворота", які регулюють потік інформації в мережі. Ці "ворота" дозволяють мережі вибірково "запам'ятовувати" або "забувати" інформацію, що є ключовим для їхньої ефективності у вирішенні складних завдань, таких як обробка мови, прогнозування часових рядів та інших задач, де потрібно управління довготривалою пам'яттю.

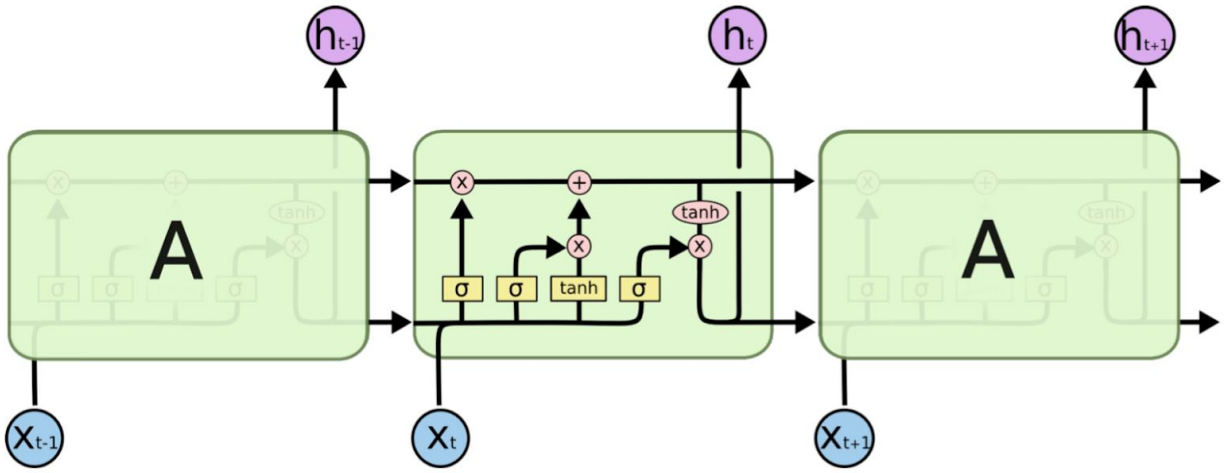


Рис. 2.6 – Схематичне зображення двонаправленої LSTM

Спочатку в LSTM нейронній мережі вирішується яка інформація буде відсіяною – це робиться за допомогою сигмоїд функції з використанням попереднього стану клітини h_{t-1}.

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

Наступним кроком є вирішення яку інформацію потрібно записати до стану клітини. Даний крок складається з двох частин:

- за допомогою сигмоїд функції визначаються які ваги потребують оновлення;

- за допомогою функції *tanh* обраховується вектор "кандидатів" які можуть бути доданими до стану клітини.

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$
$$\hat{C}_t = \tanh\left(W_C\,[h_{t-1}, x_t] + b_C\right)$$

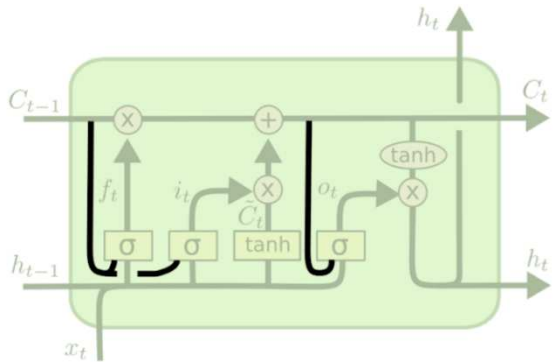
Після цього потрібно обчислити новий стан клітини C_t, з урахуванням попереднього стану C_{t-1}.

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

Тепер потрібно визначити яку інформацію потрібно передати в наступний стан. Ця інформація буде базуватись на відфільтрованій інформації стану клітини.

$$o_t = \sigma\left(W_o\,[h_{t-1}, x_t] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Наведені вище формули описують роботу звичайної LSTM нейронної мережі, проте в неї існують модифікації, на рисунку 7 зображена одна з них.



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \;+\; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \;+\; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \;+\; b_o\right)$$

Рис. 2.7 – Так звана Peephole LSTM

2.2.2 Архітектура вентильного рекурентного вузла

Вентильний рекурентного вузола (англ. Gated Recurrent Unit, GRU) [41] – це нове покоління рекурентних нейронних мереж, яке за своєю структурою дуже схоже на LSTM. GRU позбулася стану клітини й натомість використовує прихований стан для передачі інформації. Особливість GRU полягає в тому, що вона має тільки два шари фільтрації інформації – шар скидання і шар оновлення.

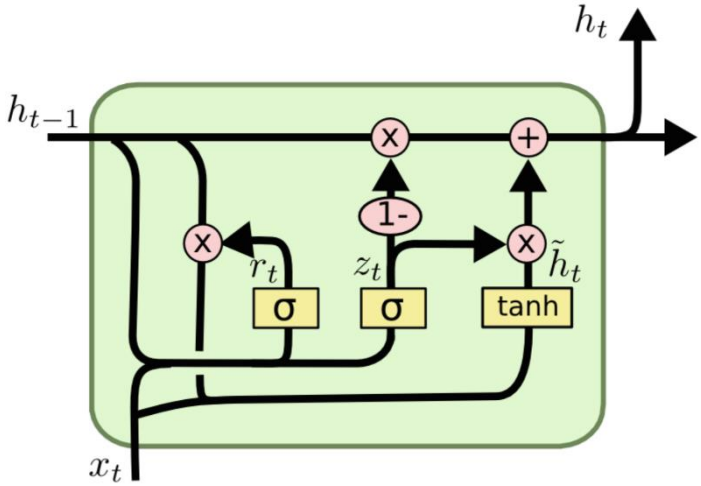


Рис. 2.8 – Схематичне зображення GRU

Всі обрахунки починаються з шару оновлення:

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

Шар оновлення працює майже так само як шар забування та вхідний шар в LSTM – вирішує яку інформацію потрібно позбутись й яку інформацію потрібно залишити. Варто зазначити що GRU дозволяє знизити ймовірність затухання градієнтів.

Шар скидання використовується для того щоб визначити яку попередню інформації потрібно позбутися. Обчислюється він як:

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

Для оновлення поточного стану клітини використовується інформація з шару скидання:

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

Наступне що потрібно зробити – порахувати пам'ять клітини в поточний момент часу:

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

Нескладно помітити те, що кількість обрахунків, що відбуваються в GRU, менша ніж в LSTM, а це, в свою чергу, впливає на швидкість тренування. Як вже було зазначено вище, GRU позбавлена проблеми затухання градієнтів, оскільки модель зберігає відповідну інформацію і передає її в наступний крок.

2.3 1D згорткові нейронні мережі

Згорткові нейронні мережі (англ. Convolutional Neural Networks, CNN) завоювали популярність в галузі комп'ютерного зору завдяки їхній ефективності в розпізнаванні образів. Однак їх можливості не обмежуються лише цим, і вони активно використовуються для обробки текстових даних. При роботі з текстовими даними CNN використовують одновимірні згортки для визначення корисних признаків. Однією з ключових особливостей згорткових мереж є їхня здатність автоматично виявляти важливі локальні признаки в даних завдяки операції згортки. Це дозволяє створювати більш компактні та виразні представлення текстових даних.

В основі CNN лежать такі поняття, як згортка та пулінг. Згортка дозволяє видобути локальні признаки з вхідних даних. При обробці тексту фільтри можуть бути налаштовані на виявлення окремих слів, n-грам або навіть синтаксичних конструкцій. Пулінг використовується для зменшення розміру даних та вибору найважливіших признаків.

1D згортки часто використовуються для обробки одновимірних послідовностей, таких як текст. У контексті тексту, одновимірна згортка зазвичай призначена для виявлення патернів в послідовностях слів або символів. Для спрощення, припустимо, що у нас є вектор x довжини N, який представляє вхідну послідовність, і фільтр (ядро згортки) w довжини M.

Операція 1D згортки визначається наступним чином:

$$y[t] = \sum_{m=0}^{M-1} x[t-m] \cdot w[m]$$

де $y[t]$ – це вихідний вектор, t пробігає значення від $M-1$ до $N-1$.

Варто зауважити, що розмір вихідного вектора буде $N-M+1$, якщо не використовувати доповнення нулями (zero-padding). Доповнення нулями може бути використане для того, щоб зберегти розмір вихідного вектора таким же, як і вхідний. У цьому випадку, операція згортки з доповненням нулями виглядає так:

$$y[t] = \sum_{m=0}^{M-1} x[t - m + \frac{M-1}{2}] \cdot w[m]$$

де t пробігає значення від 0 до $N - 1$.

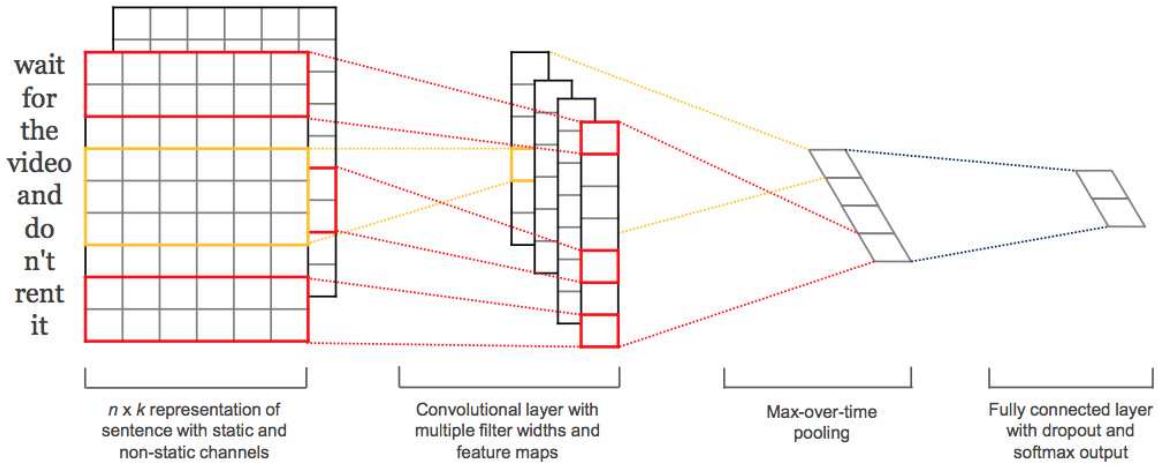


Рис. 2.9 – Робота 1D згорткової мережі для тексту

Після операції згортки, часто застосовуються активаційні функції, такі як ReLU (Rectified Linear Unit), для додавання нелінійності до моделі. Пулінгові операції також можуть бути використані після згортки для зменшення розмірності вихідного вектору, зберігаючи при цьому найбільш важливі признаки.

1D згорткові мережі можуть бути дуже ефективними в задачах, пов'язаних з текстом, як ваша дисертаційна робота, забезпечуючи швидкий та ефективний метод виявлення локальних структур і залежностей у великих текстових датасетах.

По-перше, важливо підкреслити, що 1D згортки є особливо корисними для обробки послідовностей з варіабельною довжиною. Вони дозволяють моделі фокусуватися на локальних залежностях, зменшуючи потребу в глобальних рекурентних зв'язках, які можуть бути витратними з точки зору обчислень.

Також варто зазначити, що згорткові мережі зазвичай легше інтерпретувати, ніж рекурентні мережі. Завдяки зосередженості на локальних патернах, можна візуалізувати, які саме частини вхідної послідовності активують певні нейрони. Це може бути корисно не тільки для зрозуміння, як працює модель, але і для виявлення потенційних слабких місць чи недоліків в даних.

Додатково, згорткові мережі часто виявляються менш схильними до перенавчання, особливо коли використовуються методи регуляризації, такі як dropout чи weight decay. Це робить їх відмінним вибором для задач, де доступний обмежений набір даних.

Не менш важливим є той факт, що згорткові мережі легко масштабуються. Вони можуть бути проектовані так, щоб працювати з послідовностями різної довжини без зміни архітектури, що є великою перевагою в задачах, де довжина вхідних даних може суттєво варіюватися.

Крім того, комбінація згорткових та рекурентних слоїв також може принести користь. Згорткові шари можуть відповідати за виявлення локальних патернів, тоді як рекурентні шари можуть бути використані для зберігання інформації про глобальний контекст. Це створює гібридну модель, яка може ефективно обробляти велике розмаїття задач обробки тексту.

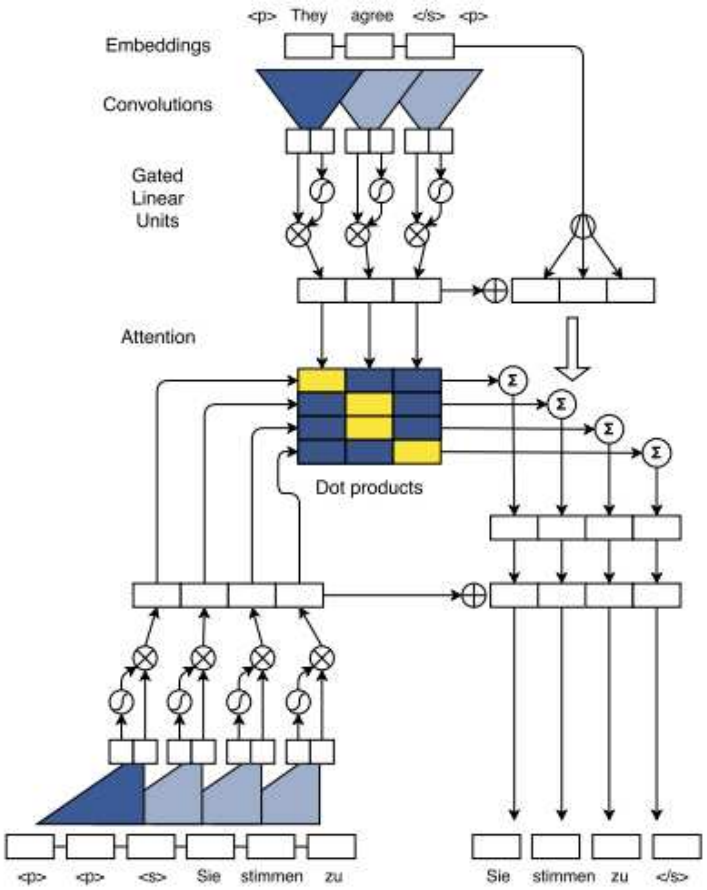


Рис. 2.10 – Архітектура повністю згорткового кондер-декодеру

У роботі Convolutional Sequence to Sequence Learning [42] робиться важливий внесок у розуміння того, як згорткові мережі можуть бути адаптовані для задач на послідовностях. В основі цієї роботи лежить концепція заміни рекурентних мереж на етапі енкодера і декодера згортковими мережами. За допомогою механізму уваги та спеціалізованих згорткових шарів, ця модель не тільки підтримує довгу пам'ять, як це роблять RNN, але й дозволяє паралелізацію обчислень, що важливо для швидкої обробки великих об'ємів тексту.

Ця модель екодер-декодер показала, що згорткові мережі можуть бути не лише ефективними, але і конкурентними за швидкістю та якістю роботи, порівняно з рекурентними мережами. Вона відкрила двері для використання згорткових мереж в ряді інших задач генерація тексту включно з умовною.

Згорткові мережі також відомі своєю ефективністю з точки зору обчислень, особливо при використанні сучасних графічних процесорів. Це може бути дуже важливим, коли мова йде про обробку великих об'ємів тексту в реальному часі.

У підсумку, згорткові нейронні мережі представляють собою потужний інструмент для обробки текстових даних. Вони не тільки дозволяють автоматично і ефективно виявляти локальні признаки в тексті, але і забезпечують високу швидкість і гнучкість, що робить їх відмінним вибором для різноманітних задач, від машинного перекладу до корекції та генерації тексту.

2.4 Архітектура трансформер

Архітектура трансформера, яка користується великою популярністю в обробці природної мови, є унікальною завдяки своєму повному зосередженню на механізмі уваги. Відмовившись від традиційних рекурентних та згорткових підходів, трансформер ефективно використовує цей механізм для виявлення складних взаємозв'язків у тексті, значно підвищуючи якість обробки мовних даних.

2.4.1 Механізм уваги

Механізм уваги (attention) [43] бере свій початок від механізмів людської уваги, які дозволяють нам фокусуватися на певних аспектах візуальних зображень або корелювати слова у мові. Людська зорова увага дозволяє нам зосередитися на певному сегменті образу з високою роздільною здатністю, в той час як навколишні регіони сприймаються в низькій роздільній здатності. Наприклад, коли ми дивимося на хвіст тварини, як зображено на рисунку 2.11, ми можемо все ще сприймати навколишній контекст, але з меншою деталізацією.

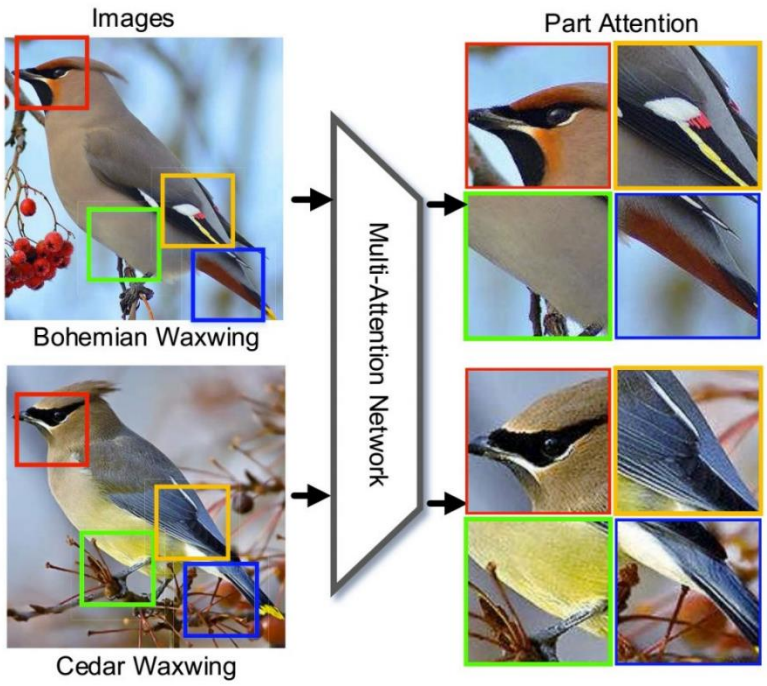


Рис. 2.11 – Увага згорткових нейронних мереж при класифікації видів пташок

Цей принцип можна застосувати до обробки мови. Наприклад, бачачи слово "eating", ми інтуїтивно очікуємо на наявність об'єкта, який споживається, і наше розуміння речення буде зосереджене на пошуку цього об'єкта. Механізм уваги в нейронних мережах моделює цю поведінку, дозволяючи моделі зосереджувати «увагу» на релевантних частинах вхідних даних, що покращує обробку і розуміння контексту, особливо в задачах обробки природної мови та комп'ютерного зору.

Рис. 2.12 – Приклад використання attention в реченні

Не обмежуючи загальності механізм attention, у глибокому навчанні, можна інтерпретувати як вектор важливості. Для того, щоб передбачити або вивести один елемент, такий як піксель у зображенні чи слово в реченні, ми оцінюємо, використовуючи attention вектор, як сильно елемент співвідноситься з іншими елементами. Зокрема, беремо суму значень елементів, зважених attention вектором, як апроксимацію.

Модель seq2seq [44] народилася в області моделювання мови. В широкому сенсі вона спрямована на перетворення вхідної послідовності (джерела) на нову (цільову) (обидві послідовності можуть мати довільну довжину). Приклади використання даного перетворення включають в себе машинне перекладання між різними мовами в текстовий або аудіо формати, підтримання діалогу запитання-відповідь або розбір речень у граматичних деревах.

Модель seq2seq зазвичай має архітектуру encoder-decoder, яка складається з наступних частин:

- Енкодер обробляє вхідну послідовність і стискає інформацію у вектор контексту (також відомий як вкладання речення або вектор «думки») фіксованої довжини. Вважається, що це представлення буде гарним сумуванням змісту всієї вихідної послідовності.
- Декодер ініціалізується вектором контексту, щоб видавати перетворений висновок. В ранніх роботах використовувався тільки останній стан мережі енкодера як початковий стан декодера (рисунок 11).

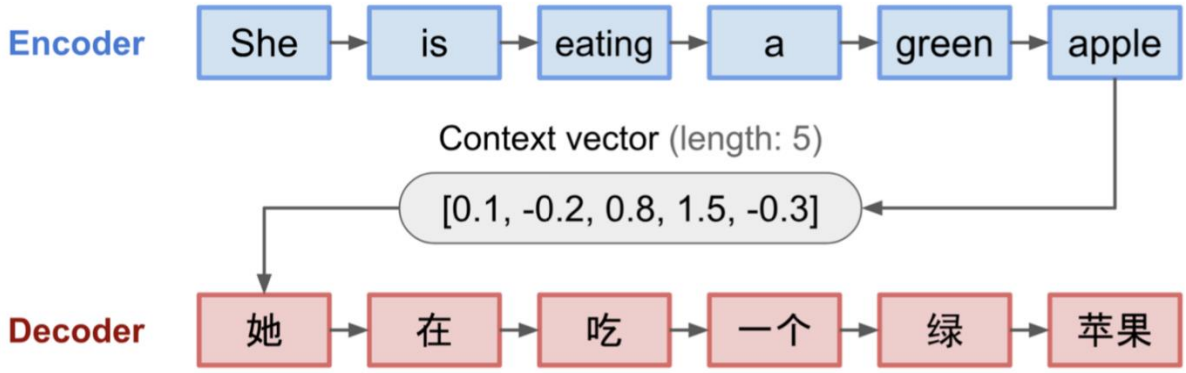


Рис. 2.13 – Схематичне зображення моделі encoder-decoder для перекладу з англійської мови на китайську

Оскільки енкодер та декодер є рекурентними нейронними мережами, то зазвичай використовують LSTM або GRU.

Основним та відомим недоліком дизайну нейронних мереж з використанням контекстного вектора фіксованої довжини є їх обмежена здатність запам'ятовувати довгі речення. Такі моделі часто забувають початкові частини інформації після обробки всього вхідного тексту, особливо в контексті машинного перекладу. Саме ця проблема і спонукала до розробки механізму уваги (attention).

Механізм уваги був спроектований, щоб поліпшити здатність моделей запам'ятовувати довгі послідовності при перекладі. На відміну від підходу, який використовує один контекстний вектор, згенерований з останнього прихованого стану енкодера, механізм уваги створює зв'язки між вектором контексту та кожним елементом вхідної послідовності. Важливість кожного з цих зв'язків визначається індивідуально для кожного елемента вхідної послідовності, завдяки чому модель може "зосередитися" на більш релевантних частинах тексту.

Ця можливість використання всієї вхідної послідовності в контекстному векторі зменшує ризик забування важливої інформації на початку тексту. Таким чином, механізм уваги ефективно підтримує зв'язок між початковою та цільовою послідовностями, поліпшуючи здатність моделі до точного перекладу та збереження контексту. По суті, вектор контексту використовує три частини інформації:

- приховані стани енкодеру;
- приховані стани декодеру;
- попередній вектор контексту.

Таким чином, модель намагається вибрати відповідні токени у вхідному реченні, враховуючи всю доступну інформацію на даний момент часу: приховані векторні представлення вхідних токенів; токени, які були вже передбачені та контекст.

$$softmax(FFN([Q;K]))$$

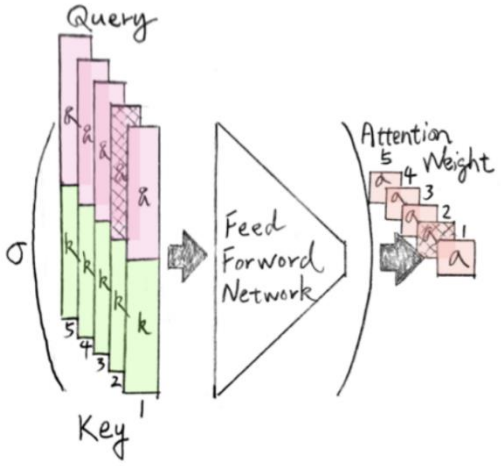


Рис. 2.14 – Адитивний механізм attention

Нехай є вхідна послідовність x довжини n й вихідна послідовність y довжини m: $x = [x_1, x_2, \dots, x_n]$, $y = [y_1, y_2, \dots, y_m]$. Нехай енкодер – це двонаправлена LSTM нейронна мережа з прямим прихованим станом h_i^f та оберненим прихованим станом h_i^b. Проста конкатенація цих обох станів векторів є станом енкодеру. Мотивація полягає в тому, щоб включити як попередні так і наступні слова в анотацію одного слова.

$$h_i = \left[h_i^{f\,T};\, h_i^{b\,T}\right]^T, i = 1, \dots, n$$

Нейронна мережа декодеру має прихований стан $s_t = f(s_{t-1}, y_{t-1}, c_t)$ для вхідного слова, який має індекс $t = 1, \dots, m$, де вектор контексту c_t – сума прихованих станів вхідної послідовності зважена механізмом attention:

$$\mathbf{c}_t = \sum_{i=1}^{n} \alpha_{t,i} \mathbf{h}_i \,,$$

$$\alpha_{t,i} = \text{align}\,(y_t, x_i) = \frac{\exp(\text{score}(s_{t-1}, \mathbf{h}_i))}{\sum_{j=1}^{n} \exp(\text{score}(s_{t-1}, \mathbf{h}_j))}.$$

Модель зв'язків присвоює t, i парі вхідних значень на позиції i та вихідного значення на позиції t (y_t, x_i), базуючись на інформації про те, як вони разом співвідносяться. Множина {t, i} є вагами уваги, що визначають скільки кожного вхідного прихованого стану слід враховувати для кожного вихідного значення. Ваги вирівнювання підбираються нейронною мережею прямого поширення з єдиним прихованим шаром (тобто звичайною лінійною моделлю) і ця мережа навчається разом з іншими частинами моделі. Таким чином, користуючись нелінійною функцією активації tanh, функцію оцінки можна записати в наступній формі:

$$\text{score}(\mathbf{s}_t,\, \mathbf{h}_i) = \mathbf{v}_\alpha^T \tanh(\mathbf{W}_\alpha[s_t;\, \mathbf{h}_i]),$$

де обидва v_α та W_α є ваговими матрицями, які слід тренувати в моделі зв'язків.

Матриця результатів вирівнювання (рис. 2.15) є гарним побічним продуктом, який явно показує кореляцію між вхідними та цільовими словами. Таким чином, ми можемо зрозуміти на які слова оригінальної (source) мови дивиться модель при написанні певних слів мовою перекладу (target).

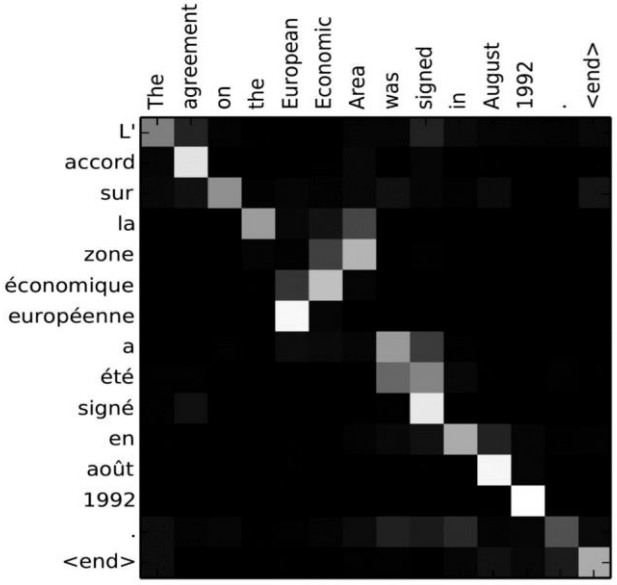


Рис. 2.15 – Матриця механізму attention для перекладу з французької на англійську

В результаті досліджень було показано, що завдяки механізму attention моделі зі збільшенням довжини речень зберігають свою якість перекладу як зображено на рисунку 2.16. Таким чином, проблема обмеженого векторного представлення у машинному перекладі була подолана завдяки механізму attention.

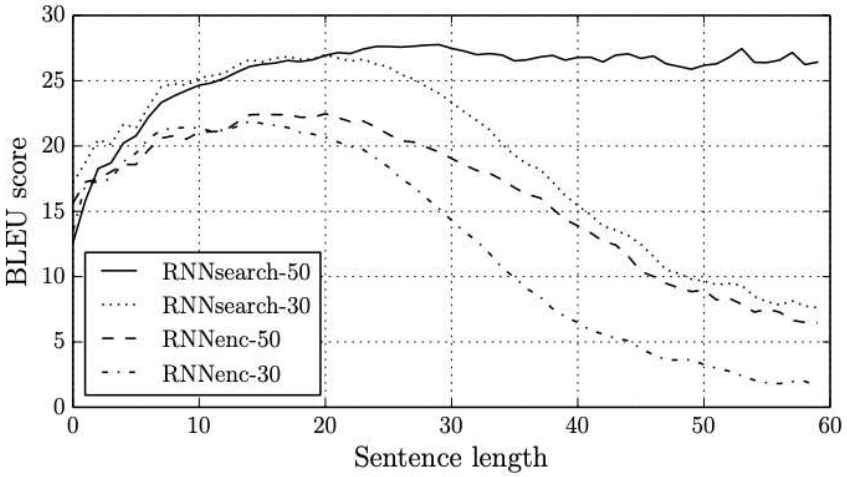


Рис. 2.16 – Залежність якості перекладу моделей від довжини речення у моделей з використанням механізму attention (RNNsearch) та без (RNNenc)

2.4.2 Attention is all you need

Архітектура трансформера, яка була запропонована у революційній роботі "Attention is all you need" [35], кардинально змінила підходи у сфері обробки природної мови (NLP), вносячи в неї інновації, які підвищили ефективність та точність моделей. Відмова від традиційної рекурентної структури на користь механізму мультиголової уваги (multi-head dot-product attention) є ключовою особливістю цієї архітектури.

Цей новаторський підхід заснований на принципі dot-product self-attention, що дозволяє моделі одночасно обробляти різні частини вхідного тексту, зосереджуючись на важливих словах та фразах без залежності від їхнього положення у реченні. У цьому контексті, механізм уваги діє як функція, що обробляє Запити (Queries) та пари Ключів (Keys) – Значень (Values), де всі ці елементи є векторами. Вихід механізму уваги розраховується як зважена сума

Значень, де ваги визначаються через функцію сумісності між Запитом і відповідним Ключем.

Механізм мультиголової (multi-head) уваги є розширенням цього принципу, де вхідна послідовність одночасно обробляється декількома "головами" уваги, кожна з яких здатна зосереджуватися на різних аспектах інформації. Це дозволяє моделі краще розуміти складні взаємозв'язки в тексті, оскільки кожна голова може сприймати різні семантичні властивості або відносини між словами.

Нехай розмірність Ключів та Запитів – d_k, а розмірність Значень – d_v. Тоді функція сумісності обчислюється як вихід функції softmax від скалярного добуток Запиту і Ключа поділеного на корінь квадратний з d_k.

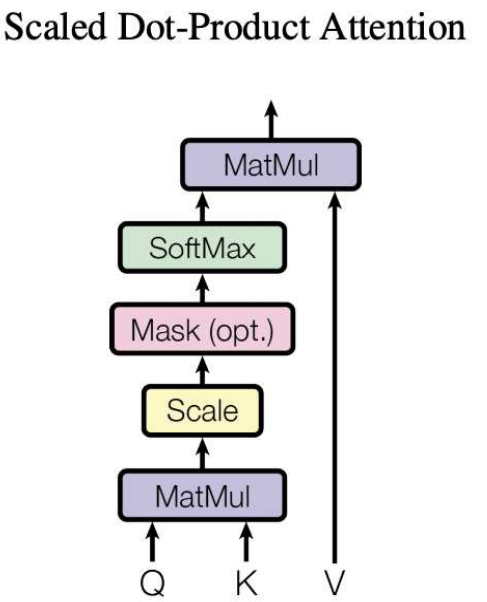


Рис. 2.17 – Схематичне поетапне зображення dot-product attention

На практиці ми можемо рахувати функцію Attention для всього набору Запитів, “запакованих” у матрицю Q одночасно. В той же час ми можемо “упакувати” Ключі та Значення у матриці K та V відповідно. Тоді результат роботи dot-product attention рахується наступним чином:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

У підході multi-head dot-product attention автори замість того, щоб рахувати функцію уваги від Запитів, Ключів, Значень з розмірностями d_{model}, пропонують

проектувати Запити, Ключі, Значення h разів різними лінійними шарами, які вивчаються під час навчання, до розмірностей d_k, d_k та d_v відповідно. Для кожної такої проекції будемо рахувати dot-product attention паралельно. Тоді на виході ми отримаємо h векторів розмірністю d_v. Далі ці вектори конкатенуються і проектуються знову до розмірності dmodel:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

де матриці проекцій W^O, W_i^Q , W_i^K, W_i^V мають розмірності $(h \cdot d_v, d_{model})$, (d_{model}, d_k), (d_{model}, d_k), (d_{model}, d_v) відповідно.

На практиці використовують паралельно $h = 8$ attention шарів (голів) й для кожного з них $d_k = d_v = \frac{d_{model}}{h} = 64$.

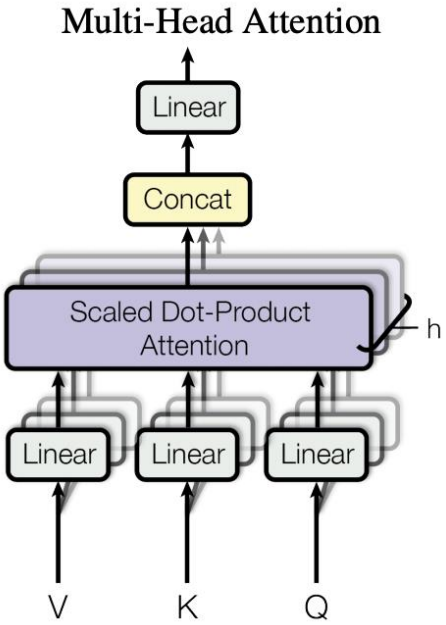


Рис. 2.18 – Схематичне зображення блоку Transformer

У transformer архітектурі multi-head dot-product self-attention використовується у 3 різні способи:

1. У encoder-decoder attention шарах Запити приходять з попередніх шарів декодеру, в той час як Ключі й Значення беруться з останнього шару енкодеру. Це дозволяє кожній позиції в декодері враховувати кожну позицію

енкодеру. Це імітує типовий механізми уваги encoder-decoder у моделях типу seq2seq, який ми розглядали у попередньому розділі.

2. Енкодер містить self-attention шари. У self-attention усі Запити, Ключі та Значення приходять з одного місця, у цьому випадку з попередніх шарів енкодеру. Кожна позиція екодера враховує всі позиції екодеру попереднього шару.
3. Схожим чином шари self-attention у декодері дозволяють всім позиціям поточного шару враховувати всі позиції попереднього шару до поточної позиції включно. Це робиться для збереження авторегресивності (на кожному кроці модель враховує тільки попередні позиції – токени) завдяки маскуванню уваги наступних позицій (визначенням їх як ∞).

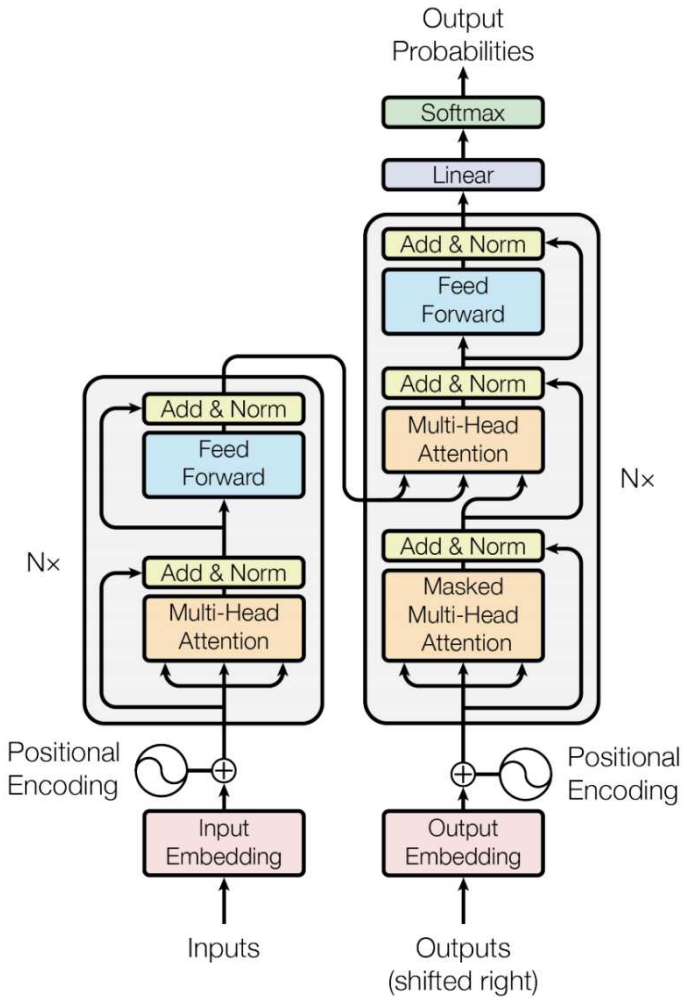


Рис. 2.19 – Повне зображення архітектури Transformer

На додачу до шарів уваги у архітектурі transformer кожен шар у енкодері та декодері також містить повнозв'язну нейронну мережу. Вона застосовується до кожної позиції окремо і паралельно та складається з двох лінійний шарів з функцією активації ReLU між ними:

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Лінійні перетворення є однаковими для всіх позицій, але в той же час різні для різних шарів. Також цей шар можна описати як згортку з розміром ядра 1. Розмірність входу і виходу мережі однакова – $d_{model} = 512$, в той час як проміжний шар вектор має розмірність $d_{ff} = 2048$.

Оскільки модель не містить ні рекурентності, ні згорток, тому для того, щоб розуміти порядок послідовності, потрібно передати інформацію про відносні та абсолютні позиції токенів в реченні. Для цього до вхідних представлень токенів додаються представлення позицій (Positional Encoding) на вході у енкодер та декодер. Представлення позицій мають таку ж розмірність, як і представлення токенів d_{model} для того, щоб їх можна було поелементно додати.

В даній роботі для представленнь позицій використовуються як синусні, так і косинусні функції різної частоти:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

де pos - це позиція, а i – це розмірність вектора представлення. Такі функції були вибрані через гіпотезу, що вони дозволяють враховувати відносні позиції, оскільки для будь-якого фіксованого відступу k, PE_{pos+k} може бути отриманий як лінійна функція від PE_{pos} .

Глобально, енкодер складається з $N = 6$ однакових шарів. У кожному шарі є два підшари. Перший – це multi-head dot-product self-attention, а другий – повнозв'язна нейронна мережа, що застосовується для кожної позиції окремо. Також між кожними двома підшарами застосовується residual connection разом з layer normalization. Таким чином, вихід кожного шару – це $LayerNorm(x + Sublayer(x))$, де $Sublayer$ – поточний підшар.

Аналогічно декодер складається з $N = 6$ однакових шарів. На додачу до двох підшарів, присутніх у екодері, декодер також має третій підшар, encoder-decoder attention. І також застосовується residual connection разом з layer normalization.

2.5 Великі мовні моделі

Великі мовні моделі зіграли важливу роль в області обробки природної мови. За свою архітектуру вони використовують трансформери. Основна ідея трансформера — це механізм уваги, який дозволяє моделі фокусуватися на різних частинах вхідної послідовності при генерації виходу.

GPT-3 (Generative Pre-trained Transformer 3) [45] вважається першим відомим прикладом великої мовної моделі. Вона є третьою ітерацією моделі GPT від OpenAI. Вона привернула велику увагу завдяки своїм неймовірно великим розміром та здатністю виконувати широкий спектр завдань без спеціалізованого доналаштування.

Великі мовні моделі, які базуються на архітектурі GPT, стали настільки потужними, що здатні виконувати не тільки задачі класифікації тексту, але і більш складні задачі, такі як написання коду, створення поетичних творів, анотація наукових статей, і навіть побудова складних логічних міркувань.

Існує набір причин, чому ця модель стала революційною для сфери NLP та AI у цілому:

1. GPT-3 є однією з найбільших нейронних мереж, які коли-небудь створювалися, із 175 мільярдами параметрів. Для порівняння, GPT-2 мала 1,5 мільярда параметрів. Цей великий розмір дозволяє моделі зберігати значну кількість інформації та використовувати її для генерації тексту.
2. Оскільки GPT-3 була навчена на великих об'ємах текстових даних, вона "знає" багато фактів про світ, включаючи історичну, наукову та культурну інформацію. Проте слід зазначити, що її "знання" є статичними і базуються на даних до моменту завершення навчання.

3. Однією з найбільш вражаючих особливостей GPT-3 є її спроможність виконувати завдання "за один раз" (zero-shot), коли модель отримує інструкцію без конкретних прикладів, "за декілька разів" (few-shot), коли моделі надаються декілька прикладів, і "за багато разів" (many-shot), коли моделі надаються багато прикладів.
4. GPT-3 показала високу ефективність у багатьох застосуваннях: переклад мови, відповіді на питання, написання поезії, створення програмного коду, і так далі. Ця загальність є результатом її великого розміру та способу навчання.
5. Незважаючи на свою потужність, GPT-3 може допускати помилки або генерувати інформацію, яка може бути неточною, упередженою або небезпечною. Це підкреслює необхідність обережного підходу при використанні таких моделей.

ChatGPT є однією з ітерацій моделі GPT, яка була оптимізована специфічно для діалогових систем. Вона здатна вести змістовний діалог, враховуючи контекст попередніх повідомлень, щоб генерувати більш відповідні відповіді.

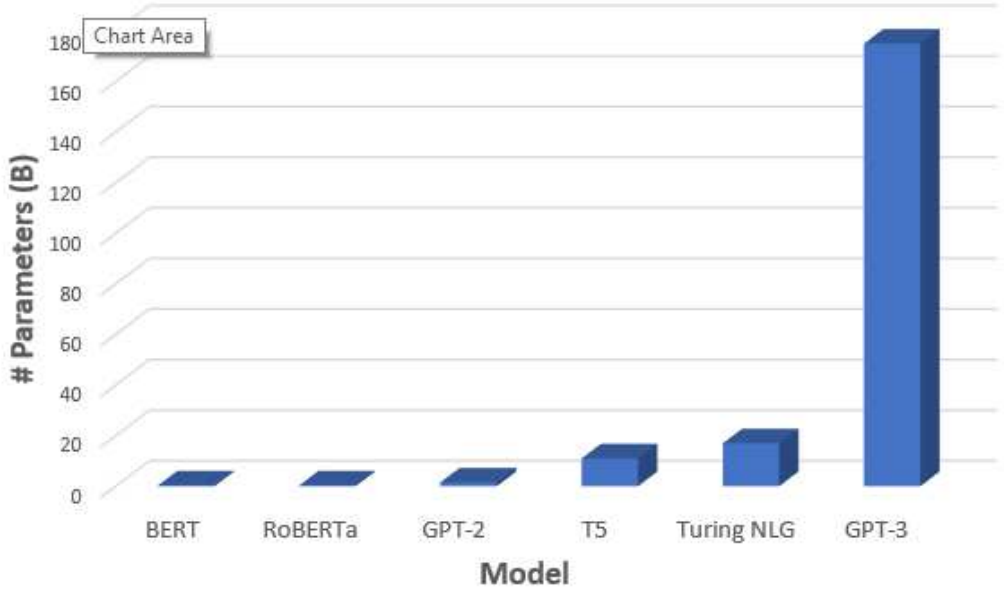


Рис. 2.20 – Порівняння кількості параметрів GPT-3 з найбільшими на той момент NLP моделями

Основним внеском у створенні ChatGPT є методика навчання, яка використовує поєднання кількох стратегій. Перша частина – це стандартне навчання з учителем, де модель навчається передбачати наступне слово в тексті, використовуючи великі набори даних. Друга частина включає в себе метод навчання з підкріпленням на основі людського зворотного зв'язку (англ. Reinforcement Learning from Human Feedback, RLHF), де модель дооптимізовується за допомогою зворотного зв'язку від людей.

RLHF – це методика, де модель проходить процес тонкого налаштування на основі зворотного зв'язку від людей. Замість того, щоб повністю покладатися на традиційний метод навчання з учителем, RLHF дозволяє використовувати дійсний зворотний зв'язок від людських користувачів, щоб покращити роботу моделі в реальних сценаріях. Це включає в себе виявлення і виправлення помилок, які модель може робити, а також адаптацію до більш специфічних або незвичних запитів.

RLHF стоїть на основі сучасних підходів до навчання мовних моделей, оскільки він дозволяє моделям навчатися безпосередньо від зворотного зв'язку людей, замість того, щоб покладатися виключно на великі набори даних.

1. **Процес збору даних:** Початковий набір даних створюється за допомогою взаємодії моделі з користувачами. Люди взаємодіють з моделлю, задаючи їй питання або даючи інструкції, а модель відповідає.
2. **Оцінка зворотного зв'язку:** Отримані відповіді моделі потім оцінюються іншими людьми. Оцінювачам може бути представлено декілька варіантів відповідей, і їм потрібно вибрати найкращий.
3. **Оптимізація за допомогою навчання з підкрілпенням:** З використанням оцінок з попереднього кроку модель навчається вибирати найкращі відповіді. Це досягається за допомогою методів навчання з підсиленням, де відповіді, які отримали вищий рейтинг, надають моделі вищий "нагороду".

Формально це можна представити наступним чином. Припустимо, a_t - це дія, яку модель вибирає на кроці t, а r_t – це відповідний нагороду, який модель отримує за цю дію. Мета моделі полягає в максимізації очікуваного нагороду:

$$\mathbb{E}\left[\sum_{t=1}^{T} r_t\right],$$

де T – це кількість кроків взаємодії. Алгоритми reinforcement learning, такі як Proximal Policy Optimization (PPO), можуть бути використані для оптимізації моделі з метою максимізації цього нагороду.

Proximal Policy Optimization (PPO) є однією з ключових технік в області навчання з підсиленням і часто використовується для тренування моделей в задачах з великим простором дій, таких як мовні моделі. Основна ідея PPO полягає в тому, щоб виконувати оптимізацію політики (стратегії) моделі, обмежуючи зміни в політиці від одного кроку оптимізації до іншого, щоб запобігти нестабільності в навчанні.

Визначимо відношення між новою та старою політикою як:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$$

Тоді основний об'єктив PPO може бути записаний як:

$$L^{PP0}(\theta) = \mathbb{E}_{s,a\sim\pi_{\theta_{old}}}[\min\left(r_t(\theta)A_{\theta_{old}}(s,a), clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_{\theta_{old}}(s,a)\right]$$

де $\pi_\theta(a|s)$ – ймовірність дії a в стані s за політикою з параметрами θ, $A_{\theta_{old}}(s,a)$ – функція переваги дії a в стані s за старою політикою, ϵ – гіперпараметр, який контролює, наскільки сильно нова політика може відрізнятися від старої на кожному кроці оптимізації.

Основний компонент цього об'єктиву – це відношення між новою та старою політикою, яке обмежується функцією "clip", щоб гарантувати, що зміни в політиці залишаються пропорційними і не призводять до нестабільності. Завдяки цьому PPO є ефективним та стабільним методом навчання з підсиленням, який можна використовувати для тренування великих мовних моделей, таких як ChatGPT.

Вперше RLHF у сучасній формі був описаний у InstructGPT, що є варіантом моделі GPT, який був створений для виконання конкретних інструкцій в тексті. Замість того, щоб просто генерувати текст на основі поданих запитів, InstructGPT була навчена виконувати дії відповідно до конкретних інструкцій, що робить її особливо корисною для задач, які потребують конкретних дій чи відповідей.

Цей підхід базується на тому ж методі RLHF, що й ChatGPT, але з акцентом на виконання конкретних інструкцій. Робота InstructGPT детально розглядає методику навчання, результати та можливі застосування цього підходу.

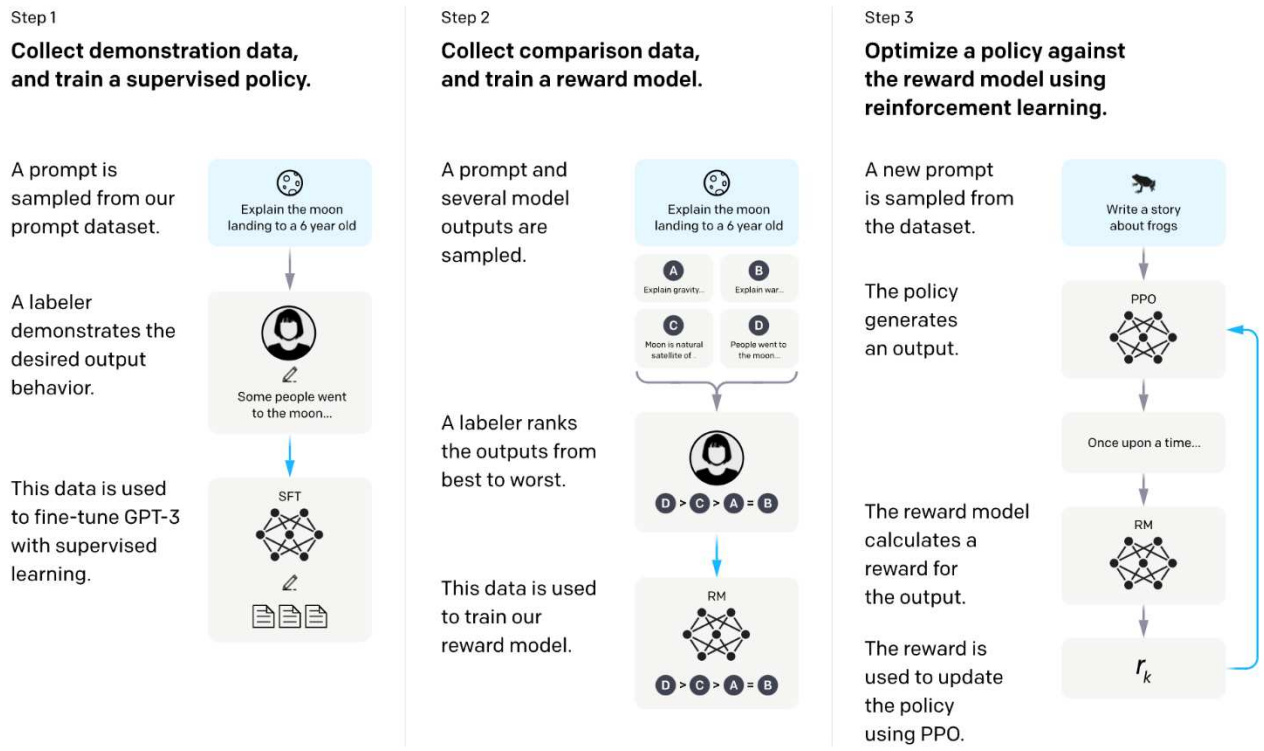


Рис. 2.21 – Повний процес тренування InstructGPT, включаючи метод навчання з підкріпленням на основі людського зворотного зв'язку

У підсумку, як ChatGPT, так і InstructGPT представляють собою значущі кроки в розвитку мовних моделей, і обидві використовують RLHF для дооптимізації на основі зворотного зв'язку від людей, що підвищує їх ефективність у реальних сценаріях.

Вплив великих мовних моделей на сучасне NLP та AI вцілому важко переоцінити. Їхня здатність до швидкого адаптування до нових задач, масштабування та виконання завдань на рівні, який раніше був недосяжним для машин, революціонує сферу технологій. Вони вже демонструють не тільки вражаючі результати в академічних завданнях, але й активно втілюються в комерційних продуктах, надаючи користувачам високоякісні інструменти для комунікації, аналітики та створення контенту. Проте, разом з потенційними перевагами, вони несуть і виклики, такі як упередження, проблеми безпеки та етики. Для збереження прогресу та побудови довірливого майбутнього в сфері AI, необхідний баланс між інноваціями, етичним підходом та відповідальністю дослідників та розробників. Таким чином, доки ми рухаємося вперед, важливо пам'ятати про відповідальне та збалансоване використання цих потужних інструментів.

РОЗДІЛ 3. ПОБУДОВА НЕЙРОННИЇ МЕРЕЖІ ДЛЯ ЗАДАЧ ПИСЬМОВОГО АСИСТЕНТУ

Побудова єдиної нейронної мережі вирішення задач письмового включає велику кількість кроків. Задля об’єктивності роботи та досягнення високої якості необхідно покрити різноманітні аспекти, класичні для розробки моделей глибокого навчання. Серед них:

- побудова процедури оцінювання якості роботи;
- робота з даними (підготовка нових та обробка існуючих);
- тренування спеціалізованих моделей та вибір архітектури;
- побудова універсальної нейронної мережі та підбір гіперпараметрів;
- оптимізація швидкості роботи моделі.

3.1 Інтегрований підхід до оцінювання якості роботи письмового асистента

Розробка нейронних мереж, здатних виконувати складні задачі, такі як виправлення граматичних та орфографічних помилок, перефразування та спрощення тексту, є нетривіальною задачею. Основним завданням при розробці таких систем є не тільки створення ефективних алгоритмів, але й побудова надійних механізмів оцінювання їхньої якості. Це важливо, оскільки якість та надійність роботи нейронних мереж прямо впливає на їх придатність до використання в реальних умовах.

Надійне оцінювання якості нейронних мереж вимагає розробки комплексних метрик, які здатні адекватно оцінювати якість генерованого тексту. Це включає не тільки перевірку граматичної коректності, але й оцінку збереження смислу, стилістичної відповідності та навіть творчості тексту. Виклик полягає у тому, що ці аспекти часто суб'єктивні та важко формалізувати, що вимагає інноваційного підходу до створення таких метрик.

Також стійка система оцінювання відіграє ключову роль у процесі ітеративного вдосконалення нейронних мереж. Вона дозволяє визначати ефективність внесених змін, відстежувати прогрес у часі та виявляти нові

можливості для покращення. Особливо це важливо в умовах, коли мережі виконують складні та багатогранні задачі, такі як генерація та обробка тексту. Таким чином, розвиток надійних методів оцінювання є невід'ємною частиною процесу створення високоякісних та ефективних нейронних мереж.

3.1.1 Метод комбінованої оцінки для якості роботи на задачах умовної генерації тексту

Найбільш прямолінійним і зрозумілим підходом є побудова оцінки на базі існуючих метрик для задач умовної генерації тексту. Ми розглянули такі методи у розділі 2. Нагадаємо, які метрики використовуються для яких задач з нашого списку.

Таблиця 3.1 – Метрики для оцінки якості умовної генерації тексту у розрізі задач письмового асистенту

Метрика	Виправлення граматичних помилок (GEC)	Спрощення тексту	Перефразування тексту
BLEU		X	X
ROUGE			X
METEOR			X
SARI		X	X
FKGL		X	
MaxMatch	X		
TER			X
ERRANT	X		
BERT-score		X	

Як бачимо, існує доволі велика кількість метрик оцінювання. Деякі з них перетинаються у декількох задачах, деякі є унікальними (специфічними) для певних задач. У цьому плані виділяється виправлення граматичних та

орфографічних помилок, де MaxMatch та похідна ERRANT використовується ексклюзивно.

Оцінка задач генерації тексту є доволі популярною і водночас складною темою. Досі не існує надійного методу для жодної задач, а тому метрики в більшості випадках виступають опосередкованим способом апроксимацію людського оцінювання. Деякі метрики, що активно використовуються у сучасних роботах, працюють доволі погано і використовуються «по інерції» задля порівняння з вже існуючими працями. Тому існує окремий напрямок для оцінки метрик.

Так у роботі [46] стверджується, що BLEU є поганою метрикою для оцінки якості задач перефразування та спрощення тексту. Для аргументації, наводиться приклад, коли не переписувати оригінальний текст дає кращу метрику BLEU у порівнянні з спробами високоякісної моделі, високо оціненої людьми-анотаторами у контрасті з автоматичною метрикою.

Також ставиться під сумнів метрика FKGL, яка не використовує переписування розмічені людьми, а лише користується оригінальним реченням та згенерованим системою.

Існує аналіз [10] який запевняє, що для системи виправлення помилок часто оцінюються лише за загальною продуктивністю, оскільки гіпотези системи не анотуються у метриці MaxMatch. Проте це може вводить в оману, і система, яка загалом працює погано, насправді може перевершувати інші системи за певними типами помилок. Це важливо, оскільки надійна спеціалізована система насправді є бажанішою, ніж посередня загальна система. Проте без аналізу типів помилок ця інформація залишається повністю невідомою. Саме тому рекомендується застосовувати ERRANT, який враховує цю проблему.

Проаналізувавши подібну літературу і врахувавши бажання мати можливість робити порівняльний аналіз, було прийняти рішення обрати по найбільш надійній метриці для кожної системи.

METEOR, вважається однією з найнадійніших метрик для оцінювання задач перефразування тексту, завдяки своєму унікальному підходу до аналізу

відповідності між оригіналом та перефразованим текстом. На відміну від інших метрик, таких як BLEU, TER, METEOR враховує не тільки точність відповідності слів, але й семантичну згуртованість фраз. Вона включає синонімію та гнучке відповідання слів, що дозволяє більш точно оцінити, наскільки добре перефразований текст передає смисл оригіналу.

SARI є однією з найбільш надійних метрик для оцінювання задачі спрощення тексту. Основна перевага SARI полягає в її здатності оцінювати якість спрощення тексту, порівнюючи його не тільки з розміченими спрощеними варіантами, але й з оригінальним текстом. Це дозволяє вимірювати, наскільки ефективно система зберігає первісний зміст, при цьому роблячи його більш доступним і зрозумілим.

SARI аналізує три аспекти спрощення: додавання, вилучення та заміна слів. Це дає більш всебічне розуміння процесу спрощення, оскільки оцінює, наскільки адекватно система використовує кожен з цих методів. Наприклад, метрика оцінює не тільки чи були вилучені складні слова, але й чи були вони замінені більш зрозумілими синонімами, а також чи були додані додаткові пояснення там, де це необхідно.

ERRANT, метрика для оцінювання виправлення граматичних помилок, вважається однією з найнадійніших у своїй сфері. Її особливість полягає у вмінні детально категоризувати помилки, роблячи аналіз глибшим і більш змістовним. ERRANT не просто фіксує факт виправлення, але й аналізує його тип, що дозволяє детально розуміти, які саме аспекти граматики краще опрацьовані в системі. Це особливо важливо, оскільки дозволяє розробникам зосередитись на специфічних областях для покращення, а також надає більш глибоке розуміння про те, як різні типи помилок впливають на загальну якість тексту. Такий підхід переводить оцінювання виправлення помилок з кількісного на якісний рівень, надаючи засоби для точнішої калібрування та удосконалення систем виправлення помилок.

Таким чином маємо: ERRANT для задачі виправлення граматичних та орфографічних помилок, SARI для задачі спрощення тексту, METEOR для задачі перефразування тексту. Як згадано вище, оскільки модель має бути універсальною,

потрібна єдина метрика оцінки для ітеративного вдосконалення нейронної мережі. Прямолінійним рішенням було б використання простого середнього:

$$AssistantScore = \frac{1}{3}ERRANT + \frac{1}{3}SARI + \frac{1}{3}METEOR$$

Такий підхід не враховує двох нюансів:

- масштаб метрик – покращення на 1 одиницю однієї метрики з точки зору однієї задачі не еквівалентно покращенню на ту саму одинцю іншої. Це може призвести до того, що для оптимізації середнього буде краще сконцентруватися на тих задачах, де одиниця метрики зростає найбільше;
- важливість задач для письмового асистенту – суб'єктивно задачі виправлення помилок, спрощення та перефразування тексту можуть бути не рівноцінними для середньостатистичного користувача.

Почнемо з важливості задач. Під час написання дисертаційної роботи не вдалося знайти наукових статей, які опитують користувачів на предмет головних (найважливіших) якостей письмового асистенту. Найкраще що вдалося зробити, врахувати кількість запитів writing assistant у комбінаціями з назвами задач. Це є досить суб'єктивним підходом, але в той же час отримані результати добре узгоджують з уявленням авторів про задачі.

Для того щоб коефіцієнти важливості задач мали обмежений ефект було прийняте рішення зробити їх у діапазоні [1; 2]. У такій конфігурації найбільша різниця між задачами буде 2 рази, що має гарантувати не виродження комбінованої метрики у одну з вхідних.

Отже, задачі отримали такі коефіцієнти важливості:

- **2** для задачі виправлення граматичних і орфографічних помилок – виправлення помилок є ключовим для забезпечення читабельності та професійності тексту, особливо в академічних, ділових та офіційних контекстах;
- **1.4** для перефразування тексту – задача є важливою для забезпечення різноманітності виразу і погляду зі «сторони», а також зробити текст більш

зрозумілим та привабливим для читачів; у той же час задача не є критично важливою для письмового асистенту;

- **1** для задачі спрощення тексту – спрощення тексту важливе для забезпечення його доступності широкій аудиторії, однак ця задача може бути менш критичною, ніж виправлення помилок або перефразування.

Наступним є нормалізація метрик. Задля визначення коефіцієнтів було вирішено розглянути 90% систем презентованих для задач за останні 5 років. Щоб уникнути аномалій і зберегти правильний масштаб, 10% найгірших результатів було не враховано у цій роботі.

Для метрики ERRANT на текстовій вибірці W&I+LOCNESS задачі виправлення граматичних та орфографічних помилок найкраща система це ESC [47], яка є ансамблем декількох систем має оцінку 79.9, у той час як прямолінійний підхід з мережею архітектури трансформер у 2019 році мав оцінку 69.5, що більш ніж на 10 одиниць нижче.

Щодо метрики SARI на текстовій вибірці Turk Corpus для спрощення тексту, досі найкращою моделлю є модель MUSS, яка має оцінку 42.5 одиниць. Схожим до baseline підходу для GEC є натренований трансформер у роботі 2018 року [48]; він отримав 33.9 одиниць, що говорить метрика SARI є менш чутливою і має менший «розкид» у порівнянні з задачею виправлення помилок.

Для задачі перефразування ми використовуємо набір даних MSCOCO з метрикою METEOR. Існує певна кількість нюансів пов’язаних з використанням версій та вибірок MSCOCO, тому варто обирати роботи що згадують фіксовану конфігурацію. Таким чином найкращою моделлю є попередньо натренований BART з роботи Task-specific pre-training improves models for paraphrase generation [49], який має 26.2 одиниць. Робота простим натренованим трансформером 2020 року має 18.5. Таким чином дельта є ще меншою у порівнянні з метриками наборів даних попередніх задач.

Таким чином, якщо ми візьмемо коефіцієнт задачі виправлення граматичних помилок за 1, то задача спрощення тексту матиме $\frac{10.4}{8.6} = 1.2$, а задача перефразування $\frac{10.4}{7.7} = 1.35$.

Після аналізу літератури кожної з задач та статей по письмовому асистенту, якщо врахувати це в комбінації з коефіцієнтами важливості задач, маємо наступну картину для комбінованої метрики.

$$AssistantScore = 2 \cdot ERRANT + 1 \cdot 1.35 \cdot SARI + 1.4 \cdot 1.2 \cdot METEOR$$

$$AssistantScore = 2 \cdot ERRANT + 1.35 \cdot SARI + 1.68 \cdot METEOR$$

Отже, побудовано метод комбінованої оцінки якості систем на задачах письмового асистенту.

3.1.2 Використання якірної системи попарних порівнянь за допомогою великих мовних моделей

Іншою потенційною метрикою, яку можна використати для аналізу систем - письмових асистентів є якірної системи попарних порівнянь за допомогою великих мовних моделей. Цей підхід є дуже новим і на момент написання роботи тільки набуває популярності.

В останні роки великі мовні моделі (LLM) знайшли своє місце не тільки як інструменти для генерації та обробки тексту, але й як ключовий елемент в оцінці якості інших моделей машинного навчання. Цей підхід виходить за рамки традиційних методів оцінювання, де використовуються стандартні набори даних і метрики. Великі мовні моделі дозволяють аналізувати якість виходів інших систем на більш глибокому рівні, враховуючи такі фактори, як контекст, семантика та навіть творчість тексту.

Одним з інноваційних методів, який набирає популярності в цьому контексті, є попарне порівняння систем. Воно полягає в створенні ситуацій, де дві різні системи представляють свої виходи на одне й те саме завдання, і ці виходи потім оцінюються порівняно один з одним. Такий підхід дозволяє не тільки оцінити

ефективність кожної системи окремо, але й виявити їхні відносні сильні та слабкі сторони.

Центральною частиною методу попарного порівняння є використання "якірної" системи. Ця система служить як стандарт або відправна точка для оцінки інших систем. Зазвичай вибір якірної системи базується на її репутації, наявності відповідних даних для порівняння, або її визнаної ефективності в певній області. Використання такої системи як орієнтир дозволяє створити об'єктивну основу для порівняння і допомагає уникнути суб'єктивізму в оцінці.

Для ілюстрації, розглянемо застосування цього методу в контексті оцінки задач письмового асистента. Припустимо, що основні функції такого асистента включають виправлення граматичних помилок, перефразування та спрощення тексту. Ми можемо обрати якірну систему, яка добре справляється, наприклад, з виправленням помилок, і порівнювати її ефективність з іншими системами, що спеціалізуються на перефразуванні або спрощенні тексту.

У такому порівнянні ми могли б використовувати різні набори даних, що відображають різні вимоги до якості тексту. Наприклад, для оцінки виправлення помилок ми могли б використовувати академічні тексти з певною кількістю заздалегідь введених помилок, тоді як для перефразування могли б використовувати тексти з літератури або журналістські статті. Це дозволить не тільки оцінити, наскільки добре кожна система справляється зі своїм завданням, але й зрозуміти, як вони порівнюються одна з одною в різних контекстах.

Також важливо розглянути гнучкість кожної системи. У контексті письмового асистента, це може означати оцінку того, наскільки легко система може бути адаптована до різних стилів письма або до специфічних вимог користувачів. Наприклад, система, яка чудово справляється з академічними текстами, може не бути такою ефективною для літературних або неформальних стилів. Це порівняння допоможе визначити, які системи найкраще підходять для різноманітних сценаріїв застосування.

Головним кандидатом у цій роботі на роль великої мовної моделі, що оцінює якість переписування, є ChatGPT-3.5. Цей вибір зумовлений унікальними

можливостями ChatGPT-3.5, що включають високу здатність до розуміння контексту та генерації відповідного тексту. Його глибоке навчання на широкому спектрі даних дозволяє ефективно оцінювати варіативність та якість перефразування, виправлення помилок та спрощення тексту. ChatGPT-3.5 може не тільки виявляти граматичні та синтаксичні помилки, але й оцінювати змістовність та стилістичну відповідність тексту, що робить його надзвичайно цінним інструментом в процесі оцінювання.

При оцінці якості перефразування, ChatGPT-3.5 може аналізувати не тільки лексичну відповідність між оригінальним текстом та його перефразованим варіантом, але й глибше розуміти структурну та семантичну згуртованість. Це означає, що модель здатна розглядати, чи зберігається первісний сенс, чи використовуються адекватні синоніми, та чи є зміни в тексті відповідними до контексту.

Що стосується виправлення помилок, ChatGPT-3.5 має можливість виявляти та виправляти широкий спектр помилок, від простих орфографічних до складних граматичних структур. Це робить його ідеальним інструментом для оцінки якості роботи інших систем, які спеціалізуються на граматичному виправленні.

Коли мова йде про спрощення тексту, ChatGPT-3.5 може оцінити, наскільки ефективно текст було спрощено, не втрачаючи при цьому ключового змісту та суті повідомлення. Модель може перевірити, чи були складні фрази та терміни замінені більш доступними аналогами, а також чи збережена логічна послідовність ідей.

Далі, важливою частиною дослідження є порівняння результатів, отриманих ChatGPT-3.5, з виходами інших моделей або систем. Це дозволяє не тільки визначити рівень компетентності ChatGPT-3.5 у контексті кожної конкретної задачі, але й зрозуміти, як різні підходи впливають на кінцевий результат.

Згідно з існуючими дослідженнями, під час використання ChatGPT для оцінювання якості роботи інших систем можуть виникати певні обмеження. Однією з основних проблем є те, що ChatGPT може працювати субоптимально при прямому виставленні оцінок. Це пов'язано з його здатністю до генерації тексту на

основі величезного масиву даних, але не завжди з можливістю точно оцінити нюанси чи якість цих виходів з точки зору сторонньої системи.

У зв'язку з цим, попарне порівняння виступає як значно надійніший метод. При попарному порівнянні дві системи оцінюються на основі їхніх відповідей на одне й те саме запитання чи завдання, дозволяючи безпосередньо порівняти їхню продуктивність. Такий підхід дозволяє уникнути потенційних проблем суб'єктивності, які можуть виникнути при використанні однієї системи для оцінювання іншої.

Додатково, щоб уникнути позиційного упередження, важливо випадково визначати позиції для кожного кандидата в процесі оцінювання. Позиційний упередження може виникнути, якщо одна система завжди оцінюється першою, оскільки це може несвідомо впливати на оцінку якості. Випадкове розміщення кандидатів гарантує, що кожна система має рівні шанси бути оціненою першою або останньою, що зменшує ризик позиційного упередження.

Цей підхід дозволяє більш точно визначити сильні та слабкі сторони кожної системи в контексті конкретних задач. Наприклад, одна система може бути більш ефективною у виправленні граматичних помилок, тоді як інша може краще справлятися з перефразуванням або спрощенням тексту. Використання попарного порівняння дає можливість об'єктивно оцінити ці аспекти та виявити, яка система краще виконує певні завдання.

Для попарного порівняння систем з використанням системи-якоря (нехай це буде система Y), ви можете використовувати наступну формулу для порівняння двох систем (нехай це будуть система 1 S_1 і система 2 S_2). Основна ідея полягає в тому, щоб порівняти, як часто кожна система "перемагає" або "програє" в порівнянні з системою-anchor.

1. **Порахувати кількість перемог проти якоря:** Для кожної системи S_1 і S_2, необхідно порахувати кількість випадків, коли система показала кращі результати, ніж anchor система Y. Це можна виразити як $W(S_1, Y)$ і $W(S_2, Y)$, де W означає кількість перемог.

2. **Порахувати кількість поразок проти якоря:** Аналогічно, потрібно порахувати кількість випадків, коли кожна система показала гірші результати, ніж система-якір. Це можна виразити $L(S_1, Y)$ і $L(S_2, Y)$, де L означає кількість поразок.
3. **Розрахунок загального рейтингу:** Далі необхідно обчислити рейтинг кожної системи відносно якоря, використовуючи формулу:

$$R(S) = \frac{W(S,Y)}{W(S,Y) + L(S,Y)}$$

де $R(S)$ - це рейтинг системи S відносно якоря.

4. **Порівняння рейтингів:** Далі стає можливо порівняти рейтинги $R(S_1)$ і $R(S_2)$. Система з вищим рейтингом вважається більш ефективною у порівнянні з якірною системою.
5. **Статистичний аналіз:** Залежно від кількості даних, можна також застосувати статистичні методи для перевірки значущості різниці між рейтингами.

Цей метод дозволяє не тільки визначити, яка система краще працює у порівнянні з якірною системою, але й надає кількісну оцінку цієї ефективності. Такий підхід може бути особливо корисним, коли ви хочете порівняти системи, які виконують подібні завдання, але розроблені за різними принципами чи технологіями.

Для врахування всіх трьох задач письмового асистенту, необхідно проганяти вхідний текст через кожну з них і далі просити ChatGPT зробити оцінку кожної з переписувань проти якоря, передаючи паралельно опис задачі, аби модель знала на що звертати увагу.

Таким чином, хоча ChatGPT-3.5 є перш за все потужним інструментом для генерації тексту, використання якірного попарного порівняння з випадковим визначенням позицій кандидатів є часто більш надійним методом для оцінки якості

різних систем письмового асистента у порівнянні з існуючими автоматичними метриками на фіксованих наборах даних. Особливо це стосується того, коли мова йде про точність та об'єктивність оцінки.

3.2 Генерація синтетичних виправлень великими мовними моделями для подальшої дистиляції

Наступним кроком у побудові письмового асистенту є видобування якісних тренувальних даних задля того щоб модель, представлена у роботі, мала високий рівень якості згенерованих переписувань.

Генерація синтетичних даних за допомогою великих мовних моделей, таких як ChatGPT-3.5, відкриває нові можливості для досліджень у сфері штучного інтелекту та обробки природної мови. Ця техніка дозволяє створювати великі набори даних, які можуть бути використані для тренування та вдосконалення моделей машинного навчання (дистиляція даних), зокрема в контексті письмових асистентів.

У цій дисертаційній роботі використано ChatGPT-3.5 для генерації переписувань окремо для трьох основних задачах: виправленні граматичних та орфографічних помилок, спрощенні тексту та перефразуванні. Як вхідні дані було використано існуючі набори даних, що забезпечує важливі умови:

- текст для задачі виправлення помилок має описки і відповідно може бути суттєво виправлений;
- текст для задачі спрощення тексту містять складні академічні або технічні терміні і можуть бути відповідно спрощені.

Такий підхід дозволяє не тільки швидко створити великий обсяг релевантних даних, але й забезпечити їх різноманітність та складність.

Аналогічно, лишається можливість і генерувати текст також за допомоги великої мовної мережі. Для виправлення граматичних та орфографічних помилок можна використовувати ChatGPT-3.5 для створення текстів з помилками. Вхідний текст (prompt) може виглядати так: "Write a short paragraph on the topic of ecology,

including specific grammatical errors". Такий текст потім можна використовувати як вхідні дані для системи, яка спеціалізується на виправленні помилок.

У задачі спрощення тексту ChatGPT-3.5 може генерувати складні абзаци, які потім потребують спрощення. Prompt може бути таким: "Create a detailed description of quantum physics for experts in the field", а потім цей текст може бути використаний для створення спрощеної версії.

Втім, у рамках дисертаційної роботи було використано перший підхід, де модель генерувала лише переписування, а як вхідні речення використовувалися існуючі великі набори задач, згенеровані напівавтоматично. Вибір такого підходу, де модель ChatGPT-3.5 використовується для генерації лише переписувань, має певні переваги.

По-перше, це дозволяє забезпечити високу якість вхідних даних, оскільки вони базуються на реальних і вже існуючих текстах. По-друге, це сприяє створенню різноманітних і складних сценаріїв, які можуть бути використані для глибшого тренування і вдосконалення моделі. Наприклад, вхідні дані для задачі виправлення помилок можуть містити текст з різними рівнями складності і різними типами помилок, що дає можливість моделі адаптуватися до широкого спектру випадків використання.

Важливим аспектом використання синтетичних даних є їхнє подальше перевірення та валідація. Хоча ChatGPT-3.5 є потужною моделлю, вона все ж може генерувати вихідні дані, що містять неточності або неідеальні переписування. Тому, в ідеальному сценарії, важливо включати крок перевірки якості синтетичних даних.

Крім того, при використанні синтетичних даних слід бути уважними до потенційного введення упередженості. Моделі, як ChatGPT-3.5, навчені на великих обсягах текстів, можуть відображати існуючі упередження в цих текстах. Таким чином, важливо забезпечити, щоб набір даних для тренування був різноманітним та збалансованим, щоб уникнути посилення цих упереджень у вашій універсальній моделі.

Таблиця 3.2 – Описи задачі спрощення тексту для ChatGPT-3.5 і середня оцінка

Опис задачі	Оцінка
Please simplify the following text to make it easier to understand:	4.5
Rewrite this paragraph in simpler terms for a younger audience:	4.8
Convert the following complex text into simple, straightforward language:	4.6
Simplify this text to make it more accessible for non-native English speakers:	4.7
Transform the following content into easy-to-understand language:	4.0
Rewrite this passage using simpler vocabulary and sentence structures:	4.2
Make the following text less complex and more straightforward:	4.3

Таблиця 3.3 – Описи задачі виправлення граматичних та орфографічних помилок тексту для ChatGPT-3.5 і середня оцінка

Опис задачі	Оцінка
Correct grammar for the following text:	4.1
Identify and fix any grammatical mistakes in this paragraph:	3.8
Revise this text to correct any grammatical errors:	3.7
Edit the following sentences for proper grammar and syntax:	4.0
Rectify any grammatical inaccuracies in this text:	3.6
Improve this text by correcting only necessary grammatical errors:	4.5
Amend the following passage to fix essential grammar issues:	4.3

Таблиця 3.4 – Описи задачі перефразування для ChatGPT-3.5 і середня оцінка

Опис задачі	Оцінка
Rewrite the following text with different wording but the same meaning:	4.5
Paraphrase this paragraph to convey the same message in a different way:	4.8
Rephrase the following sentences while maintaining their original intent:	4.6
Alter the wording of this text to express the same idea differently:	4.3

Change the structure and words of the following passage but keep the same meaning:	4.7
Provide a different version of this text with the same context and message:	4.2
Reword this content while preserving its original concept and information:	4.4

Отже, у рамках дисертаційної роботи було побудовано 3 синтетичних набори даних для кожної з задач письмового асистенту: задачі виправлення граматичних і орфографічних помилок, спрощення та перефразування тексту. Для кожної задачі взято по 10 000 вхідних речень з існуючих наборів даних: ParaNMT для перефразування тексту, WikiAll для спрощення тексту та Lang8 для задачі виправлення помилок.

Набори даних були профільтровані від неякісних (garbage) речень, далі вибрані випадковим чином, щоб відповідати вибраній кількості. Далі для кожної задачі було використано по 7 текстів з описом задачі (prompts). Далі на 20-ти прикладах було вручну оцінено якість переписувань по шкалі від 1 до 5.

Це було зроблено з метою визначення описів, які підходять для генерації тренувальних прикладів найкращим чином, використовуючи велику мовну модель ChatGPT.

Як можна побачити з даних у таблиця 3.2-3.4 більшість prompts працює доволі добре і ChatGPT-3.5 виконує гарну роботу в незалежності від того, як сформована задача, хоча й все ж й існує різниця. Винятком є лише задача виправлення помилок; справа у тому, що велика кількість переписувань крім власне виправлення граматичних та орфографічних помилок також перефразовують текст, що є виходить поза рамки задачі. З метою виправити це обрані описи які наголошують на виправленні лише обов'язкових помилок. Саме вони й отримали найкращі оцінки.

Після вибору найкращого опису задачі для моделі згідно оцінювання, використовуючи OpenAI API було у автоматичному режимі надіслано 10 000 текстів для кожної задачі. Середня кількість GPT-3.5 токенів на запит, в залежності

від речення була приблизно 60 токенів у сумі (генерація біля 25-ти). На час написання дисертаційної роботи сумарна вартість генерації склала близько 10 USD, що доволі дешево для 30 000 переписувань.

Приклад діалогу:

User: Rewrite this paragraph in simpler terms for a younger audience:
Stanford has been widely considered to be one of the most prestigious universities in the entire world.
ChatGPT: Stanford is known as one of the best universities in the world.

Також варто зазначити, що модель не завжди генерує лише переписування, а часто ще діалогові повідомлення на кшталт “Sure. Here’s your text: <TEXT>” або “The revised version: <TEXT>. The edits that were made by the system: …”. Тобто потрібно ще витягнути з відповіді моделі саме переписуваний текст. Непогано працюють евристики, такі як вибирати строку після двокрапки у попередньому рядку, тощо. Але вони все ж не дають гарантованого результату.

Для вирішення проблеми було запропоновано 2 підходи:

- самостійно додавати “The revised version:” як частину відповіді асистенту. У Completion API існує можливість давати моделі продовжувати свою власну репліку, маючи попередню історію. Таким підходом ми ніби змушуємо модель почати відповідь певним чином;
- використовувати інструмент functions. Ця функція дозволяє гарантовано отримувати JSON файл певного формату у відповідь. Користувач має можливість задати поля та детально описати кожне з них. У нашому випадку вибудемо використовувати наступну схему: `{"rewritten_text": {"type": "string" ,"description": "User text rewritten according to the requirements."}}`

Обидва підходи продемонстрували високе покриття «видобутого» тексту переписувань. Задля надійності і економії токенів було вирішено використовувати підхід з JSON схемою.

Далі згенеровані набори даних будуть згадуватися як ParaphraseGPT, SimplifyGPT та CorrectGPT для задач перефразування, спрощення тексту та виправлення помилок відповідно, або AssistGPT, щоб згадати всі 30 тисяч синтетичних прикладів. У подальших розділах буде розглянута якість згенерованого набору даних з точки зору якості роботи нейронних мереж натренованих з їх використанням.

У підсумку, інтеграція ChatGPT-3.5 в процесі створення синтетичних даних для тренування письмового асистента є стратегічним вибором. Цей підхід дозволяє створювати більш об'ємні та різноманітні тренувальні набори даних за адекватні грошові витрати, що є ключовим для розробки високоефективної та адаптивної моделі.

3.3 Тренування спеціалізованих нейронних мереж з огляду на вибір архітектури

Вибір архітектури нейронної мережі та вагів (у випадку попередньо натренованої моделі) є одним з ключових елементів у розробці ефективних систем обробки природної мови, зокрема, при створенні письмового асистента. Це рішення має значний вплив на загальну продуктивність, точність та адаптивність системи до конкретних задач.

Архітектура нейронної мережі визначає загальну структуру моделі, включаючи кількість шарів, типи шарів (повнозв'язні, згорткові, рекурентні, трансформери), та спосіб, яким ці шари взаємодіють. Правильний вибір архітектури є критичним для забезпечення здатності моделі ефективно вчитися та адаптуватися до специфіки задач, таких як виправлення помилок, спрощення тексту, або перефразування. У Розділі 2 ми детально розглянули типи нейронних мереж та як вони працюють.

Ваги моделі також є важливими. Попередньо натреновані моделі можуть значно прискорити процес навчання моделі та покращити її здатність до генералізації. Наприклад, GPT-3 або BERT, які були натреновані на великих

наборах даних, можуть надати попереднє розуміння мови, що є надзвичайно корисним для більшості задач обробки тексту. Це особливо важливо у випадках, коли для тренування моделі недостатньо власних даних.

У той же час, використання попередньо натренованих вагів також має свої виклики. Одним з них є можливість упередженості або архітектурні обмеження, які могли бути внесені в ці моделі під час їхнього первинного навчання. Таке упередження може бути неочевидним, але воно може значно вплинути на вихідні результати. Тому важливо проводити додаткову валідацію та тестування моделі, щоб забезпечити її надійність та об'єктивність.

У якості першого етапу ми будемо тренувати окремі моделі під конкретні задачі. Цей крок потрібен задля того, щоб зрозуміти, де ми знаходимося відносно існуючих праць з точки зору якості. Також важливим аспектом є як багато і чи взагалі втратить універсальна модель, здатна вирішувати всі задачі, відносно окремих моделей.

У якості базових моделей, як ми будемо розглядати 4 моделі:

- LSTM з увагою Luong – екодер-декодер рекурентна нейронна мережа з модифікованим алгоритмом уваги для зв’язку між енкодером та декодером, 3 шарами як для кодування та декодування та прихованим шаром у 512 нейронів, word2vec для BPE представлень з словника;
- повністю згорткова нейронна мережа – екодер-декодер CNN з 4-ма шарами з розміром ядра 512 нейрони та шириною у 3 токени, 2-ма шарами з розміром ядра 1024 та шириною 3, 1-м шаром з розміром ядра 512 та шириною 2048 та шириною 1, word2vec для BPE представлень з словника;
- базова модель трансформер – екодер-декодер з відповідною архітектурою: 6 шарів як для кодування та декодування, прихованим шаром у 512 нейронів, fast forward шаром 2048, word2vec для BPE представлень з словника;
- базова модель BART – попередньо натренована модель базової архітектури трансформер з трохи більшим прихованим шаром 768 на задачі "denoising" –

відновленні тексту, де в процесі навчання моделі подається початковий текст із випадково зміненими або видаленими частинами.

Усі моделі мають схожу кількість параметрів (близько 110 мільйонів), а тому порівняння між ними є чесним з точки зору розміру моделей.

Такий вибір розмір обґрунтований тим, що ціль цієї роботи знайти баланс у розмірі моделей між потужністю (якістю) та швидкістю роботи (ефективністю). Більші моделі, хоча й можуть бути потужнішими у вирішенні певних задач, часто вимагають значно більше обчислювальних ресурсів та часу для навчання та використання. Це може бути неприйнятним у середовищах з обмеженими ресурсами або коли швидкість відгуку є критичною.

З іншого боку, менші моделі, хоча й менш потужні, надають перевагу у швидкості та економічності, що робить їх більш привабливими для застосувань у реальному часі та у середовищах з обмеженими обчислювальними можливостями. Однак, важливо зазначити, що навіть при однаковій кількості параметрів, різні архітектури можуть мати різну продуктивність у специфічних задачах. Це пов'язано з тим, що архітектура та спосіб навчання моделі істотно впливають на те, як модель може вчитися та адаптуватися до різних типів даних.

Тому, під час вибору моделі для конкретної задачі, важливо враховувати не тільки кількість параметрів, але й специфіку задачі, наявність даних та обчислювальні ресурси. Це допоможе забезпечити, що вибрана модель оптимально відповідає потребам задачі та середовища, в якому вона буде використовуватися.

Таблиця 3.5 – Обрані нейромережеві архітектури моделей

Модель	Кількість шарів	Прихований шар	Розмір словника	Fast forward шар	Кількість параметрів
LSTM	3	512	32 000	-	115 950 000
Fully CNN	5	512-2048	32 000	-	106 900 000
Transformer	6	512	50 000	2048	110 000 000
BART	6	512	32 000	2048	139 000 000

Для кожної задачі поділимо набір даних на тренування та валідацію. У кожному випадку, будемо починати з наступної конфігурації:

- розмір групи (batch size) = 32;
- темп навчання (learning rate) = 0.0005;
- оптимізатор Adam (Adam optimizer): $\beta1 = 0.9$, $\beta2 = 0.98$, $\varepsilon = 1e\text{-}9$;
- кількість епох (number of epochs) = 5;
- обмеження градієнту (gradient clipping) = 1.0;
- відсоток відсіву (dropout rate) = 0.1;
- кроків розігріву (warm-up steps) = 4000;
- згладжування міток (label smoothing) = 0.1.

Далі в залежності від якості на валідаційній вибірці буде підлаштовувати гіперпараметри.

Тепер, можна сформувати тренувальні набори даних для кожної з задач письмового асистента та перейти до тренування.

3.3.1 Тренування нейромережевої моделі для задачі виправлення граматичних та орфографічних помилок

Таблиця 3.6 – Обрана комбінація тренувальних даних для задачі виправлення помилок

Набір даних	Оригінальна к-ть прикладів	% змінених речень	Обрана к-ть прикладів
Lang-8	947 344	52.5%	50 000
FCE	34 490	62.4%	34 490
NUCLE	56 958	38.0%	56 968
W&I+LOCNESS	34 304	67.3%	60 000

Для задачі виправлення орфографічних помилок використаємо комбінацію наборів даних описаних у розділі 1.4. Це тренувальні частини National University of Singapore Corpus of Learner English (NUCLE), Lang-8 Learner Corpora, First Certificate in English (FCE) та W&I+LOCNESS. Оскільки Lang-8 – найбільший набір даних з найменшою якістю анотації, було вирішено обмежити кількість прикладів з цього набору даних до 50 000, попередньо профільрувавши їх. W&I+LOCNESS навпаки найбільш якісний, тому його будемо використовувати більш за допомогою процедури upsample, збільшивши кількість майже вдвічі (до 60 000).

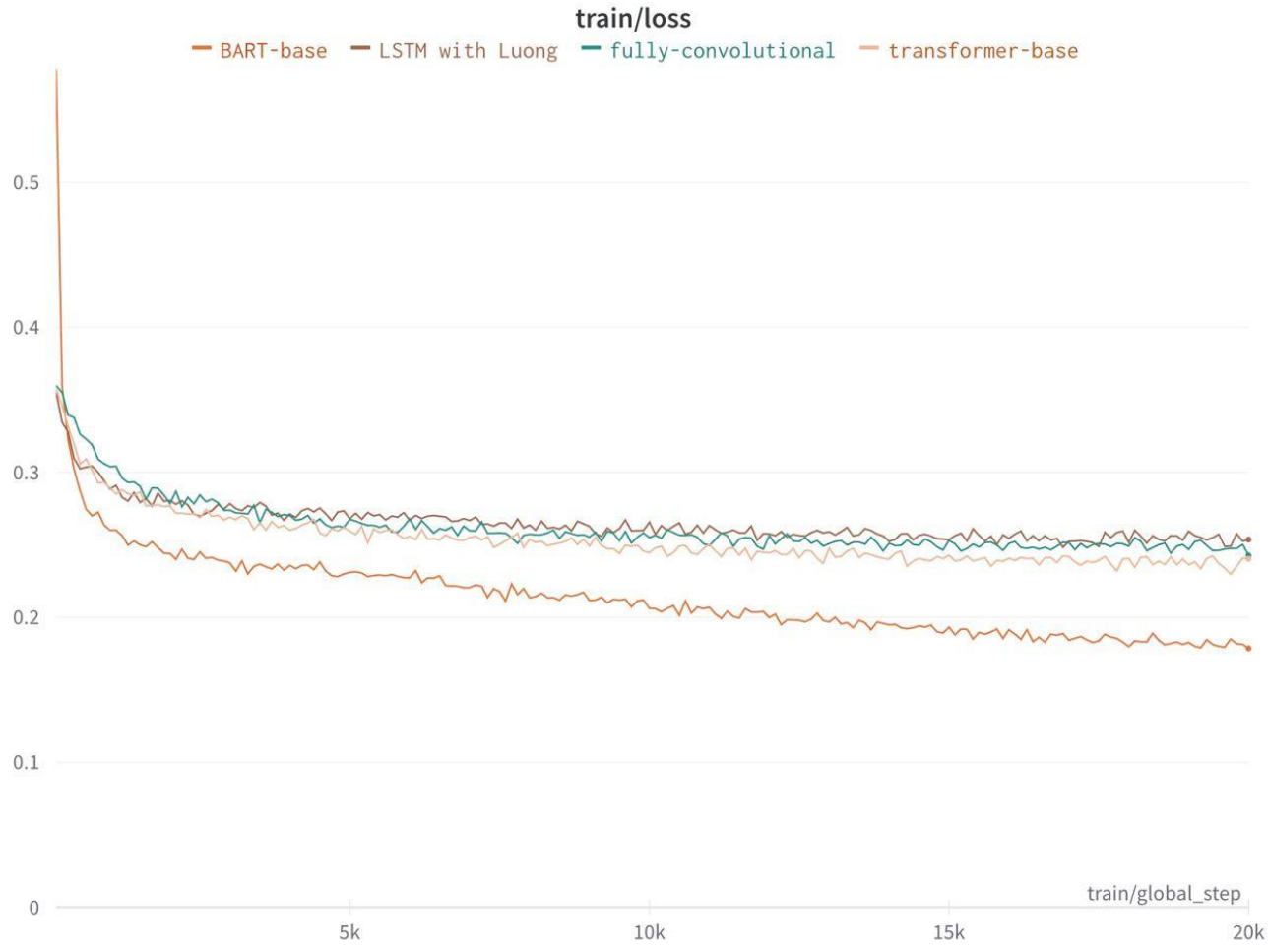


Рис. 3.1 – Функція втрат для обраних моделей зі стандартним набором гіперпараметрів для задачі виправлення помилок

Валідаційні частини мають FCE та W&I+LOCNESS. Їх ми будемо застосовувати для підбору гіперпараметрів. Тестові (evaluation) частини мають

NUCLE та W&I+LOCNESS. Вони будуть використані для оцінки якості моделі. Для NUCLE стандартною метрикою якості є MaxMatch, тоді як для W&I+LOCNESS – це ERRANT. Обидві метрики детально описані у розділі 1.2. Як згадувалося у розділі 3.1, для фінальної комбінованої метрики ми будемо використовувати ERRANT на W&I+LOCNESS.

Рис. 3.2 – Утилізація графічного процесора для тренування моделей для задачі виправлення помилок

Будемо починати з стандартного набору гіперпараметрів з попереднього підрозділу. З розрахунку на розмір даних, кількість оновлень моделі оберемо 20 000.

Як бачимо, BART-base консистентно демонструє кращу якість під час тренування на тренувальній функції втрат (чим менше значення тим краще). На другому місці з невеликим відривом transformer-base. Схожі результати мають LSTM with Luong та fully-convolutional модель.

Використання графічного інтерфейсу під час тренування є доволі ефективним – на рівні 80%. Не заважаючи на те що пам’ять використовується майже на максимум обрахунки все-одно не досягають 90%. Це може бути пов'язано з обмеженнями апаратного забезпечення або неефективністю алгоритмів оптимізації pyTorch [50]. Щоб покращити результати, можна спробувати оптимізувати керування пам'яттю або переглянути алгоритми тренування. Також варто розглянути можливість застосування розподіленого обчислення, що дозволить розподілити навантаження та ефективніше використовувати ресурси.

Для кожної моделі було проведено декілька ітерацій з метою покращення якості та більш якісно підбору гіперпараметрів. Отримані найкращі моделі зображені у таблиці 3.7.

Таблиця 3.7 – Результати тренування моделей обраних архітектур на задачі виправлення помилок і порівняння з існуючими моделями

Система	NUCLE test (MaxMatch)			W&I+LOCNESS test (ERRANT)		
	Precision	Recall	F0.5	Precision	Recall	F0.5
Zhao [51]	67.7	40.6	59.8	-	-	-
Kiyono [52]	67.9	44.1	61.3	65.5	59.4	64.2
GECToR	77.5	40.1	65.3	79.2	53.9	72.4
Rothe [53]	-	-	68.7	-	-	75.9
CNN	66.5	40.2	58.8	65.5	52.3	62.4
LSTM	66.9	40.5	59.2	66.7	53.1	63.4
Transformer	67.7	40.6	59.7	67.9	53.9	64.5
BART	67.8	44.7	**61.4**	68.2	58.5	**66.0**

Як видно з таблиці 3.7, попередньо натренована модель BART демонструє стабільно кращу якість у порівнянні з випадково ініціалізованими моделями. Особливо сильно це помітно на тестовому наборі NUCLE, де різниця скала більше ніж 1.5 одиниць F0.5. У обох тестових наборах помітною є суттєва різниця саме у повноті метрик. Це означає, що покриття переписувань є кращим у моделі BART. Наша гіпотеза полягає у тому, що завдяки попередньому тренуванню, модель вже доволі непогано розуміє природню мову, а не вчить це з 0. Зокрема це помітно по тому як швидко падає функція втрат під час тренування.

Різниця ж між випадково ініціалізованими моделями є меншою, оскільки сам по собі процес тренування є досить схожим. Схожою є і кількість параметрів, яку мають моделі і відповідна й їх «потужність». Трансформер показує себе трохи, особливо на тестовому наборі W&I+LOCNESS.

Слід зазначати, що порівняння не є ідеальним та залежить від підбору гіперпараметрів. Все ж, результати співпадають з очікуванням. Попереднє тренування стабільно покращує якість роботи моделей, надаючи моделям універсальну інформацію про мову, що використовується як основа для подальшого спеціалізованого навчання. Такий підхід значно збільшує загальну ефективність моделей, оскільки вони вже мають базове розуміння мовних структур та контекстів перед тим, як бути адаптованими до конкретних задач. Це дозволяє уникнути необхідності вчити модель від початку для кожної нової задачі, економлячи час та ресурси. Попереднє тренування, особливо у випадку великих нейронних мереж, може значно покращити їх здатність до генералізації, підвищуючи якість та точність їх відповідей у більш складних або незнайомих сценаріях.

3.3.2 Тренування спеціалізованої нейромережевої моделі для задачі перефразування тексту

Перейдемо до задачі перефразування тексту. Цей раз спробуємо побудувати власне попереднє тренування, яка вже довело свою ефективність для задачі виправлення граматичних та орфографічних помилок.

У якості тренувального мікса будемо використовувати 2 найвідоміші набори даних описані у Розділі 1.3. Це Quora Question Pairs (QQP) та Microsoft Common Objects in COntext (MSCOCO). Не заважаючи на меншу кількість наборів перефразування у MSCOCO, він має більшу кількість на перефразувань на набір, що робить його помітно більшим за QQP. Нам це підходить, оскільки він має трохи кращу якість.

MSCOCO має валідаційну та тестову частини акуратно побудовані авторами набору даних Microsoft. У той же час MSCOCO з початку створення Kaggle змагання був один великим тренувальним корпусом, оскільки закрита частина набору даних ніколи не була відкрита загалу. У роботі Bilateral multi-perspective matching for natural language sentence [54] автори поділили його на тренувальну, валідаційну та тестову частину. Ми використовуємо цей поділ далі у нашій роботі.

У якості метрик будемо використовувати BLEU, TER та METEOR, детально описані у розділі 1.2. Якщо BLEU та METEOR визначають що модель має кращу якість у випадку підвищення значень цих метрик, то TER – навпаки: чим нижче, тим краще. Це важливо пам’ятати під час оцінювання. Також важливо нагадати, що для фінальної комбінованої метрики ми будемо використовувати METEOR на тестовому наборі даних MSCOCO.

Будемо починати з стандартного набору гіперпараметрів з попереднього підрозділу. Для можливості порівняння з задачею виправлення помилок, також оберемо 20 000, як кількість оновлень моделі.

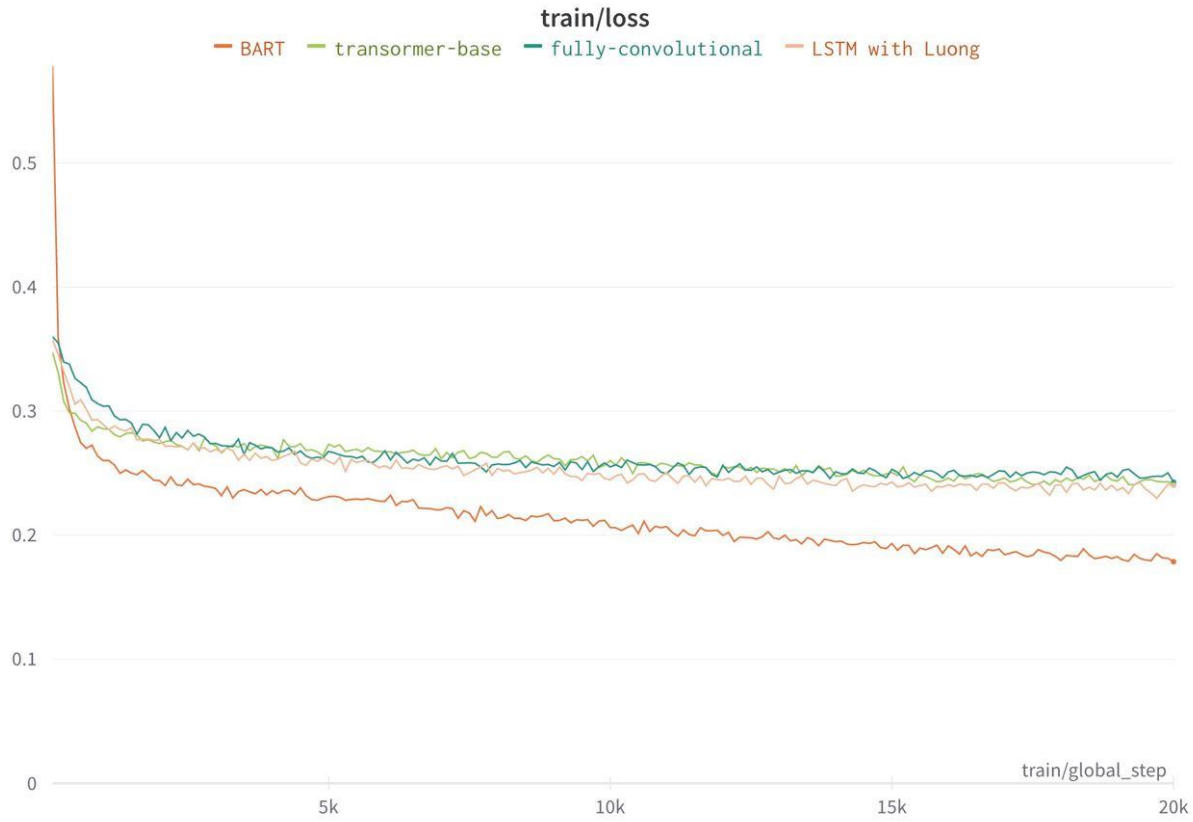


Рис. 3.3 – Функція втрат для обраних моделей зі стандартним набором гіперпараметрів для задачі перефразування тексту

Як бачимо, отримані графіки дуже схожі на ті, що були отримані для задачі виправлення граматичних та орфографічних помилок. BART показує помітно кращу якість, в той час як випадково ініціалізовані моделі мають дуже схожі функції втрат, з вищим значенням за попередньо натреновану модель.

Як можна помітити з Таблиці 3.8, відповідний тренд по метрикам зберігається. Попередньо натренована модель BART помітно випереджає випадково ініціалізовані моделі з точки зору всіх наявних метрик. В свою чергу, різниця між згортковою, рекурентною та мережею трансформер є менш помітною в порівнянні з задачею виправлення помилок – важко обрати архітектуру яка найкраще працює на даній задачі.

Далі спробуємо побудувати власне попереднє тренування та порівняти його з існуючою моделлю BART. Для попереднього тренування ми виберемо великий та різноманітний набір даних, який включатиме текст з різних джерел та тематик,

щоб забезпечити модель широким спектром мовного матеріалу. Мета полягатиме в тому, щоб навчити модель розуміти та обробляти мову на глибокому рівні, охоплюючи різноманітність семантичних та синтаксичних структур. Це дозволить моделі розпізнавати та генерувати більш точні та релевантні відповіді в подальшому.

Таблиця 3.8 – Результати тренування моделей обраних архітектур на задачі перефразування тексту

Система	QQP test			MSCOCO test		
	BLEU	TER	METEOR	BLEU	TER	METEOR
rLSTM [55]	28.4	59.1	30.2	26.9	63.3	24.4
CGMH [56]	22.5	65.0	27.0	17.3	72.6	21.9
MCPG [57]	24.1	64.5	31.8	16.5	73.5	23.2
PTS [57]	25.6	58.7	31.4	17.0	69.9	22.8
CNN	27.9	59.9	30.7	25.3	61.5	24.9
LSTM	27.5	59.7	30.1	25.0	61.1	24.6
Transformer	28.7	58.6	31.5	25.2	60.6	24.5
BART	**30.5**	**57.3**	**32.8**	**28.2**	**57.6**	**26.0**

Можна спробувати побудувати власне універсальне попереднє тренування, але зазвичай це довгого складного тренування та складного збору з багатьма нюансами. Це потребує величезних фінансових та часових витрат. Натомість, спробуємо побудувати попереднє тренування спеціалізоване на поточній задачі. Для цього нам знадобиться великий набір даних задачі перефразування з можливо

гіршою якістю. Ідеальним кандидатом для цього є ParaNMT, детально розглянутий у розділі 1.3.

Нагадаємо, що ParaNMT містить більше 50 млн пар англійською, згенерованих нейромережевим перекладом паралельних корпусів для задач машинного перекладу. Тобто це величезний набір даних відносно фінальних наборів даних, хоча й з нижчою якістю, оскільки він анотований не людьми, а іншими нейронними мережами автоматично.

Таким чином, процедура виглядає наступним чином:

1. Попереднє тренування на наборі даних ParaNMT (1 епоха)
2. Тонке налаштування на міксі MSCOCO + QQP
3. Оцінювання якості роботи моделі на MSCOCO та QQP

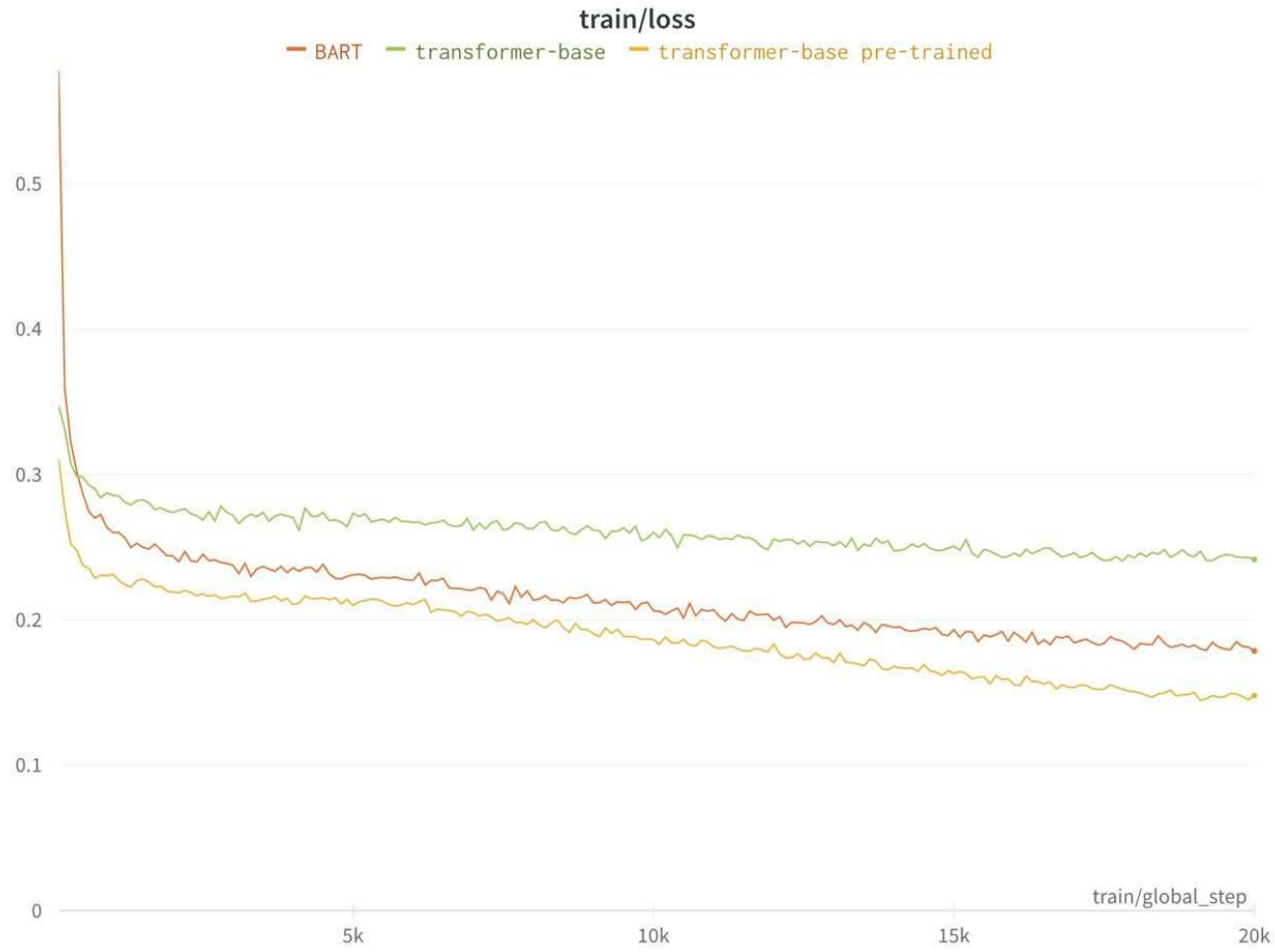


Рис. 3.4 – Функція втрат для попередньо натренованих та випадково ініціалзованих моделей для задачі перефразування тексту

Як бачимо з графіку, попереднє тренування на наборі даних ParaNMT спрацювало з точки зору функції втрат. Функція втрат спеціалізовної попередньо тренованої моделі дуже швидко падає, оскільки вона вже адаптована до задачі парафразування, і залишається нижчою за функцію універсальної попередньо натренованої моделі BART впродовж всього тренування.

Таблиця 3.9 – Результати тренування попередньо натренованих моделей (*) обраних архітектур на задачі перефразування тексту.

Система	QQP test			MSCOCO test		
	BLEU	TER	METEOR	BLEU	TER	METEOR
resid LSTM	28.4	59.1	30.2	26.9	63.3	24.4
CGMH	22.5	65.0	27.0	17.3	72.6	21.9
MCPG	24.1	64.5	31.8	16.5	73.5	23.2
PTS	25.6	58.7	31.4	17.0	69.9	22.8
CNN	27.9	59.9	30.7	25.3	61.5	24.9
LSTM	27.5	59.7	30.1	25.0	61.1	24.6
Transformer	28.7	58.6	31.5	25.2	60.6	24.5
CNN*	29.5	57.8	32.6	27.7	57.8	25.7
LSTM*	29.2	58.1	32.6	26.7	59.0	25.5
Transformer*	**30.6**	**57.4**	**33.2**	27.6	**57.6**	**26.2**
BART	**30.5**	**57.3**	32.8	**28.2**	**57.6**	26.0

Варто зазначити, що нижча функція втрат під час тренування не гарантує кращі метрики на тестових наборах даних. Це може бути пов'язано з явищем перенавчання (overfitting), коли модель занадто точно адаптується до навчального набору даних, втрачаючи здатність до узагальнення на нових даних. Таким чином, важливим аспектом є збалансування між зниженням функції втрат на тренувальних даних і забезпеченням генералізації моделі. Крім того, варто проводити ретельний аналіз помилок на тестових наборах, щоб виявити можливі недоліки моделі та скорегувати їх. В кінцевому рахунку, цілісний підхід до оцінювання та тестування є ключовим для створення надійної та ефективної моделі машинного навчання.

У Таблиці 3.9 можна побачити, що дійсно моделі попередньо навчені на ParaNMT показують суттєво кращу якість, а модель архітектури трансформер взагалі має результати чи не кращі за модель BART. Це помітно як на наборах даних MSCOCO, так і на QQP, тобто результат є консистентним (стабільним). З іншої сторони нижча функція втрат давала очікування, що ParaNMT моделі можуть давати навіть кращі результати. Саме тому важливо перевіряти фінальну якість моделей на окремих тестувальних наборах даних, які не перетинаються з тренувальними.

Перевагою моделі BART є очевидно той факт, що вона універсальна і, як ми вже побачили, показує так само високу якість на задачі виправлення граматичних та орфографічних помилок.

Також неочікуваним фактом є те що натреновані моделі є кращі за більшість вже існуючих робіт. Ймовірною причиною цього є різниця в методології оцінювання та тим фактом, що задача не є на стільки популярною як виправлення помилок і кількість існуючих робіт є значно меншою.

3.3.3 Тренування спеціалізованої нейромережевої моделі для спрощення тексту

Нарешті, перейдемо до задачі спрощення тексту. Як згадувалося у Розділі 1.5, наборів даних існує невелика кількість у порівнянні з вищезгаданими задачами.

Отримання набору Newsela виявилося проблематичним, оскільки дані не є у вільному доступі, а отримуються за запитом. Під час написання дисертаційної роботи, відповідь авторів не вдалося отримати. Саме тому єдиним набором даним використаним для задачі є WikiLarge.

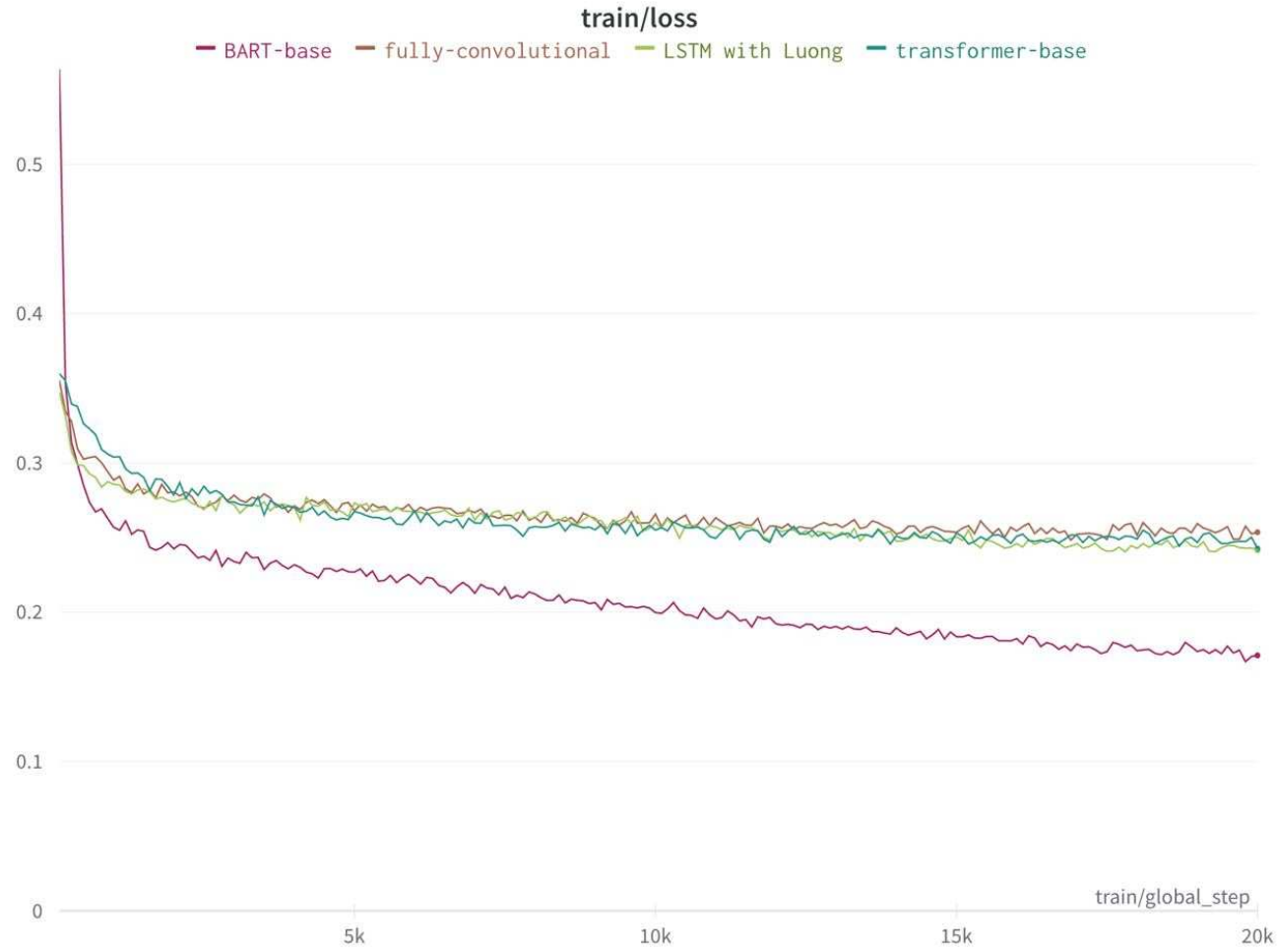


Рис. 3.4 – Функція втрат для обраних моделей зі стандартним набором гіперпараметрів для задачі спрощення тексту

Для оцінки моделей для спрощення тексту будемо використовувати TurkCorpus, окремий тестовий набір даних з 8 переписуваннями, проанотованими анотаторами з Mechanical Turck для кожного речення. Застосування множинних переписувань для кожного речення дозволяє оцінити гнучкість та адаптивність моделей до різних стилів спрощення. Такий підхід допомагає отримати всебічне

уявлення про здатність моделей адекватно трансформувати складний текст у його спрощену версію, зберігаючи при цьому ключові інформаційні елементи.

Як бачимо з графіку, маємо чітку закономірність, яка повторюється серед всіх трьох задач для задач письмового асистенту. BART показує помітно кращу якість з точки зору тренувальної функції втрат, в той час як випадково ініціалізовані моделі мають дуже схожі результати, з вищими значеннями за попередньо натреновану модель.

Далі оцінимо якість натренованих моделей на тестувальному наборі даних TurkCorpus. Метрики, які зазвичай використовуються при оцінці якості (BLUE, FKGL та SARI) описані у Розділі 1.2.

Таблиця 3.10 – Результати тренування моделей обраних архітектур на задачі спрощення тексту

Система	TurkCorpus		
	BLEU	FKGL	SARI
SBMT [7]	72.4	7.3	40.0
ACCESS [58]	-	7.3	41.4
MUSS	-	7.6	42.5
TST	-	7.9	41.5
CNN	68.3	8.2	36.8
LSTM	70.3	7.7	37.2
Transformer	70.4	7.5	37.5
BART	70.3	7.9	39.6

У Таблиці 3.10 наведено результати оцінювання на наборі даних TurkCorpus. Як можна помітити, тут перевага BART не є такою відчутною. З іншої сторони, існує низка робіт, яка пояснює неефективність більшості метрик для задач спрощення тексту, як ми вже згадували у розділі 3.1. Ключовою ж (і найбільш надійною) метрикою є SARI. Як бачимо, тут різниця виглядає помітною.

Хоча BART показує кращі результати за обрані архітектури, різниця у порівнянні з state-of-the-art підходами все ще помітна. Не дивлячись на це, саме ця модель продемонструвала стабільний результат на всіх 3-х задачах, перевершивши випадково ініціалізовані згорткові, рекурентні моделі та архітектуру трансформер. А тому, саме цю модель ми будемо використовувати як основу для нашого письмового асистенту.

3.4 Побудова універсальної моделі письмового асистенту

Тепер, коли ми добре розуміємо кожну задач, їх метрики та якість роботи спеціалізованих моделей запропонованих архітектур у порівнянні з існуючими роботами, можна починати роботу над єдиною моделлю, здатною виконувати всі 3 вибрані задачі письмового асистенту. У розділі 3.3 було вирішено використовувати саме модель BART як базову.

У цій дисертаційній роботі ми розглянемо декілька підходів для побудови універсальної мульти-задачної моделі. Серед них:

1. Адаптери – це невеликі, навчені модулі, які вставляються між шарами попередньо натренованої моделі. Вони дозволяють налаштувати модель під специфічні задачі, не змінюючи основних вагів. Це може бути ефективним способом адаптації моделі до кожної з трьох задач, мінімізуючи при цьому необхідність повторного тренування великих частин моделі;
2. Спільний енкодер - спеціалізований декодер: цей підхід передбачає використання одного енкодера для зчитування та розуміння тексту та декількох декодерів для кожної задачі окремо. Така структура дозволяє

моделі зосередитися на спільному розумінні тексту, але при цьому мати спеціалізовані інструменти для кожної задачі;

3. Спеціальний токен для визначення задачі (control codes): Введення спеціального токена в якості частини вхідних даних, який вказує на конкретну задачу, може бути ефективним способом навчання моделі розрізняти між задачами. Цей токен може бути використаний для активації відповідних шарів або модулів у моделі, залежно від потрібної задачі.

Використання цих трьох підходів може створити дуже гнучку та потужну систему, яка буде здатна адекватно реагувати на різні вимоги та особливості кожної з трьох задач.

3.4.1 Використання модулів-адаптерів для тренування універсальної нейронної мережі

Адаптери в нейронних мережах представляють собою новаторський підхід до тонкої настройки глибоких нейронних мереж, особливо в сфері обробки природної мови. Вони дозволяють вносити невеликі зміни в попередньо натренованій моделі, які можуть виявитися дуже ефективними для специфічних завдань, таких як виправлення граматичних помилок, спрощення тексту або перефразування.

Цей метод є особливо цінним, оскільки дозволяє використовувати потужні, вже попередньо натреновані моделі, такі як BART або T5, і адаптувати їх під конкретні завдання, не вимагаючи повного перенавчання моделі. Таким чином, використання адаптерів може істотно скоротити час і ресурси, необхідні для тренування, зберігаючи при цьому ефективність великих моделей.

Основна ідея адаптерів полягає в тому, що ви вставляєте додаткові шари між існуючими шарами попередньо натренованої моделі. Ці додаткові шари - адаптери - відносно малі і тренуються окремо, при цьому ваги основної моделі залишаються незмінними. Це означає, що ви можете використовувати загальне розуміння мови,

яке надає попередньо натренована модель, і одночасно налаштовувати її на специфічні завдання за допомогою адаптерів.

Адаптери, як правило, є досить простими в архітектурі. Вони можуть складатися всього з декількох повнозв'язних шарів, але, незважаючи на свою простоту, вони можуть значно підвищити ефективність моделі для конкретної задачі. Завдяки своїй гнучкості і ефективності, адаптери можуть бути використані для широкого спектру завдань, від перекладу тексту до генерації тексту, виправлення помилок і багато іншого.

Однією з ключових переваг адаптерів є їх модульність. Ви можете тренувати окремі адаптери для різних задач, а потім легко вставляти або видаляти їх з моделі без необхідності повторного тренування всієї моделі. Це робить адаптери ідеальними для ситуацій, де вам потрібно швидко адаптувати модель під різні завдання.

Ще однією перевагою є те, що адаптери дозволяють зберегти знання, вже накопичені в моделі. Попередньо натреновані моделі, такі як BERT або GPT, вже мають глибоке розуміння мови завдяки навчанню на великих наборах даних. Адаптери дозволяють використовувати це розуміння і адаптувати його під конкретні завдання, не втрачаючи загальних знань.

Використання адаптерів особливо є корисним для даної роботи у контексті мультизадачності. Замість того, щоб створювати окремі моделі для кожної задачі, ви можете використовувати одну і ту ж основну модель і просто міняти адаптери в залежності від завдання. Це не тільки економить час і ресурси, але і дозволяє моделі накопичувати знання з різних доменів.

Концепція використання адаптерів у контексті ефективного переносу навчання в нейронних мережах для задач обробки природної мови отримала значну увагу після публікації статті "Parameter-Efficient Transfer Learning for NLP" [59]. У цій роботі дослідники зосереджуються на створенні методів, які дозволяють адаптувати великі попередньо натреновані моделі до конкретних завдань з мінімальною кількістю додаткових параметрів.

Детально зображено схему адаптеру на Рисунку 3.5. Шар архітектури трансформер складається з двох основних компонентів: багатоголової уваги (multi-headed attention) та двох лінійних (feed-forward) шарів, розділених шаром нормалізації (layer norm). Адаптер, який вставляється в трансформерний шар, зазвичай містить два додаткові лінійні шари, які оточують нелінійну активаційну функцію.

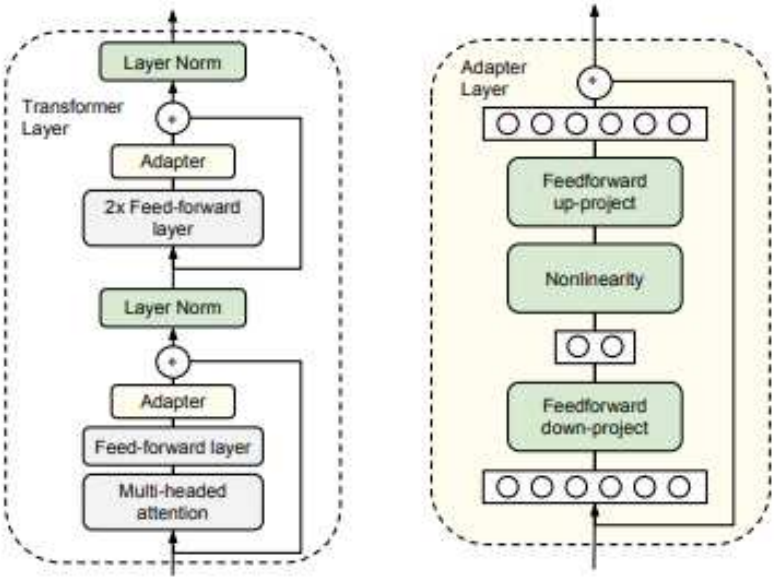


Рис. 3.5 – Схематичне зображення використання модуля-адаптеру у архітектурі трансформер

Перший лінійний шар в адаптері — це шар проекції вниз (down-project), який зменшує розмірність даних назад до розмірності, сумісної з вимірами вхідних даних трансформерного шару. Після шару проекції вгору йде нелінійність, як правило, це активаційна функція, така як ReLU або GELU. Ця нелінійна активація додає моделі можливість вловлювати складніші відносини в даних.

Після нелінійного шару йде другий лінійний шар, шар проекції вгору (up-project), який збільшує розмірність вхідних даних до більш високої вимірності, що визначається параметрами моделі. Ця структура дозволяє адаптерам модифікувати сигнали, які проходять через трансформерний шар, не порушуючи загальну архітектуру моделі.

Важливою частиною архітектури адаптера є використання skip-connection, також відомого як residual connection. Skip-connection дозволяє вхідним даним адаптера обходити два лінійні шари і нелінійність, ідучи прямо до виходу адаптера. Це забезпечує, що навіть після внесення специфічних для задачі коректив у вектори ознак, модель все ще здатна використовувати первісне репрезентативне знання, яке було в моделі до застосування адаптера.

У наші роботі ми перевикористовуємо підхід з роботи "Parameter-Efficient Transfer Learning for NLP", додаючи модулі-адаптери двічі у кожен енкодер шар трансформеру та тричі (також після cross-attention) для кожного декодер шару моделі BART. Таким чином для моделі BART base виходить $6 * 2 + 6 * 3 = 30$ адаптерів у сумі.

Ми також використовуємо різні конфігурації адаптерів, змінюючи розмір прихованого (між down-project та up-project). Повторюючи існуючі роботи у напрямку, ми пробуємо набір варіантів h_{size}: $\{8, 64, 256\}$. Більший розмір надає модулю більшу виразність, у той же час збільшуючи кількість додаткових параметрів.

Таблиця 3.11 – Результати тренування моделі BART з використанням модулів-адаптерів та без

	ERRANT	SARI	METEOR	AssistantScore	# параметрів
окремі моделі	66.0	39.6	26.0	229.1	417 M
адаптер 8	45.3	31.2	18.3	163.4	139 M
адаптер 64	51.4	32.4	18.2	177.2	142 M
адаптер 256	60.4	34.5	22.4	205.6	151 M

Таким чином кількість доданих параметрів в залежності від шару буде дорівнювати

$$30 \cdot 2 \cdot d_{model} \cdot h_{size} = 30 \cdot 2 \cdot 768 \cdot h_{size} = 47\,280 \cdot h_{size}$$

Як бачимо з Таблиці 3.11, якість роботи моделей, що використовують адаптери сильно залежить від кількості параметрів закладених у ці модулі. Найменший розмір прихованого шару адаптеру 8 додає менше пів мільйона параметрів, але якість роботи такої моделей є помітно нижчою. З іншою сторони адаптер 256 додає 12 мільйонів параметрів, що відповідає 8.7% від всього розміру моделі BART-base, але в той же час різниця між таким підходом і 3-ма окремими моделями є значно меншою.

З точки зору комбінованої якості на один параметр архітектура, адаптер 256 демонструє найкращий результат. У той же час важливим нюансом є той факт, що для того щоб виконати усі задачі письмового асистента на вхідному тексті потрібно прогнати модель окремо на кожній задачі з окремим набором адаптерів. Тобто немає частин (результатів), які можна перевикористати.

3.4.2 Застосування підходу з спільним кодування та спеціалізованим декодуванням

Підхід із спільним енкодером і різними декодерами для різних задач в обробці природної мови може бути ефективним з точки зору часу виконання та використання ресурсів. Центральна ідея полягає у тому, що один енкодер зчитує та аналізує вхідний текст, конвертуючи його в універсальні векторні репрезентації, які потім використовуються множиною декодерів для генерування відповідей або виконання специфічних задач. Кожен декодер оптимізований для конкретної задачі, будь то виправлення помилок, перефразування, чи спрощення тексту. Наприклад, такий підхід був застосований у роботі Instance-specific decoders for multi-task NLP [60].

Енкодер виконує складну роботу з розуміння мови, дозволяючи декодерам зосередитись на своїх унікальних задачах. Це розділення обов'язків ефективно,

оскільки кодування вхідних даних є найбільш ресурсоємною частиною процесу, і коли це робиться один раз, декодери можуть ефективно генерувати вихідні дані без повторення цієї трудомісткої роботи. Таким чином, використання спільного енкодера може значно зменшити обчислювальні витрати і покращити швидкість відгуку системи.

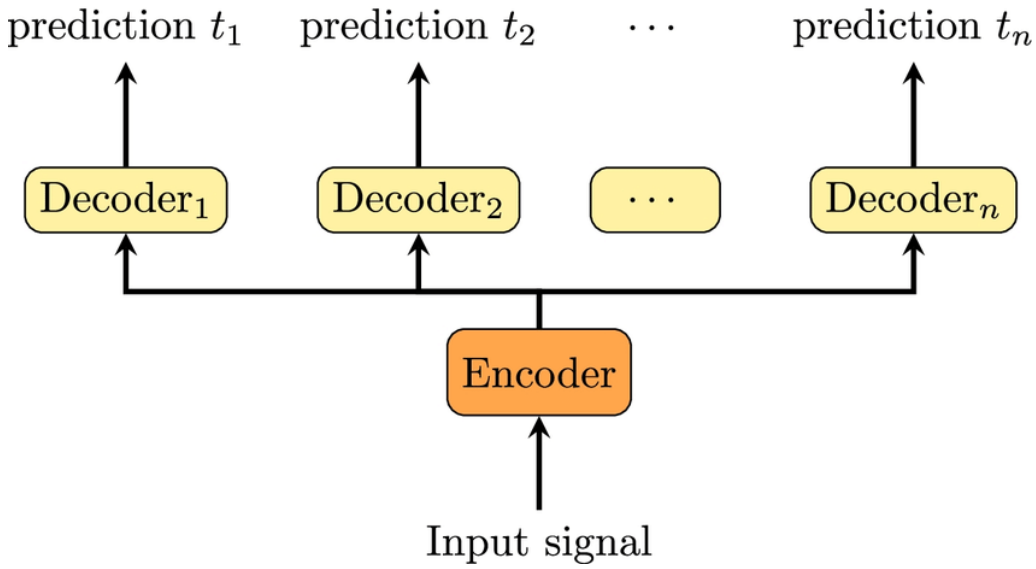


Рис. 3.6 – Схематичне зображення підходу зі спільним енкодером та спеціалізованими декодерами

Кожен спеціалізований декодер може бути адаптований до своєї задачі, що дозволяє більш точно реагувати на вимоги цієї задачі. Наприклад, декодер для виправлення помилок може зосередитись на граматиці та правописі, в той час як декодер для перефразування може зосередитись на збереженні значення оригінального тексту, але зміні його структури. Це розділення дозволяє кожному декодеру ефективно використовувати ресурси та бути оптимізованим для кращої продуктивності.

Такий підхід також забезпечує велику гнучкість у розширенні системи. Якщо з'являється потреба в новій задачі, можна додати новий декодер без необхідності зміни чи перенавчання енкодера. Це робить систему стійкою до масштабування і дозволяє швидко відповідати на нові вимоги або виклики.

Оскільки кодування вхідного тексту виконується один раз енкодером, вектори кодування можуть бути перевикористані для різних задач. Це усуває необхідність повторного кодування тексту для кожної задачі, що значно скорочує час, необхідний для виконання передбачень. Репрезентації, створені енкодером, можуть бути кешовані, дозволяючи швидкий доступ до них при використанні декодерів.

У порівнянні з адаптерами, які вносять зміни на рівні окремих шарів моделі, спільний енкодер використовує однаковий процес кодування для всіх задач. Це відрізняється від ситуації, коли адаптери потребують, щоб кожен шар моделі був адаптований та перенавчений для кожної конкретної задачі. Адаптери забезпечують більшу гнучкість на рівні шарів використовуючи відносно невелику кількість параметрів, в той час як спеціалізовані декодери вимагають більшу к-ть унікальних параметрів.

Спільний енкодер-спеціалізований декодер підхід мінімізує цю витрату часу, оскільки кодування виконується один раз, а декодери можуть працювати паралельно або бути викликаними за необхідності, без необхідності повторного проходження через енкодер. Це робить підхід зі спільним енкодером ідеальним для розробки високопродуктивних систем обробки природної мови, де для однієї вхідної послідовності потрібно виконати всі декодер-задачі одночасно.

Стосовно тренування енкодеру, то маємо кілька опцій:

1. Використовувати його як, тобто покладатися на те що універсальні ваги BART енкодеру є достатньо хороші і не потребують додаткового тренування;
2. Тренувати його для однієї з задач і далі заморозити для інших; як основну можна взяти виправлення граматичних та орфографічних помилок, оскільки вона має найбільшу кількість даних;
3. Натренувати його на всіх задач, а далі усереднити; таким чином буде отриманий компроміс між задами.

Не дивлячись на більшу кількість доступних параметрів для тренування, підхід зі спільним енкодером та спеціалізованими декодерами показує у найкращому випадку схожу якість у порівнянні з BART моделлю з адаптерами 256. Останній є більш оптимальним з точки зору якості на один параметр.

Таблиця 3.12 – Результати тренування моделі BART з використанням спільного енкдору та спеціалізованими декодерами

	ERRANT	SARI	METEOR	AssistantScore	# параметрів
окремі моделі	66.0	39.6	26.0	229.1	417 M
спільний енкодер (1)	59.4	34.2	20.3	199.1	251 M
спільний енкодер (2)	66.0	32.4	18.5	206.8	251 M
спільний енкодер (3)	60.3	34.1	18.9	198.4	251 M

Цікавим є те, що ідея з усередненням параметрів енкодеру не спрацювала у порівнянні з просто застосування стандартних вагів BART. Можливо це означає, що після тренування ваги моделей кожної з задач занадто різні і усереднення не дає змістовних результатів. Можливо процес кодування послідовностей з попереднього тренування BART вже дуже добре підходить для задач письмового асистенту.

За рахунок великого значення коефіцієнту задачі виправлення помилок у комбінованій метриці, оптимізація цієї в першу чергу приносить найкращий результат. В такому підході ми трохи жертвуємо задачами перефразування і спрощення тексту. У той же час, падіння на цих задач не є суттєвим, що скоріш за все означає, що кодування виправлення помилок є прийнятним (і може бути використано) для інших задач.

3.4.3 Контрольні токени задач для тренування універсальної моделі письмового асистенту

Підхід із використанням спеціальних токенів для визначення задачі, відомих як control codes, дозволяє одній моделі виконувати різні завдання обробки тексту, такі як переклад, виправлення помилок, або генерацію тексту, з використанням однакової архітектури. Ключовим аспектом цього підходу є додавання до вхідних даних спеціального токена, який слугує як індикатор конкретної задачі, яку модель має виконати.

У цьому контексті, такі токени керування діють подібно до команд, які можуть змінювати поведінку моделі, спрямовуючи її на виконання конкретних задач. Наприклад, у випадку з моделлю, яка займається як перекладом, так і виправленням тексту, можна було б використати токен "[SIMPLIFY]" для активації режиму перекладу, і токен "[CORRECT]" для активації режиму виправлення помилок. Такі токени використовувалися і великих мовних моделях, таких як CTRL: A Conditional Transformer Language Model for Controllable Generation [61].

Моделі архітектури трансформер особливо добре підходять для цього підходу, оскільки вони вже здатні розуміти контекст та важливість позиціонування токенів у вхідному тексті. Це робить їх гарними кандидатами для розуміння та відповідної реакції на токени керування. Таким чином, модель може, наприклад, виконувати задачі виправлення помилок у тексті, генерації відповідей, або стиснення інформації, в залежності від того, який токен керування їй подано.

Важливим нюансом використання control codes в нейронних мережах, особливо тих, що базуються на трансформерах і використовують механізм BPE (Byte Pair Encoding), є додавання цих токенів безпосередньо до словника моделі. Це робиться для того, щоб запобігти їх розбиттю на менші частини, як це відбувається в процесі стандартного токенізування.

У випадку, якщо спеціальний токен не включений до словника моделі, він може бути розділений на менші субтокени, що може призвести до втрати його семантичного значення і, як наслідок, до зменшення ефективності моделі.

Наприклад, якщо модель тренується розпізнавати токен "[TRANSLATE]" як індикатор для перекладу тексту, але цей токен не включений до словника, він може бути поділений на "[", "TRANSL", "ATE", "]", що робить його нерозпізнаваним для моделі як єдиний спеціальний токен.

Однією з ключових переваг підходу є те, що він не вимагає додавання значної кількості нових параметрів до моделі. Це відрізняє цей підхід від інших методів, таких як використання адаптерів або спеціалізованих шарів, які можуть суттєво збільшити загальний розмір моделі.

Таблиця 3.13 – Результати тренування моделі BART з використанням контрольних токенів

	ERRANT	SARI	METEOR	AssistantScore	# параметрів
окремі моделі	66.0	39.6	26.0	229.1	417 M
1 контрольний токен	57.4	30.7	19.2	188.5	139 M
3 контрольні токени	58.4	30.8	21.2	194.0	139 M
5 контрольних токенів	58.6	30.7	21.0	193.9	139 M

У підході з токенами керування основні зміни, які вносяться в модель, полягають у додаванні невеликої кількості нових токенів до словника. Оскільки кожен такий токен керування займає лише невеликий об'єм пам'яті, загальний вплив на розмір моделі мінімальний. Це робить підхід особливо привабливим для застосувань, де обмеження пам'яті або обчислювальні ресурси є критичними факторами.

Основний параметр, який можна змінювати це кількість токенів для кожної конкретної задачі. Це дозволяє точно налаштувати поведінку моделі, забезпечуючи більшу адаптивність та гнучкість. Наприклад, для задачі перекладу можна ввести серію токенів, таких як "[TRANSLATE]", "[TRANSLATE_1]", "[TRANSLATE_2]", ..., "[TRANSLATE_N]".

Цей метод дає можливість збільшити "експресивну силу" задачі, надаючи моделі більш гнучкі можливості для реагування на вимоги задачі. Це дозволяє одній моделі ефективно впоратися з широким спектром завдань, використовуючи різні стратегії обробки для кожного випадку.

З таблиці 3.13 можна помітити, що не дивлячись, що контрольні токени дають гірші результати ніж деякі конфігурації адаптерів та спеціалізованих декодерів, вони майже не потребують доданих параметрів. Результати все-одно наближаються інших підходів, а з точки зору якості на один параметр підхід взагалі демонструє найкращу якість.

3.4.4 Покращення універсальних моделей письмового асистенту за допомогою даних згенерованими великими мовними моделями

Отже, у попередніх секціях підрозділу ми спробували різні підходи по побудові універсальної моделі здатної вирішувати множину задач письмового асистента. Було проведено порівняння з випадком, коли для кожної задачі використовується окрема модель і, очікувано, цей підхід дає найкращу якість, хоча є досить неефективним з точки зору ефективності параметрів.

Досі ми не використовували новий набір даних, створений рамках даної роботи. Мова йде про синтетичні дані, згенеровані великою мовною моделлю ChatGPT у Розділі 3.2. Використання цих даних потенційно може зменшити різницю між універсальними моделями та спеціалізованими.

У рамках дисертаційної роботи, ми спробували різні конфігурації тренування з використанням синтетичним даних: поетапне тренування, комбінування даних у різних наборах. Далі ми презентуємо результати з комбінування даних, оскільки такий підхід показав найкращі результати на тестових даних.

Результат у Таблиці 3.14 підтверджує що дані згенеровані великими мовними моделями суттєво покращують якісь абсолютно всіх розглянутих моделей. Це також показує що великі мовні моделі досягли приголомшуючі результати в останні роки, перевершуючи за якістю роботу багато існуючих state-of-the-art

моделей та будучи при цьому універсальним інструмент, що покриваю велику множину задач.

Таблиця 3.14 – Результати використання синтетичних даних для тренування універсальних та спеціальних моделей

	синтет. дані	ERRANT	SARI	METEOR	Assistant Score	# парам.
окремі моделі		66.0	39.6	26.0	229.1	417 M
окремі моделі	X	71.3	41.9	29.4	248.5	417 M
адаптер 256		60.4	34.5	22.4	205.6	151 M
адаптер 256	X	65.2	36.1	25.3	221.6	151 M
спільний енкодер (2)		66.0	32.4	18.5	206.8	251 M
спільний енкодер (2)	X	71.3	35.3	24.8	232.0	251 M
5 контрольних токенів		58.6	30.7	21.0	193.9	139 M
5 контрольних токенів	X	59.1	32.1	21.8	199.8	139 M

Також можна помітити, що суттєво покращилися результати універсальних моделей, особливо підходів адаптер 256 та «спільний енкодер (2)». Останній показує кращу якість ніж окремі моделі до використання синтетичних даних. Таким чином, згенеровані у рамках цієї дисертаційної роботи дані дозволяють надолужити втрати по якості генерації пов’язані з побудовою більш універсальних моделей.

У той же час, використання адаптерів з прихованим шаром 256 лишається найбільш ефективним підходом з точки зору якості на кількість параметрів. Такий результат у метриці є саме тим, чого ми хочемо досягти у дисертаційній роботі. Переваги пов’язані зі швидкодією у підході зі спільним енкодером працюють тільки тоді, коли необхідно виконати декілька задач для одного й того самого вхідного речення, що не часто трапляється у задачах письмового асистенту, де виправлення застосовують послідовно. Якість підходу з контрольними токенами,

не дивлячись на ефективність, не є достатньою. Саме тому у наступних розділах ми концентруємося на підході з адаптерами 256.

3.5 Оптимізація швидкості роботи універсальної нейронної мережі для задач письмового асистенту

Отже, тепер, коли ми визначилися з вибором фінальної архітектури та якістю моделі, можна перейти до останнього етапу – оптимізації. Ми прагнемо забезпечити, щоб наша модель працювала швидко і ефективно. Письмовий асистент повинен надавати швидкі виправлення, не сповільнюючи роботу користувачів.

3.5.1 Використання бібліотеки ONNX для оптимізації швидкості роботи моделей

Одним з основних напрямків оптимізації нашої моделі є використання Open Neural Network Exchange (ONNX) [62], універсального формату для нейронних мереж. ONNX дозволяє перетворювати моделі з різних фреймворків машинного навчання у стандартний формат, що значно полегшує їхнє використання та інтеграцію в різні системи. Ця технологія не тільки забезпечує гнучкість у використанні моделі на різних платформах, але й оптимізує її виконання, дозволяючи моделі працювати швидше і ефективніше.

ONNX є важливим елементом в екосистемі машинного навчання, оскільки він спрощує процес перенесення моделей між різними фреймворками. ONNX забезпечує уніфікований формат файлів для зберігання нейронних мереж, що дозволяє розробникам використовувати моделі, навчені за допомогою одного фреймворку, у інших фреймворках без необхідності перенавчання або значної переробки коду.

Ця можливість перетворення і сумісності є особливо корисною в умовах, коли розробники хочуть використовувати передові нейронні мережі, розроблені в академічних дослідженнях або в інших фреймворках, для вирішення реальних

бізнес-задач. ONNX усуває бар'єри, що виникають через різницю в інструментах та дозволяє використовувати передові досягнення машинного навчання без прив'язки до конкретного фреймворку.

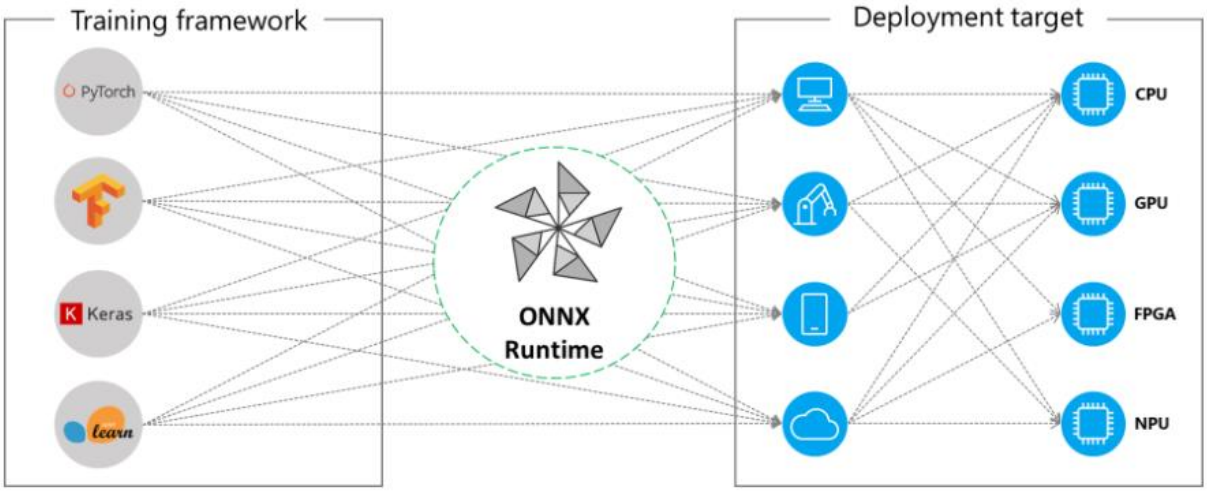


Рис. 3.7 – Застосування ONNX. У нашому випадку ми переносимо нейронну мережу натреновану на PyToch для використання на GPU

Однією з ключових переваг ONNX є його підтримка широкого спектру фреймворків. Наприклад, моделі, розроблені у PyTorch, TensorFlow, Keras або Microsoft Cognitive Toolkit, можуть бути легко перетворені до формату ONNX. Це робить ONNX ідеальним вибором для компаній та розробників, які хочуть максимізувати використання своїх моделей у різноманітних додатках і сервісах.

ONNX також відіграє важливу роль у оптимізації продуктивності. Коли модель перетворена до формату ONNX, вона може бути додатково оптимізована для конкретних обчислювальних платформ, таких як GPU або FPGA. Це означає, що модель може бути налаштована для ефективного використання ресурсів конкретного обладнання, що підвищує її швидкість та ефективність.

ONNX також підтримує концепцію графічних оптимізацій, де обчислювальні графи моделі можуть бути трансформовані для підвищення ефективності. Ці оптимізації можуть включати в себе скорочення непотрібних операцій, об'єднання

шарів для зменшення затримки та використання більш ефективних алгоритмів для певних математичних операцій.

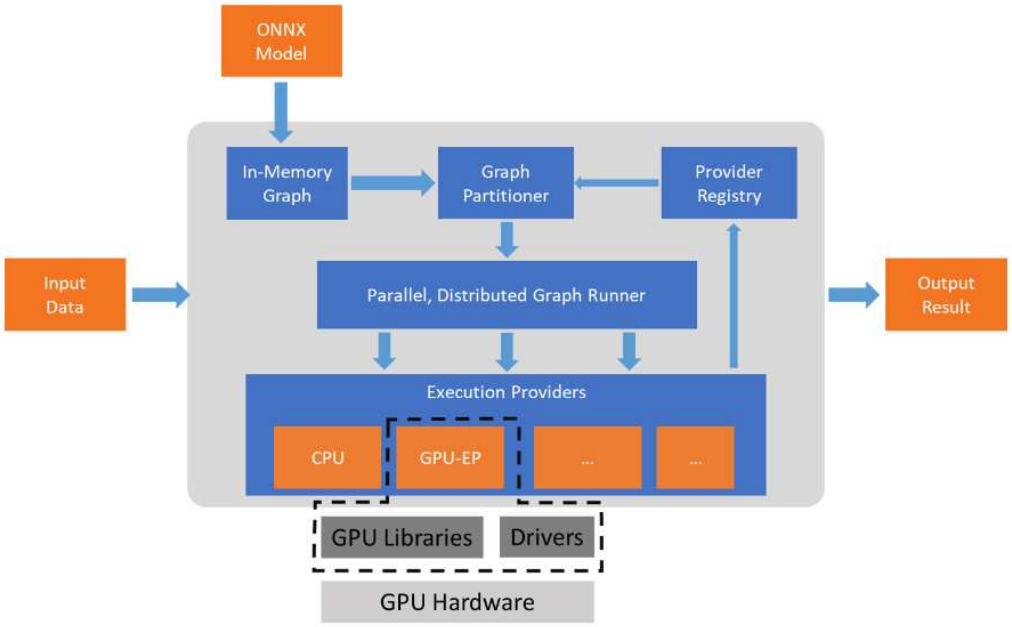


Рис. 3.8 – Опис основних принципів (функцій) роботи ONNX

Таким чином, ONNX представляє собою потужний інструмент, який забезпечує гнучкість, швидкість та ефективність для нейронних мереж, підвищуючи якість та швидкість розробки та впровадження моделей машинного навчання. Це робить ONNX незамінним ресурсом у сучасному швидкозмінному світі машинного навчання та штучного інтелекту.

Ми будемо використовувати `torch.onnx.export` для конвертації моделі у формат ONNX (.onnx). Цей підхід використовує TorchScript бек-енд і потребує прогнати тензор для того, щоб побудувати граф. При чому не важливі вхідні значення, лише формат має бути int (так, як на вхід подаються індекси з BPE словника) та відповідна формат тензора (shape), щоб прохід був математично можливим.

3.5.2 Квантизація для вагів нейронних мереж для оптимізації швидкості роботи

Наступним кроком для покращення швидкості роботи моделі є квантизування вагів [63]. Квантизація є процесом зменшення точності чисел, які використовуються для зберігання та обробки вагів і активацій у моделях. Головна мета квантизації полягає в тому, щоб зменшити обсяг пам'яті, який займає модель, та підвищити швидкість її обчислень, що є важливим для впровадження моделей в обмежені обчислювальні середовища, такі як мобільні пристрої або вбудовані системи, а також для оптимізації швидкодії на серверах.

Однак квантизація також має свої виклики. Зменшення точності даних може призвести до втрати інформації, що потенційно може вплинути на точність моделі. Тому важливо ретельно тестувати квантизовану модель, щоб переконатися, що вона зберігає прийнятний рівень точності для заданого застосування.

Процес квантизації зазвичай включає в себе два основних етапи: калібрування та перетворення. Під час калібрування модель проходить через набір даних, щоб визначити оптимальні параметри для квантизації (наприклад, мінімальні та максимальні значення для кожного шару). На етапі перетворення ці параметри використовуються для заміни Float32 ваг та активацій на їхні Int8 еквіваленти [64].

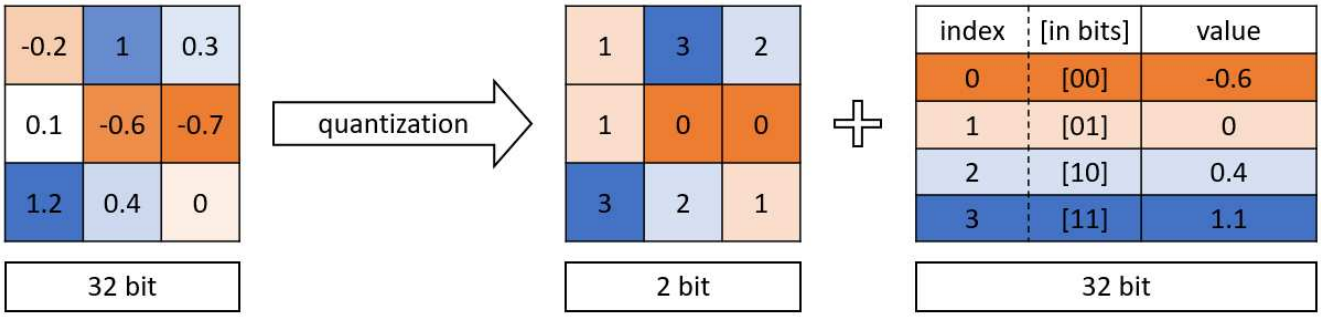


Рис. 3.9 – Схематичний приклад роботи квантизації

Існують різні рівні та методи квантизації, але двома з найпопулярніших є перехід від Float32 (32-бітні числа з плаваючою комою) до FP16 (16-бітні числа з плаваючою комою) і до Int8 (8-бітні цілі числа).

Розглянемо спочатку особливості першого підходу – FP16 (Half-Precision Floating-Point):

- FP16 займає половину пам'яті, порівняно з Float32, що дозволяє збільшити пропускну спроможність і зменшити використання пам'яті.
- Особливо ефективний для використання на GPU, які оптимізовані для обробки FP16, дозволяючи швидше виконувати операції з нейронними мережами.
- Хоча FP16 забезпечує меншу точність порівняно з Float32, для багатьох завдань цього достатньо, і втрати в точності часто незначні.

Далі перейдемо до наступного підходу – Int8 (8-Bit Integer Quantization), який теж є доволі популярним:

- Int8 квантизація зменшує точність до 8-бітних цілих чисел, що забезпечує ще більшу економію пам'яті та швидкість обчислень.
- Використання Int8 може бути особливо ефективним у вбудованих системах та мобільних пристроях, де обмеження пам'яті та обчислювальної потужності є критичними.
- Однак перехід до Int8 вимагає ретельного калібрування та тестування, оскільки більш значна зміна точності може вплинути на поведінку та точність моделі.

Застосування квантизації у ONNX моделях є ключовою стратегією для покращення ефективності виконання нейронних мереж, особливо коли мова йде про застосування моделей у середовищах з обмеженими обчислювальними ресурсами, таких як мобільні пристрої чи вбудовані систем.

ONNX, як універсальний формат для нейронних мереж, підтримує квантизацію, дозволяючи трансформувати високоточні моделі в більш компактні та ефективні версії. При застосуванні квантизації у моделях ONNX, важливо провести ретельне калібрування, щоб забезпечити, що зниження точності не призведе до значної втрати у якості виконання задачі. Калібрування часто включає в себе процес збору статистики про розподіл даних у моделі, щоб визначити оптимальні діапазони для квантизації. Після калібрування, модель може бути перетворена до квантизованої форми, готової до впровадження.

Таблиця 3.15 – Швидкість роботи моделей в залежності від застосування бібліотеки ONNX та квантизації

Формат моделі	Точність вагів	Швидкість генерації (мс)	Assistant Score
PyTorch	FP32	1.68	221.6
ONNX	FP32	1.59	221.6
ONNX	FP16	1.24	221.5
ONNX	INT8	0.91	220.3

Квантизація може бути особливо корисною для моделей, які використовуються в додатках реального часу, де швидкість відгуку є критичною, наприклад, в застосунках для перекладу мови, розпізнавання мовлення або комп'ютерного зору. В цих випадках, квантизовані ONNX моделі можуть забезпечити потрібну продуктивність без значної втрати у точності.

Важливо також враховувати, що квантизація може бути не завжди придатна для деяких типів задач або даних. Наприклад, у випадках, де необхідна висока точність (наприклад, у медичних додатках), може бути доцільніше зберегти вищу точність даних.

Для оцінки швидкості роботи моделі, ми будемо використовувати кількість оброблених речень на секунду з набору даних NUCLE. Швидкість умовної

генерації залежить в тому числі від кількості згенерованих токенів, а тому відрізняється в залежності від задачі. Наприклад, задача виправлення помилок в середньому повертає більше токенів ніж було у вхідній послідовності, задача перефразування залишає її незмінною, а спрощення тексту очікувано зменшує вхідну кількість токенів. Швидкість ж генерації 1 токена залишається незмінною, тому це є хорошим орієнтиром для оцінки роботи моделі.

Також зафіксуємо і апаратне забезпечення. Для усіх експериментів зі швидкістю було використано NVIDIA Tesla V100. Кількість речень що оброблюються паралельно (batch size) становить 100. Нагадаємо, що у попередньому розділі ми вирішили використовувати модель BART з адаптерами розмірністю прихованого шару 256.

Далі коли ми обраховуємо швидкість генерації, то рахуємо час на генерацію одного токена, враховуючи паралельний батчинг. Тобто якщо якщо ми за 10 секунд згенерували 20 токенів з батчем 30. То швидкість генерації $= \frac{10}{20 \cdot 30}$

Як бачимо і бібліотека ONNX, і квантизація помітно пришвидшують швидкість роботи обраної моделі письмового асистенту. В той же час, якість на більшості етапів не втрачається (або втрачається мінімально), а розмір моделі зменшується у декілька разів, що робить її цілком придатною для використання на мобільних пристроях. Також слід зазначити, що з квантизацією пам’ять на GPU вивільнюється, а тому можна суттєво збільшувати розмір групи (batch size), що має зробити швидкість генерації ще кращою.

Таким чином, у даній дисертаційні роботі, ми отримали ефективну та якісну нейронну мережу здатну виконувати 3 задачі письмового асистента: виправлення граматичних та орфографічних помилок, спрощення тексту та перефразування. Усі 3 задачі модель виконує на відносно високому рівні, показуючи кращі результати за велику кількість окремих спеціалізованих моделей.

ВИСНОВКИ

На основі результатів, отриманих протягом роботи над дисертаційним дослідженням, можна зробити наступні висновки:

1. Детальний аналіз трьох задач умовної генерації тексту (виправлення граматичних та орфографічних помилок, спрощення тексту та перефразування) показав ефективність існуючих наборів даних, метрик оцінки якості та провідних практик для цих задач. Найкращі підходи мають певні спільні риси та ідеї, які були застосовані у роботі.
2. Комбінований метод оцінки для якості роботи систем на задачах умовної генерації тексту дозволяє надійно оцінювати, як добре модель виконує задачі письмового асистенту, враховуючи важливість задач для письмового асистенту, якість існуючих тестувальних наборів даних та чутливість обраних метрик.
3. Новий набір даних для тренування письмових асистентів, зібраний використовуючи останні здобутки у напрямку обробки природних мов - анотацію за допомогою великих мовних моделей, суттєво покращує якість розглянутих моделей незалежно від архітектури. Якість залежить від вибору вхідних описів для кожної з трьох задач.
4. Спеціалізовані моделі, розроблені для кожної з обраних задач, демонструють перевагу над іншими методами, особливо у плані адаптації до специфічних умов завдань і тренувальних даних. Найбільш перспективні архітектури нейронних мереж для умовної генерації тексту: RNN, CNN, трансформер, BART.
5. Універсальна нейронна мережа, здатна виконувати різноманітні завдання письмового асистента, відкриває можливості для підвищення ефективності роботи. Архітектура адаптер дозволяє створення єдиної мережі без значної втрати якості.
6. Фінальна універсальна модель успішно оптимізована для забезпечення швидшої роботи без значних втрат у якості. Використання технологій,

таких як ONNX та квантизація вагів, сприяє ефективному впровадженню
цих моделей для практичного застосування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hilbert, M., & Lopez, P. (2011). The World's Technological Capacity to Store, Communicate, and Compute Information. *Science*, *332*(6025), 60-65. https://doi.org/10.1126/science.1200970.
2. *ChatGPT*. chat.openai.com. (2023). Retrieved 1 June 2023, from https://chat.openai.com/.
3. *AI Writing Assistant Software Market Size, Share, Trends & Forecast*. www.verifiedmarketresearch.com. (2023). Retrieved 12 July 2023, from https://www.verifiedmarketresearch.com/product/ai-writing-assistant-software-market/
4. Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 311–318. https://doi.org/10.3115/1073083.1073135
5. Lin, C. Y. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, 74–81.
6. Banerjee, S., & Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, 65–72.
7. Xu, W., Napoles, C., Pavlick, E., Chen, Q., & Callison-Burch, C. (2016). Optimizing statistical machine translation for text simplification. In *Transactions of the Association for Computational Linguistics*, *4*, 401-415. https://doi.org/10.1162/tacl_a_00107
8. Kincaid, J. P., Fishburne Jr, R. P., Rogers, R. L., & Chissom, B. S. (1975). Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel.
9. Dahlmeier, D., & Ng, H. T. (2012). Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter*

of the Association for Computational Linguistics: Human Language Technologies, 568-572.

10. Bryant, C. J., Felice, M., & Briscoe, E. (2017). Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 793–805. https://doi.org/10.18653/v1/P17-1074
11. Snover, M., Dorr, B., Schwartz, R., Micciulla, L., & Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, 223–231.
12. Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., & Artzi, Y. (2019). Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*. https://doi.org/10.48550/arXiv.1904.09675
13. *Quora Duplicate Questions | Kaggle*. www.kaggle.com. (2023). Retrieved 13 June 2023, from https://www.kaggle.com/code/aymenmouelhi/quora-duplicate-questions/notebook
14. Wieting, J., & Gimpel, K. (2017). ParaNMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. *arXiv preprint arXiv:1711.05732*. https://doi.org/10.18653/v1/P18-1042
15. Cheng, S. C., Chen, J. J., Yang, C., & Chang, J. S. (2018, August). LanguageNet: Learning to Find Sense Relevant Example Sentences. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, 99–102.
16. Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, 740–755. https://doi.org/10.48550/arXiv.1405.0312
17. Fu, Y., Feng, Y., & Cunningham, J. P. (2019). Paraphrase generation with latent bag of words. *Advances in Neural Information Processing Systems*, 32. https://doi.org/10.48550/arXiv.2001.01941

18. Guo, Y., Liao, Y., Jiang, X., Zhang, Q., Zhang, Y., & Liu, Q. (2019). Zero-shot paraphrase generation with multilingual language models. *arXiv preprint arXiv:1911.03597*. https://doi.org/10.48550/arXiv.1911.03597
19. Patro, B. N., Kurmi, V. K., Kumar, S., & Namboodiri, V. P. (2018). Learning semantic sentence embeddings using sequential pair-wise discriminator. *arXiv preprint arXiv:1806.00807*. https://doi.org/10.48550/arXiv.1806.00807
20. Dahlmeier, D., Ng, H. T., & Wu, S. M. (2013, June). Building a large annotated corpus of learner English: The NUS corpus of learner English. In *Proceedings of the eighth workshop on innovative use of NLP for building educational applications*, 22–31.
21. Tajiri, T., Komachi, M., & Matsumoto, Y. (2012, July). Tense and aspect error correction for ESL learners using global context. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 198–202.
22. Yannakoudakis, H., Briscoe, T., & Medlock, B. (2011, June). A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, 180–189.
23. Bryant, C., Felice, M., Andersen, Ø. E., & Briscoe, T. (2019, August). The BEA-2019 shared task on grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, 52–75. https://doi.org/10.18653/v1/W19-4406
24. Omelianchuk, K., Atrasevych, V., Chernodub, A., & Skurzhanskyi, O. (2020, July). GECToR–Grammatical Error Correction: Tag, Not Rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, 163–170. https://doi.org/10.18653/v1/2020.bea-1.16
25. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32. https://doi.org/10.48550/arXiv.1906.08237

26. Zhao, W., Wang, L., Shen, K., Jia, R., & Liu, J. (2019). Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. *arXiv preprint arXiv:1903.00138*. https://doi.org/10.18653/v1/N19-1014
27. Stahlberg, F., & Kumar, S. (2021). Synthetic data generation for grammatical error correction with tagged corruption models. *arXiv preprint arXiv:2105.13318*. https://doi.org/10.48550/arXiv.2105.13318
28. Zhao, Y., Chen, L., Chen, Z., & Yu, K. (2020, April). Semi-supervised text simplification with back-translation and asymmetric denoising autoencoders. In *Proceedings of the AAAI Conference on Artificial Intelligence (34, 05)*. 9668–9675. https://doi.org/10.48550/arXiv.2004.14693
29. Xu, W., Callison-Burch, C., & Napoles, C. (2015). Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, *3*, 283–297. https://doi.org/10.1162/tacl_a_00139
30. Martin, L., Fan, A., De La Clergerie, É., Bordes, A., & Sagot, B. (2020). MUSS: multilingual unsupervised sentence simplification by mining paraphrases. *arXiv preprint arXiv:2005.00352*. https://doi.org/10.48550/arXiv.2005.00352
31. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., ... & Zettlemoyer, L. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*. https://doi.org/10.48550/arXiv.1910.13461
32. Zhao, S., Meng, R., He, D., Andi, S., & Bambang, P. (2018). Integrating transformer and paraphrase rules for sentence simplification. *arXiv preprint arXiv:1810.11193*. https://doi.org/10.18653/v1/D18-1355
33. Ganitkevitch, J., Van Durme, B., & Callison-Burch, C. (2013, June). PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 758–764.
34. Omelianchuk, K., Raheja, V., & Skurzhanskyi, O. (2021, April). Text Simplification by Tagging. In *Proceedings of the 16th Workshop on Innovative*

Use of NLP for Building Educational Applications, 11–25. https://doi.org/10.48550/arXiv.2103.05070

35. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*. https://doi.org/10.48550/arXiv.1706.03762
36. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*. https://doi.org/10.48550/arXiv.1301.3781
37. Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing*, 1532–1543. https://doi.org/10.3115/v1/D14-1162
38. Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, *5*, 135–146. https://doi.org/10.48550/arXiv.1607.04606
39. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533–536.
40. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.
41. Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*. https://doi.org/10.48550/arXiv.1412.3555
42. Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. (2017, July). Convolutional sequence to sequence learning. In *International conference on machine learning*, 1243–1252. https://doi.org/10.48550/arXiv.1705.03122
43. Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*. https://doi.org/10.48550/arXiv.1409.0473

44. Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, *27*. https://doi.org/10.48550/arXiv.1409.3215

45. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, *33*, 1877-1901. https://doi.org/10.48550/arXiv.2005.14165

46. Mathur, N., Baldwin, T., & Cohn, T. (2020). Tangled up in BLEU: Reevaluating the evaluation of automatic machine translation evaluation metrics. *arXiv preprint arXiv:2006.06264*. https://doi.org/10.18653/v1/2020.acl-main.448

47. Qorib, M., Na, S. H., & Ng, H. T. (2022, July). Frustratingly easy system combination for grammatical error correction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1964–1974. https://doi.org/10.18653/v1/2022.naacl-main.143

48. Vu, T., Hu, B., Munkhdalai, T., & Yu, H. (2018). Sentence Simplification with Memory-Augmented Neural Networks. In *Proceedings of NAACL-HLT*, 79–85. https://doi.org/10.18653/v1/N18-2013

49. Skurzhanskyi, O., & Marchenko, O. (2022, December). Task-specific pre-training improves models for paraphrase generation. In *Proceedings of the 2022 6th International Conference on Natural Language Processing and Information Retrieval*, 44–48. https://doi.org/10.1145/3582768.3582791

50. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, *32*. https://doi.org/10.48550/arXiv.1912.01703

51. Zhao, W., Wang, L., Shen, K., Jia, R., & Liu, J. (2019). Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. *arXiv preprint arXiv:1903.00138*. https://doi.org/10.18653/v1/N19-1014

52. Kiyono, S., Suzuki, J., Mita, M., Mizumoto, T., & Inui, K. (2019). An empirical study of incorporating pseudo data into grammatical error correction. *arXiv preprint arXiv:1909.00502*. https://doi.org/10.18653/v1/D19-1119

53. Rothe, S., Mallinson, J., Malmi, E., Krause, S., & Severyn, A. (2021). A simple recipe for multilingual grammatical error correction. *arXiv preprint arXiv:2106.03830*. https://doi.org/10.48550/arXiv.2106.03830

54. Wang, Z., Hamza, W., & Florian, R. (2017). Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*. https://doi.org/10.48550/arXiv.1702.03814

55. Prakash, A., Hasan, S. A., Lee, K., Datla, V., Qadir, A., Liu, J., & Farri, O. (2016). Neural paraphrase generation with stacked residual LSTM networks. *arXiv preprint arXiv:1610.03098*. https://doi.org/10.48550/arXiv.1610.03098

56. Miao, N., Zhou, H., Mou, L., Yan, R., & Li, L. (2019, July). Cgmh: Constrained sentence generation by metropolis-hastings sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence (33, 01)*, 6834–6842. https://doi.org/10.48550/arXiv.1811.10996

57. Fabre, B., Urvoy, T., Chevelu, J., & Lolive, D. (2021, April). Neural-Driven Search-Based Paraphrase Generation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, 2100–2111. https://doi.org/10.18653/v1/2021.eacl-main.180

58. Martin, L., De La Clergerie, É. V., Sagot, B., & Bordes, A. (2020, May). Controllable Sentence Simplification. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, 4689–4698. https://doi.org/10.48550/arXiv.1910.02677

59. Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., ... & Gelly, S. (2019, May). Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, 2790–2799. https://doi.org/10.48550/arXiv.1902.00751

60. Ivison, H., & Peters, M. E. (2022). Hyperdecoders: Instance-specific decoders for multi-task NLP. *arXiv preprint arXiv:2203.08304*. https://doi.org/10.48550/arXiv.1910.02677
61. Keskar, N. S., McCann, B., Varshney, L. R., Xiong, C., & Socher, R. (2019). Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*. https://doi.org/10.48550/arXiv.1909.05858
62. *GitHub - onnx/onnx: Open standard for machine learning interoperability*. github.com. (2023). Retrieved 25 June 2023, from https://github.com/onnx/onnx/tree/main.
63. Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., & Keutzer, K. (2022). A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, 291–326. https://doi.org/10.48550/arXiv.2103.13630
64. Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., ... & Wu, H. (2017). Mixed precision training. *arXiv preprint arXiv:1710.03740*. https://doi.org/10.48550/arXiv.1710.03740

ДОДАТКИ

Додаток А

Список праць здобувача за темою дисертації

Статті у наукових фахових виданнях України:

1. Skurzhanskyi O., & Marchenko A. (2021). Review of neural approaches for conditional text generation. *Bulletin of Taras Shevchenko National University of Kyiv. Series: Physics and Mathematics*, *1*, 102–107. https://doi.org/10.17721/1812-5409.2021/1.13
2. Скуржанський О., Марченко О., & Анісімов, А. (2024). Спеціалізоване попереднє навчання нейромережевих моделей на синтетичних даних для покращення генерації перефразувань. *Кібернетика та системний аналіз, том 60*, 3–12. https://doi.org/10.34229/KCA2522-9664.24.2.1
3. Skurzhanskyi O., & Marchenko A. (2023). Neural approaches for writing assistant tasks. *Bulletin of Taras Shevchenko National University of Kyiv. Series: Physics and Mathematics*, *2*, 232–238. https://doi.org/10.17721/1812-5409.2023/2.40

Праці, які засвідчують апробацію матеріалів дисертації:

4. Omelianchuk, K., Atrasevych, V., Chernodub, A., & Skurzhanskyi, O. (2020). GECToR–Grammatical Error Correction: Tag, Not Rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, 163–170. https://doi.org/10.18653/v1/2020.bea-1.16
5. Omelianchuk, K., Raheja, V., & Skurzhanskyi, O. (2021). Text Simplification by Tagging. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications,* 11–25. https://doi.org/10.48550/arXiv.2103.05070
6. Skurzhanskyi, O., & Marchenko, O. (2022). Task-specific pre-training improves models for paraphrase generation. In *Proceedings of the 2022 6th International Conference on Natural Language Processing and Information Retrieval*, 44–48. https://doi.org/10.1145/3582768.3582791
7. Skurzhanskyi, O. (2023). Distillation of Large Language Models for Text Simplification. *Modeling, control and information technologies: Proceedings of VI*

International scientific and practical conference, 230–231. https://doi.org/10.31713/MCIT.2023.071

Праці, які додатково відображають наукові результати дисертації:

8. Скуржанський О. (2021). Збагачення семантичного представлення тексту в нейронних мережах морфологічною інформацією для задачі виправлення граматичних та орфографічних помилок. *Systems and measures of artificial intelligence AIIS'2021*, 104–110.