КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра теоретичної кібернетики

Кваліфікаційна робота
на здобуття ступеня бакалавра
за спеціальністю 122 Комп'ютерні науки
на тему:

СУМАРИЗАЦІЯ ТЕКСТУ ЗА ДОПОМОГОЮ АРХІТЕКТУРИ ТРАНСФОРМЕРУ

Виконав студент 4-го курсу
Омельченко Роман Олексійович _____ (підпис)

Науковий керівник:
доцент, кандидат фіз.-мат. наук
Трохимчук Ростислав Миколайович

_____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри теоретичної кібернетики
«___» _____
2021 р., протокол № ___
Завідувач кафедри
проф. Крак Ю.В. _____
(підпис)

Київ – 2022

ЗМІСТ

ВСТУП

Суммаризація – це процес скорочення набору певних даних, коли за допомогою певних обчислень створюється підмножина, що зберігає основну суть початкового змісту.

Зараз існує два основних способи суммаризувати тексти: абстрактивний і екстактивний підходи. Це одна з найважливіших галузей обробки природної мови.

Зачатки походження цієї сфери лежать у спробах розв’язку задач машинного перекладу під час Другой світової війни. Дещо більш активного розвитку галузь обробки природної мови почала набувати ближче до 1950-х років завдяки науковим розробкам у сфері сигтаксичної теорії мови, а також не в останню чергу новим парсинговим алгоритмам.

Перша публікація в цій галузі датується 1957 роком використовує певні статистичні техніки. Інтерес до досліджень у сфері сумаризації текстів значно виріс починаючи з 2015 року.

До 2016 року основним і найпотужнішим методом узагальнення одного чи кількох документів було узагальнення на основі шаблонів.

У 2016 з’явився латентний семантичний аналіз(LSA), що комбінуючись з невід’ємною матричною факторизацією перевершував попередні підходи, хоча і не замінив їх повністю, а часто навіть поєднувався з ними.

До 2020 року переважна частина досліджень на цю тему велась про екстрактивну сумаризацію текстів і методи машинного навчання теж були переважно спрямовані на цей напрям наукового пізнання.

Але приблизно з 2020 року основний фокус науковців почав зсуватись у бік абстрактивної сумаризації та сумаризації у реальному часі.

Актуальність роботи зумовлена зростаючим інтересом світової наукової спільноти та software-індустрії в цілому до теми обробки природної мови(Natural Language Processing) та появи нових технологічних підходів до синтезу та аналізу текстів та мовлення.

Також приводів обрати саме цю тему як провідну для даної бакалаврської випускної роботи додає стрімкий розвиток нових рішень у галузі машинного навчання, таких як моделі з архітектурою трансформеру, що прийшли на зміну рекурентним штучним нейромережам.

Метою роботи є вивчення підходів до аналізу і подальшої сумаризації документів, а також порівняння ефективності різних моделей сумаризації. Переважна увага у даній роботі буде приділена саме абстрактивним методам, як більш цікавим з точки зору досліджень.

Завдання: обрати декілька Multilanguage-моделей, проаналізувати їх будову, знайти серед них не навчену заздалегідь для роботи з сумаризацією українською мовою модель, навчити її сумаризувати тексти українською мовою та проаналізувати ефективність її роботи, можливі недоліки та аспекти і особливості практичного застосування. Вказати можливі шляхи покращення майбутніх результатів.

Об’єкт дослідження: Multilanguage-модель для аналізу природної мови.

Методи розроблення: технічні засоби на мові програмування високого рівня Python.

Можливі сфери застосування: академічна сфера, засоби спрощення навчального процесу.

Обсяг і структура роботи: робота складається зі вступу, постановки задачі, чотирьох розділів, майбутніх покращень та висновку. Повний обсяг роботи складає 52 сторінки. Список використаних джерел містить 21 найменування.

ПОСТАНОВКА ЗАДАЧІ

Основною задачу, яку я ставлю перед собою під час написання цієї роботи, це налаштувати певну модель архітектури трансформера, не преднавчену для роботи з українськими текстами, для автоматичного створення резюме текстів українською мовою і проаналізувати її ефективність і можливі проблеми, які можуть завадити її роботі.

РОЗДІЛ 1. ПОНЯТТЯ СУМАРИЗАЦІЇ, ІСТОРІЯ І ТЕОРЕТИЧНЕ ПІДҐРУНТЯ ПИТАННЯ.

1.1.1 Поняття сумаризації

Сумаризація – це процес побудови більш стислої версії для певного набору даних, із включенням у неї найбільш важливих даних зі змісту кожного з документів.

Сумаризація може бути виконаною як для тексту, так і для медіафайлів, таких як зображення та відео. Сумаризація тексту спрямована на пошук найбільш інформативних речень та слів у документі. Задача сумаризації зображень розв’язується підходами пов’язаними з пошуком зображень з певного набору зображень таким чином, щоб знайдена послідовність найкраще характеризувала початкову множину зображень. У свою чергу задача сумаризації відеофайлів витягує найважливіші кадри з відеопотоку.

Історія сумаризації текстів це один із основних напрямків практичного ужитку методів обробки природної мови(ОПМ). Отже щоб краще розібратися у питанні оглянемо історію ОПМ.

Історичним зачином для ОПМ було завдання машинного перекладу у часи Другої світової війни. Але ефективних алгоритмів для перекладу довгий час не існувало. У 1980-х та 1990-х з підвищенням інтересу до досліджень у галузі штучного інтелекту, дослідники почали більше цікавитись питаннями обробки природної мови. У цей час разом з розвитком ОПМ, пов’язані з ним теми, такі як статистична обробка природних мов, витягування інформації та автоматичної сумаризації також набули розповсюдження у науковій спільноті.

Спочатку текстова сумаризація текстів була виконувана виключно з використанням алгоритмів заснованих на правилах(rule-based algorythms), які працювали оцінюючи різні частини тексту відповідно до їх важливості. Такий підхід мав назву оцінювач важливості і використовував концепцію ієрархічної пропозиційної мережі(Hierarchical Proposition Network), де певні концептуальні одиниці отримували числові подання речень для оцінки їх важлиості.

На сьогоднішній момент існує два підходи до автоматичної сумаризації текстів: екстрактивний та абстрактивний.

1.1.2 Екстрактивна сумаризація

Суть цього методу – це витягнути дані з документу без жодних змін. Обрані з тексту зразки інформації включають ключові фрази, або речення і заголовки, для зручної розмітки документа, які разом складають стислу репрезентацію вхідних файлів, причому зазвичай саме речення, оскільки їх витягування допомагає зберегти правильність і цілісність граматичної структури текстового резюме.

Тобто по суті ми виокремлюємо певну підмножину речень з початкового тексту, у випадку текстової сумаризації.

Екстрактивна сумаризація складається х трьох незалежних етапів:

1. Конструкція проміжного подання тексту.

Існує два таких типи підходів, базованих на проміжних поданнях: предметна репрезентація, та індикаторна репрезентація.

Предметна репрезентація трансформує текст в проміжне подання з урахуванням тем, обговорених у тексті. Основні техніки які використовуються для такого підходу поділяються на:

- імовірнісні підходи
- підходи з визначенням тематичних слів
- латентний семантичний аналіз
- Байєсівські моделі тем.

Індикаторна репрезентація представляє кожне речення як список формальних ознак – індикаторів – важливості, наприклад:

- довжина речення

- позиція у документі
- включення певних фраз
- тощо.

2. Оцінювання речень на основі проміжного подання.

Коли побудована проміжна репрезентація, кожному реченню присвоюється оцінка важливості. У випадку предметної репрезентації оцінка речення репрезентує як добре речення доводить найважливіші думки з тексту.

У індикаторній репрезентації оцінка обчислюється об’єднанням показників зважених індикаторів.

3. Побудова стислого подання тексту з певної кількості речень.

Сумаризатор обирає k найбільш важливих речень щоби скласти резюме тексту. Деякі підходи використовують жадібні алгоритми щоби вибрати найбільш важливі речення, деякі розглядають вибір речень як оптимізаційну задачу, де обирається набір речень з урахуванням умови максимізації важливості та узгодженості висловлювань, та мінімізації повторень.

1.1.3 Абстрактивна суммаризація. Модель трансформеру

Цей вид сумаризації використовується переважно для текстових файлів. Абстрактивні методи будують семантичні конструкції оригінального тексту і використовують їх для створення резюме схожим способом, яким це виконав би людський мозок.

Абстрактивні методи трансформують текст перефразуванням початкових фрагментів таким чином стискаючи його краще за екстрактивні. Звичайно такі методи є набагато складнішими за екстрактивні і потребують використання обробки природної мов*и.*

Донедавна найкращим способом враховувати часові залежності у послідовностях при обробці текстів були рекурентні нейронні мережі. Але існує архітектура лише з механізмами уваги, яка може покращити результати у задачах перекладу, сумаризації і не тільки. Називається вона моделлю трансформера.

Трансформер це модель глибокого навчання, яка унаслідувала механізм уваги, за допомогою якого вона успішно відділяє важливі фрагменти тексту.

Нижче на рисунку 1 продемонстровано схему архітектури трансформеру.

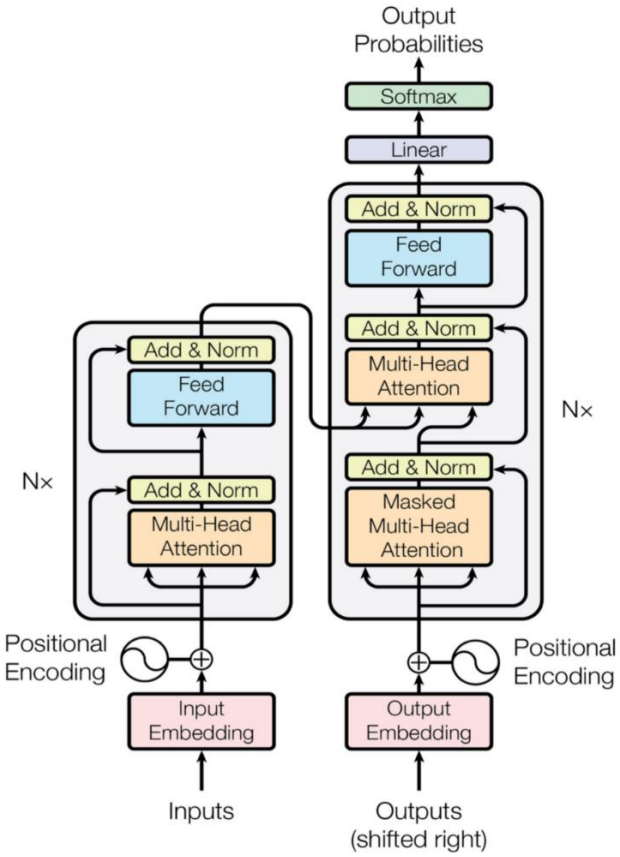


Рисунок 1 - схема архітектури трансформеру

Механізм уваги дозволяє розуміти контекст для кожного елемента із вхідної послідовності, тому на відміну від рекурентних нейронних мереж трансформер може обробляти весь вхідний файл одночасно.

Як більш ранні моделі sequence-to-sequence моделі, трансформер містить у своїй будові архітектуру енкодер-декодер:

- Енкодер складається з кодувальних шарів, кожен з яких послідовно обробляє вхідні дані.
- Декодер складається з декодувальних шарів, які послідовно розкодовують вивід енкодера.

І декодер і енкодер складаються з модулів, які можуть розміщуватись один поверх іншого декілька разів. Можна побачити з наведеного вище малюнка, що у їх

будові переважно знаходяться шари Multi-Head Attention та нейронної мережі прямого поширення.

Вхідні і вихідні дані спочатку вбудовуються у n-вимірний простір. Так як у будові трансформеру немає рекурентних нейронних мереж, то і відносні позиції слів додаються у якості вбудованих подань кожного слова, ці подання будуть n-вимірними векторами.

Механізм уваги зважує для кожного вхідного елементу послідовності релевантність усіх інших елементів, та враховує це під час генерації вихідних даних. Кожен шар декодера має окремий механізм уваги, що черпає інформацію з результатів роботи попередніх шарів декодеру до того як на вхід декодера буде передана інформація з кодувань.

У свою чергу і шари декодера, і шари енкодера мають нейронну мережу прямого поширення для додаткової обробки вхідних даних і містять залишкові з’єднання та кроки нормалізації шарів.

Енкодер відображує вхідну послідовність символьної репрезентації тексту (x1,….,xn) на множину послідовних представлень z=(z1,…,zn). Декодер отримує собі на вхід множину z і на її основі генерує вихідну послідовність (y1,…,ym) символів поелементно у кожну одиницю часу.

На кожному кроці модель лишається авторегрессивною, тобто отримує на вхід попередньо згенеровані символи як додаткові ввідні дані, щоби на їх основі згенерувати свій подальший текстовий вихід.

Енкодер складається з множини потужності N=6 однакових шарів, кожен шар має два підшари:

- Перший з вищеописаних містить Multi-Head механізм самоуваги
- другий це проста позиційна нейронна мережа прямого поширення, за яким імплементується залищковий зв’язок за кожним із підшарів, після яких слідує нормалізація шару.

Тобто вихідні дані з кожного підшару це LayerNorm(x + Sublayer(x)), де Sublayer(x) це функція, реалізована кожним підшаром. Задля облегшення реалізації та функціонування цих залишкових з’єднань усі підшари у моделі, як і вбудовуючі шари виробляє вихідні дані розмірності $d_{model} = 512$.

Отже, основна мета кожного шару енкодера це генерувати такі кодування, які показують зв’язки певних частин тексту між собою. Потім попередній шар передає кодування на наступний шар у якості вхідних даних.

Декодер також як і енкодер складається з множини потужності N=6 ідентичних шарів, але на додачу до двох підшарів енкодера також має у своїй конструкції і третій підшар, який реалізовує операцію Multi-Head Attention над вихідними даними з шарів енкодера. Аналогічно до енкодера у декодера імплементовано залишкові зв'язки навколо обох підшарів, за якими слідує обов'язкова нормалізація шарів. У підшарі самоуваги у стеку шарів декодера механізм самоуваги дещо модифіковано, щоби забезпечити недосяжність доступу до наступних позицій за допомогою маски уваги(Attention Mask). Це маскування у купі з фактом, що вихідні вбудовування зміщені на одну позицію забезпечує залежність передбачень щодо згенерованих вихідних даних на позиції *i* лише від позицій менших за *i* , тобто лише від відомих даних.

Тобто, у свою чергу функція кожного з шарів на етапі декодера отримувати кодування енкодера у якості вхідних даних і генерувати вихідну текстову послідовність на основі енкодерної інформації про контекст та взаємозв'язки.

1.1.4 Концепція Multi-Head Attention

Механізм уваги у трансформері можна представити рівнянням (8):

(8)

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

Де Q – це матриця запитів(векторних репрезентацій одного слова з послідовності, розмірності d_q), K – ключі (векторна репрезентація усіх слів у послідовності, розмірності d_k) і V – це значення(векторна репрезентація усіх слів у послідовності, розмірності d_v).

Для пари енкодер-декодер та для модулів Multi-Head Attention V складається з тієї ж послідовності слів, що і Q, але для модуля уваги, який враховує послідовності енкодера та декодера V відрізняється від послідовності, представленої Q.

Ми обчислюємо точкові добутки запитів Q зі всіма ключами K, ділимо кожен з них на квадратний корінь з розмірності d_k та проганяємо через активаційну функцію softmax для отримання вагових коефіцієнтів значень у межах від 0 до 1.

Двома найбільш широкоуживаними функціями уваги є адитивна увага та точково-добуткова увага.

Вони є майже ідентичними, та адитивна увага відрізняється лише відсутністю фактору масштабування $1/\sqrt{d_k}$.

Адитивна увага обраховує функцію сумісності з використанням нейронної мережі прямого поширення з одним прихованим шаром.

Хоча ці дві функції теоретично мають однакову швидкодію, але точково-добуткова увага на практиці є набагато більш ефективною з точки зору затрат пам’яті та часу на обчислення, оскільки може бути виконана з використанням високооптимізованих бібліотечних функцій для матричного множення.

Нижче на рисунку 2 наведена схема алгоритму обчислення точково-добуткової уваги.

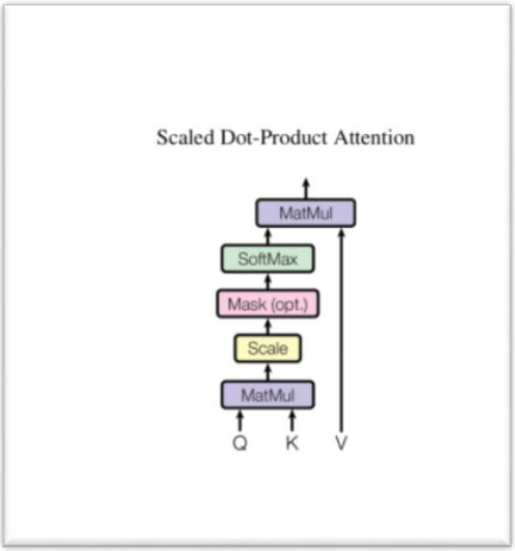


Рисунок 2 – Точково-добуткова увага

І хоча для невеликих значень d_k ці дві уваги поводяться схожим чином з точки зору продуктивності, але на більших об’ємах даних очевидною стає перевага точково-добуткової уваги.

Так як для великих значень розмірності d_k точкові добутки набувають великої амплітуди значень, то для запобігання потрапляння значень функції softmax у зони з дуже малими градієнтами проводиться масштабування точкового добутку на $1/\sqrt{d_k}$

Тобто ми можемо сказати, що значення у V множаться та сумуються з вагами уваги *a*, які визначаються формулою(9):

(9)

$$a = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)$$

Вона означає, що усі вагові коефіцієнти моделі трансформеру *a* визначені впливом на кожне слово з послідовності Q усіх інших слів, що представлені множиною K.

Окрім того, до вагів застосовується активаційна функція Softmax для нормалізації значень і отримання розподілу між 0 та 1.

Ці вагові коефіцієнти потім використовуються для усіх слів з послідовності V.

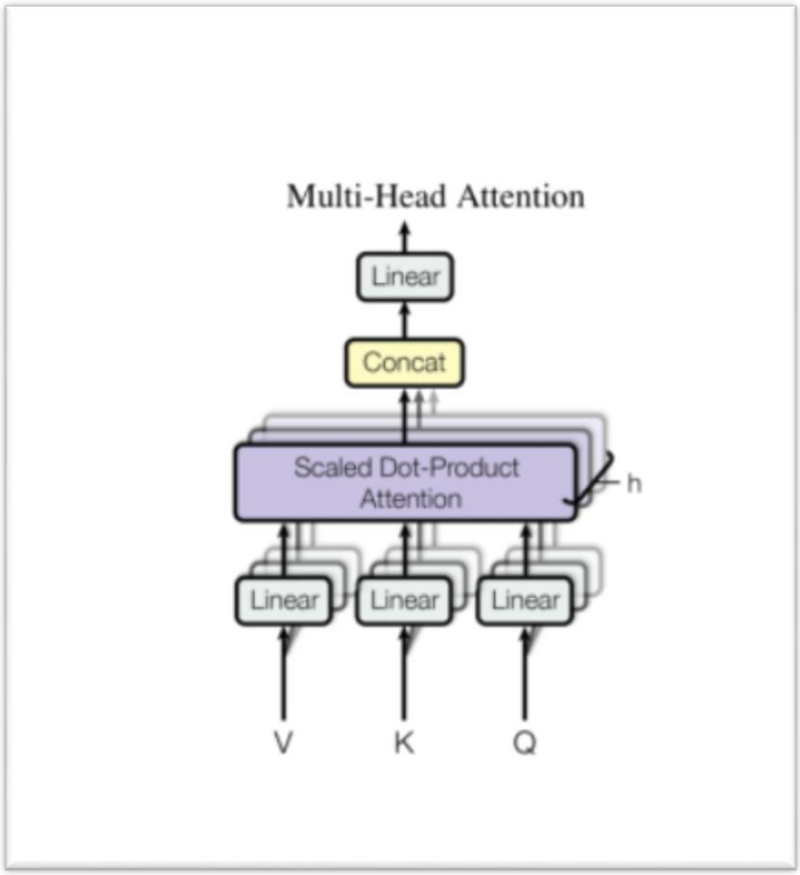
Замість того, щоб виконувати одночасно лише одну функцію уваги з ключами, значеннями та запитами розмірності d_{model} вбачається доцільнішим h разів лінійно спроєктувати ключі, значення та запити на різні результати тренувань моделі розмірностей d_k, d_q, d_v відповідно.

Тож Multihead-attention механізм може бути розподілений на декілька паралельно виконуваних механізмів задля оптимізації швидкодії та покращеної ефективності.

Механізм уваги виконується кілька разів, приймаючи на вхід лінійні проекції множин Q, K, V, що відкриває системі можливість навчатися на різних поданнях Q, K, V і таким чином покращує результати роботи.

Схему роботи механізму багатоголової уваги продемонстровано рисунком 3

Рисунок 3 – Багатоголова увага



Лінійні репрезентації отримуються шляхом матричного множення Q, K, V на матрицю вагів W, яка є результатом тренування моделі трансформеру.

Кожну лінійно спроєктовану версію множини параметрів функції уваги подаємо у дану функцію на вхід і запускаємо паралельне виконання, з отриманням у подальшому результат розмірності d_v, після чого усі отримані дані з'єднуємо за допомогою операції конкатонації і ще раз лінійно проєктуємо їх на множини ключів, значень і запитів, це і буде результатом роботи функції Multihead-attention.

Формула MultiHead-attention (10):

(10)

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

Де проекції це d_{model} d_k параметричні матриці $W_i^Q \epsilon \mathbb{R}^{d(model)\times d(k)}$, $W_i^K \epsilon \mathbb{R}^{d(model)\times (dk)}$, $W_i^V \epsilon \mathbb{R}^{d(model)\times d(v)}$ і $W_i^O \epsilon \mathbb{R}^{hd(v)\times d(k)}$

Матриці Q, K, V різнитимуться у відповідності до місцезнаходження модулів уваги у структурі моделі, у залежності від того чи належать вони енкодеру, чи декодеру, чи-то розміщені між декодером і енкодером. Причиною цьому є те, що іноді нам треба мати доступ до цілої вхідної послідовності енкодера, а іноді нам досить частини вхідної послідовності для декодера.

Модуль Multi-Head Attention, з'єднуючи разом у єдину систему енкодер і декодер, має враховувати вхідну послідовність енкодера разом з вхідною послідовністю декодера до певної позиції.

Архітектура моделі трансформеру використовує ідею Multi-Head Attention трьома різними способами:

1. У декодерно-енкодерному шарі запити надходять з попереднього шару декодера, а ключі пам'яті та значення надходять з вихідної інформації енкодера. Це дозволяє кожній позиції декодера мати доступ до даних у всіх позиціях вхідної послідовності тексту. Ця поведінка є досить схожою на поведінку типічного енкодерно-декодерного механізму уваги у моделях типу послідовність-до-послідовності.
2. Енкодер містить шари самоуваги. Це такий шар, де усі ключі, значення та запити надходять з одного і того ж самого місця, а в даному разі це саме вихідні дані попереднього шару енкодера. Кожна позиція з енкодера має доступ до усіх позицій попереднього шару енкодера.
3. Аналогічно до другого пункту, декодер також має шари самоуваги, які дозволяють кожному елементу мати доступ до усіх попередніх позицій включаючи позицію даного елементу. Ми маємо переконатись, що декодер не має доступу до інформації зліва задля забезпечення властивості авторегресивності, для цього ми всередині імплементації функції точкового добутку маскуємо усі непотрібні з'єднання присвоюваням їм значення мінус нескінченності(- ∞) на етапі введення вхідних даних у функцію активації softmax.

Після Multi-Head Attention модулів у енкодері і декодері структурно розміщується точкова нейронна мережа прямого поширення. Параметри цієї компактної нейромережі прямого поширення ідентичні для кожної позиції. Їх можна описати як лінійне перетворення застосоване до кожного елемента із вхідної текстової послідовності.

послідовність на основі енкодерної інформації про контекст та взаємозв’язки.

1.1.5 Позиційні нейронні мережі прямого поширення

На додачу до підшарів з реалізацією механізму уваги кожен з підшарів стеку шарів енкодера та декодера містить повнозв’язну нейронну мережу прямого поширення.

Ця мережа застосовується до кожної позиції ідентичним чином і застосовується незалежно. Модель цієї мережі складається з двох лінійних перетворень, із застосуванням між ними активаційної функції ReLU. що може бути представлене рівнянням (11):

(11)

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

Лінійні трансформації на різних позиціях лишаються однаковими, але у залежності від їх приналежності до певного шару моделі трансформеру вони використовують відповідно і різні вхідні параметри.

1.1.6 Вбудовування

Так як у переважаючій більшості випадків моделі, що мають собі на меті перетворення однієї текстової послідовності символів у іншу, модель архітектури трансформеру, як одна із sequence-to-sequence моделей, за основу бере використання попередньо навчені вбудовування для конвертації вхідних та вихідних послідовностей токенів у вектори розмірності d_{model} .

Також у даній моделі використовуються лінійні трансформації та функція активації Softmax, задля реалізації завдання конвертації вихідних даних декодера у згенеровані імовірності наступних токенів.

У архітектурі трансформеру використовується спільна матриця вагових коефіцієнтів для двох вбудовуючих шарів та лінійної трансформації, передуючої використанню активаційної функції Softmax.

У вбудовуючих шарах моделі ці вагові коефіцієнти множаться на $\sqrt{d_{model}}$.

1.1.7 Кодування позицій

Так як до складу основних композиційних елементів моделі архітектури трансформеру не входить жодного шару, що реалізував би рекурентний чи згортковий механізм функціонування нейронної мережі, вбачається необхідним уведення вхідної інформації щодо відносної чи абсолютної позиції токенів у вхідній послідовності, щоби ми мали змогу відслідковувати порядок елементів тексту.

Тож для розв'язання цієї проблеми до вхідних вбудовувань додаються позиційні кодування на дні стеків декодера та енкодера. Позиційні кодування будуть мати розмірність d_{model} , таку ж саму, як і вбудовування, тому вони можуть додаватись один до одного.

Існує багато різних способів обрати підходящий шлях обробки і передачі моделі sequence-to-sequence даних про позиційне кодування. У архітектурі трансформеру для реалізації процесу кодування позицій текстових елементів використовуються синусоїди і косинусоїди різних періодів, формули (11) та (12):

(11)

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

(12)

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

У даних формулах *pos* слугує позначенням позиції елемента, а *i* відповідно позначення розмірності. Тобто можна сказати, що кожен вимір позиційного кодування відповідає певній синусоїді. Довжини хвиль синусоїдних функцій формують числовий ряд геометричної прогресії від 2π до 10000*2π.

Дана функція була обрана через її здатність облегшити моделі навчитися отримувати доступ до даних за відносними позиціями, так як для будь-якого зафіксованого зміщення k, PE_{pos+k} можна подати як лінійну функцію PE_{pos}.

1.1.8 Порівняння аспектів функціонування та обґрунтування раціональності використання механізму самоуваги у моделі архітектури Трансформеру

Якщо порівнювати різні особливості та аспекти шарів самоуваги зі згортковими та рекурентними, що часто використовуються для відображення однієї послідовності символьних репрезентацій $(x_1, \ldots, x_n)$ на іншу послідовність однакової довжини $(z_1, \ldots, z_n)$, де x_i , $z_i \in \mathbb{R}^d$, такі як приховані шари у складі конструкції типічного перетворювача послідовностей, наприклад енкодер та декодер, то знайдемо три основних причини для того щоб вважати механізм самоуваги кращим:

1. Загальна складність обчислень на кожному шарі моделі архітектури трансформеру.
2. Кількість обчислень, які можна паралелізувати, яка вимірюється за найменшою кількістю необхідних послідовних операцій, задіяних у процесі.
3. Довжина шляху між залежностями, розміщеними на великих дистанціях у нейронній мережі.

Пошук локації та аналітичне вивчення далеких залежностей моделі найголовнішою задачею у багатьох завданнях перетворення та конвертації послідовностей символів.

Одним з найголовніших чинників, прямо впливаючих на даний процес, - це довжина шляхів, якими мають долати відстані прямі та зворотні інформаційні сигнали нейронної мережі.

Чим компактнішими та коротшими будуть шляхи між будь-якою комбінацією позицій із вхідних та вихідних текстових послідовностей, тим більш простим вбачається ров’язання проблеми пошуку розміщення далеких зв’язків моделі.

Для кращої ілюстрації максимального розмаху шляху між двома вхідними та вихідними позиціями наведемо наступну таблицю:

Тип шару	Складність операцій проводимих у певному шарі мережі	Послідовні операції	Максимальна довжина шляху
Самоуваги	$O(n^2*d)$	O(1)	O(1)
Рекурентний	$O(n*d^2)$	O(n)	O(n)
Згортковий	$O(k*n*d^2)$	O(1)	$O(\log_k(n))$
Самоуваги з обмеженнями	O(r*n*d)	O(1)	O(n/r)

Таблиця 1: приведені максимальні довжини шляхів, складність обчислень на один шар мережі, мінімальна кількість послідовних операцій для кожного типу шарів, *n* позначає довжину послідовності символів, *d* – це представлення розмірності, *k* – це ядрова кількість згорток, *r* – розмір вибірки сусідніх елементів у обмеженому механізмі самоуваги

Тобто як зазначено у таблиці вище, шари самоуваги з'єднують усі позиції за константну кількість виконаних послідовних операцій, у той самий час як рекурентні шари потребують аж O(n) послідовних операцій.

Отже у термінах алгоритмічної ефективності комп'ютерних обчислень шари самоуваги є набагато більш результативними з точки зору оптимізації швидкодії ніж рекурентні шари, у тих випадках, коли довжина n є меншою за репрезентацію розмірності параметрів d, а це найчастіше так і виходить у репрезентаціях речень, які використовуються сучасними моделями для машинного перекладу.

Але коли ми маємо справу з задачами, що потребують значно більшої довжини речень, як наприклад задача сумаризації, то тоді розв'язок потребує додаткової оптимізації шляхом обмеження розміру вибірки сусідніх елементів до розміру r у

вхідній послідовності, відцентрованій навколо відповідної позиції у вихідній послідовності.

Такі дії ведуть до збільшення максимальної довжини шляху до O(n/r).

Єдиний шар згорткової нейронної мережі, який має ширину ядра $k < n$, не є здатним з'єднати всі пари вхідних та вихідних позицій.

Для того щоб мати змогу їх об'єднати треба мати множину з потужністю O(n/k) згорткових шарів у випадку суміщених ядер, або ж потужнітю $O(\log_k(n))$ у разі якщо мати справу з розширеними згортками, збільшуючи довжину найдовшого шляху між будь-якими двома позиціями у нейронній мережі.

Згорткові шари у переважній більшості практичних випадків більш ресурсоємкі (у фактор k разів) за рекурентні шари штучних нейронних мереж. Але, розділені згортки здатні певним чином зменшити складність обчислень до рівня $O(k*n*d + n*d^2)$.

Навіть для $k = n$ складність обчислень у штучній нейронній мережі з роздільними згортками буде приблизно еквівалентною використанню комбінації з шару самоуваги та точкової мережі прямого поширення, яка і використовується у моделі трансформеру.

Також однією з непрямих переваг використання саме механізму самоуваги буде набагато краща і ясніша з точки зору інтерпретуємості модель.

Це теж робить застосування саме даного механізму, а не інших аналогічних йому підходів, більш бажаним для даної архітектури.

РОЗДІЛ 2. ОГЛЯД ІСНУЮЧИХ ПІДХОДІВ

У своїй роботі я вирішив зосередитись на моделях, інтегрованих у бібліотеку huggingface однойменної компанії. Серед основних моделей, інтегрованих у неї варто виділити такі:

- Bert
- RoBERTa
- T5(Text-to-Text Transfer Transformer)

2.1.1 BERT

Дослідження словесних репрезентацій, які можуть бути активно застосовані при вирішенні різних проблем програмування, було і є активною сферою наукового пошуку впродовж як мінімум останніх десятиліть, та включало в себе як методи засновані на використанні нейронних мереж, так і за допомогою не нейронних підходів.

Попередньо натреновані словесні вбудовування стали основою сучасних підходів до рішень задач з аналізу природної мови. Вони є значно кращими у ефективності в порівнянні з вбудовуваннями навченими з нуля. Для того, щоб виконати завдання попереднього навчання векторів словесних вбудовувань, було використано мовні моделювання «зліва-напрво», а також ознаки для виокремлення правильно і неправильно вжитих слів у правому та лівому контекстах.

Такі підходи були узагальненими на більші сутності, такі як вбудовування речень та абзаців.

Модель ELMo та її попередники узагальнюють традиційні словесні вбудовування дослідженнями з витягування контекстно-залежних ознак з мовних моделей «зліва-направо» та «справа-наліво». Контекстуальне представлення кожного маркера є конкатенацією уявлення зліва направо і справа наліво.

BERT(Bidirectional Encoder Representations from Transformers) – це фреймворк для машинного навчання з відкритим кодом для обробки природної мови (NLP).

BERT розроблено, щоб допомогти комп’ютерам зрозуміти значення неоднозначної мови в тексті, використовуючи навколишній текст для встановлення контексту.

Фреймворк BERT був попередньо навчений з використанням тексту з Вікіпедії, і його можна точно налаштувати за допомогою наборів даних і відповідей.

У роботі даного фреймворку є два основні кроки:

1. попереднє навчання(pre-training)
2. точне налаштування(fine-tuning).

Під час попереднього навчання модель тренується на нерозмічених даних, пов'язаних з різними завданнями. Для точного налаштування необхідним є спочатку ініціалізувати модель BERT за допомогою попередньо навчених параметрів а також усіма параметрами, що будуть налаштовуватися підзадачами з використанням маркованих даних.

Кожне підзавдання має окремі точно налаштовані моделі, навіть якщо вони ініціалізовані однаковими попередньо навченими параметрами.

Архітектура моделі BERT представлена багаторівневим двонаправленим енкодером, базованим на оригінальній реалізації з бібліотеки tensor2tensor.

Кількість шарів, тобто блоків трансформеру позначено як L, розмір прихованих шарів як H, а номер голів самоуваги A.

Розмір моделі BERTBASE:

- L=12
- H=768
- A=12
- Total Parameters=110 M

Розмір моделі BERTLARGE:

- L=24
- H=1024
- A=16
- Total Parameters = 340 M

Параметри BERTBASE були обрані для однакового розміру з OpenAI GPT, щоб ці дві моделі можна було порівнювати за ефективністю.

Однак трансформер BERT використовує двонаправлену самоувагу, а трансформер GPT використовує самоувагу з обмеженнями, у якій кожен токен може звертатися лише до свого лівого контексту.

Процес претренування і налаштування моделі приведено на малюнку 4.

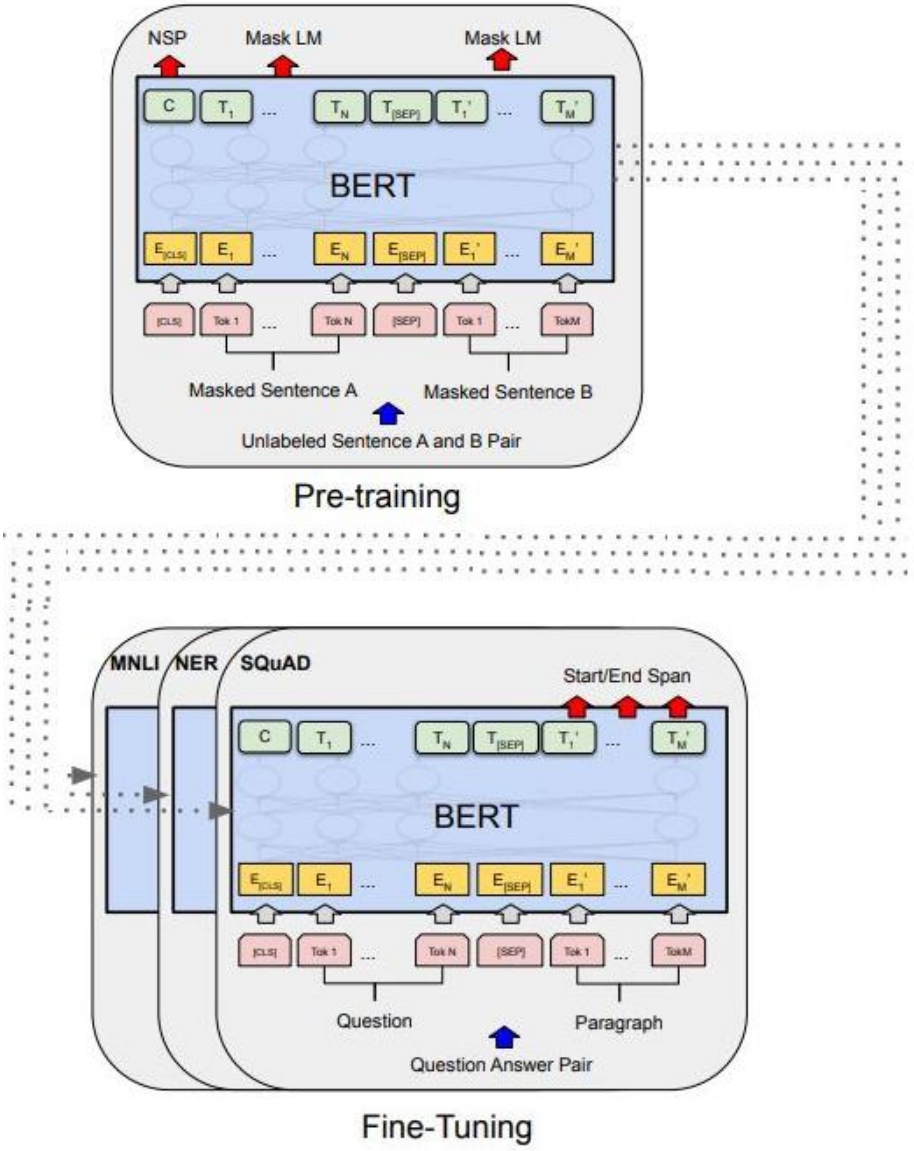


Рисунок 4 претренування і налаштування BERT

Вище наведені загальні процедури попереднього навчання та тонкого налаштування для BERT. Крім вихідних шарів, для попереднього навчання використовується така ж архітектура як і для тонкого налаштування.

Для ініціалізації моделі для різних підзадач використовуються однакові попередньо навчені параметри для їх підгонки внутріншім оптимізатором моделі архітектури BERT.

Під час точного налаштування усі параметри проходять процедуру настройки паралельно.

Щоби BERT міг успішно вирішувати різні підзадачі, репрезентація вхідних даних має бути здатною до репрезентації як єдиного вхідного речення, так і пар речень у одній послідовності токенів.

BERT використовує вбудовування WordPiece, загальний розмір словника якого складає 30000 токенів.

[CLS] – це спеціальний символ, що додається перед кожним прикладом введення.

Фінальний прихований шар, що відповідає токену [CLS], використовується для агрегації репрезентації послідовностей для задач класифікації.

Пари речень разом складаються у єдину послідовність. Речення розрізняються у два способи:

- [SEP] – це спеціальний роздільник, що наприклад розділяє запитання та відповіді
- До кожного токену додається навчене вбудовування, що позначає його належність до речення A чи B.

Ми позначаємо вхідні вбудовування як E, фінальний прихований вектор спеціального токену [CLS] як $C \in \mathbb{R}^H$, фінальний прихований вектор для i-того вхідного токена як $T_i \in \mathbb{R}^H$.

Для заданого токена вхідне представлення складається з агрегації відповідних токенізаційних, сегментарних та позиційних вбудовувань.

Зразок цього процесу наведено на рисунку 5.

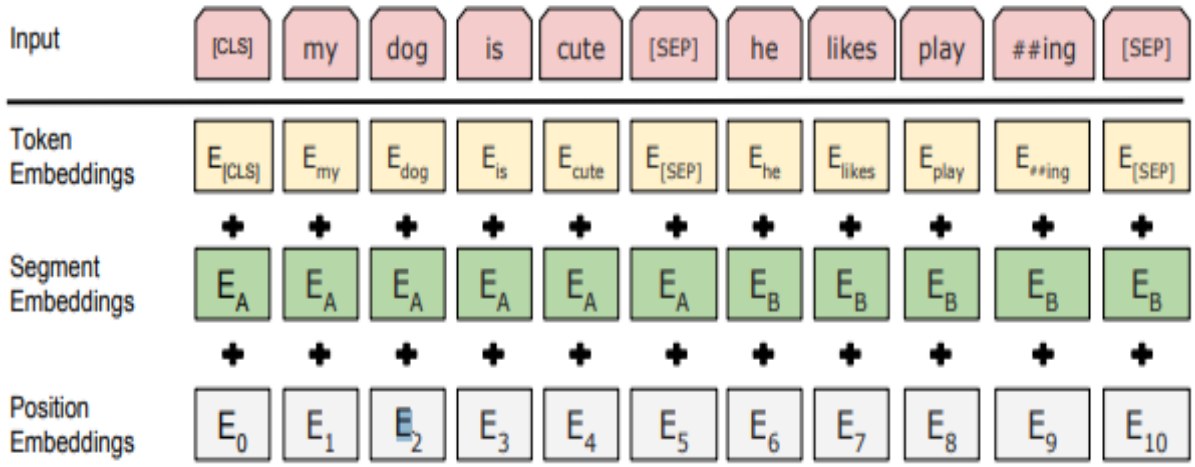


Рисунок 5 зразок вбудовувань

2.1.2 RoBERTa

Архітектура моделі BERT має певні суттєві недоліки, які були певним чином опрацьовані у його дорobленій версій від Facebook – RoBERTa(Robustly Optimized BERT Pretraining Approach).

Основними можливими полями покращення ефективності перформансу моделі BERT є:

1. Маскування під час навчання

Маскування виконується тільки під час попередньої обробки даних, з чого випливає наявність лише однієї статичної маски. Тому однакові маски вхідних даних були використані у моделі на кожній епосі.

2. Прогнозування наступного речення

Оригніальний формат ввідних даних використаний у BERT це SEGMENT-PAIR+NSP LOSS. Тобто у ньому кожні вхідні дані мають пару сегментів, де кожен може містити кілька речень, але загальна довжина обмежена 512 токенами.

Помічено було, що деякі речення шкодять ефективності системи на підзадачах, що згідно з гіпотезою дослідників траплялося через нездатність моделі відстежувати далекі залежності.

3. Кодування тексту

У оригінальній імплементації моделі BERT використовується словник символьного рівня BPE розміром 30000.

BERT використовує метод WordPiece для варіанту бінарного кодування пар у мовних моделях.

4. Розмір тренувального фрагменту

Початковий BERT натренований на 1 мільйоні кроків з розміром тренувальної партії 256 послідовностей, що показує можливість для покращення.

Для покращення описаних вище якостей моделі BERT було проведено такі дії:

1. Заміна статичного маскування на динамічне:

Щоб не маскувати одне і те ж саме слово декілька разів, команда Facebook використала динамічне маскування. Тренувальні дані повторювались до 10 разів, і кожного разу замасковане слово було іншим, отже речення було тим самим, але замасковані слова щоразу були іншими.

2. Відмова від прогнозування наступного речення.

У випадку використання SENTENCE-PAIR+NSP кожні вхідні дані містили пари речень, виокремлених з неперервної частини одного або декількох документів.

У випадку відмови від нього:

- У повних реченнях кожні ввідні дані зберігалися з повними реченнями, виокремленими з цілісного документу або з перетинів документів, загальна довжина складала 512 токенів.
- У документних реченнях вхідні дані сконструйовані схожим чином з повними реченнями, за винятком того, що вони не можуть перетинати межі документів.

Вхідні дані виокремлені булі кінців документа можуть бути коротшими за 512 символів.

Тому автори моделі RoBERTa динамічно збільшують розмір тренувального фрагменту у таких випадках щоби досягти аналогічного об'єму токенів, як і у випадку повних речень.

За результатами дослідження функції генерації наступного речення, відмова від даної особливості реалізації моделі здатна певним чином покращити її роботу на навчальних підзадачах.

На рисунку 6 наведено порівняльна ефективність моделей з та без генерації наступного речення:

Model	SQuAD 1.1/2.0	MNLI-m	SST-2	RACE
Our reimplementation (with NSP loss):				
SEGMENT-PAIR	90.4/78.7	84.0	92.9	64.2
SENTENCE-PAIR	88.7/76.2	82.9	92.1	63.0
Our reimplementation (without NSP loss):				
FULL-SENTENCES	90.4/79.1	84.7	92.5	64.8
DOC-SENTENCES	90.6/79.7	84.7	92.7	65.6
$\text{BERT}_{\text{BASE}}$	88.5/76.3	84.3	92.8	64.3
$\text{XLNet}_{\text{BASE}}$ (K = 7)	–/81.3	85.8	92.7	66.1
$\text{XLNet}_{\text{BASE}}$ (K = 6)	–/81.0	85.6	93.4	66.7

рисунок 6 – порівняння ефективності з та без генерації наступного речення

3. Тренування з великим фрагментом одночасно тренованих об’єктів із датасету

Тренування моделі за допомогою більшої кількості об’єктів у тренувальній партії покращує результати у розв’язку бажаних задач та збільшує точність моделі.

Великі фрагменти також простіше паралелізувати за розподіленого паралельного навчання.

4. Byte-Pair кодування

Використовуємо бінарне кодування необроблених символів замість символів Юнікоду.

Використано обрізаний до 50000 екземплярів словник підслів The BPE subword vocabulary.

Навіть у обрізаному стані це все ще більше, ніж кількість слів у базовому словнику моделі BERT.

Не дивлячись на невелике погіршення кінцевих результатів у деяких випадках, цей метод дозволяє універсалізувати схему кодування і не використовувати попередню обробку та правила токенізації вхідних даних.

Приклад такого кодування наведено на рисунку 7:

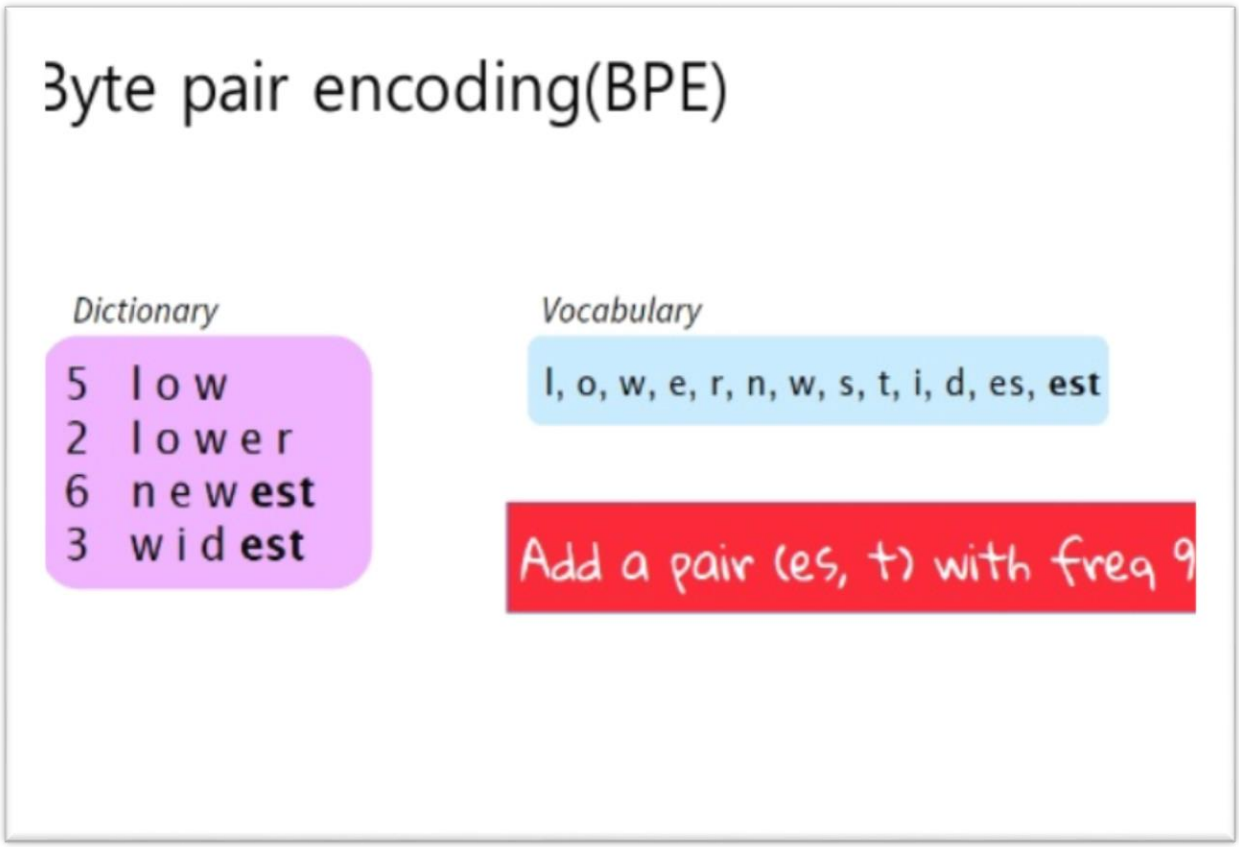


Рисунок 7 – бінарні кодування

5. Збільшення об'єму тренувальних даних

Тренування моделі BERT на більших датасетах відчутно збільшує ефективність роботи. По цій причині розмір тренувальної вибірки було збільшено до 160 гігабайтів нестиснутого тексту.

Усі ці фундаментальні зміни призвели до значного покращення результатів роботи моделі. З появою таких моделей як BERT і RoBERTa можна казати про наближення моделей для обробки природної мови до людського рівня розуміння мови.

Нижче у таблиці 2 наведено порівняльний аналіз деяких основних якісних характеристик моделей BERT та RoBERTa:

	BERT	RoBERTa
Розмір вибірки параметрів моделі (у мільйонах одиниць)	BASE: 110 LARGE: 340	BASE: 110 LARGE: 340
Час затрачений на тренування моделі	BASE: 8 x V100 x 12 днів LARGE: 64 чіпи TPU x 4 дні (чи 280 x V100 x 1 день)	LARGE: 1024 x V100 x 1 день, тобто приблизно у 4-5 разів довше за BERT
Робота моделі	Показує найкращі результати з наявних на момент 2018 року	Кращі, ніж у BERT на 2-20%
Дані на яких навчено модель	16 гігабайтний датасет Books Corpus та дані з Вікіпедії. 3.3 мільярди слів	160 гігабайтів (16 гігабайтів BERT та ще 144 додаткових даних
Метод, на якому ґрунтується робота моделі	BERT(Двосторонній енкодер)	У своїй базі має BERT без генерації наступного речення

Таблиця 2

2.1.4 Модель трансформеру T5

T5(Text-to-Text Transfer Transformer) - це модель архітектури трансформеру, що відкриває можливість працювати з великим числом задач пов’язаних з перетворенням однієї послідовності тексту в іншу.

T5 можна назвати універсальною моделлю.

Ця модель дозволяє робити як машинний переклад, так і класифікацію, регресійні задачі(наприклад, передбачити наскільки схожими є два речення між собою), такі sequence-to-sequence задачі як сумаризація документів.

На рисунку 8 наведено схему архітектури моделі T5.

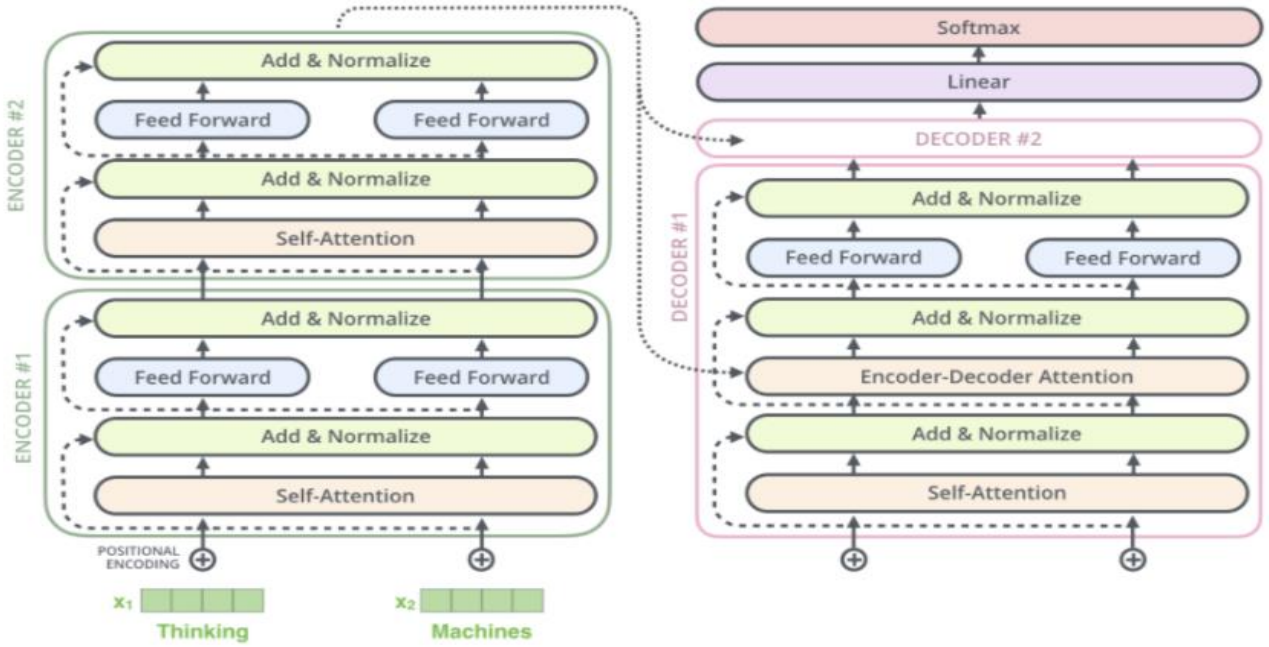


Рисунок 8 – архітектура T5

Структура T5 дуже схожа з базовими ідеями архітектури трансформеру і представляє собою просто базові декодерно-енкодерні шари з механізмам самоуваги.

Модель T5 використовує текст з різних веб-сторінок витягнутий за допомогою common crawl. Автори моделі використовують досить просту евристичне фільтрування даних: T5 прибирає усі рядки, що не закінчуються кінцевим розділовим знаком. Також T5 видаляє рядки де використовується слово javascript і будь-які сторінки з фігурними дужками.

Він дедуплікує датасет шляхом вибору повзаючого вікна із 3 фрагментів речень і дедуплікує їх так, що тільки одне з них з'являється в датасеті. У кінці модель має справу з 750 гігабайтами обраного тексту на англійській мові.

Наступний крок це пошук непідконтрольної мети задля того щоби модель мала можливість навчатися на нерозмічених даних декількома способами. У початковому тексті кілька слів викинуто з унікальними сторожовими маркерами.

Слова викидаються незалежно, із нормальним імовірнісним розподілом. Модель натренована так, щоб передбачувати в основному сторожові маркери, щоби окреслити та знайти викинутий текст. Цей процес продемонстровано на рисунку 9.

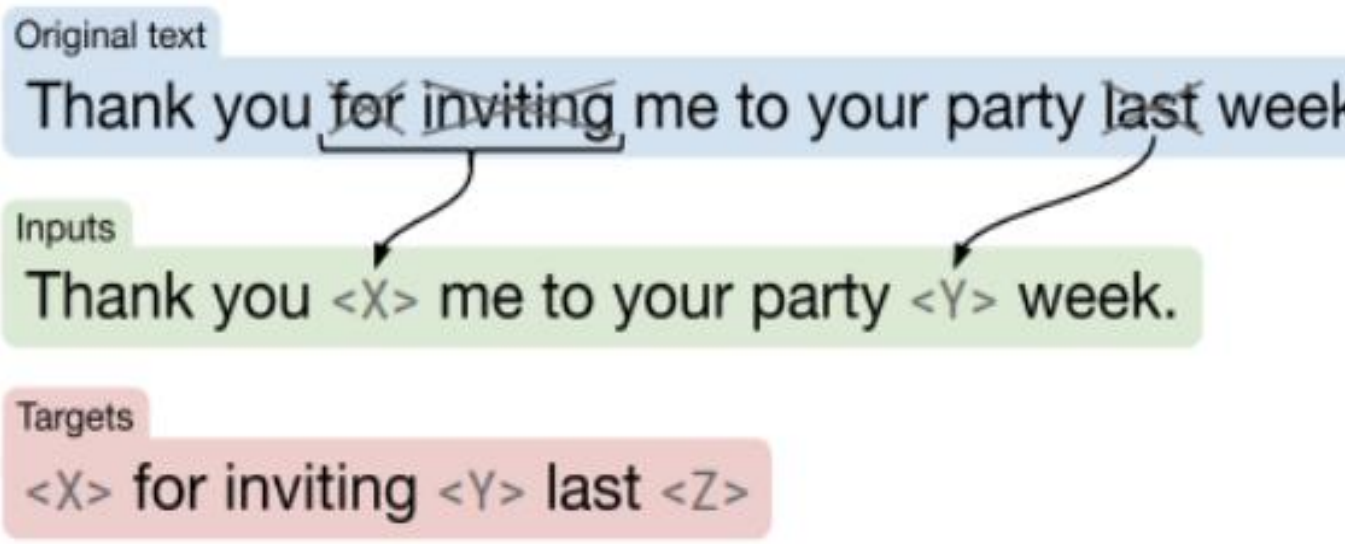


Рисунок 9 – робота з текстом T5

Попередньо навчена модель базована на архітектурі трансформера з енкодером-декодером розмірами еквівалентними до BERT-BASE. Попереднє навчання проводилось на датасеті common crawl з метою прибирання зайвого шуму тексту і тривало 2^{19} кроків на 2^{35}, або ж ~348 токенах з інверсним кореневим порядком навчання.

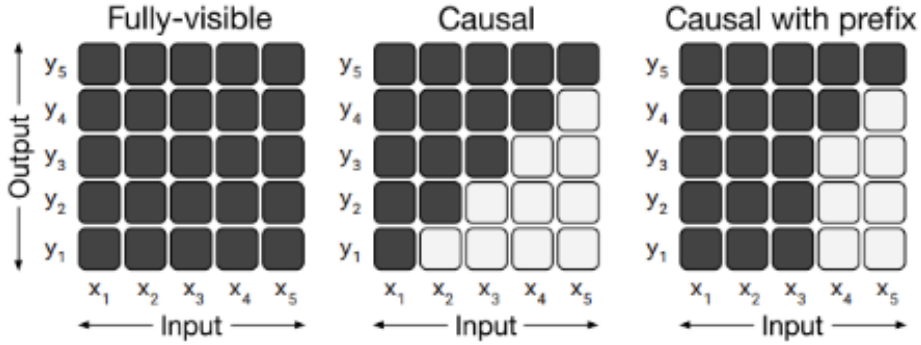
Задачі з тонкої настройки включали GLUE, CNN/DM(CNN / Daily Mail), SQuAD, SuperGLUE, задачі з перекладу, такі як: WMT14 EnDe, WMT14 EnFr, та WMT14 EnRo.

Також ця модель має варіанти архітектури:

1. Повністю спостережувана маска – це такий тип маски, де кожен вихідний елемент має доступ до читання кожного вхідного.

2. Звичайна маска – дає добрі результати для прогнозування послідовностей, адже модель не може дивитись у наступні елементи для передбачення.
3. Звичайна маска з префіксом – дозволяє моделі дивитись на перший біт вхідної послідовності і далі модель починає прогнозувати наступні елемени вхідної послідовності.

Далі на рисунках 10,11 приведені паттерни маски уваги, а нижче – відповідні їм варіанти архітектури моделі.



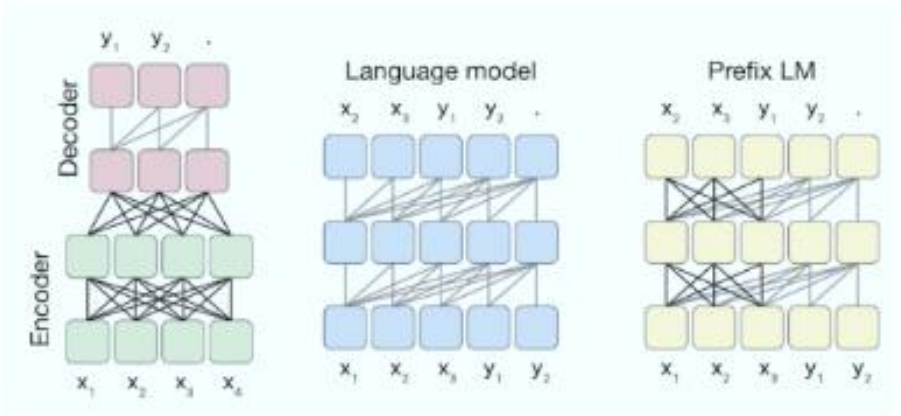


Рисунок 10, 11 – патерни масок уваги і відповідні їм моделі архітектури

Модель є мультизадачною. Замість навчання без учителя та тонкої настройки, модель навчена на багатозадачності. Є декілька стратегій змішування завдань:

- Кожне завдання може тренуватись однаковий проміжок часу
- Зважувати кожен тренувальний датасет за кількістю тренувальних екземплярів, щоби не перетренувати мережу на малому датасеті і не недотренувати на великому

Через те, що завдання навчання без учителя досить великі, варто

встановлювати штучний ліміт кількості тренувань на кожному окремому завданні.

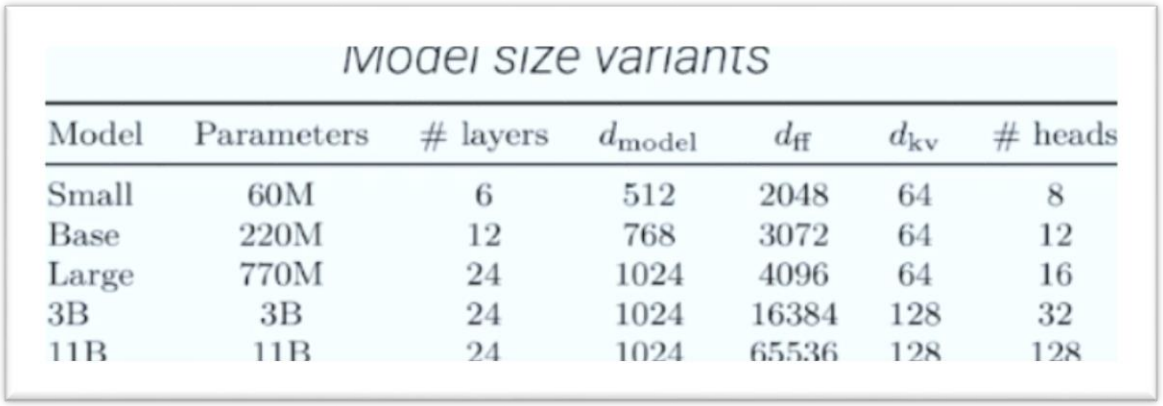
Також можна взяти прорції кількості екземплярів, та визначити температуру на них, щоби наблизити їх до однорідності.

Також окрім мультизадачності є також і інші техніки тренування моделі.

Автори цієї моделі натренували її на 5 варіантах:

- T5-small
- T5-base
- T5-large
- T5 з $3*10^9$ параметрами
- T5 з $11*10^9$ параметрами

Нижче на рисунку 12 наведені варіанти моделі T5

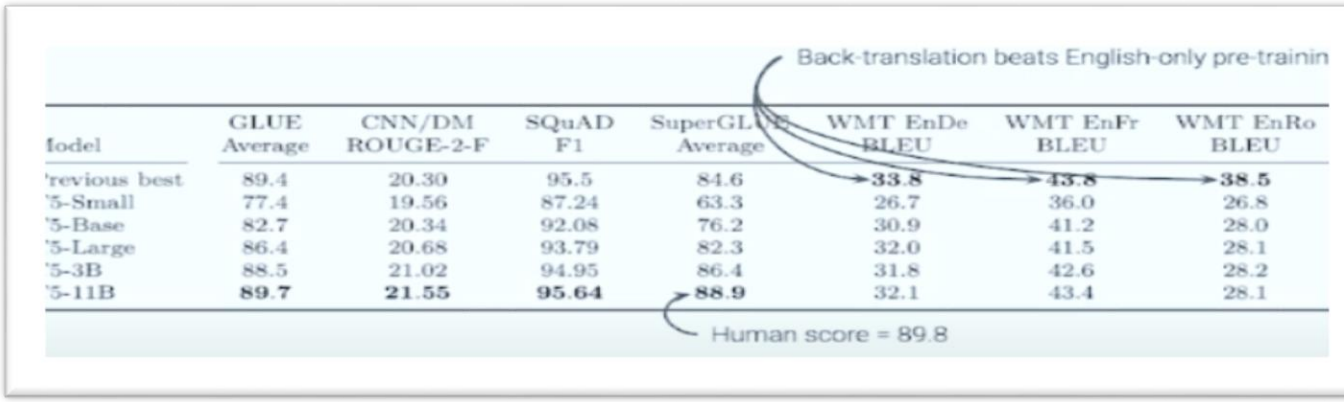


Model size variants

Model	Parameters	# layers	d_{model}	d_{ff}	d_{kv}	# heads
Small	60M	6	512	2048	64	8
Base	220M	12	768	3072	64	12
Large	770M	24	1024	4096	64	16
3B	3B	24	1024	16384	128	32
11B	11B	24	1024	65536	128	128

рисунок 12 – варіанти моделі T5

Результати роботи різних за розміром варіантів моделі на різних завданнях і датасетах продемонстровано рисунком 13:



Model	GLUE Average	CNN/DM ROUGE-2-F	SQuAD F1	SuperGLUE Average	WMT EnDe BLEU	WMT EnFr BLEU	WMT EnRo BLEU
Previous best	89.4	20.30	95.5	84.6	**33.8**	**43.8**	**38.5**
T5-Small	77.4	19.56	87.24	63.3	26.7	36.0	26.8
T5-Base	82.7	20.34	92.08	76.2	30.9	41.2	28.0
T5-Large	86.4	20.68	93.79	82.3	32.0	41.5	28.1
T5-3B	88.5	21.02	94.95	86.4	31.8	42.6	28.2
T5-11B	**89.7**	**21.55**	**95.64**	**88.9**	32.1	43.4	28.1

рисунок 13 – варіанти моделі та їх ефективність

РОЗДІЛ 3. ЗАПРОПОНОВАНИЙ ПІДХІД ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1 Вибір інструментів реалізації

У процесі вибору підходу до розв’язку поставленої задачі, я вирішив проаналізувати у порівнянні підходи, описані у минулому розділі.

Порівнюючи ефективність роботи на датасеті MS MARCO(датасет, що складається з 8.8 мільйонів уривків і 1 мільйону пошукових запитів) за метрикою MRR@10, можна побачити, що ефективність різних типів моделі T5 значно покращується зі зростанням кількості параметрів моделі.

Також помітним є перевага навіть базової версії T5 над моделлю BERT за представленою метрикою. Це представлено рисунком 14:

		MS MARCO Passage	
	# Params	Dev	Test
BM25	-	0.184	0.186
+ BERT-large	340 M	0.372	0.365
+ T5-base	220 M	0.381	-
+ T5-large	770 M	0.393	-
+ T5-3B	3 B	**0.398**	0.388

Рисунок 14 – порівняння ефективності моделей BERT та T5 за метрикою MRR@10

Зі зростанням загальної кількості тренувальної вибірки датасету MS MARCO для моделі, модель T5-BASE поводить себе кращим чином даних під час розв’язку задач передбачених цим датасетом, ніж BERT-BASE після навчання на аналогічних вибірках.

Цікавим є також значна різниця у роботі на невеликих об’ємах даних: на критично малих датасетах(мова йде про приблизно 1000 екземплярів) модель T5 набагато краща за результатами роботи за BERT-BASE.

Зі збільшенням тренувального датасету різниця між моделями поступово зменшується, але на дуже великих вибірках даних(530 тисяч елементів) все ще зберігається перевага T5. На рисунку 15 показано графік покращення метрики

MRR@10 від тисяч одиниць у тренувальному наборі моделі:

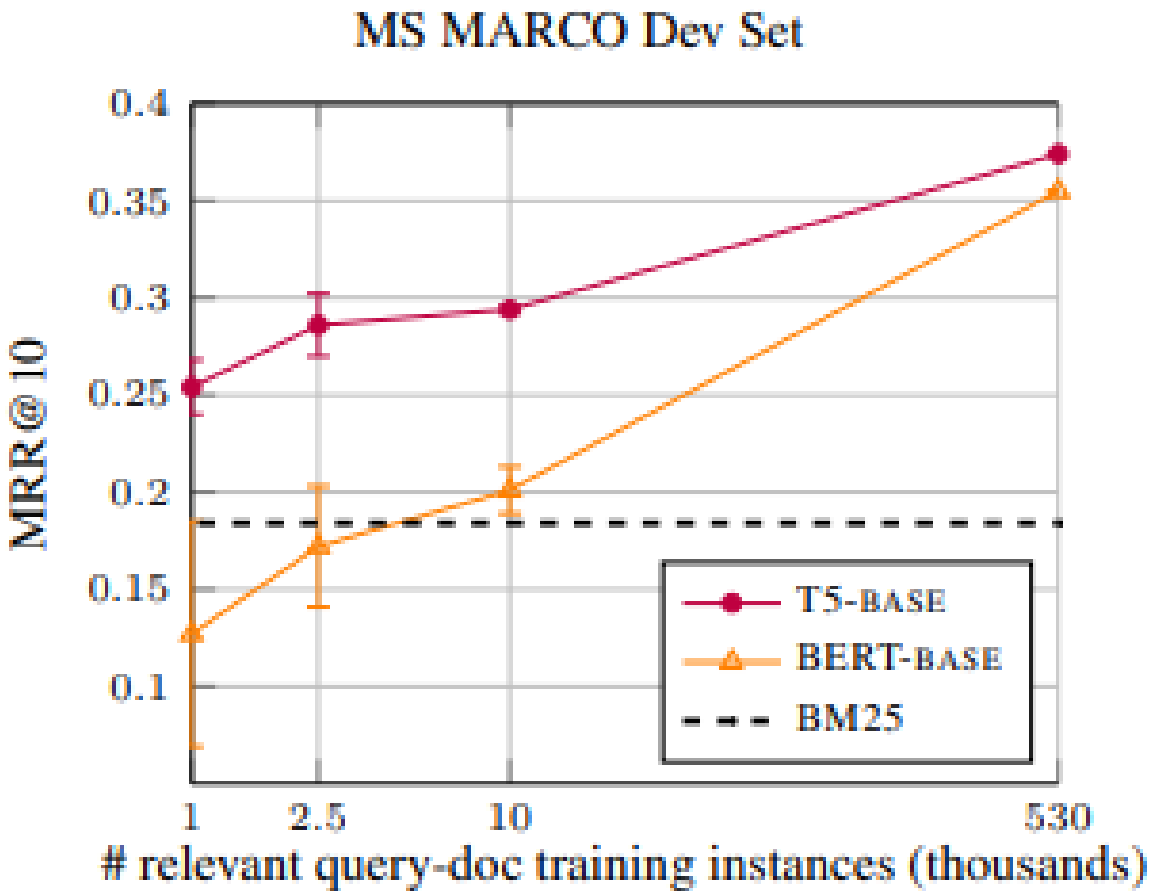


Рисунок 15 – Залежність покращення результатів від тисяч тренувальних екземплярів

На датасеті Robust04 моделі T5 також показали себе дещо краще за BERT.

За певних особливостей тренувань модель T5-BASE може видавати кращі результати за версії реалізації моделі BERT з великою і максимальною кількістю параметрів.

Також модель T5 здатна обійти у ефективності розв’язку багатьох завдань і багатьох наступників BERT, та просто його покращені версії.

Однією з причин цьому слугує велика кількість параметрів моделі T5 та її структурні особливості архітектури. Так, велика версія T5 має майже утричі більше параметрів, ніж її попередник BERT. Порівняння роботи на датасеті Robust04 наведено на рисунку 16:

	Robust04		
Model	AP	nDCG@20	Jdg@20
Birch	0.3697	0.5325	-
BERT–MaxP	-	0.5453	-
PARADE	-	0.5713	-
BM25	0.2531	0.4240	0.9770
+ T5-base	0.3279	0.5298	0.9158
+ T5-large	0.3288	0.5345	0.8906
+ T5-3B	**0.3876**	**0.6091**	0.9632
BM25 + RM3	0.2903	0.4407	0.9764
+ T5-base	0.3340	0.5532	0.9058
+ T5-large	0.3382	0.5287	0.8840
+ T5-3B	**0.4062**	**0.6122**	0.9588

Рисунок 16

Порівнюючи ефективність T5 та RoBERTa під час вирішення запитань з багатьма варіантами вибору, який порушує багато аспектів здорового глузду, можна побачити, що за аналогічних об’ємів даних RoBERTa справляється дещо краще за T5, це продемонстровано рисунком 17:

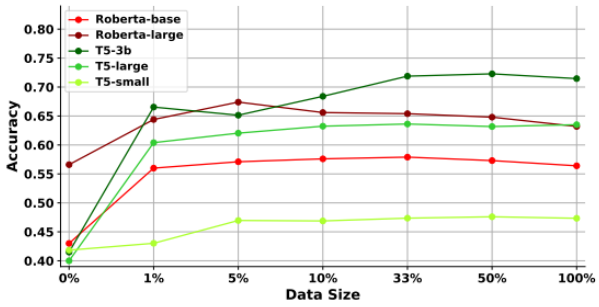


Рисунок 17

Також навчання саме RoBERTa дозволяє досягти меншої втрати за однакову кількість епох, продемонстровано рисунком 18:

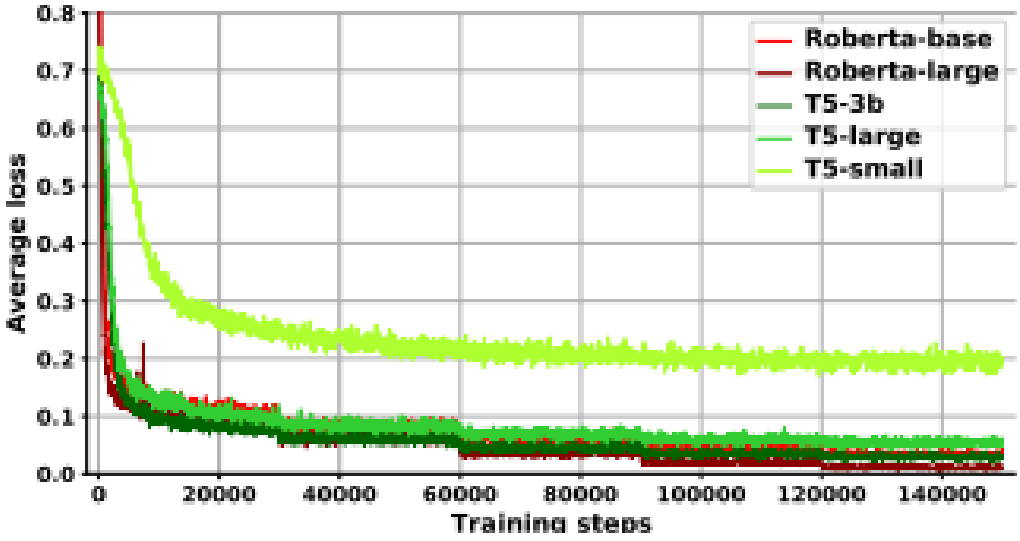


Рисунок 18

Наведений нижче малюнок 19 показує, що моделі краще поводяться на питаннях із несхожими відповідями, і коли навчені на більшій кількості даних. Ще помітним стає, що T5 справляється з короткими питаннями краще за менших об’ємів даних.

Model	Data Size	Similarity				Length				Vocabulary overlap			
		25%	50%	75%	100%	25%	50%	75%	100%	25%	50%	75%	100%
Roberta	0%	53.6	63.9	71.9	**80.9**	63.0	66.3	65.1	75.9	62.3	71.3	69.9	66.7
	1%	56.6	**73.5**	75.6	78.7	**68.8**	**69.6**	68.0	78.0	**68.6**	71.3	74.3	70.2
	5%	**60.3**	72.4	**76.5**	80.4	66.7	68.3	**72.1**	**82.6**	**68.6**	**73.5**	**74.9**	**72.6**
	10%	58.2	71.1	73.2	79.8	67.1	65.9	70.2	79.1	68.0	72.4	70.6	71.3
	33%	58.8	72.0	74.7	78.0	68.0	68.7	70.6	76.3	66.9	71.7	72.8	72.2
	50%	57.1	70.4	73.9	80.2	66.4	66.1	71.2	77.8	65.8	70.0	**74.9**	70.9
	100%	55.6	68.3	69.3	74.8	62.7	65.2	66.9	73.0	63.4	65.9	71.9	66.7
T5-3b	0%	48.8	48.3	51.9	51.5	50.1	50.2	50.1	50.0	47.1	52.6	51.2	49.6
	1%	61.7	73.9	73.6	78.7	71.2	70.0	70.2	76.5	68.2	70.4	76.9	72.4
	5%	60.3	72.4	71.9	79.3	68.4	68.7	67.8	79.1	64.1	70.7	77.6	71.7
	10%	65.4	75.9	75.4	82.6	70.6	74.8	72.5	81.3	69.9	74.1	77.8	77.4
	33%	67.3	**77.2**	80.0	82.4	**73.4**	74.6	**77.8**	81.1	70.4	78.7	79.7	78.0
	50%	**68.8**	**77.2**	**80.2**	**84.3**	73.2	**76.3**	77.3	**83.7**	72.1	**79.1**	**80.2**	**79.1**
	100%	68.2	76.1	78.2	84.1	72.3	71.9	76.5	79.3	**78.9**	77.2	75.8	76.7

Рисунок 19

3.2 Уточнення задачі

На основі аналізу підходів, наведеного вище, я вирішив обрати модель T5.

На це є такі основні причини:

1. Вона більш універсальна і за правильного підходу до навчання справляється зі своїми завданнями краще за інші моделі.
2. Вона попередньо навчена для завдання сумаризації.
3. За відносно невеликих датасетів показує результати кращі за аналогічні у BERT. А за певних умов та розміру параметрів навчається краще за RoBERTa.

Так як модель T5 не була преднавчена для роботи з сумаризацією українською мовою, я вирішив у рамках своєї дипломної роботи провести тонку настройку моделі T5 для уможливлення її використання для розв’язку задач автоматичної сумаризації українською мовою.

Для цього я обрав датасет XL-SUM, на якому була преднавчена багатомовна версія моделі T5. Він складається з архіву новин з BBC на 43 мовах, а особливої зручності йому додає те, що у кожної статті є написане людиною-автором резюме тексту.

Рисунок 20 - зразок навчальних даних

	id	url	title	summary	text
518	141010_kikhtenko_profile_sx	https://www.bbc.com/ukrainian/politics/2014/10...	Олександр Кіхтенко: новий генерал-губернатор Д...	Президент Петро Порошенко звільнив з посади го...	summarize: Новим керівником Донеччини став ген...
213	050129_kruty_anniv	https://www.bbc.com/ukrainian/domestic/story/2...	Річниця загибелі 300 студентів під Крутами	29 січня виповнюється 87 річниця героїчної заг...	summarize: Україна вже два роки відзначає цю р...
713	160522_euro_2016_send_off_or	https://www.bbc.com/ukrainian/sport/2016/05/16...	У Києві провели футбольну збірну на Євро-2016	У Києві на Михайлівській площі кілька тисяч ук...	summarize: У офіційній церемонії взяли участь,...
1129	news-45483128	https://www.bbc.com/ukrainian/news-45483128	"Самопомич" у Київраді саморозпустилася: всі д...	Фракція Київради "Самопомич" саморозпустилась ...	summarize: Сергій Гусовський про звинувачення ...
6	090513_court_elections_oh	https://www.bbc.com/ukrainian/domestic/story/2...	Суд скасував вибори президента 25 жовтня	Конституційний суд України скасував постанову ...	summarize: Таке рішення суду називають очікува...
392	120917_usa_media_review_ko	https://www.bbc.com/ukrainian/politics/2012/09...	Кіно і вибори: американські ЗМІ про події на Б...	Донедавна майже всі американські медіа мали ко...	summarize: Наталка Хижняк Для BBC Україна, Ваш...
155	141117_vs_heavenly_hundred	https://www.bbc.com/ukrainian/news_in_brief/20...	Допомогу отримала 91 сім'я загиблих з "Небесно...	Матеріальну допомогу від держави вже отримала ...	summarize: "Сьогодні ми маємо 100 осіб, які є ...
222	140410_usa_russia_east_ko	https://www.bbc.com/ukrainian/politics/2014/04...	США і Росія говорять про мирне розв'язання кри...	Держсекретар США Джон Керрі та його російський...	summarize: Будівля ОДА в Донецьку перебуває у ...
787	080827_miliband_ukraine_it	https://www.bbc.com/ukrainian/news/story/2008/...	Мілібенд прибув в Україну говорити про Грузію	Британський міністр закордонних справ Дейвід М...	summarize: Напередодні візиту британський міні...
775	140619_ri_rasmussen_border_troops	https://www.bbc.com/ukrainian/news_in_brief/20...	НАТО констатує нове скупчення російських військ...	Генсек НАТО Андерс Фог Расмуссен заявив, що мо...	summarize: "Якщо їх розгорнули, аби перекрити ...

3.3 Аналіз отриманих результатів

Після настройки моделі на етапі валідації моделлю було згенеровано 250 текстових резюме.

Нижче продемонструю деякі з вдалих, на мою думку, прикладів автоматичної сумаризації у таблиці 3:

Згенероване резюме до тексту статті	Справжнє резюме
Мнагрополтика прогнозу зниження врожаю украньско пшениц.	Урожай зерна в Украи цього року буде нижчим вд прогнозованого урядом влтку.
Уряд вропейського Союзу очолив створення нового уряду.	Через 2 мсяц псля позачергових парламентських виборв, де жодна партя не набрала абсолютно бльшост, у Серб утворено новий уряд.
Ураган "Сенд", який залишив пвденно-сходному узбережж Куби, становить смертну загрозу.	Ураган "Сенд" рухаться на Флориду та Багамськ острови псля того, як вирував на Гат, Куб та Ямайц, де забрав життя близько 20 людей.
Украньск вйськов пдтримали перший пункт обов'язкового припинення вогню та спостережних пунктв на Донбас.	Украньск вйськов представили комплекс заходв щодо демлтаризац зони в район Широкиного, повдомля прес-служба мнстерства оборони.
Британя загрожу "жорсткий брекзит" через 23 роки.	Британський парламент проголосував за поправку, яка передбача нове вдтермнування "брекзиту". Прем'р-мнстр Борис Джонсон виступав категорично проти такого ршення.
Силовики готов стрляти побиття людей пд час мжнародних виборв у Блорус.	У Мнську тривають акц проти результатв президентських виборв свавлля правоохоронцв.
Украньск православн греко-католицько церкви визнали голодомор 1932-33 рокв актом геноциду украньского народу.	Глави УПЦ КП, УГКЦ та УРКЦ закликають парламент визнати Голодомор 1932-33 рокв актом геноциду украцв Див.
Див розповв програму Б-Б-С та слухачв.	Украньска Служба Б-Б-С провадить нтерактивн програми, у яких можуть взяти участь наш слухач.
Книга року ВВС-2013 представля кожну з книжок "довгого списку" (претендентв на звання Книги року).	Книжкова премя ВВС Украна Книга року ВВС-2013 представля кожну з книжок "довгого списку" (претендентв на звання Книги року).
Коалця ухвалила закон про вибори, як будуть визнан за пропискою.	Президент Ющенко пдписав ухвален коалцйною бльшстю у парламент поправки до закону про вибори.
Головний директор Нацонального банку Украни Богдан Данилишин повдомив, що фскальн витрати вд кризи банквського сектору становить 38% ВВП через фнансово кризи.	38% ВВП - такими загальн економчн втрати вд банквсько кризи 2014-2016 рокв. Прям фскальн витрати на очищення банквсько системи коштували кран 14% ВВП.
Техасський штат Австрал отримав бльше сотн скарг на голки в полуниц.	Полця Австрал заарештувала пдлтка, який ззнався в тому, що пдкладав голки у полуницю. Ягоди з металевими голками, як продавали у мсцевих магазинах, ранше спровокували панку по всй кран
Названо число випадкв коронаврусу на ранок 23 березня у Украи. Кількість переважна, що пдхопили 25 тисяч людей.	У п'ятницю, 20 березня, в Украи зарестрували 15 нових випадкв захворювання на коронаврус.

В Британ обговорюють запровадження стандартизованих пачок цигарок.	Стандартизован пачки для цигарок рятують життя, оскльки зменшують кількість людей, як починають курити.
Дослдники з Королвського коледжа Лондона вилкували першим сигналом хвороби Хвороба Паркнсона, який може бути вражен дофамном.	Вчен повдомили, що виявили ранн ознаки хвороби Паркнсона, що проявляються ще за 15-20 рокв до появи перших симптомв захворювання.
вропейський угрупування "Надзвичайн кроки" викликали головний позов у справ вирубування заповдного лсу.	Найвищий суд вропи наказав Польщ негайно припинити масове вирубування дерев у найстаршому вропейському лс.
Олексй Навальный - кандидат в посаду мера Москви.	Напередодн винесення вироку у справ "Кровлсу" росйського опозицонера та блогера Олекся Навального зарестрували кандидатом у мери Москви.
Пвнчна Корея каже, що плану змусити ядерне випробування з боку державних державних структур.	Пвнчна Корея всупереч закликам мжнародно спльноти вперше випробувала ядерну зброю, поширивши стурбованість у регон.
Ангела Меркель заявила, що самт С може закликати Росю до припинення вйськових дй у Сир.	вропейська Рада засудила атаки Рос на цивльне населення в Сир, проте лдери С не змогли домовитись щодо нових санкцй проти РФ.
Техасський мормон Мтт Ромн був кандидатом в президенти США.	Мтт Ромн забезпечив соб мсце кандидата у президенти США вд Республканськопарт на виборах, як вдбудуться у листопад.
У Львов в лквдац авар, яка тривала з загибел людей, й отрулися на нудоту.	Мнстерство з надзвичайних ситуацй Украни заявля, що хд лквдац наслдкв фосфорно авар на Льввщин перебува пд контролем державно комс рятувальникв, в
Американський космчний шаттл "Атлантс"" сприйняв новий пуск з космодрому на мис Канаверал.	Астронавти з американського космчного шатла" Атлантс" вийшли на шсть годин у вдкритий космос на орбтальнй Мжнароднй космчнй станц, де буде встановлено дв нов сонячн батаре.
Дональд Трамп запровадив надзвичайний стан - "потужнш гумантарна криза на нашому пвденному кордон".	Дональд Трамп вперше виступив з спецальним зверненням до американськонац у прямому ефр з Овального кабнету Блого дому. Президент США вкотре закликав пдтримати його проект з зведення стни на кордон з Мексикою.
США призначили афроамериканця Остна першим темношкрим на посад глави Пентагону.	Переможець президентських виборв у США Джо Байден плану призначити очльником Пентагону вдставного генерала Ллойда Остна.
На Майдан Незалежност у Кив вдбувся мтинг "маршу за мпчмент", який проходить перед СЗО СБУ.	Прихильники Мхела Саакашвл у недлю проводять в Кив "марш за мпчмент".
"Лверпуль" утрет пробился до Лги вропейських чемпонв, а "Реал" здобув звичних голв.	спанський "Реал" у драматичному фнал у Кив перемг англйський "Лверпуль" 3:1 та здобув вже 13-й в стор найпрестижнший клубний трофей вропи.
У недлю у Шарлоттсвл вдбулись акц протесту, як викликали участь у маршах ультраправих та боротьб з неонацистськими активстами.	Президент США Дональд Трамп зазнав критики з лав власно республканськопарт за сво коментар з приводу сутичок пд час протеств ультраправих у штат Врджиня.
Юля Тимошенко заявила, що втратили великого лдера Велико Британ Маргарет Тетчер у вц 87 рокв.	Премр-мнстр Велико Британ Девд Кемерон назвав день смерт Маргарет Тетчер "дуже сумним днем для крани", а Юля Тимошенко шкоду, що не зможе провести в останнй шлях.

Таблиця 3

Тепер продемонструю деякі невдалі на мою думку приклади текстового резюме налаштованої моделі, представлені у таблиці 4:

Згенероване резюме до тексту статті	Справжнє резюме
"Мовчанн ягнят" - серйний фльм в стор.	Неперевершений за свою напруженстю трилер водночас глибокий соцальний нарис про об'ктивацю жнки в чоловчому свт - фльм "Мовчання ягнят" далеко випередив свй час, поясню кнокритик BBC
Британським дтям витрачають на телевзор сьогодн часу, який обира саме виршити.	Нов рекомендац щодо охорони здоровя, оприлюднен в Англ, радять дтям, як мають надмрну вагу або страждають вд ожирння, вести облк свого харчування та фзично активност.
читачв Tesla та SpaceX оголосили, що компаню Cambridge Analytica за отримання 50 млн даних.	лон Маск видалив сторнки Tesla та SpaceX з соцмереж Facebook.
Рося ма намр заврити "гумантарну паузу" у Алеппо по 11 годин на день, заявив мнстр закордонних справ Рос Володимир Путн.	Мнстр оборони Рос Сергй Шойгу заявив, що режим припинення вогню в Алеппо - так звана "гумантарна пауза" - продовжена на добу.
McDonald's (The Lion King), який хоче був власником у майбутнй частин ресторанв у Кита, обця фгурувати його з з франшизою.	McDonald's погодився продати 80% свого бзнесу у Кита та Гонконгу. Це частиною планв компан з розширення ресторанних франших по всьому свту.
Б-Б-С оголосив, що "РосУкрЕнерго' може не знайти вд компан 'Газпром'.	Оглядач британського журналу 'Економст' Едвард Лукас, який щойно видав книгу 'Нова холодна вйна' про взамини Рос з Заходом, в нтерв'ю Б-Б-С заявив, що новий газовий конфлкт Украни з Росю
Блорус розпочав спробу постачання нафти до блоруських труб.	вросоюз вимага вд Рос та Блорус негайних пояснень причин перебов у постачанн нафти одним з головних трансконтинентальних трубопроводв.
Для з'явився перший лтак для Украньско арм.	Мноборони Украни уклало контракт з ДП "Антонов" на три нов вйськово-транспортн лтаки Ан-178 для арм.
Закон опозицйних зборах в Укран довколо вводить публчний мтинг на Манежнй площ.	Росйський президент Володимир Путн пдписав змни до закону, як забороняють проводити мтинги, демонстрац та пкети в нчний час.
Затримали Антона Скибу, якого затримали за "екстремзм" - на м'я.	Представники "Донецько народно республки" захопили в Донецьку журналста-фрлансера Антона Скибу, який спвпрацював з американським телеканалом CNN.
Президент Украни Вктор Янукович заявив, що "хоче жити швидко дуже добре".	Вктор Янукович заявив, що реформи в Укран будуть продовжуватись, та застерг вд того, аби очкувати вд курсу на реформування швидких результатв.
Донецьку прогнали колоню полонених, яка шла з Донецька на День Незалежност. На вулиц вдкрили военв, щоб украньск сепаратисти.	Мжнародн правозахисники заявили, що акця, влаштована проросйськими сепаратистами в Донецьку, порушу Женевську конвенцю, яка вимага гуманного шанобливого ставлення до вйськовополонених.

Псля того як "Дорнь 17" почалися вд Британ.	Бля узбережжя пвденноанглйського графства Кент почалися роботи з пднятя на поверхню з морського дна ункального лтака Друго свтово вйни - бомбардувальника "Дорнь 17".
Об'мн рукави-буфи у 2016 роц був дуже великими.	Буфи, волани, батик тонка смужка. Кореспондентка BBC Culture з питань моди розповда про головн тренди у чоловчому жночому одяз в 2016 роц.
ноземна преса пише про таке:	21 березня захдна преса пише про моврну схему вдмивання росйських грошей британськими банками, розслдування ФБР щодо можливих зв'язкв штабу Дональда Трампа з росянами можливсть вйськового конфл
Пише "Фокус" з перемогою президента Украни Петра Порошенка.	Стратегчн плани Петра Порошенка, його потенцйн опоненти на виборах 2019 року та сумнвний "напад патротизму" жителв Донбасу, що перехали до Рос, - теми укранських тижневикв.
Укранський урядовець Юрй Продан повдомив, що газом з вропи може забезпечити постачання газом з Заходу.	Украна готуться до оскарження чинно газово угоди з Росю у Стокгольмському арбтраж, заявив у парламент мнстр енергетики Юрй Продан.
Укранська жнка Геннадя Курочка, у якому конкурс "Мс свту" розлучила жнку з дтьми та призувачуть вол у понедлок организатори.	У соцмережах жваво обговорюють скандал навколо конкурсу "Мс Украна", а саме - дисквалфкацю цьогорчно переможниц Веронки Ддусенко.
Футболюзя "Свобода висловлювання у рзних частинах" взяла участь у розмов проекту "Ув'язнен за мистецтво".	Американський актор Джонн Депп долучився до кампан за свободу слова з закликом звльнити укранського режисера Олега Сенцова, ув'язненого в Рос.
Украньск вропейськ виступники змусили каспйську нафту з одесського ринок.	Вдкриттям нафтопроводу Баку - Джейхан Захд здйснив черговий крок до давно омряно мети - зменшити свою нафтову залежнсть вд обмежено клькост джерел сировини, якими зараз насамперед нестабльний Близь
Хоча "Подорожей Гуллвера" виявили мумфковане тло: зрст деяких гномв, яких звинуватили у зловживанн деяких мешканцв, перевищу столтне столтне тло.	Зрст ранцв, як жили в поселенн Махунк на сход рану, до початку XX столття не пере вищу вав одного метра.
талйський палац вдновляться в занедбаному дерев'янскому склепнням.	Насолодитися деякими з найдивовижнших архтектурних пам'яток можна, лише глянувши вгору. Оглядач BBC Culture роздивляться найкрасивш стел свту.
Захдна крига антарктиц танет швидше, нж з дендом.	Антарктика скида сво крижан покриви швидше, нж очкувалося.
Полця Сан-Лус зростанням клькост тунеля стеження за ваном Лопесом, якого заарештували за недбаного будинку.	У США виявили секретний тунель для контрабанди наркотикв з Мексики: вн об'днав приватний мексиканський будинок закинутий ресторан KFC в американському штат Арзона. Довжина пдземного ходу - близько 180 м.
Закон опозицйних зборах в Укран довколо вводить публчний мтинг на Манежнй площ.	Росйський президент Володимир Путн пдписав змни до закону, як забороняють проводити мтинги, демонстрац та пкети в нчний час.

Михайло Погребинський сказав, що на його заяву, якщо на його заяву, то його не зважа, що вн був сумнвачений боць.	Для укранського кервництва заява Ернеста Нусера про вдставку з посади мського голови Мукачевого була повною несподванкою.
Президент Путн застерг проти Рос, яке вважа дяльнсть спостергачв на виборах у США.	Сполучен Штати засудили переслдування опозицйних полтикв демонстрантв у Рос напередодн запланованих на недлю парламентських виборв.
Наглядову Раду ран оприлюднили спроби вдбгати на занепоконня, як називають здатися у першому тур голосуванн.	Орган, що нагляда за перебгом виборв в ран, наказав частково перерахувати голоси псля першого туру президентських виборв, що вдбувся минуло п'ятниц.

Таблиця 4

Як можна бачити з наведених вище результатів, система не декодує три літери українського алфавіту – і, ї, є.

Ця проблема пов'язана з відсутністю цих літер у базовому словнику використаної моделі, адже я не зміг знайти у відкритому доступі словник векторних представлень вбудовувань слів українскої мови з архіву бібліотеки Sentencepiece.

Також інша проблема – явна недотренованість моделі. Хоча модель і демонструє певне розуміння далеких залежностей у тексті, але все ще видно, що для кращої роботи датасет треба збільшувати.

Вона виникла через певну обмеженість технічних ресурсів для тренування, так як безкоштовна версія Google Collab не дає достатньо ресурсів для навчання моделі на всьому наявному об'ємі даних.

Через це максимально можливий розмір датасету для навчання без переповнення пам'яті графічного прискорювання GPU у моєму випадку складає близько 1000 навчальних файлів і 30 епох, що досить мало для тонкої настройки моделей такого виду.

Значення метрики втрати стабільно зменшувалося у процесі тонкого налаштування моделі на обраному датасеті.

Загалом для даної кількості параметрів моделі архітектури T5 та порівняно малого датасету отримані результати я вважаю непоганими.

РОЗДІЛ 4. ПРОГРАМНА РЕАЛІЗАЦІЯ

Інсталюємо необхідні модулі

```
!pip install sentencepiece
!pip install transformers
!pip install rich[jupyter]
```

Код для виводу зразків даних у відформатованому табличному форматі:

```
def show_dataframe(d_set):
  #вивід даних

  Rev =Console()
  tab = Table(Column("Оригінал", justify="center" ), Column("Резюме", justify="ce
nter"), title="Зразок",pad_edge=False, box=box.ASCII)

  for i, row in enumerate(d_set.values.tolist()):
    tab.add_row(row[0], row[1])

  Rev.print(tab)
```

Для виводу логів навчання у вигляді таблиць використаємо модуль rich.table та вбудовані у нього класи Column, Table, визначимо консольний логер:

```
scr=Console(record=True)
```

Визначимо таблицю для логів тренувань:

```
write_log_learning = Table(Column("Епоха", justify="center" ),
                           Column("Крок", justify="center"),
                           Column("Метрика втрати", justify="center"),
                           title="Тренувальний статус",pad_edge=False, box=box.ASCII
)
```

Налаштуємо графічне прискорення GPU:

```
# GPU
from torch import cuda
device = 'cuda' if cuda.is_available() else 'cpu'
```

Для зручної роботи з користувацьким датасетом сторимо клас датасету для зчитування даних та передачі даталоадеру:

```
class MyDataSetClass(Dataset):

  def __init__(self, dataset, model_tokenizer, original_size, summ_size, original
_text, summ_text):
    self.tokenizer = model_tokenizer
    self.data = dataset
    #data
    self.source_len = original_size
    self.summ_len = summ_size
    self.target_text = self.data[summ_text]
    self.source_text = self.data[original_text]

  def __len__(self):
    return len(self.target_text)

  def __getitem__(self, index):
    source_text = str(self.source_text[index])
    target_text = str(self.target_text[index])
    source_text = ' '.join(source_text.split())#попередня обробка тексту
    target_text = ' '.join(target_text.split())#обробка тексту
    #кодування оригіналу
    original_text = self.tokenizer.batch_encode_plus([source_text], max_length= s
elf.source_len, pad_to_max_length=True, truncation=True, padding="max_length", re
turn_tensors='pt')
    #кодування резюме
    human_summary = self.tokenizer.batch_encode_plus([target_text], max_length= s
elf.summ_len, pad_to_max_length=True, truncation=True, padding="max_length", retu
rn_tensors='pt')


    original_ids = original_text['input_ids'].squeeze()
    original_mask = original_text['attention_mask'].squeeze()
    human_ids = human_summary['input_ids'].squeeze()
    human_mask = human_summary['attention_mask'].squeeze()

    return {
        'source_ids': original_ids.to(dtype=torch.long),
        'source_mask': original_mask.to(dtype=torch.long),
        'target_ids': human_ids.to(dtype=torch.long),
        'target_ids_y': human_ids.to(dtype=torch.long)
    }
```

Для тренування використаємо функцію learn() з параметрами моделі, переданими з головної функції:

```
def learn(running_ep, model_tok, trainable_model, device, d_loader, optimization)
:
  trainable_model.train()
  for _,data in enumerate(d_loader, 0):
    y = data['target_ids'].to(device, dtype = torch.long)
    y_ids = y[:, :-1].contiguous()
    lm_labels = y[:, 1:].clone().detach()
    lm_labels[y[:, 1:] == model_tok.pad_token_id] = -100
    #механізм уваги
    s_ids = data['source_ids'].to(device, dtype = torch.long)
    s_mask = data['source_mask'].to(device, dtype = torch.long)

    res = trainable_model(input_ids = s_ids, attention_mask = s_mask, decoder_inp
ut_ids=y_ids, labels=lm_labels)
    err = res[0]

    if _%5==0:
      write_log_learning.add_row(str(running_ep), str(_), str(err))
      scr.print(write_log_learning)

    optimization.zero_grad()
    err.backward()
    optimization.step()
```

Функція для генерації машинного резюме, та його порівняння з оригінальним:

```
def checking(val_ep, model_tok, val_model, device, d_loader):

  val_model.eval()
  generated = []
  human_written = []
  with torch.no_grad():
      for _, data in enumerate(d_loader, 0):
          human_wr_summary = data['target_ids'].to(device, dtype = torch.long)
          s_ids = data['source_ids'].to(device, dtype = torch.long)
          src_msk = data['source_mask'].to(device, dtype = torch.long)
          model_gen_token_ids = val_model.generate(
              input_ids = s_ids,
              attention_mask = src_msk,
              max_length=150,
              num_beams=2,
              repetition_penalty=2.5,
              length_penalty=1.0,
              early_stopping=True
              )
          machine_summary = [model_tok.decode(g, skip_special_tokens=True, clean_
up_tokenization_spaces=True) for g in model_gen_token_ids]
```

```
            human_summary = [model_tok.decode(t, skip_special_tokens=True, clean_up
_tokenization_spaces=True) for t in human_wr_summary]
            if _%10==0:
                scr.print(f'Completed {_}')

            generated.extend(machine_summary)
            human_written.extend(human_summary)
    return generated, human_written
```

Визначимо функцію для тренування моделі – T5Trainer().

```
def T5finetuner(fine_tune_set, original_text, shortened_text, parameters,
                output_dir="/content/drive/MyDrive/outpt/"):
```

Встановимо випадкові сіди та детермінований Pytorch для відтворюваності:

```
    torch.manual_seed(parameters["SEED"])  # Встановлення випадкових сідів та

    np.random.seed(parameters["SEED"])

    torch.backends.cudnn.deterministic = True #  детермінованого pytorch для ві
    дтворюваності
```

Визначимо токенізатор для кодування тексту, визначимо модель та передамо її на графічний прискорювач GPU:

```
    scr.log(f"""[Model]: Завантаження {parameters["FinetuneMod"]}...\n""")     # л
    оги


    model_tokenizer = T5Tokenizer.from_pretrained(parameters["FinetuneMod"]) # то
    кенізатор для кодування тексту


    fine_tunable_ = T5ForConditionalGeneration.from_pretrained(parameters["Finetu
    neMod"]) # Визначаємо модель. Використаємо модель t5-
    base і додамо шар мовної моделі зверху для генерації резюме

    fine_tunable_ = fine_tunable_.to(device)  # передаємо модель на графічний при
    скорювач GPU


    scr.log(f"[Data]: Читання даних...\n") # логи


    dataframe1 = fine_tune_set[[original_text, shortened_text]] # Завантажуємо да
    тасет
```

```
show_dataframe(dataframe1.head(2)) #виводимо його
```

Завантажуємо датасет:

```
dataframe1 = fine_tune_set[[original_text, shortened_text]] # Завантажуємо да
тасет
show_dataframe(dataframe1.head(2)) #виводимо його
```

Розбиваємо датасет на тренувальні та валідаційні екземпляри, 80 відсотків випадковим чином віддаємо на тренувальні дані, решта йде на валідацію.

```
learning_prop = 0.8 # 80% - тренувальний набір

learning_set = dataframe1.sample(frac=learning_prop, random_state=parameters[
"SEED"])      # Створюємо датасет і даталоадер
check_set = dataframe1.drop(learning_set.index).reset_index(drop=True)
learning_set = learning_set.reset_index(drop=True)

scr.print(f"Повний датасет: {dataframe1.shape}")
scr.print(f"Тренувальний датасет: {learning_set.shape}")
scr.print(f"Тестовий датасет: {check_set.shape}\n")
```

Створюємо екземпляри класу датасету

```
learn_dataset = MyDataSetClass(learning_set, model_tokenizer, parameters["ori
ginal_size"],
parameters["summary_size"], original_text, shortened_text) # Створюємо тренув
альний датасет для подальшого створення даталоадерів
check_dataset = MyDataSetClass(check_set, model_tokenizer, parameters["origin
al_size"],
parameters["summary_size"], original_text, shortened_text) # Створюємо валіда
ційний датасет для подальшого створення даталоадерів
```

Визначаємо параметри для даталоадерів:

```
learning_parameters = {
      'batch_size': parameters["learning_batch"],  # Визначаємо параметри для
   створення даталоадерів
    'shuffle': True,
```

```
        'num_workers': 0
    }

    check_parameters = {
        'batch_size': parameters["validation_batch"],  # Визначаємо параметри для
      створення даталоадерів
        'shuffle': False,
        'num_workers': 0
    }
```

Створення даталоадерів для тестування та валідації

```
    learning_dataset_upl = DataLoader(learn_dataset, **learning_parameters) #Ство
    рення даталоадеру для тестування.
    checking_dset_upl = DataLoader(check_dataset, **check_parameters) #Створення
    даталоадеру для валідації.
```

Визначимо оптимізатор, який буде налаштовувати ваги нейронної межі під час тренування:

```
      algorythm_optimization = torch.optim.Adam(params=fine_tunable_.parameters()
      , lr=parameters["learn_speed"]) # Визначимо оптимізатор
```

Цикл тренування моделі:

```
    scr.log(f'[Ініціалізація тонкої настройки...\n') # початок циклу тренування

    for tr_ep in range(parameters["epochs"]):
        learn(tr_ep, model_tokenizer, fine_tunable_, device, learning_dataset_upl
, algorythm_optimization)  # тренування

    scr.log(f"[Зберігання моделі]...\n")

      model_save_path = os.path.join(output_dir, "model_files") #  Зберігання мод
      елі після тренування
      fine_tunable_.save_pretrained(model_save_path)
      model_tokenizer.save_pretrained(model_save_path)
```

Оцінка тестового датасету:

```
      scr.log(f"[Ініціалізація валідації]...\n")
```

```
    for val_ep in range(parameters["val_ep"]):   # оцінка тестового датасету
              model_gen_output, human_written = checking(val_ep, model_tokenizer,
       fine_tunable_, device, checking_dset_upl)
        final_dataframe = pd.DataFrame({'Generated Text': model_gen_output, 'Actu
        al Text': human_written})
        final_dataframe.to_csv(os.path.join(output_dir, 'predictions.csv'))

    scr.save_text(os.path.join(output_dir, 'logs.txt'))

    scr.log(f"[Валідація завершена.]\n")
          scr.print(f"""[Model] Модель збережено у @ {os.path.join(output_dir, "m
      odel_files")}\n""")
    scr.print(f"""[Validation] Згенеровані елементи збережено у @ {os.path.join(o
    utput_dir, 'predictions.csv')}\n""")
      scr.print(f"""[Logs] Логи збережено @ {os.path.join(output_dir, 'logs.txt')
      }\n""")
```

Передаємо моделі параметри для навчання. Для покращення результатів роботи сумаризатора я взяв преднавчену для російської мови модель T5.

```
model_parameters={
    "FinetuneMod":"UrukHan/t5-russian-summarization",
    "learning_batch":4,          # розмір тренувальної партії
    "validation_batch":3,          #  розмір валідаційної партії
    "epochs":30,               # кількість тренувальних епох
    "val_ep":1,                 # кількість валідаційних епох
    "learn_speed":5e-5,          # темп навчання
    "original_size":512,  # Максимальна довжина вхідного тексту
    "summary_size":90,    # Максимальна довжина резюме
    "SEED": 42

}
```

Запускаємо навчання:

```
T5finetuner(dataframe=dataframe1[:], source_text="text", target_text="summary", m
odel_params=model_params, output_dir="/content/drive/MyDrive/outpt/")
```

ПОДАЛЬШІ ПОКРАЩЕННЯ

Власне, для покращення результатів навчання моделі T5 у майбутніх роботах я спробую розв’язати описані проблеми моєї реалізації тонкого налаштування та оптимізувати процес навчання задля покращення результатів.

У випадку появи на ринку технологій більш досконалої моделі для завдання сумаризації не виключено, що я використаю її для наступних досліджень.

Тож я бачу такі шляхи покращення майбутньої ефективності моделі:

1. Отримати доступ до файлів векторних словників української мови SentencePiece.
2. Навчити модель на більш досконалому обладнанні та на більшому об’ємі даних, можливо краще оптимізувати використання GPU та розмір тренувальної партії.
3. Використовувати версії моделі з більшою кількістю параметрів, наприклад T5-Large або T5-3B.
4. Використовувати більш нові за реалізацією версії архітектури T5.

Даний список ще може бути розширений застосуванням новіших за T5 і кращих за результатами роботи моделей.

ВИСНОВОК

Підсумовуючи усю проведену роботу можна сказати, що методи і технології сумаризації конче необхідні у сучасному світі, адже їх розвиток прямо пов’язаний з розвитком систем обробки природної мови. На даний момент існують такі два основних напрямки розвитку технології сумаризації як абстрактивна та екстрактивна сумаризація.

Екстрактивна сумаризація ґрунтується на витягуванні з тексту певних його складових частин без змін. Найчастіше це речення. Витягування відбувається на основі багатьох ознак, таких як довжина речення, позиція у документі, включення певних фраз, частота певних змістовних термів. Іноді застосовуються методи латентного семантичного аналізу.

Абстрактивна сумаризація ґрунтується на застосуванні моделей глибокого навчання, таких як згорткові мережі, або архітектура трансформеру. Вони дозволяють досягти більшого рівня новизни резюме тексту.

Основним принципом архітектури трансформеру є багатошарова модель енкодер-декодер з механізмом самоуваги. Це дозволяє моделі враховувати контекст при генерації вихідної послідовності.

Основними моделями трансформеру, використовуваними зараз є:

- Модель BERT
- RoBERTa
- T5
- Тощо

У роботі було реалізовано процедуру тонкого налаштування моделі трансформеру T5 на україномовному датасеті, але через технічну обмеженість провести не вдалося реалізувати її повний потенціал на практиці. Навіть незважаючи на це, модель непогано генерує резюме, а її недоліки були розглянуті та проаналізовані у підрозділі 2.2.2.

Вказані також напрямки майбутньої роботи над цією моделлю з метою подальшого покращення результатів роботи і дослідження.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D Trippe, Juan B Gutierrez, and Krys

2. Kochut. 2017. Text summarization techniques: A brief survey. arXiv preprint arXiv:1707.02268 (2017).

3. Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and

4. Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. arXiv preprint arXiv:1607.07086 (2016).

5. Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014).

6. Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. 2016. End-to-end attention-based large vocabulary speech recognition. In Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on. IEEE, 4945–4949.

7. Lalit Bahl, Peter Brown, Peter De Souza, and Robert Mercer. 1986. Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'86., Vol. 11. IEEE, 49–52.

8. Hareesh Bahuleyan, Lili Mou, Olga Vechtomova, and Pascal Poupart. 2018. Variational Attention for Sequence-toSequence Models. In COLING.

9. David Balduzzi and Muhammad Ghifary. 2016. Strongly-typed recurrent neural networks. In Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48. JMLR. org, 1292–1300.

10. Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In Advances in Neural Information Processing Systems. 1171–1179.

11. Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. Journal of machine learning research 3, Feb (2003), 1137–1155.

12. Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks 5, 2 (1994), 157–166.

13. Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. 1996. A maximum entropy approach to natural language processing. Computational linguistics 22, 1 (1996), 39–71.

14. Neelima Bhatia and Arunima Jaiswal. 2016. Automatic text summarization and it's methods-a review. In Cloud System and Big Data Engineering (Confluence), 2016 6th International Conference. IEEE, 65–72.

15. Samuel Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. 632–642.

16. James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2016. Quasi-recurrent neural networks. arXiv preprint arXiv:1611.01576 (2016).

18. Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, 2009.

20. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

21. William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In Proceedings of the Third International Workshop on Paraphrasing (IWP2005), 2005.