ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

МП – машинний переклад;

БШП – багатошаровий перпцетрон;

ЗНМ – згорткова нейронна мережа;

РНМ – рекурсивна нейронна мережа;

RNN – Recurrent Neural Network – рекурентна нейронна мережа;

GRU-RNN – Gated Recurrent Unit Recurrent Neural Network – рекурентна нейронна мережа з вбудованою періодичною зупинкою;

GPU – Graphics Proccesing Unit – графічний процесор;

CNN – Cable News Network – кабельна новинна мережа;

TF – term frequency – статистичний показник, що використовується для оцінки важливості слів у контексті документа;

IDF – inverse document frequency – статистичний показник, що використовується для оцінки важливості слів у контексті документа;

MVC – Model-View-Controller – архітектурний шаблон;

API – Application Programming Interface - прикладний програмний інтерфейс;

ORM – Object-Relational Mapping – технологія програмування;

ВСТУП

Протягом значного проміжку часу в усьому світі швидкими темпами збільшується кількість інформації. Люди кожного дня сприймають та фільтрують вхідний потік інформації, що надходить з різних джерел: робота, побутові проблеми, популярні джерела інформації тощо. Після винайдення мережі Інтернет кількість такої інформації стала стрімко зростати, з’явилася велика кількість сервісів для надання користувачам всього необхідного для комфортного життя.

Кожен день люди покладаються на широкий спектр джерел, щоб бути інформованими: від новин до постів в соціальних мережах чи результатів пошуку. Здатність розробити моделі машинного навчання, які можуть забезпечувати автоматичне зведення довгих текстів до анотації, є ключовим рішенням для можливості сприйняття великих обсягів інформації за короткий час.

Абстрактне анотування тексту є задачею генерування заголовку чи короткого резюме (анотації), що містить декілька речень та захоплює істотно важливі ідеї статті чи уривку. Прикметник “абстрактний” використовується для позначення анотації, яка є не просто вибором кількох існуючих речень, а є стислим перефразуванням основного змісту документа, у ході якого потенційно використовуються словник, невидимий в джерелі документа.

Ця задача може бути розглянута як відображення вхідної послідовності слів з документа-джерела до цільової послідовності слів – анотації. У недалекому минулому, моделі глибокого навчання, що називаються послідовність-до-послідовності, були успішними у вирішенні багатьох проблем, таких як машинний переклад, розпізнання мови та відеорозміщення субтитрів. В рамках моделі послідовність-до-послідовності, дуже релевантною моделлю до нашої задачі є рекурентна нейронна мережа енкодер-декодер модель, запропонована у 2014 році, що на той час забезпечувала найкращу продуктивність у машинному перекладі (МП).

Не зважаючи на схожість, абстрактне анотування текстів це зовсім інша прблема, ніж МП. На відміну від МП, анотування зазвичай дуже коротке та не залежить дуже сильно від довжини вхідного документа. Також ключова складність у анотуванні – стиснути вхідні дані таким чином, щоб не втратити розуміння, у той час як МП очікується без втрат.

У даній роботі реалізовані такі основні речі:

1. Застосовується готова енкодер-декодер рекурентна нейронна мережа для анотування текстів, що була першопочатково придумала для машинних перекладів на двох різних англомовних корпусах.
2. Використовуються нові моделі, що застосовувались не тільки для машинного перекладу, завдяки ним досягається підвищення продуктивності.
3. Використаний новий набір даних для забезпечення абстрактоно анотування текстів, що вкладається в кілька речень.

1. АНАЛІЗ НЕЙРОННИХ МЕРЕЖ

Нейронна мережа (штучна нейронна мережа) — це математична модель, а також її програмна та апаратна реалізація, побудовані за принципом функціювання біологічних нейронних мереж — мереж нервових клітин живого організму. Це поняття виникло при вивченні процесів, які відбуваються в мозку, та при намаганні змоделювати ці процеси. Першою такою спробою були нейронні мережі У. Маккалока та У. Піттса. Після розробки алгоритмів навчання отримувані моделі стали використовуватися в практичних цілях: в задачах прогнозування, для розпізнавання образів, в задачах керування тощо.

ШНМ являють собою систему з'єднаних між собою простих обробників (штучних нейронів), які взаємодіють. Такі обробники зазвичай є доволі простими (особливо в порівнянні з процесорами, що застосовуються в персональних комп'ютерах). Кожен обробник подібної мережі має справу лише з сигналами, які він періодично отримує, і сигналами, які він періодично надсилає іншим обробникам. І тим не менш, будучи з'єднаними в достатньо велику мережу з керованою взаємодією, такі локально прості обробники разом здатні виконувати доволі складні завдання.

Нейронні мережі не програмуються в звичайному розумінні цього слова, вони навчаються. Можливість навчання — одна з головних переваг нейронних мереж перед традиційними алгоритмами. Технічно, навчання полягає в знаходженні коефіцієнтів зв'язків між нейронами. В процесі навчання нейронна мережа здатна виявляти складні залежності між вхідними даними й вихідними, а також здійснювати узагальнення. Це означає, що в разі успішного навчання мережа зможе повернути правильний результат на підставі даних, які були відсутні в навчальній вибірці, а також неповних та/або “зашумлених”, частково спотворених даних.

Загальна схема простої нейроннї мережі (рисунок 1.1):

1. Зеленим кольором позначено вхідні нейрони.
2. Блакитним – приховані нейрони.
3. Жовтим – вихідний нейрон.

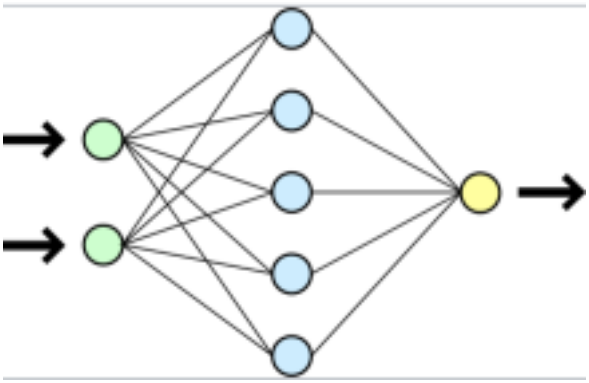


Рисунок 1.1 – Схема простої нейромережі

Нейронні мережі можна класифікувати:

1. За типом вхідної інформації:
   - 1.1 Аналогові нейронні мережі – використовують інформацію у формі дійсних чисел;
   - 1.2 Двійкові нейронні мережі – оперують з інформацією, представленою в двійковому вигляді.
2. За характером навчання:
   - 2.1 Навчання з учителем – відомі вихідні результати нейронної мережі;
   - 2.2 Навчання без вчителя – нейронна мережа опрацьовує тільки вхідні дані та самостійно формує вихідні результати. Такі мережі називають самоорганізаційними;
   - 2.3 Навчання з підкріпленням – система призначення штрафів і заохочень від середовища.

1.1 Характеристика згорткової нейронної мережі

В машинному навчанні згорткова нейронна — це такий тип штучної нейронної мережі прямого поширення, в якому схему з'єднання її нейронів натхнено організацією зорової кори тварин, окремі нейрони якої впорядковано таким чином, що вони реагують на області, які покривають зорове поле, частково перекриваючись[1]. Згорткові мережі було натхнено біологічними процесами[2], вони є варіаціями багатошарових перцептронів, розробленими для використання мінімальної попередньої обробки. Вони мають широке застосування в розпізнаванні зображень та відео, рекомендаційних системах та обробці природної мови.

Згорткові мережі можуть включати шари локальної або глобальної підвибірки, які поєднують виходи кластерів нейронів. Вони також складаються з різних комбінацій згорткових та повноз'єднаних шарів, із застосуванням поточкової нелінійності в кінці кожного шару. Для зниження числа вільних параметрів та покращення узагальнення вводиться операція згортки на малих областях входу. Однією з головних переваг згорткових мереж є використання спільної ваги у згорткових шарах, що означає, що для кожного пікселя шару використовується один і той же фільтр (банк ваги); це як зменшує обсяг необхідної пам'яті, так і поліпшує продуктивність.

В той час як традиційні моделі багатошарового перцептрону (БШП) успішно застосовувалися для розпізнавання зображень, через повну зв'язність між вузлами вони страждають від прокляття розмірності, і, отже не дуже добре масштабуються на зображення вищих роздільностей.

Наприклад, у наборі CIFAR-10 зображення мають розмір лише 32×32×3 (ширина 32, висота 32, 3 канали кольору), тому один повноз'єднаний нейрон у першому прихованому шарі звичайної нейронної мережі матиме 32*32*3 = 3 072 вагові коефіцієнти. Проте зображення 200×200 призведе до нейронів, що мають 200*200*3 = 120 000 вагових коефіцієнтів.

Такі мережеві архітектури не беруть до уваги просторову структуру даних, розглядаючи вхідні пікселі, що є далеко і близько один від одного, на рівних засадах. Очевидно, що повна зв'язність нейронів у рамках розпізнавання зображень є марнотратною, а величезна кількість параметрів швидко веде до перенавчання.

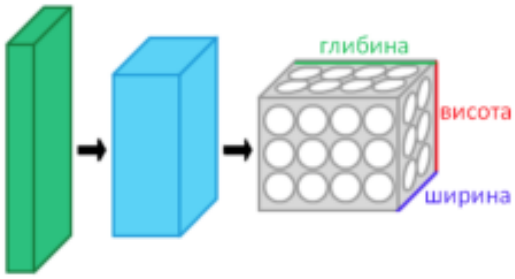


Рисунок 1.2 – Шари ЗНМ, розташвані в 3 вимірах

Згорткові нейронні мережі є біологічно натхненими варіантами багатошарових перцептронів, розробленими для імітації поведінки зорової кори. Ці моделі пом'якшують виклики, поставлені архітектурою БШП, використовуючи сильну просторово локальну кореляцію, присутню в природних зображеннях. На противагу до БШП, ЗНМ мають наступні відмітні ознаки:

1. Тривимірні ємності нейронів. Шари ЗНМ мають нейрони, впорядковані в 3 вимірах: ширина, висота та глибина. Нейрони всередині шару є з'єднаними з невеликою областю попереднього шару, що називається рецептивним полем. Для формування архітектури ЗНМ складаються різні типи шарів, як локально, так і повноз'єднані.
2. Локальна з'єднаність: відповідно до концепції рецептивних полів, ЗНМ використовують просторово локальну кореляцію шляхом застосування схеми локальної з'єднаності між нейронами сусідніх шарів. Ця архітектура таким чином забезпечує, що навчені “фільтри” виробляють найсильніший відгук до просторово

локального вхідного образу. Складання багатьох таких шарів веде до нелінійних “фільтрів”, що стають все більше “глобальними” (тобто, чутливими до більшої області піксельного простору). Це дозволяє мережі спочатку створювати добрі представлення дрібних деталей входу, а потім збирати з них представлення більших областей.

3. Спільні ваги: В ЗНМ кожен фільтр відтворюється на усьому зоровому полі. Ці відтворені вузли використовують спільну параметризацію (вектор ваги та упередження) та формують карту ознаки. Це означає, що всі нейрони в заданому згортковому шарі виявляють в точності одну й ту ж саму ознаку. Відтворені вузли таким чином дозволяють ознакам бути виявленими незалежно від їхнього положення в зоровому полі, забезпечуючи таким чином властивість інваріантності відносно зсуву.

Разом ці властивості дозволяють згортковим нейронним мережам досягати кращого узагальнення на задачах бачення. Також допомагає й поділ ваги, різко зменшуючи кількість вільних параметрів, яких треба навчатися, знижуючи таким чином вимоги до пам'яті для роботи мережі. Зниження обсягу пам'яті уможливлює тренування більших, потужніших мереж.

1.1.1 Згортковий шар

Архітектура ЗНМ формується стосом різних шарів, що перетворюють вхідний об'єм на вихідний об'єм (що, наприклад, зберігає рівні відношення до класів) за допомогою диференційованої функції. Зазвичай застосовується декілька різних типів шарів.

Згортковий шар є основним будівельним блоком ЗНМ. Параметри шару складаються з набору фільтрів для навчання (або ядер), які мають невеличке рецептивне поле, але простягаються на всю глибину вхідного об'єму. Протягом прямого проходу кожен фільтр здійснює згортку по ширині

та висоті вхідного об'єму, обчислюючи скалярний добуток даних фільтру та входу, і формуючи 2-вимірну карту активації цього фільтру. В результаті мережа навчається, які фільтри активуються, коли вони бачать певний конкретний тип ознаки у певному просторовому положенні у вході.

Складання активаційних карт усіх фільтрів уздовж виміру глибини формує повний вихідний об'єм згорткового шару. Таким чином, кожен запис у вихідному об'ємі може також трактуватися як вихід нейрону, що дивиться на невеличку область у вході, та ділить параметри з нейронами тієї ж активаційної карти.

1.1.2 Локальна з'єднаність

При опрацюванні входів високої розмірності, таких як зображення, недоцільно з'єднувати нейрони з усіма нейронами попереднього об'єму, оскільки така архітектура мережі не бере до уваги просторову структуру даних. Згорткові мережі використовують просторово локальну кореляцію шляхом забезпечення схеми локальної з'єднаності між нейронами сусідніх шарів: кожен нейрон з'єднується лише з невеликою областю вхідного об'єму. Обшир цієї з'єднаності є гіперпараметром, що називається рецептивним полем нейрону. З'єднання є локальними в просторі (вздовж ширини та висоти), але завжди поширюються вздовж усієї глибини вхідного об'єму. Така архітектура забезпечує, щоби навчені фільтри виробляли найсильніший відгук до просторово локальних вхідних образів.

1.1.3 Просторова організація

Розмір вихідного об'єму згорткового шару контролюють три гіперпараметри:

1. Глибина вихідного об'єму контролює кількість нейронів шару, що з'єднуються з однією й тією ж областю вхідного об'єму. Всі ці

нейрони вчитимуться активуватися для різних ознак входу. Наприклад, якщо перший згортковий шар бере як вхід сире зображення, то різні нейрони вздовж виміру глибини можуть активуватися в присутності різних орієнтованих контурів, або плям кольору.

2. Крок контролює те, як стовпчики глибини розподіляються за просторовими вимірами (шириною та висотою). Коли кроком є 1, то глибинні стовпчики нейронів розміщуються в просторових положеннях на відстані лише 1 просторової одиниці один від одного. Це веде до сильного перекриття рецептивних полів між стовпчиками, а також до великих вихідних об'ємів. І навпаки, якщо застосовуються більші кроки, то рецептивні поля перекриватимуться менше, й отримуваний в результаті вихідний об'єм матиме менші просторові розміри.
3. Іноді зручно доповнювати вхід нулями по краях вхідного об'єму. Розмір цього нульового доповнення є третім гіперпараметром. Нульове доповнення дозволяє контролювати просторовий розмір вихідних об'ємів. Зокрема, іноді бажано точно зберігати просторовий розмір вхідного об'єму.

Просторовий розмір вихідного об'єму може обчислюватися як функція від розміру вхідного об'єму W, розміру ядрового поля нейронів згорткового шару K, кроку, з яким вони застосовуються S, і величини нульового доповнення P, що застосовується на краях. Формула для обчислення того, скільки нейронів «уміщається» до заданого об'єму, задається як

Якщо це число не є цілим, то кроки встановлено неправильно, і нейрони не може бути розміщено вздовж вхідного об'єму симетричним чином. Загалом, встановлення нульового доповнення в $P = (K - 1) / 2$, коли

кроком є S = 1, забезпечує, щоби вхідний та вихідний об'єми мали однаковий просторовий розмір. Хоча взагалі використання всіх нейронів попереднього шару не є абсолютно обов'язковим, наприклад, ви можете вирішити використовувати лише частину доповнення.

1.1.4 Спільне використання параметрів

Оскільки всі нейрони в єдиному зрізі ділять між собою одну й ту ж параметризацію, то прямий прохід у кожному зрізі глибини згорткового шару може бути обчислено як згортку вагових коефіцієнтів нейронів з вхідним об'ємом (звідси й назва: згортковий шар). Таким чином, є звичним називати набори вагових коефіцієнтів фільтром (або ядром), який згортається із входом. Результатом цієї згортки є активаційна карта, і набір активаційних карт для кожного з різних фільтрів складається разом вздовж виміру глибини для отримання вихідного об'єму. Спільне використання параметрів сприяє інваріантності архітектури ЗНМ відносно зсуву.

Важливо зауважити, що допущення спільного використання параметрів іноді може не мати сенсу. Особливо в тому разі, коди вхідні зображення до ЗНМ мають певну особливу центровану структуру, в якій ми очікуємо навчання зовсім різних ознак у різних просторових положеннях. Одним із практичних прикладів є коли вхід є обличчями, що було відцентровано в зображенні: ми можемо очікувати, що вчитимемося різних особливих ознак очей та волосся в різних частинах зображення. В такому разі є звичним пом'якшувати схему спільного використання параметрів, і натомість просто називати шар локально з'єднаним шаром.

1.1.5 Підвибірковий шар

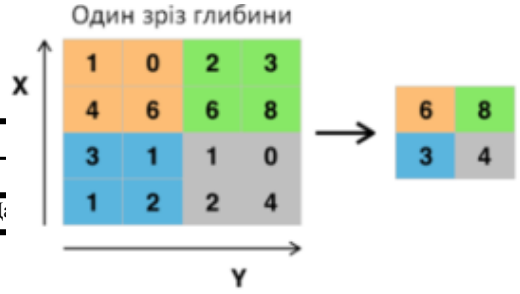


Рисунок 1.4 – Максимізаційна підвибірка із фільтром 2x2 та кроком = 2

Іншим важливим поняттям ЗНМ є підвибірка, яка є різновидом нелінійного зниження дискретизації. Існує декілька нелінійних функцій для реалізації підвибірки, серед яких найпоширенішою є максимізаційна підвибірка. Вона розділяє вхідне зображення на набір прямокутників без перекриттів, і для кожної такої підобласті виводить її максимум. Ідея полягає в тому, що якщо ознаку було знайдено, то її точне положення не так важливе, як її грубе положення відносно інших ознак. Функцією підвибіркового шару є поступове скорочення просторового розміру представлення для зменшення об'єму параметрів та обчислень у мережі, і відтак також для контролю перенавчання. В архітектурі ЗНМ є звичним періодично вставляти підвибірковий шар між послідовними згортковими шарами. Операція підвибірки забезпечує один із різновидів інваріантності відносно зсуву.

Підвибірковий шар діє незалежно на кожен зріз глибини входу, і зменшує його просторовий розмір. Найпоширенішим видом є підвибірковий шар із фільтрами розміру 2×2, що застосовуються з кроком 2, який знижує дискретизацію кожного зрізу глибини входу в 2 рази як за шириною, так і за висотою, відкидаючи 75 % активацій. Кожна операція взяття максимуму (англ. *MAX*) в такому разі братиме максимум з 4 чисел. Розмір за глибиною залишається незмінним.

На додачу до максимізаційної підвибірки, підвибіркові шари можуть також виконувати й інші функції, такі як усереднювальна підвибірка та навіть *L2*-нормова підвибірка. Історично усереднювальна підвибірка застосовувалася часто, але останнім часом впала в немилість у порівнянні з дією максимізаційної підвибірки, робота якої на практиці виявилася кращою.

Через агресивне скорочення розміру представлення (що є корисним лише для менших наборів даних для контролю перенавчання) поточною тенденцією в

літературі є використання менших фільтрів[3], або відмова від підвибіркового шару взагалі.

1.1.6 Повноз'єднаний шар

Насамкінець, після кількох згорткових та максимізаційно підвибіркових шарів, високорівневі міркування в нейронній мережі здійснюються повноз'єднаними шарами. Нейрони у повноз'єднаному шарі мають повні з'єднання з усіма активаціями попереднього шару, як це можна бачити у звичайних нейронних мережах. Їхні активації відтак може бути обчислювано матричним множенням, за яким слідує зсув упередження.

1.1.7 Шар втрат

Шар втрат визначає, як тренування мережі штрафує відхилення між передбаченими та справжніми мітками, і є, як правило, останнім шаром у мережі. В ньому можуть використовуватися різні функції втрат для різних завдань. Багатозмінна логістична втрата застосовується для передбачення єдиного класу з K взаємно виключних класів. Сигмоїдна перехресно-ентропійна втрата застосовується для передбачення K незалежних значень імовірності в проміжку 0, 1]. Евклідова втрата застосовується до регресії дійснозначних міток ∞, ∞].

1.2 Характеристика рекурентної нейронної мережі

Рекурентні нейронні мережі — це клас штучних нейронних мереж, у якому з'єднання між вузлами утворюють орієнтований цикл. Це створює внутрішній стан мережі, що дозволяє їй проявляти динамічну поведінку в часі. На відміну від нейронних мереж прямого поширення, РНМ можуть використовувати свою внутрішню пам'ять для обробки довільних послідовностей входів. Це робить їх застосовними до таких задач, як

розпізнавання несегментованого неперервного рукописного тексту[4] та розпізнавання мовлення[5].

1.2.1 Повнорекурентна мережа

Це основна архітектура, розроблена в 1980-х роках: мережа нейроноподібних вузлів, кожен з орієнтованим з'єднанням до кожного іншого вузла. Кожен з вузлів має змінну в часі дійснозначну активацію. Кожне з'єднання має змінювану дійснозначну вагу. Деякі з вузлів називаються входовими вузлами, деякі — виходовими, а решта — прихованими вузлами. Більшість із наведених нижче архітектур є окремими випадками.

Для постановок керованого навчання з дискретним часом тренувальні послідовності входових векторів стають послідовностями активацій входових вузлів, по одному вектору на кожен момент часу. В кожен заданий момент часу кожен не входовий вузол обчислює свою поточну активацію як нелінійну функцію від зваженої суми активацій всіх вузлів, від яких до нього надходять з'єднання. Для деяких із виходових вузлів на певних тактах можуть бути задані вчителем цільові актвації. Наприклад, якщо входова послідовність є мовленнєвим сигналом, що відповідає вимовленій цифрі, то кінцевий цільовий вихід у кінці послідовності може бути міткою, яка класифікує цю цифру. Для кожної послідовності її похибка є сумою відхилень усіх цільових сигналів від відповідних активацій, обчислених мережею. Для тренувального набору численних послідовностей загальна похибка є сумою похибок усіх окремих послідовностей.

У постановках навчання з підкріпленням не існує вчителя, який надавав би цільові сигнали для РНМ, натомість час від часу застосовується функція пристосованості або функція винагороди для оцінювання

продуктивності РНМ, яка впливає на її входовий потік через виходові вузли, з'єднані з приводами, що впливають на середовище.

1.2.2 Рекурсивні нейронні мережі[6]

Рекурсивна нейронна мережа створюється рекурсивним застосуванням одного й того ж набору ваг до диференційовної графоподібної структури шляхом обходу цієї структури в топологічній послідовності. Також, такі мережі зазвичай тренують зворотним режимом автоматичного диференціювання. Їх було введено для навчання розподілених представлень структури, таких як терміни логіки. Окремим випадком рекурсивних нейронних мереж є самі РНМ, чия структура відповідає лінійному ланцюжкові. Рекурсивні нейронні мережі застосовували до обробки природної мови. Рекурсивна нейронна тензорна мережа використовує функцію компонування на основі тензорів для всіх вузлів дерева.

1.2.3 Мережі Елмана та Джордана

Мережа Хопфілда становить історичний інтерес, хоч вона й не є загальноприйнятою РНМ, оскільки її побудовано не для обробки послідовностей зразків. Вона натомість вимагає стаціонарних входів. Вона є РНМ, в якій усі з'єднання є симетричними. Винайдена Джоном Хопфілдом 1982 року, вона гарантує, що її динаміка збігатиметься. Якщо з'єднання тренуються із застосуванням геббового навчання, то мережа Хопфілда може працювати як робастна асоціативна пам'ять, стійка до змін з'єднань.

Одним із варіантів мережі Хопфілда є двоспрямована асоціативна пам'ять. ДАП має два шари, кожен з яких можна використовувати як входовий, щоби викликати асоціацію й виробити вихід на іншому шарі.

Як правило, рекурентний багатошаровий перцептрон складається з ряду каскадованих підмереж, кожна з яких складається з декількох шарів[7]

вузлів. Кожна з цих підмереж є мережею прямого поширення повністю, крім останнього шару, який може мати зворотні зв'язки всередині себе. Кожни з цих підмереж під'єднується лишу зв'язками прямого поширення.

1.2.4 Тренування

Градієнтний спуск

Щоби мінімізувати загальну похибку, може застосовуватися градієнтний спуск для зміни кожної ваги пропорційно похідній похибки по відношенню до цієї ваги, за умови, що нелінійні функції активації є диференційовними. Для здійснення цього в 1980-х і на початку 1990-х років було розроблено різні методи Полом Вербосом, Рональдом Вільямсом, Тоні Робінсоном, Юргеном Шмідгубером, Зеппом Хохрайтером, Бараком Перлмуттером та іншими.

Стандартний метод називається «зворотне поширення в часі», або ЗПЧ, і є узагальненням зворотного поширення для мереж прямого поширення, і, як і той метод, є зразком автоматичного диференціювання в режимі зворотного накопичення, або принципу мінімуму Понтрягіна. Обчислювально витратніший інтерактивний варіант називається «рекурентне навчання в реальному часі», або РНРЧ, і є зразком автоматичного диференціювання в режимі послідовного накопичення зі складеними векторами тангенсів. На відміну від ЗПЧ, цей алгоритм є локальним в часі, але не локальним у просторі[8].

В цьому контексті локальний у просторі означає, що вектор ваг вузла може бути уточнено лише із застосуванням інформації, що зберігається в з'єднаних вузлах та самому вузлі, так що складність уточнення одного вузла є лінійною по відношенню до розмірності вектору ваг. Локальний в часі означає, що уточнення відбуваються неперервно (інтерактивно), і залежать лише від найнещодавнішого такту, а не від декількох тактів у межах заданого

обрію часу, як у ЗПЧ. Біологічні нейронні мережі видаються локальними як у часі, так і в просторі.

Недоліком РНРЧ є те, що для рекурсивного обчислення часткових похідних від має часову складність O(кількість прихованих × кількість ваг) на такт для обчислення матриць Якобі, тоді як ЗПЧ займає лише O(кількість ваг) на такт, ціною, проте, зберігання всіх прямих активацій в межах заданого обрію часу.

Існує також інтерактивний гібрид ЗПЧ і РНРЧ з проміжною складністю, і є варіанти для неперервного часу. Головною проблемою градієнтного спуску для стандартних архітектур РНМ є те, що градієнти похибки зникають експоненційно швидко з розміром часової затримки між важливими подіями. Як спробу подолання цих проблем було запропоновано архітектуру довгої короткочасної пам'яті разом з гібридним методом навчання ЗПЧ/РНРЧ.

Крім того, інтерактивний алгоритм, що називається причинним рекурсивним зворотним поширенням, реалізує та поєднує разом парадигми ЗПЧ та РНРЧ для локальної рекурентної мережі. Він працює з найзагальнішими локально рекурентними мережами. Алгоритм ПРЗП може мінімізувати глобальну похибку; цей факт призводить до поліпшеної стійкості алгоритму, забезпечуючи об'єднавчий погляд на методики градієнтних обчислень для рекурентних мереж із локальним зворотним зв'язком.

Цікавий підхід до обчислення градієнтної інформації в РНМ довільних архітектур, що запропонували Ван та Буфе, ґрунтується на діаграмному виведенні графів плину сигналу для отримання пакетного алгоритму ЗПЧ, тоді як Камполуччі, Унчіні та Піацца запропонували його швидку інтерактивну версію на основі теореми Лі для обчислення чутливості мереж.

1.2.5 Методи глобальної оптимізації

Тренування ваг у нейронній мережі можливо моделювати як нелінійну задачу глобальної оптимізації. Цільову функцію для оцінки пристосованості або похибки певного вагового вектора може бути сформовано таким чином: Спершу ваги в мережі встановлюються відповідно до цього вагового вектора. Далі, мережа оцінюється за тренувальною послідовністю. Як правило, для представлення похибки поточного вагового вектора використовують суму квадратів різниць між передбаченнями та цільовими значеннями, вказаними в тренувальній послідовності. Потім для мінімізації цієї цільової функції може бути застосовано довільні методики глобальної оптимізації.

Найуживанішим методом глобальної оптимізації для тренування РНМ є генетичні алгоритми, особливо в неструктурованих мережах. Спочатку генетичний алгоритм кодується вагами нейронної мережі в наперед визначеному порядку, коли один ген у хромосомі представляє одне зважене з'єднання, і так далі; вся мережа представляється єдиною хромосомою. Функція пристосованості обчислюється наступним чином: 1) кожна вага, закодована в хромосомі, призначається відповідному зваженому з'єднанню мережі; 2) потім тренувальний набір зразків представляється мережі, яка поширює вхідні сигнали далі; 3) до функції пристосованості повертається середньоквадратична похибка; 4) ця функція потім веде процес генетичного відбору.

Популяцію складають багато хромосом; таким чином, багато різних нейронних мереж еволюціюють, поки не буде досягнуто критерію зупинки. Поширеною схемою зупинки є: 1) коли нейронна мережа засвоїла певний відсоток тренувальних даних, або 2) коли досягнуто мінімального значення середньоквадратичної похибки, або 3) коли було досягнуто максимального числа тренувальних поколінь. Критерій зупинки оцінюється функцією пристосованості при отриманні нею оберненого значення середньоквадратичної похибки з кожної з нейронних мереж під час тренування. Отже, метою генетичного алгоритму є максимізувати функцію пристосованості, знизивши таким чином середньоквадратичну похибку.

Для пошуку доброго набору ваг можуть застосовуватися й інші методики глобальної (та/або еволюційної) оптимізації, такі як імітація відпалу та метод рою часток.

1.3 Автокодувальник

Автокодувальник, автоасоціатор або мережа діабло — це штучна нейронна мережа, що використовується для навчань ефективних кодувань. Метою автокодувальника є навчитися представленню (кодуванню) набору даних, зазвичай задля зниження розмірності. Нещодавно концепція автокодувальника стала застосовуватися ширше для навчання породжувальних моделей даних.

Архітектурно найпростішою формою автокодувальника є не-рекурентна нейронна мережа прямого поширення, що є дуже подібною до багатошарового перцептрону (БШП) із вхідним шаром, вихідним шаром та одним або декількома прихованими шарами, що з'єднують їх. Відмінності між автокодувальниками та БШП, однак, полягають в тому, що в автокодувальнику вихідний шар має таку ж кількість вузлів, як і вхідний шар, і в тому, що замість тренування передбаченню цільового значення Y для

заданих входів *X*, автокодувальники тренують відбудові їхніх власних входів *X*. Таким чином, автокодувальники є моделями спонтанного навчання.

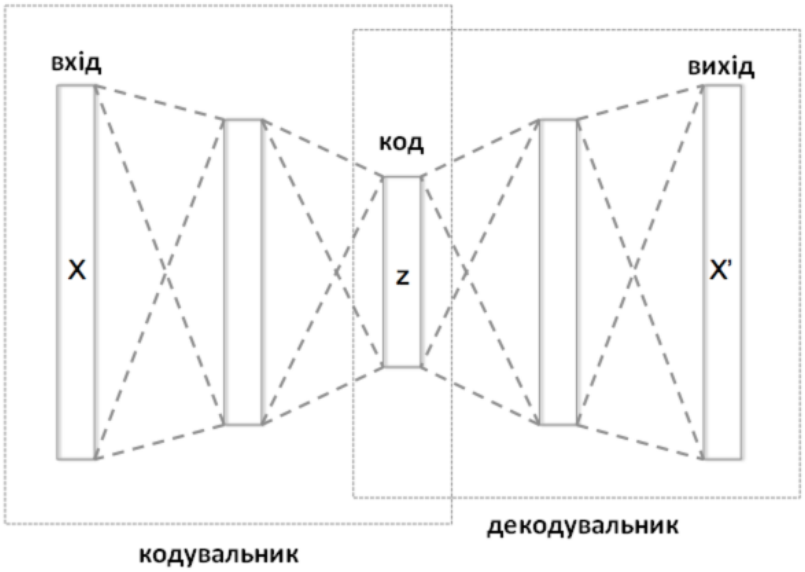


Рисунок 1.5 – Схематична структура автоасоціатора з трьома повноз'єдними прихованими шарами

Автокодувальник завжди складається з двох частин, кодувальника та декодувальника, які може бути визначено як переходи та , так, що:

: X F

: F X

В найпростішому випадку, коли є лише один прихований шар, автокодувальник бере вхід x і відображує його на z :

Його зазвичай називають кодом або латентними змінними. Тут є поелементною активаційною функцією, такою як сигмоїдна функція або зрізаний лінійний вузол. Після цього z відображується на відбудову такої ж форми, як і x:

Тренування

Алгоритм тренування автокодувальника може бути узагальнено як

для кожного входу *x*: виконати прохід прямого поширення для обчислення активацій на всіх прихованих шарах, а потім і на вихідному шарі, щоби отримати вихід . Виміряти відхилення від входу x (зазвичай застосовуючи квадратичну похибку), зворотно поширити цю похибку мережею та виконати уточнення вагових коефіцієнтів.

Автокодувальники часто тренують із застосуванням одного з багатьох варіантів зворотного поширення (таких як метод спряжених градієнтів найшвидший спуск тощо). І хоча вони часто є досить дієвими, існують фундаментальні проблеми із застосуванням зворотного поширення до тренування мереж із багатьма прихованими шарами. Щойно похибки зворотно поширюються до перших кількох шарів, вони стають дуже маленькими та незначними. Це означає, що мережа майже завжди навчатиметься відбудови усереднення всіх тренувальних даних. І хоча більш передові методи зворотного поширення (такі, як метод спряжених градієнтів) до певної міри можуть розв'язувати цю проблему, вони все одно призводять до дуже повільного процесу навчання, та слабких розв'язків. Цій проблемі можна зарадити застосуванням початкових вагових коефіцієнтів, що є наближенням кінцевого розв'язку. Процес пошуку цих початкових вагових коефіцієнтів часто називають попереднім тренуванням.

Джефрі Хінтон розробив методику попереднього тренування для тренування багатошарових «глибинних» автокодувальників. Цей метод полягає в розгляді кожної сусідньої пари з двох шарів як обмеженої машини Больцмана, так, що попереднє тренування наближує добрий розв'язок, а потім у застосуванні методики зворотного поширення для тонкого налаштування результатів. Ця модель має назву глибинна мережа переконань.

2. ВИБІР МОДЕЛЕЙ ТАРЕАЛІЗАЦІЯ ПРОГРАМИ

2.1 Схожі роботи

Велика кількість робіт в анотуванні заключалась в добуванні ключових речень у вхідному документі і перетворення їх на анотацію.

Люди зі своєї сторони здатні перефразовувати оригінальну історію власними словами. Людська анотація як така є абстрактною і рідко містить оригінальні речення з документа. Задача абстрактного анотування була стандартизована, використовуючи змагання DUC-2003 і DUC-2004. Дані для цих задач складались з сюжетів новин з багатьма анотаціями на кожен сюжет, згенерованими людьми. Найкраща система у задачі DUC-2004 називалась TOPIARY і використовувала комбінацію лінгвістично мотивованих технік стиснення і навчання без вчителя алгоритми визначення теми, що додавали ключові слова, витягнені з статті, до стиснутого виходу. Деякі інші роботи задачі абстрактного анотування включають використання традиційних підходів машинного перекладу, стиснення, використовуючи зважені правила трансформації дерева, квазісинхронні граматичні підходи.

Разом з появою глибокого навчання, як явної альтернативи для багатьох задач нейролінгвістичного програмування (NLP), дослідники почали враховувати цей фреймворк як привабливу, повну керованих даних альтернативу абстрактної анотації. В поспіху автори використовували конвулюційні моделі для кодування джерела і чутливої до контексту нейронної мережі, щоб генерувати анотацію. В розширенні цієї роботи використовувалать подібна конвулюційна модель для кодування, але замінений декодер на рекурентну нейронну мережу, дав подальші покращення у продуктивності на обидвох датасетах.

У моїй роботі використовується рекурентна нейронна мережа на вхідних та тренувальних даних. Крім того, щоб провести експерименти для

порівняння, запропонований новий набір даних для вхідного документа, на якому встановлені еталонні номера.

Нижче проаналізовані подібності і відмінності моделі, яку використовую я і моделей в схожих роботах анотації.

Багатофункціональний кодувальник. Лінгвістичні особливості такі як морфологічна розмітка, іменовані об’єкти, а також TF та IDF інформація були використані в багатьох підходах до узагальнення, але вони є новими у контексті підходів глибокого навчання для абстрактного анотування.

Модель перемикання генератор-вказівник. Ця модель поєднує в собі добувний та абстрактний підходи до узагальнення в окремому наскрізному фреймворку. Вказівні мережі таокж використовувались раніше для проблем рідкісних слів в контексті машинного перекладу, але додавання перемикача в нашу модель дозволило встановити баланс між тим, коли бути вірним джерелу (наприклад, для іменованих сутностей та слів, що не входять в словник для тренування) і коли дозволено бути креативним. Я вважаю, що такий процес, можливо, імітує те, як людина складає узагальнення.

Модель ієрархічної уваги. Раніше запропонована енкодер-декодер модель використовує увагу тільки на рівні речень. Новизна мого підходу полягає у спільному моделюванні уваги на обидвох рівнях: речень та слів, де увага на рівні слів знаходиться під впливом уваги на рівні речень, таким чином захопивши поняття важливості речень і важливість слів в рамках цих речень.

2.2 Енкодер-декодер рекурентна нейронна мережа

Базова модель відповідає моделі нейронного машинного перекладу. Енкодер складається з двобічної GRU-RNN[9], в той час як декодер складається з односпрямованої GRU-RNN з таким самим прихованим станом розміру як і енкодер та механізму уваги над джерелом прихованого стану.

Також декодер включає в себе максимально м’який шар над словником тренування для генерування слів.

На додачу до базової моделі, до задачі анотації адаптовано великий словниковий запас. У моєму підході декодер-словник кожної міні партії є обмежений словами у документі-джерелі. Крім цього, найбільш частовживані слова додаються до цільового словника допоки словник не досяг фіксованого розміру. Метою даної методики є зменшення розміру максимально м’якого рівня декодера, який є головним обчислювальним місцем. Ця технологія також пришвидшує зближення, зосередивши зусилля моделювання тільки на словах, які мають важливе значення для даного прикладу. Цей метод особливо добре підходить для анотування, оскільки більшість слів анотування приходять, в будь якому випадку, з вхідного документа.

2.3 Захоплення ключових слів, використовуючи багатофункціональний кодувальник

У анотуванні одним з ключових завдань є визначити основні поняття і ключові об’єкти в документі, навколо якого розгортається історія. Для того, щоб досягти цієї мети, потрібно буде виходити за рамки вбудованих слів на основі подання вхідного документа і частково захоплювати додаткові лінгвістичні особливості, такі як частини в промові, іменовані сутності та статистичний показник важливості слів. Тому необхідно створити додаткові, базовані на перегляді, матриці вкладення для словника кожного тегу-типу, аналогічного вкладенням для слів. Для безперервних функцій, таких як визначення важливості слова, потрібно конвертувати їх в категоріальні значення, дискретизувавши їх в фіксоване число контейнерів і використовувати одну гарячу репрезентацію, щоб вказати, в який контейнер вони потрапили. Це дозволяє зіставити їх в матрицю. Нарешті, для кожного слова у вихідному документі, просто переглядаються свої вкладення на всіх пов’язаних тегах і об’єднується їх в єдиний довгий вектор (рисунок 2.1).

2.4 Моделювання рідкісних/невидимих слів, використовуючи перемикання декодер/вказівник

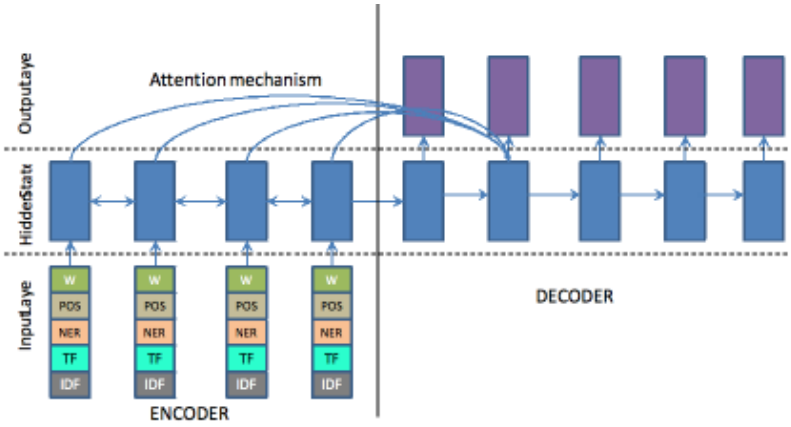


Рисунок 2.1 – Багатофункціональний кодувальник

Часто у анотуванні ключові слова чи іменовані сутності у текстовому документі, що є центральними для узагальнення, можуть бути невидимими чи рідкісними з натренованих даних. Оскільки словниковий запас декодера фіксується під час навчання, він не може емітувати ці невидимі слова. Замість цього, найбільш поширеним способом їх обробки є генерування заповнювача. Однак, це не призводить до розбірливого узагальнення. В анотуванні, інтуїтивним способом впоратися з такими словами, є вказати на їх місце у вхідному документі. Це поняття моделюється за допомогою перемикання декодер-вказівник архітектури, що графічно показана на рисунку 2.2. У цій моделі декодер обладнаний перемикачем, який вирішує чи використовувати генератор чи вказівник на кожному часовому кроці. Якщо ввімкнене перемикання, декодер видає слово з його словника слів для тренування в звичайному режимі. Проте, якщо перемикач вимкнений, декодер генерує вказівник на одну з словесних позицій в джерелі. Слово на місці розташування вказівника потім копіюється до анотації. Перемикач моделюється як сигмоїдна активаційна функція над лінійним шаром, базованим на всьому доступному контексті на кожному часовому кроці як показано нижче.

– ймовірність вмикання перемикання на і-тому шляху декодера, - прихований стан, – вбудований вектор емісії від початкового кроку, – зважений контекст-вектор

, , , , – параметри перемикання. Використовується розподіл уваги по позиціях слів у документі як розподіл на зразку, де вказівник.

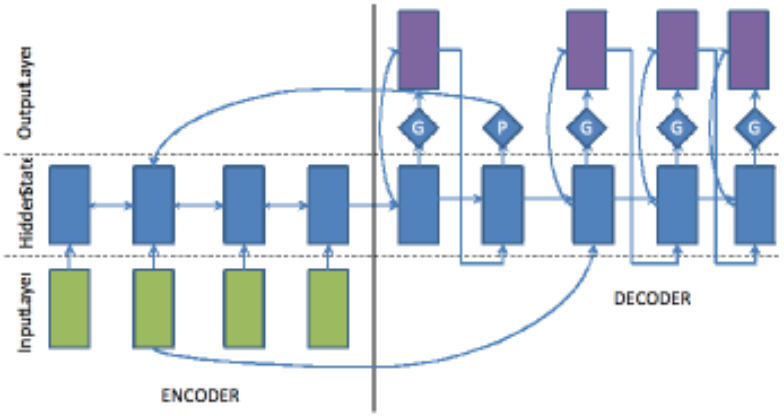
У рівнянні, наведеному вище, – значення вказівника на і-ту позицію слова у анотації, зразки якого отримані з розподілу уваги над документом словесних позицій , де – ймовірність і-того часового кроку у декодері, що вказує на і-ту позицію в документі, та – прихований стан декодера на позиції .

У час тренування, запропонована модель точно вказує на інформацію, коли слова анотування не існують в словнику слів для тренування. Коли слово, що знаходиться поза словником для тренування, з'являється в анотації на багатьох позиціях документа, розривається зв'язок, коли воно з'явилось вперше. У часі тренування, я оптимізовую умовну логарифмічну функцію правдоподібності, показану нижче, додатковим регулюванням штрафів.

x і y – анотація і документ слів відповідно, - індикатор функції, що встановлений в 0 всякий раз, коли слово на позиції *i* в анотуванні є виключенням з словника декодера. На час тестування модель автоматично визначає на кожному часовому етапі чи слід генерувати вказівник, базований на ймовірності перемикання . Ми легко використовуємо максимальний

аргумент ймовірності чи генерацію вказівника на кращий вихід на кожному часовому кроці.

Механізм вказівника може бути більш надійним в обробці рідкісних слів, тому що він використовує кодування прихованого стану репрезентації рідкісних слів, щоб визначити, на яке слово з документа вказати. Від тоді, коли прихований стан залежить від усього контексту слова, модель здатна точно вказати на невидимі слова, хоча вони не з’являються в словнику для



тренування.

Рисунок 2.2 – Модель декодер/вказівник перемикання

2.5 Захоплення ієрархічної структури документа ієрархічною увагою

У датасетах, коли вхідний документ дуже довгий, на додачу до визначення ключових слів в документі, також важливо визначити ключові речення, за допомогою яких може бути складена анотація. Ця модель використовує дві двобічні рекурентні нейронні мережі: одну на рівні слів, іншу на рівні речень, щоб захопити обидва рівні значущості. Механізм уваги працює на обидвох рівнях одночасно. Увага на рівні слів повторно зважується за відповідною увагою на рівні речень і перенормовується, як показано нижче:

де - вага уваги на рівні слів на -тій позиції в документі, – ідентифікатор речення на -тій позиції слова, - вага уваги на рівні речень для -ного речення в джерелі, - кількість слів у вхідному документі, – перенесена увага на -тій позиції слова. Перенесена увага використовується для обчислення зваженого контекстного вектора, який йде в якості вхідного прихованого стану декодера. Крім цього, я з'єдную додаткові позиційні вкладення з прихованим станом на рівні речень рекурентної нейронної мережі з моделлю позиційної важливості в реченні і документі. Тому ця архітектура моделює ключові речення так само, як і ключові слова в цих реченнях. Графічна репрезентація цієї моделі показана на рисунку 2.3.

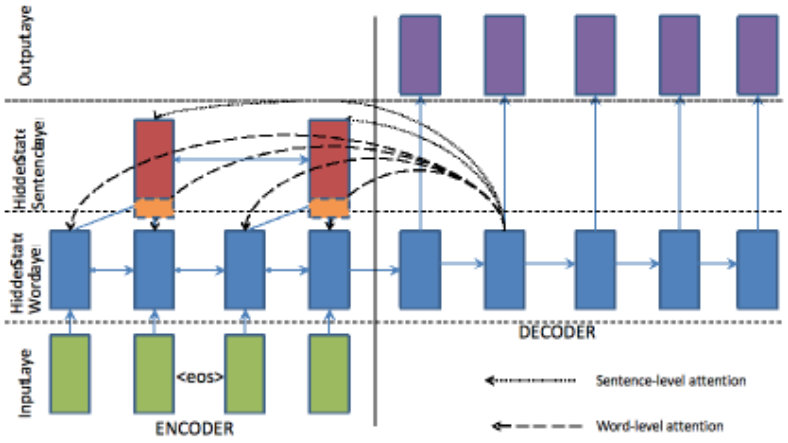


Рисунок 2.3 – Ієрархічний енкодер з ієрархічною увагою

2.5 Реалізація вибраної моделі

Для реалізації я вибрала модель послідовність до послідовності з ієрархічною увагою. Я реалізувала клас моделі *class Seq2SeqModel*. Цей клас реалізує багаторівневу рекурентну нейронну мережу, що складається з

енкодера і базованого на увазі декодера. У класі можлива зміна конфігурації для тренування з LSTM на GPU і навпаки. Клас приймає на вхід такі параметри:

1. Розмір словника вхідного документа;

2. Розмір словника для тренування;

3. Список пар (I, O), де I задає максимальну довжину входу, що буде оброблена в цьому бакеті, а O задає максимальну довжину виходу. Екземпляри для тренування, що мають вхідну довжину більшу, ніж I, чи екзепляри виходу, що мають довжину більшу, ніж O, будуть перенесені в наступний бакет і відповідно доповнені. Припускаємо, що список відсортовано, наприклад [(2, 4), (8, 16)];

4. Кількість вузлів на кожному рівні моделі;

5. Кількість слоїв моделі;

6. Норма градієнта (градієнти будуть обрізані максимально, відповідно до цієї норми);

7. Розмір партій, що використовуватимуться під час навчання. Констукція моделі не залежить від розміру партій, тому його можна змінити після ініціалізації, якщо це зручно, наприклад, для декодингу;

8. Початкова швидкість навчання;

9. Падіння швидкості навчання, коли це необхідно;

10. Використання LSTM (якщо встановлено true, використовється для тренування LSTM замість GPU);

11. Кількість зразків для вибірки;

12. Конфігурація тільки вперед (якщо встановлено true, не будується зворотній прохід в моделі);

13. Тип даних для зберігання внутрішніх змінних.

Після ініціалізації зчитуються дані (використовуються декілька кортежів для ефективності) з файла вхідного документа і файла анотацій. Ці дані поміщаються в кортежі. За зчитування відповідає метод *read_data*. Він

приймає на вхід шлях до файлу вхідного документа, шлях до файлу анотації і максимальний розмір рядків для зчитування (якщо цей параметр = 0, то обидва файли будуть прочитані повністю). Файл анотації повинен бути вирівняним з файлом вхідного документа.

Метод повертає список довжин. Отриманий датасет dataset[n] містить список пар (source, target), зчитаних з файлів, які вписуються в розмір кортежів, тобто len(source) < buckets[n][0] і len(target) < buckets[n][1], де source і targer – джерело і анотація відповідно. Джерело і анотація є списком ідентифікаторів токенів.

Після цього створється акожен рівень моделі, заданого розміру. Далі зчитуються дані з кортежів (отриманих в методі *read_data*) і вираховуються їх розміри. Розмір кортежів являє собою список зростаючих чисел від 0 до 1, які будуть використовуватись для вибору кортежа. Довжина [scale[i], scale[i+1]] пропорційна розміру і-того кортежа, що бере участь у процесі тренування.

Далі розпочинається цикл тренування, який умовно нескінченни (*while True*). В цикл генерується рандомне число та за допомогою нього обирається кортеж, який буде використовуватись в тренування на даному часовому кроці, Для кожної ітерації циклу тренування, ми заміряємо час виконання від початку до кінся (*end_time – start_time*), щоб показати час, який витрачається на одну ітерацію циклу. Це дозволяє стежити за продуктивністю моделі (вона збільшкється чи падає).

З рандомно згенерованого кортежа отримуємо клас *tensorflow.TFRecords* для вхідних даних енкодера та декодера, що робить за допомогою методу *model.get_batch.* Далі ці дані передаються до методу *model.step*.

Після цього заміряється час одного виклику модулю step (модуль *step* повертає значення функцію втрат на поточних вхідних даних). Далі вихід з модуля *step* агрегується глобальною змінною *loss,* для того, щоб розуміти

значення функції втрат на даному етапі. Також рахується кількість зроблених кроків. Маючи параметр *FLAGS.steps_per_checkpoint* ми кожен крок, який ділиться без остачі на *steps_per_checkpoin,t* зберігаємо всі параметри матриць моделі на диск для подальшого використання у інших викликах програми або з іншими програмами, бо формат, в якому зберігається модель має універсальний характер для усіх користувачів бібліотеки Tenzor Flow. Далі виводяться на екран значення усіх глобальних статистик, таких як:

1. Глобальний крок;
2. Швидкість навчання;
3. Точність моделі;
4. Час, витрачений на тренування;
5. Шлях до файлу, де збережені всі параметри матриці моделі.

Наступний крок – оцінка моделі, де так само викликається метод *step* та агрегується помилка, яка отримана з методу *step,* але вже не для даних для тренування, а на тестових даних. Після цього також виводимо точність моделі, але вже на тестових даних.

Так як метод step є основним в процесі тренування та оцінювання моделі, детально опишу його роботу. Метод приймає на вхід:

1. Session: спеціальний об’єкт фреймоворку Tenzor Flow, задяки якому викликаються усі операції;
2. Encoder_inputs: стаття, яка повинна бути узагальнена, в форматі *tensorflow.TFRecords*;
3. Dencoder_inputs: анотація, яка повинна бути отримана, в форматі *tensorflow.TFRecords*;
4. Forward_only: булевий параметр, щоб робити лише оцінювання моделі, без навчання.

На виході методу отримуємо:

1. Норму градієнтів усіх параметрів або пустий об'єкт, якщо параметр *forward_only = true*;

2. Метрику точності моделі;
3. Анотацію, яка згенерована моделлю.

Метод *step* викликає два методи фрейморку Tenzor Flow: *forward* і *backward* для моделі. Всі інші операції, такі як: пермложення матриць, застосування активаційних функцій, автодиференціювання та оновлення параметрів моделі методами оптимізації, автоматично виконує фреймворк Tenzor Flow. Я лише задаю йому архітектуру моделі.

2.6 Призначення програми

Програму можна використовувати як сторонню Python бібліотеку, яка автоматично завантажить останню найкращу натреновану модель. Цю бібліотеку можна викликати з будь-якої Python програми, як на CPU так і на GPU машинах. Програма має повну документацію, яку можна згенерувати за допомогою стандартного Python модуля sphinx.

Розроблену програму можна використовувати також у багатьох лінгвістичних компонентах. Програма може бути легко видозмінена та має ліцензію.

3. ЕКСПЕРИМЕНТИ ТА РЕЗУЛЬТАТИ

3.1. Gigaword корпус

У цій серії експериментів я використовувала анотований Gigaword корпус, як описано в Rush et al. (2015)[10]. Також використовувала скрипти для попередньої обробки даних, в результаті чого було отримано 3.8M натренованих прикладів. Скрипт також виробив близько 400 тисяч валідацій та тестових прикладів, але я створила випадково підібрану підмножину 2000 прикладів, кожен з яких використовувався для валідації та тестування. Саме на цих прикладах базуються звіти продуктивності. Крім цього був виконаний точний тестовий зразок, який використовувався в Rush et al. (2015), щоб зробити порівняння наших моделей. Також були зроблені невеликі зміни у скрипті для вилучення не тільки слів-токенів, а також системно згенерованих частин мовлення та іменованих сутностей.

Тренування. Для всіх моделей, описаних вище, використовувалися 200-вимірні слово-до-вектора вектори[11], натреновані на такому самому корпусі для ініціалізації моделі вкладень, але їм було дозволено обновлятись під час тренування. Приховані стани розмірностей енкодера і декодера були виправлені на 400 рівні у всіх експериментах. Коли я використовувала тільки перше речення документа як джерело, як це було зроблено у Rush et al. (2015), розмір словника енкодера був 119,505 і те, що декодер встановив – 68,885. Використовувались партії розміром 50 і радномно перемішані дані для тренування на кожній епосі, при сортуванні кожних 10 партій відповідно до їх довжини, щоб прискорити тренування. Я не використовувала відсів чи упорядкування, але використовувала градієнт скорочення слів. Для всіх моделей застосовувався великий словниковий трюк, де обмежується розмір словника до 2000, тому що це зменшує час тренування на кожній епосі приблизно втричі.

Декодування. У час декодування використовується променевий пошук розміру 5 для генерування узагальнення, розмір узагальнення обмежується до розміру 30 слів, так як це максимальний розмір, помічений у зразковому наборі. Було виявлено, що середня довжина анотації на всіх моделях з 7.8 до 8.3 слів, якщо все узгоджено дуже тісно з усіма валідованими послідовностями, і приблизно 8.7 слів без будь-яких конкрентих налаштувань.

Обчислювальні витрати. Я тренувала всі моделі на GPU. Більшість моделей забирало приблизно 10 годин на епоху, за винятком ієрархічної моделі, що витрачала 12 годин на епоху. Всі моделі як правило сходилися протягом 15 епох, використовуючи ранній критерій зупинки на основі вартості перевірки. Тому час тренування до сходження коливається від 6 до 8 днів, залежно від моделі. Генерування узагальнень у тестовому часі є досить швидким з пропускною спроможністю близько 20 анотацій за секунду на одному графічному процесорі (GPU), використовуючи пакети розміром 1.

Показники обчислень. Подібно до (Nallapati et. al., 2016)[12] і (Chopra et al., 2016)[13] я використовую повну довжину F1 варіантів Rounge, щоб оцінити мою систему. Незважаючи на обмежену довжину, рекол був кращим показником для більшості попередніх робіт, один з його недоліків є вибір ліміту довжини, що варіюється від корпусу до корпусу, роблячи порівняння виконання важкими для дослідників. Повнорозмірний рекол, з іншого боку, не вимагає обмеження довжини, але віддає більш довгі анотації. Повнорозмірний F1 вирішкє цю проблему так як може не пропустити довші анотації, замість того, щоб нав’язувати спеціальне обмеження довжини.

Крім того, я показую відсоток токенів у системі анотування, що з’являються в джерелі (називаємо їх “Швидкість копіювання” у Таблиці 3.1). Нижче описані експерименти та результати на корпусі Gigaword.

words-lvt2k-1sent: Це базова модель кодувальника-декодувальника з великим лексичним трюком. Ця модель навчається тільки на першому реченні з вхідного документа так, як це зроблено в Rush et al. (2015).

words-lvt2k-2sent: Ця модель ідентична моделі вище, за винятком того, що вона навчається на перших двох реченнях з вхідного документа. На цьому корпусі додані додаткові речення у вихідний файл, що, по всій видимості, допомагають підвищити продуктивність, як показано в Таблиці 3.1. Я також намагалась додавати більше речень, але продуктивність знизилась, що ймовірно зв'язано з тим, що останні речення в цьому корпусі не мають відношення до анотації.

words-lvt2k-2sent-hieratt: Оскільки використовувалось два речення з вхідного документа, я підготувала ієрархічну модель уваги, запропоновану в розділі 2.4. Як показано в Таблиці 3.1, ця модель підвищує продуктивність в порівнянні до її більш плоскої копії, автоматично вивчаючи відносну важливість перших двох речень в автоматичному режимі.

feats-lvt2k-2sent: Тут я, як раніше, треную по перших двох реченнях, але використовую розмовні частини та іменовані об'єкти в Gigaword корпусі, а також TF, IDF, щоб вхідні вкладення на стороні джерела, як описано в розділі 2.2. В цілому, вектор впровадження виріс від вхідного 100 до 155 і здійснив збільшення приросту в порівнянні з його аналогом *wordslvt2k-2sent*, як показано в Таблиці 1, демонструючи корисність синтаксису в цій задачі.

feats-lvt2k-2sent-ptr: Ця модель перемикання генератор/вказівник . описана в розділі 2.3, але, крім того, я використовую повнофункціональні вкладення на стороні документа, як в наведеній вище моделі. Експерименти показують, що ва модель здатна досягнути найкращої продуктивності на моєму тестовому наборі всіма трьома варіантами Rouge, як показано в Таблиці 3.1.

Порівняння з сучасними моделями. Я порівняла продуктивність моїх моделей words-lvt2k-1sent з сучасними моделями прикладі, створеному Rush

et al. (2015), як показано в нижній частині Таблиці 1. Я також натренувала іншу систему words-lvt5k-1sent, що має більший LVT словник розміром 5000, але також має набагато більші вхідні і цільові словники: 200 і 400 тисяч відповідно.

Причина, через яку я не оцінила найкращі моделі валідації, в тому, що цей набір тестів складався лише з одного речення з вхідного документа і не включав примітки NLP, що необхідні в цих найкращих моделях. Таблиця показує, що, не дивлячись на цей факт, моя модель перевершує модель Rush et al. (2015) ABS+ зі статистичною значимістю. Крім того мої підібрані моделі також демонструють кращу абстрактну властивість.

Я вважаю, що двонаправлений RNN, який я використовувала для моделювання джерела, захоплює більш насичену контекстну інформацію кожного слова, ніж уявлення сумарного вкладення, що використовується Rush et al. (2015) і Chopra et al. (2016) у їхніх згорткових енкодерах уваги. Крім того, явне моделювання важливої інформації, таких як безліч вхідних речень, лінгвістичні функції на рівні слів, використання механізму перемикання для вказання вхідних слів, коли це потрібно та ієрархічна увага – вирішують конкретні проблеми в анотуванні, кожен з яких збільшує продуктивність.

Таблиця 3.1

Порівняння продуктивності на різних моделях

Назва моделі	Rounge-1	Rounge-2	Rounge-L	Відсоток копіювань
Повна довжина F1 на моєму внутрішньому наборі				
words-lvt2k-1sent	34.97	17.17	32.70	75.85
words-lvt2k-2sent	35.73	17.38	33.25	79.54
words-lvt2k-2sent-hieratt	36.05	18.17	33.52	78.52
feats-lvt2k-2sent	35.90	17.57	33.38	78.92
feats-lvt2k-2sent-ptr	***36.40**	**17.77**	***33.71**	78.70
Повна довжина F1 на наборі (Rush et al., 2015)				
ABS+ (Rush et al., 2015)	29.78	11.89	26.97	91.50
words-lvt2k-1sent	32.67	15.59	30.64	74.57

RAS-Elman (Chopra et al., 2016)	33.78	15.97	31.15	
words-lvt5k-1sent	*35.30	16.64	*32.62	

3.2. DUC корпус

DUC корпус складається з двох частин: корпус 2003 року, що містить 624 документи, частини анотації, та корпус 2004 року, що містить 500 пар. Оскільки ці корпуси занадто малі для навчання великих нейронних мереж, Rush et al. (2015) натренували свої моделі на корпусі Gigaword, але об'єднали їх з додатковою логарифмічною моделлю екстрактного анотування з функціями ручної роботи, які тренуються на корпусі DUC 2003. Оригінальна нейронна модель уваги називається ABS і комбінована модель ABS+. Chopra et al. (2016) також заявляють про продуктивність своєї RASElman моделі на цьому корпусі і вважаються поточним еталоном, тому що вона перевершує всі раніше опубліковані вихідні дані, включаючи екстраверсійні та абстрактні системи на основі нейронної мережі, виміряні офіційною DUC метрикою реколу 75 байтів. На цих експериментах використовується одна і та сама метрика для оцінки моделей, але не враховуються звіти з інших систем для економії місця.

В моїй роботі я просто запускаю моделі, натреновані на Gigaword корпусі, так як їх не треба налаштовувати на валідаційному наборі DUC. Єдина зміна, яку я внесла в декодер, для того, щоб оминати кінець анотації змушувати її публікувати точно 30 слів для кожної анотації, так як офіційна оцінка цього корпусу базується на відгуку Rouge рекол з обмеженою довжиною. На цьому корпусі так само, оскільки є тільки одне речення з джерела і немає NLP приміток, запускається тільки модель words-lvt2k-1sent і words-lvt5k-1sent.

Продуктивність цієї моделі на тестовому наборі порівнюється з моделями ABS і ABS+, RASElman від (Chopra et al., 2016), а також TOPIARY

– найбільш ефективною системою на DUC-2004 у Таблиці 3.2. Я відмітила, що найкраща модель words-lvt5k-1sent перевершує RAS-Elman на двох з трьох варіантів Rouge, будучи конкурентоспроможними на Rouge-1.

Таблиця 3.2

Оцінка моделей з використанням Rouge рекол

Модель	Rouge-1	Rouge-2	Rouge-L
TOPIARY	25.12	6.46	20.12
ABS	26.55	7.06	22.05
ABS+	28.18	8.49	23.81
RAS-Elman	**28.97**	8.26	24.06
words-lvt2k-1sent	28.35	**9.46**	24.59
words-lvt5k-1sent	28.35	9.42	**25.24**

3.3 CNN/Daily Mail корпус.

Існуючі корпуси абстрактного анотування текстів, включаючи Gigaword і DUC містять тільки одне речення в кожній анотації. В цьому розділі описується новий корпус, що охоплює анотацію з багатьох речень. Щоб відтворити цей корпус, я модифікую існуючий корпус, який використовувався для задачі відповісти на питання вказання шляху (Hermann et al., 2015). У цій роботі автори використовували згенеровані людьми анотації з новин на CNN та Daily Mail веб-сайтів як питання (з одним з прихованих об’єктів), а також розповіді як відповідні уривки, з яких очікується, що система заповнить ними відповіді на питання. Автори зарелізили скрипти, які сканують, витягують та генерують пари проходів і питань з цих сайтів. З простою модифікацією скрипта я змогла відновити всі частини анотації кожної історії в оригінальному порядку для того, щоб отримати анотацію, яка складається з багатьох речень. В цілому цей корпус містить 286,817 пар для тренування, 13,368 пар для валідації і 11,487 тестових пар. Вхідні документи в наборі для тренування містять 766 слів, що захоплюють в середньому 29.74 речення, а анотацію складають 53 слова і 3.72 речення. Унікальні характеристики цього набору даних та як довгі

документи і впорядковані анотації, що складаються з декількох речень, представляють цікаві проблем, і я надіюсь на них звернуть увагу більше дослідників. Набір даних був випущений в двох версіях: один складався з фактичних імен сутностей, а в іншому екземпляри сутностей замінені на цілі імена документів, починаючи з 0. Оскільки розмір словникового запасу в анонімній версії менший, я використовувала її у всіх експериментах, описаних нижче. Я обмежила розмір вхідного словника до 150 тисяч, і словника для тренування до 60 тисяч, довжину документа-джерела і анотації до 800 і 100 слів відповідно. Використовувалось 100-розмірне вкладення word2vec, установлений прихований стан моделі розміром 200. Я також створила явні вказівники в даних для навчання, співвідносячи тільки анонімні ідентифікатори об'єктів між джерелом і вихідним результатом на аналогічних рядках, як це було зроблено для слів, що виходять за межі словника, на Gigaword корпусі.

Обчислювальні витрати. Я використовувала одну GPU для тренування всіх моделей на цьому наборі даних. В той час, як плоскі моделі (words-lvt2k і wordslvt2k-ptr) займали менше 5 годин в епоху, ієрархічна модель уваги була дуже дорогою, споживаючи майже 12.5 годин в епоху. Збіжність всіх моделей також повільніша на цьому наборі даних, порівняно з Gigaword, займаючи близько 35 епох для всіх моделей. Таким чином, години для тренування до конвергенції дорівнюють близко 7 дням для плоских моделей і майже 18 днів для моделі ієрархічної уваги. Декодування також повільніше, з пропускною здатністю 2 приклади за секунду для плоских моделей і 1.5 – для ієрархічної моделі уваги при запуску на одному графічному процесорі і для партії розміром 1.

Оцінка. Я оцінила моделі з використанням повнорозмірної метрики Rouge F1, яку я також використовувала для корпуса Gigaword, але з однією помітною різницею: як в системних так і в золотих анотаціях я розглядала кожнувиділену фігуру як окреме речення.

Результати. Результати з основного кодувальника-декодувальника уваги, а також ієрархічної модері уваги представлені в Таблиці 3.3. Хоча цей набір даних менший і складніший, ніж корпус Gigaword, цікаво відмітити, що номери Rouge знаходяться в одному діапазоні. Однак ієрархічна модель уваги, описана в розділі 2.4, перевершує базовий декодувальник уваги тільки незначним чином.

Таблиця 3.3

Виконання різних моделей на тестовому наборі CNN/Daily Mail з використанням повнорозмірної метрики Rouge-F1

Модель	Rouge-1	Rouge-2	Rouge-L
words-lvt2k	32.49	11.84	29.47
words-lvt2k-hieratt	32.75	12.21	29.01
words-lvt2k-temp-att	***35.46**	***13.30**	***32.65**

При візуальному контролі виходу системи, я помітила, що на цьому наборі даних, обидві моделі створювали анотації, які іноді містили фрази чи навіть речення, що повторюються. Оскільки в анотаціях на цьому наборі містилось декілька речень, цілком ймовірно, що декодер “забуває”, яка частина документа використовувалась при створенні більш ранніх моментів. Щоб побороти цю проблему, я використовувала модель тимчасової уваги Sankaran et al. (2016), що відслідковувала минулі ваги уваги декодера і експліцитно відмовляла його від участі в тих же частинах документа в майбутньому. Модель працює так, як показано наступною простою формулою:

—

де - ненормалізований вектор уваги ваги на *t*-тому часовому кроці декодера. Іншими словами, модель тимчасової уваги зменшує вагу уваги на поточному часовому кроці, якщо попередні ваги уваги високі на одній і тій самій частині документа.

Використовуючи цю стратегію, модель тимчасової уваги значно підвищує продуктивність як базової моделі, так і ієрархічної моделі уваги. Я також помітила, що повторення коротких анотацій, отриманих цією моделлю, менше, як це показано на прикладі в Таблиці 3.4.

Ці результати, хоча і є попередніми, повинні служити хорошій орієнтації для майбутніх досліджень цих моделей.

Таблиця 3.4

Порівняння істинної анотації з анотаціями, отриманими з різних моделей

Вхідний документ
(@entity0) wanted : film director , must be eager to shoot footage of golden lassos and invisible jets . @entity0 confirms that @entity5 is leaving the upcoming " @entity9 " movie (the hollywood reporter first broke the story) . @entity5 was announced as director of the movie in november . @entity0 obtained a statement from @entity13 that says," given creative differences , @entity13 and @entity5 have decided not to move forward with plans to develop and direct ’ @entity9 ’ together . " (@entity0 and @entity13 are both owned by @entity16 .) the movie , starring @entity18 in the title role of the @entity21 princess , is still set for release on june 00 , 0000 . it ’s the first theatrical movie centering around the most popular female superhero . @entity18 will appear beforehand in " @entity25 v. @entity26 : @entity27 , " due out march 00 , 0000 . in the meantime, @entity13 will need to find someone new for the director ’s chair.
Істинна анотація
@entity5 is no longer set to direct the first " @entity9 " theatrical movie @entity5 left the project over " creative differences " movie is currently set for 0000
words-lvt2k
@entity0 confirms that @entity5 is leaving the upcoming " @entity9 " movie @entity13 and @entity5 have decided not to move forward with plans to develop @entity0 confirms that @entity5 is leaving the upcoming " @entity9 " movie
words-lvt2k-hieratt
@entity5 is leaving the upcoming " @entity9 " movie the movie is still set for release on june 00 , 0000 @entity5 is still set for release on june 00 , 0000
words-lvt2k-temp-att
@entity0 confirms that @entity5 is leaving the upcoming " @entity9 " movie the movie is the first film to around the most popular female actor @entity18

will appear in " @entity25 , " due out march 00 , 0000

Іменовані об’єкти і числа анонімізуються скриптом попередньої обробки. Теги “<eos>” представляють границю між двома основними моментами. Модель тимчасової уваги (words-lvt2k-temp-att) вирішує проблему анотування повторень, що є в моделях words-lvt2k і words-lvt2k-hieratt, змушуючи модель уваги зосередитись на непокритих частинах документа.

3.4 Аналіз якості

В Таблиці 3.5 представлено декілька високоякісних і неякісних результатів на основі валідаційному наборі, встановленому на основі feats-lvt2k- 2sent – одній з найкращих моделей. Навіть, коли модель відрізняється від підсумкового узагальнення, її узагальнення, як правило, дуже значимі і актуальні. З іншої сторони, модель іноді неправильно інтерпретує семантику тексту і генерує анотацію з комічною інтерпретацією, як показано на прикладах низької якості в таблиці. Очевидно, що захоплення сенсу складних речень – складність цих моделей.

Таблиця 3.5

Приклади згенерованих анотацій з найкращої моделі на корпусі Gigaword

Результати анотацій хорошої якості
S: a man charged with the murder last year of a british backpacker confessed to the slaying on the night he was charged with her killing , according to police evidence presented at a court hearing tuesday . ian douglas previte , ##, is charged with murdering caroline stuttle , ## , of yorkshire , england **T**: man charged with british backpacker ’s death confessed to crime police officer claims **O**: man charged with murdering british backpacker confessed to murder
S: following are the leading scorers in the english premier league after saturday ’s matches: ## - alan shearer -lrbnewcastle united -rrb- , james beattie . **T**: leading scorers in english premier league **O**: english premier league leading scorers
S: volume of transactions at the nigerian stock exchange has continued its

decline since last week , a nse official said thursday . the latest statistics showed that a total of ##.### million shares valued at ###.### million naira -lrb- about #.### million us dollars -rrb- were traded on wednesday in #,### deals . **T**: transactions dip at nigerian stock exchange **O**: transactions at nigerian stock exchange down
Результати анотацій поганої якості
S: broccoli and broccoli sprouts contain a chemical that kills the bacteria responsible for most stomach cancer , say researchers , confirming the dietary advice that moms have been handing out for years . in laboratory tests the chemical , , killed helicobacter pylori , a bacteria that causes stomach ulcers and often fatal stomach cancers . **T**: for release at #### mom was right broccoli is good for you say cancer researchers **O**: broccoli sprouts contain deadly bacteria
S: norway delivered a diplomatic protest to russia on monday after three norwegian fisheries research expeditions were barred from russian waters . the norwegian research ships were to continue an annual program of charting fish resources shared by the two countries in the barents sea region . **T**: norway protests russia barring fisheries research ships **O**: norway grants diplomatic protest to russia
S: j.p. morgan chase ’s ability to recover from a slew of recent losses rests largely in the hands of two men , who are both looking to restore tarnished reputations and may be considered for the top job someday . geoffrey , now the co-head of j.p. morgan ’s investment bank , left goldman , sachs & co. more than a decade ago after executives say he lost out in a bid to lead that firm . **T**: # executives to lead j.p. morgan chase on road to recovery **O**: j.p. morgan chase may be considered for top job

S: вхідний документ, T: підсумкова анотація, O: вихід системи. Не дивлячись на те, що показана однакова кількість якісних і неякісних результатів, хороші показники набагато більше розповсюджені, ніж погані.

Наступний приклад, показаний на рисунку 3.1, відображає приклад виводу з моделі перемикання генератор/вказівник на корпусі Gigaword. З прикладів видно, що модель вчиться правильно використовувати вказівники не тільки для названих об’єктів, але і для багатослівних фраз. Не дивлячись на свою точність, покращення продуктивності всієї моделі незначне. Я

вважаю, що вплив цієї моделі може бути вираженим в інших умовах з більш складним розподілом рідкісних слів. В майбутньому я планую провести більше експериментів з цією моделлю.

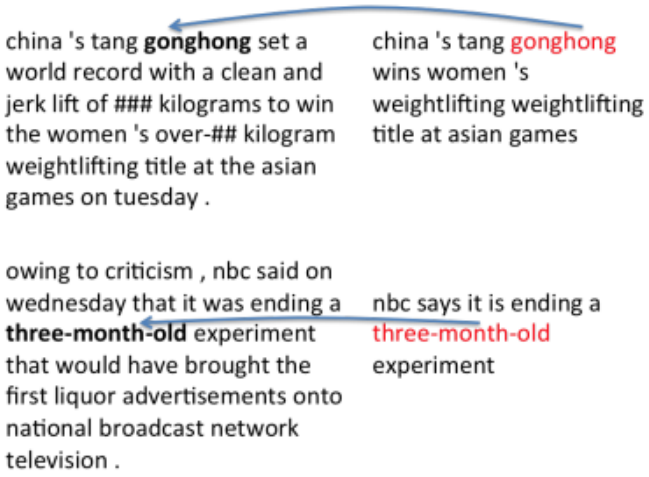


Рисунок 3.1 - Приклад виводу з комутаційних мереж генератор/вказівник

Стрілка показує, що вказівник на вихідну позицію використовувався для генерації відповідного підсумкового слова.

В даних CNN/Daily Mail, хоча мої моделі можуть створювати високоякісні анотації, що містять декілька речень, я помітила, що одне і теж речення чи фраза просто повторюється. Я вважаю, що моделі, які включають всередині себе увагу, такі як Cheng et al. (2016), можуть вирішити цю проблему, побудивши модель “запам’ятати” слова, які вона вже здійснила в минулому.

ВИСНОВКИ

Метою даного дипломного проекту була розробка нейромережевої програми для анотування текстів. Розроблена програма надає стислий опис документа, що був завантажений користувачем на мій веб-сайт.

В цій роботі я застосовувала кодувальник-декодер уваги для задачі абстрактного анотування з дуже багатообіцяючими результатами, що значно перевершують сучасні результати на двох різних наборах даних. Кожна із запропонованих моделей вирішує конкретну проблему абстрактного узагальнення, що приводить до подальшого підвищення продуктивності. Також був запропоновай новий набір даних для анотації, що містить багато речень, і встановлено на ньому контрольні цифри.

В рамках моєї майбутньої роботи, я планую зосередити свої зусилля на цих даних і побудувати більш надійні моделі для анотацій, що складаються з багатьох речень.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Convolutional Neural Networks (LaNet) – Режим доступу http://deeplearning.net/tutorial/lenet.html (дата звернення 24.04.2017). – Назва з екрану.
2. Katsuhiko Mori. Subject independent facial expression recognition with robust face detection using a convolutional neural network – Neural Networks, 2003. – 555 с.
3. Graham Benjamin. Fractional Max-Pooling – arXiv:1412.6071, 2014
4. A Novel Connectionist System for Improved Unconstrained Handwriting Recognition. – Режим доступу: http://people.idsia.ch/~juergen/tpami_2008.pdf. – Дата доступу 12.05.2016.
5. William Senior. Long short-term memory recurrent neural network architectures for large scale acoustic modeling / Proc. Interspeech., 2011. – 5с.
6. Christoph Goller. Learning task-dependent distributed representations by backpropagation through structure – Neural Networks, 1996.
7. Neural Networks as Cybernetic Systems – Електрон. дані (1 файл) – Режим доступу www.brains-minds-media.org/archive/615/bmm615.pdf (дата звернення 17.04.2017) – Назва з екрану.
8. A local learning algorithm for dynamic feedforward and recurrent networks – Електрон. дані (1 файл) – Режим доступу ftp://ftp.idsia.ch/pub/juergen/bucketbrigade.pdf/ (дата звернення 19.04.2017) – Назва з екрану.
9. End-to-End Attention-based Large Vocabulary Speech Recognition – Електрон. дані (1 файл) – Режим доступу https://arxiv.org/abs/1508.04395 (дата звернення 24.04.2017) – Назва з екрану.
10. A neural attention model for abstractive sentence summarization – Електрон. дані (1 файл) – Режим доступу

https://aclweb.org/anthology/D15-1044 (дата звернення 02.05.2017) – Назва з екрану.

11. Parse Server – Електрон. дані (1 файл) – Режим доступу https://www.back4app.com/product/parse-server (дата звернення 08.05.2017) – Назва з екрану.
12. Sequence-to-sequence rnns for text summarization – Електрон. дані (1 файл) – Режим https://openreview.net/pdf (дата звернення 10.05.2017) – Назва з екрану.
13. Abstractive sentence summarization with attentive recurrent neural networks – Електрон. дані (1 файл) – Режим доступу http://www.aclweb.org/anthology/N16-1012 (дата звернення 12.05.2017) – Назва з екрану.