МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

ЛЕКЦІЯ 6. Обробка послідовностей шляхом глибокого навчання

Львів -- 2025

Лекція зі штучного інтелекту 2025-06

Вступ

На цьому занятті ми розглянемо моделі обробки послідовностей — важливу галузь глибокого навчання, яка дозволяє комп'ютерам аналізувати, розуміти та генерувати дані, що мають часову або послідовну структуру. Ми ознайомимося з рекурентними нейронними мережами (RNN), їх варіантами (LSTM, GRU), механізмом уваги (Attention) та архітектурою Трансформерів, які революціонізували обробку природної мови та інші задачі, пов'язані з послідовностями.

Теми, що розглядаються

1. Поняття послідовних даних та моделей послідовностей
2. Обмеження звичайних нейронних мереж для послідовних даних
3. Рекурентні нейронні мережі (RNN)
4. Проблема зникаючого градієнта в RNN
5. Вдосконалені архітектури: LSTM та GRU
6. Двонаправлені RNN
7. Глибокі RNN
8. Моделі послідовність-до-послідовності (Sequence-to-Sequence)
9. Механізм уваги (Attention)
10. Архітектура Трансформерів
11. Застосування моделей послідовностей

Послідовні дані та моделі послідовностей

Що таке послідовні дані?

Послідовні дані — це дані, де порядок елементів має значення і містить важливу інформацію. На відміну від незалежних спостережень, елементи послідовності зазвичай залежать від попередніх елементів.

Приклади послідовних даних:

- **Текст**: речення, абзаци, документи
- **Часові ряди**: фінансові дані, показники датчиків
- **Аудіо**: мовлення, музика, звуки
- **ДНК та білкові послідовності**: генетичний код
- **Відео**: послідовності кадрів
- **Дані користувацької поведінки**: історія переглядів, покупок

Задачі обробки послідовностей:

- **Розпізнавання мовлення**: перетворення аудіо в текст
- **Машинний переклад**: переклад з однієї мови на іншу
- **Генерація тексту**: створення нових текстів
- **Аналіз настроїв**: визначення емоційного забарвлення тексту
- **Розпізнавання іменованих сутностей**: виявлення імен, організацій, дат у тексті
- **Прогнозування часових рядів**: передбачення майбутніх значень
- **Генерація музики**: створення нових музичних композицій

Обмеження звичайних нейронних мереж для послідовних даних

Традиційні нейронні мережі прямого поширення (feed-forward networks) мають суттєві обмеження при роботі з послідовними даними:

Проблеми звичайних нейронних мереж:

1. **Фіксована розмірність входу та виходу**:
    - Звичайні нейронні мережі приймають вхідні дані фіксованої розмірності
    - Послідовності можуть мати різну довжину
    - Навіть з максимальною довжиною та заповненням нулями (padding), це не є ефективним представленням
2. **Відсутність розділення параметрів по часових кроках**:
    - Для кожної позиції в послідовності потрібні окремі параметри
    - Це призводить до надмірної кількості параметрів
    - Неможливість узагальнення на послідовності різної довжини
3. **Відсутність пам'яті**:
    - Звичайні нейронні мережі не зберігають інформацію про попередні входи
    - Не можуть моделювати залежності між елементами послідовності
    - Не враховують контекст
4. **Однонаправленість обробки**:
    - Обробка відбувається за один прохід
    - Неможливість враховувати як попередній, так і наступний контекст одночасно

Приклад проблеми контексту:

- "Він сказав: 'Тедді Рузвельт був великим президентом.'"
- "Він сказав: 'Плюшеві ведмедики продаються зі знижкою!'"
- Дивлячись лише на перші три слова, неможливо визначити, про що йдеться далі.

Рекурентні нейронні мережі (RNN)

Рекурентні нейронні мережі (RNN) — це клас нейронних мереж, спеціально розроблений для обробки послідовних даних. Вони мають "пам'ять", яка дозволяє їм зберігати інформацію про попередні входи.

Основна ідея RNN:

- На кожному кроці часу, RNN отримує поточний вхід та попередній прихований стан
- Обчислює новий прихований стан, який передається на наступний крок
- Параметри мережі розділяються (shared) між усіма кроками часу

Математичне представлення RNN:

$$ a^{\langle t \rangle} = g(W_{aa} a^{\langle t-1 \rangle} + W_{ax} x^{\langle t \rangle} + b_a) $$

$$ y^{\langle t \rangle} = g'(W_{ya} a^{\langle t \rangle} + b_y) $$

де:

- $a^{\langle t \rangle}$ — прихований стан на кроці t
- $x^{\langle t \rangle}$ — вхід на кроці t
- $y^{\langle t \rangle}$ — вихід на кроці t
- W_{aa}, W_{ax}, W_{ya} — вагові матриці
- b_a, b_y — зміщення
- g, g' — функції активації

Спрощена нотація:

Для спрощення, матриці W_{aa} та W_{ax} часто об'єднують в одну матрицю W_a:

$$ a^{\langle t \rangle} = g(W_a [a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_a) $$

Типи RNN архітектур:

1. **Один до одного (One-to-One)**:
    - Стандартна нейронна мережа без рекурентності
    - Приклад: класифікація зображень
2. **Один до багатьох (One-to-Many)**:
    - Один вхід, послідовність виходів
    - Приклад: генерація музики з жанру
3. **Багато до одного (Many-to-One)**:
    - Послідовність входів, один вихід
    - Приклад: аналіз настроїв тексту
4. **Багато до багатьох (Many-to-Many)**:
    - Послідовність входів, послідовність виходів

- Два підтипи:
  - Однакова довжина входу і виходу (розпізнавання іменованих сутностей)
  - Різна довжина входу і виходу (машинний переклад)

Навчання RNN: зворотне поширення в часі (BPTT)

Для навчання RNN використовується алгоритм зворотного поширення в часі (Backpropagation Through Time, BPTT):

1. Виконати пряме поширення через усю послідовність
2. Обчислити функцію втрат для всієї послідовності
3. Виконати зворотне поширення помилки через усі кроки часу
4. Оновити ваги на основі градієнтів

Функція втрат для всієї послідовності:

$$ L = \sum_{t=1}^{T} L^{\langle t \rangle} $$

де $L^{\langle t \rangle}$ — втрата на кроці t.

Мовна модель та генерація послідовностей

Мовна модель — це модель, яка оцінює ймовірність послідовності слів. Вона відповідає на питання: "Наскільки ймовірним є це речення в даній мові?"

Побудова мовної моделі на основі RNN:

1. **Токенізація**: розбиття тексту на слова або підслова
2. **Створення словника**: визначення унікальних токенів
3. **Представлення слів**: перетворення слів у one-hot вектори або ембедінги
4. **Навчання RNN**: прогнозування наступного слова на основі попередніх

Математично, мовна модель обчислює:

$$ P(y^{\langle 1 \rangle}, y^{\langle 2 \rangle}, ..., y^{\langle T_y \rangle}) = \prod_{t=1}^{T_y} P(y^{\langle t \rangle} | y^{\langle 1 \rangle}, y^{\langle 2 \rangle}, ..., y^{\langle t-1 \rangle}) $$

Генерація тексту з мовної моделі:

1. Ініціалізація початкового прихованого стану
2. На кожному кроці:
   - Отримати розподіл ймовірностей для наступного слова
   - Вибрати слово з цього розподілу (жадібно або з семплюванням)
   - Використати це слово як вхід для наступного кроку
3. Повторювати, поки не буде згенеровано кінець послідовності або досягнуто максимальної довжини

Проблема зникаючого градієнта в RNN

Одна з основних проблем стандартних RNN — це проблема зникаючого градієнта (vanishing gradient problem). Під час зворотного поширення помилки через багато кроків часу, градієнти можуть стати надзвичайно малими.

Причини проблеми:

- Повторне множення на матрицю ваг при зворотному поширенні
- Якщо власні значення матриці ваг менші за 1, градієнти експоненційно зменшуються
- Це призводить до того, що мережа не може вивчити довгострокові залежності

Наслідки:

- RNN ефективно "забуває" інформацію з далекого минулого
- Не може моделювати довгострокові залежності
- Навчання стає неефективним для довгих послідовностей

Приклад проблеми: "Я виріс у Франції... Я вільно розмовляю *французькою*."

Стандартна RNN може мати труднощі зі зв'язуванням слова "французькою" зі словом "Франції", якщо між ними багато інших слів.

Вдосконалені архітектури: LSTM та GRU

Для вирішення проблеми зникаючого градієнта були розроблені вдосконалені архітектури RNN.

LSTM (Long Short-Term Memory)

LSTM — це спеціальна архітектура RNN, розроблена для моделювання довгострокових залежностей. Ключова ідея — використання "клітинного стану" (cell state), який може зберігати інформацію протягом багатьох кроків.

Компоненти LSTM:

1. **Забуваючий шлюз (Forget gate)**:
    - Вирішує, яку інформацію видалити з клітинного стану
    - $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$
2. **Вхідний шлюз (Input gate)**:
    - Вирішує, яку нову інформацію зберегти
    - $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
    - $\tilde{C}*t* = *tanh(W_C \cdot [h*{t-1}, x_t] + b_C)$
3. **Оновлення клітинного стану**:
    - Комбінує старий стан з новою інформацією

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

4. **Вихідний шлюз (Output gate)**:
   - Вирішує, яку інформацію виводити
   - $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$
   - $h_t = o_t * \tanh(C_t)$

GRU (Gated Recurrent Unit)

GRU — це спрощена версія LSTM, яка об'єднує забуваючий та вхідний шлюзи в один "шлюз оновлення" (update gate).

Компоненти GRU:

1. **Шлюз оновлення (Update gate)**:
   - Вирішує, скільки попереднього стану зберегти
   - $z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$

2. **Шлюз скидання (Reset gate)**:
   - Вирішує, скільки попереднього стану ігнорувати
   - $r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$

3. **Кандидат на новий стан**:
   - $\tilde{h}*t = *tanh(W \cdot [r_t * h*{t-1}, x_t] + b)$

4. **Оновлення стану**:
   - $h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$

Порівняння LSTM та GRU:

- **LSTM**: більш потужна, але складніша і повільніша
- **GRU**: простіша, швидша, менше параметрів
- На практиці обидві архітектури показують хороші результати, вибір залежить від конкретної задачі

Двонаправлені RNN

Стандартні RNN обробляють послідовність зліва направо, що обмежує їх здатність використовувати майбутній контекст. Двонаправлені RNN (Bidirectional RNN) вирішують цю проблему.

Принцип роботи:

- Дві окремі RNN обробляють послідовність у різних напрямках
- Пряма RNN обробляє послідовність зліва направо
- Зворотна RNN обробляє послідовність справа наліво
- Виходи обох RNN об'єднуються для отримання фінального результату

Математичне представлення:

$$ \overrightarrow{a}^{\langle t \rangle} = g(W_{a\overrightarrow{a}} \overrightarrow{a}^{\langle t-1 \rangle} + W_{ax} x^{\langle t \rangle} + b_{\overrightarrow{a}}) $$

$$ \overleftarrow{a}^{\langle t \rangle} = g(W_{a\overleftarrow{a}} \overleftarrow{a}^{\langle t+1 \rangle} + W_{ax} x^{\langle t \rangle} + b_{\overleftarrow{a}}) $$

$$ y^{\langle t \rangle} = g'(W_{y\overrightarrow{a}} \overrightarrow{a}^{\langle t \rangle} + W_{y\overleftarrow{a}} \overleftarrow{a}^{\langle t \rangle} + b_y) $$

Переваги:

- Використання як попереднього, так і майбутнього контексту
- Покращена точність для багатьох задач
- Особливо корисні для задач, де важливий повний контекст (розпізнавання іменованих сутностей, розуміння мови)

Обмеження:

- Потрібна вся послідовність перед початком обробки
- Не підходять для генерації тексту в реальному часі
- Вимагають більше обчислювальних ресурсів

Глибокі RNN

Глибокі RNN — це архітектури, які додають глибину до рекурентних мереж, стекуючи кілька рекурентних шарів.

Архітектура:

- Кілька рекурентних шарів, розташованих один над одним
- Вихід кожного шару є входом для наступного шару
- Кожен шар може мати свою власну динаміку

Математичне представлення:

$$ a^{[l]}{\langle t \rangle} = g(W_{aa}^{[l]} a^{[l]}{\langle t-1 \rangle}} + W_{ax}^{[l]} a^{[l-1]}{\langle t \rangle}} + b_a^{[l]}) $$

де l — номер шару.

Переваги:

- Здатність вивчати більш абстрактні представлення
- Покращена продуктивність для складних задач
- Ієрархічне представлення ознак

Недоліки:

- Більше параметрів для навчання
- Підвищені обчислювальні вимоги
- Складніше навчання (проблема зникаючого градієнта посилюється)

Моделі послідовність-до-послідовності (Sequence-to-Sequence)

Моделі послідовність-до-послідовності (Seq2Seq) — це архітектури, призначені для перетворення однієї послідовності в іншу, потенційно різної довжини.

Архітектура:

- **Енкодер**: обробляє вхідну послідовність і створює її представлення
- **Декодер**: генерує вихідну послідовність на основі представлення від енкодера

Процес роботи:

1. Енкодер обробляє всю вхідну послідовність
2. Останній прихований стан енкодера передається декодеру як початковий стан
3. Декодер генерує вихідну послідовність крок за кроком
4. На кожному кроці декодер використовує свій попередній вихід як вхід

Застосування:

- Машинний переклад
- Узагальнення тексту
- Відповіді на запитання
- Генерація діалогів

Проблеми:

- Інформаційне "вузьке місце" — вся інформація про вхідну послідовність має бути стиснута в один вектор
- Складність моделювання довгих послідовностей
- Втрата інформації про початок послідовності

Механізм уваги (Attention)

Механізм уваги (Attention) був розроблений для вирішення проблеми "вузького місця" в моделях Seq2Seq, дозволяючи декодеру фокусуватися на різних частинах вхідної послідовності.

Принцип роботи:

- Замість передачі лише останнього прихованого стану енкодера, зберігаються всі приховані стани
- На кожному кроці декодер обчислює ваги уваги для кожного прихованого стану енкодера

- Ці ваги визначають, наскільки декодер повинен "звертати увагу" на кожну позицію вхідної послідовності
- Зважена сума прихованих станів енкодера використовується як контекстний вектор

Математичне представлення:

1. Обчислення оцінок уваги: $$ e_{t,t'} = f(s_{t-1}, h_{t'}) $$
2. Нормалізація оцінок за допомогою softmax: $$ \alpha_{t,t'} = \frac{\exp(e_{t,t'})}{\sum_{j=1}^{T_x} \exp(e_{t,j})} $$
3. Обчислення контекстного вектора: $$ c_t = \sum_{t'=1}^{T_x} \alpha_{t,t'} h_{t'} $$
4. Оновлення прихованого стану декодера: $$ s_t = f(s_{t-1}, y_{t-1}, c_t) $$
5. Генерація виходу: $$ y_t = g(s_t, c_t) $$

Переваги механізму уваги:

- Дозволяє моделі фокусуватися на релевантних частинах вхідної послідовності
- Вирішує проблему "вузького місця" в Seq2Seq моделях
- Покращує якість перекладу та інших задач обробки послідовностей
- Забезпечує інтерпретованість — можна візуалізувати, на які частини вхідної послідовності модель звертає увагу

Типи механізмів уваги:

- **Адитивна увага**: використовує невелику нейронну мережу для обчислення оцінок уваги
- **Мультиплікативна увага**: використовує скалярний добуток для обчислення оцінок
- **Самоувага (Self-attention)**: обчислює увагу між різними позиціями однієї послідовності

Архітектура Трансформерів

Трансформери — це архітектура, яка повністю базується на механізмі уваги, без використання рекурентних або згорткових шарів.

Ключові компоненти:

1. **Самоувага з багатьма головками (Multi-head self-attention)**:
   - Дозволяє моделі фокусуватися на різних аспектах інформації
   - Паралельне обчислення кількох "голів" уваги
   - Об'єднання результатів для отримання збагаченого представлення
2. **Позиційне кодування (Positional encoding)**:
   - Оскільки в Трансформері немає рекурентності, потрібно явно кодувати позицію
   - Використовуються синусоїдальні функції різних частот

3. **Залишкові з'єднання (Residual connections) та нормалізація шарів**:
    - Полегшують навчання глибоких мереж
    - Стабілізують процес навчання
4. **Повнозв'язні feed-forward мережі**:
    - Обробляють представлення після механізму уваги
    - Застосовуються незалежно до кожної позиції

Архітектура Трансформера:

- **Енкодер**: складається з N ідентичних шарів
    - Кожен шар має підшар самоуваги та підшар feed-forward мережі
    - Кожен підшар має залишкове з'єднання та нормалізацію
- **Декодер**: також складається з N ідентичних шарів
    - Кожен шар має підшар самоуваги, підшар уваги до виходу енкодера та підшар feed-forward мережі
    - Маскована самоувага в декодері запобігає доступу до майбутніх позицій

Переваги Трансформерів:

- **Паралелізація**: всі позиції обробляються одночасно, що значно прискорює навчання
- **Моделювання довгострокових залежностей**: пряма увага між будь-якими позиціями
- **Масштабованість**: можливість створення дуже великих моделей
- **Висока якість**: встановили нові стандарти в багатьох задачах NLP

Обмеження:

- **Квадратична складність**: обчислювальна складність самоуваги зростає квадратично з довжиною послідовності
- **Високі вимоги до пам'яті**: необхідність зберігати матрицю уваги розміром n×n
- **Відсутність індуктивного зміщення**: модель повинна вивчити залежності з нуля

Застосування моделей послідовностей

Обробка природної мови (NLP):

- **Машинний переклад**: переклад тексту з однієї мови на іншу
- **Узагальнення тексту**: створення коротких резюме довгих текстів
- **Генерація тексту**: створення нових текстів заданого стилю або тематики
- **Відповіді на запитання**: пошук відповідей на запитання в тексті
- **Аналіз настроїв**: визначення емоційного забарвлення тексту
- **Розпізнавання іменованих сутностей**: виявлення імен, організацій, дат у тексті

Обробка аудіо:

- **Розпізнавання мовлення**: перетворення аудіо в текст
- **Виявлення ключових слів**: розпізнавання специфічних слів у аудіопотоці
- **Генерація музики**: створення нових музичних композицій
- **Класифікація аудіо**: визначення типу звуку або музичного жанру

Біоінформатика:

- **Аналіз послідовностей ДНК та білків**: передбачення функцій та структури
- **Прогнозування взаємодій білків**: моделювання взаємодій між молекулами

Часові ряди:

- **Прогнозування фінансових ринків**: передбачення цін акцій, валют
- **Аналіз даних датчиків**: виявлення аномалій, прогнозування відмов обладнання
- **Прогнозування погоди**: моделювання метеорологічних умов

Розпізнавання мовлення

Розпізнавання мовлення — це процес перетворення аудіосигналу в текст. Сучасні системи розпізнавання мовлення базуються на глибоких нейронних мережах.

Традиційний підхід:

- Використання фонем (базових одиниць звуку)
- Ручне проектування акустичних моделей
- Окремі компоненти для акустичного моделювання та мовного моделювання

Сучасний підхід на основі глибокого навчання:

- **End-to-end навчання**: пряме перетворення аудіо в текст
- **Використання RNN/LSTM/GRU**: моделювання часових залежностей
- **Механізм уваги**: фокусування на важливих частинах аудіо
- **CTC (Connectionist Temporal Classification)**: спеціальна функція втрат для вирівнювання послідовностей різної довжини

Архітектура CTC для розпізнавання мовлення:

1. Аудіо перетворюється в спектрограму або інші акустичні ознаки
2. Ознаки подаються на двонаправлену RNN/LSTM
3. Вихід мережі проходить через повнозв'язний шар з softmax
4. CTC функція втрат дозволяє моделі вивчити вирівнювання між аудіо та текстом

Переваги CTC:

- Не потребує точного вирівнювання між аудіо та текстом
- Дозволяє моделі вивчити вирівнювання самостійно
- Ефективно обробляє повторення та паузи

Виявлення ключових слів (Trigger Word Detection)

Виявлення ключових слів — це задача розпізнавання специфічних слів або фраз в аудіопотоці, наприклад, "Привіт, Сірі" або "Окей, Google".

Підходи до виявлення ключових слів:

1. **На основі RNN/LSTM**:
    - Аудіо перетворюється в спектрограму
    - Спектрограма подається на RNN/LSTM
    - Мережа навчається передбачати 1, коли ключове слово закінчується, і 0 в інших випадках
2. **Синтез даних для навчання**:
    - Створення великого набору даних шляхом накладання ключових слів на фонові аудіо
    - Балансування позитивних та негативних прикладів
3. **Оптимізація для низького споживання енергії**:
    - Використання легких архітектур
    - Квантизація моделей
    - Спеціалізоване апаратне забезпечення

Ембедінги слів (Word Embeddings)

Ембедінги слів — це щільні векторні представлення слів, які захоплюють семантичні та синтаксичні властивості слів.

Традиційне представлення слів:

- **One-hot кодування**: кожне слово представлене вектором з 1 на позиції слова в словнику і 0 на всіх інших позиціях
- **Проблеми one-hot кодування**:
    - Розріджені вектори великої розмірності
    - Не захоплюють семантичні відношення між словами
    - Не узагальнюються на нові слова

Ембедінги слів:

- **Щільні вектори**: зазвичай 50-300 вимірів
- **Семантично значущі**: слова з подібним значенням мають подібні вектори
- **Підтримують алгебраїчні операції**: "король" - "чоловік" + "жінка" ≈ "королева"

Методи навчання ембедінгів:

1. **Word2Vec**:
    - **CBOW (Continuous Bag of Words)**: передбачення слова за його контекстом
    - **Skip-gram**: передбачення контексту за словом
    - Використовує неглибоку нейронну мережу
2. **GloVe (Global Vectors)**:
    - Базується на матриці спільної зустрічальності слів
    - Комбінує локальний та глобальний контекст
3. **FastText**:
    - Розширення Word2Vec
    - Представляє слова як суми n-грам символів
    - Краще обробляє рідкісні слова та морфологію

Застосування ембедінгів:

- **Передача знань**: попередньо навчені ембедінги можуть покращити продуктивність моделей з обмеженими даними
- **Аналіз настроїв**: захоплення семантичних нюансів
- **Машинний переклад**: представлення слів різних мов у спільному просторі
- **Пошук документів**: семантичний пошук замість точного співпадіння

Дебіасинг ембедінгів:

- Ембедінги можуть захоплювати та підсилювати упередження з навчальних даних
- Методи дебіасингу:
    - Ідентифікація напрямків упередження (наприклад, гендерний вектор)
    - Нейтралізація упередження в певних вимірах
    - Збалансоване навчання на різноманітних даних

Порівняння RNN та Трансформерів

Переваги RNN:

- **Ефективність для коротких послідовностей**: менше параметрів
- **Явне моделювання послідовності**: природне представлення часових залежностей
- **Можливість обробки послідовностей змінної довжини**: без необхідності фіксованого розміру входу
- **Менші вимоги до пам'яті**: лінійна складність по довжині послідовності

Недоліки RNN:

- **Проблема зникаючого/вибухаючого градієнта**: складність моделювання довгострокових залежностей
- **Послідовна обробка**: неможливість паралелізації по часових кроках
- **Обмежений контекст**: практично обмежена довжина залежностей, які можна моделювати

Переваги Трансформерів:

- **Паралельна обробка**: значне прискорення навчання та інференсу
- **Необмежений контекст**: пряма увага між будь-якими позиціями
- **Масштабованість**: можливість створення дуже великих моделей
- **Стабільне навчання**: менше проблем з градієнтами

Недоліки Трансформерів:

- **Квадратична складність**: обмеження на довжину послідовності
- **Високі вимоги до пам'яті та обчислень**: потребують потужного обладнання
- **Відсутність індуктивного зміщення**: потребують більше даних для навчання
- **Фіксована довжина входу**: необхідність обрізання або розбиття довгих послідовностей

Тенденції розвитку:

- **Ефективні Трансформери**: архітектури, що зменшують квадратичну складність (Linformer, Reformer, Performer)
- **Гібридні архітектури**: поєднання переваг RNN та Трансформерів
- **Спеціалізовані архітектури**: оптимізовані для конкретних задач та доменів

Оцінка моделей послідовностей

Метрики для машинного перекладу:

- **BLEU (Bilingual Evaluation Understudy)**: вимірює n-грамну схожість між перекладом та еталоном
- **METEOR**: враховує синоніми та морфологічні варіанти
- **ROUGE**: використовується для оцінки узагальнення тексту
- **TER (Translation Edit Rate)**: мінімальна кількість редагувань для перетворення перекладу в еталон

Метрики для розпізнавання мовлення:

- **WER (Word Error Rate)**: відсоток неправильно розпізнаних слів
- **CER (Character Error Rate)**: відсоток неправильно розпізнаних символів
- **BLEU для ASR**: адаптація BLEU для оцінки транскрипцій

Метрики для генерації тексту:

- **Перплексія (Perplexity)**: міра невизначеності моделі
- **Людська оцінка**: суб'єктивна оцінка якості, зв'язності та релевантності

- **Автоматичні метрики зв'язності**: оцінка граматичної та логічної зв'язності

Практичні аспекти роботи з моделями послідовностей

Попередня обробка даних:

- **Токенізація**: розбиття тексту на слова, підслова або символи
- **Нормалізація**: приведення тексту до стандартного формату
- **Доповнення (Padding)**: вирівнювання послідовностей до однакової довжини
- **Маскування (Masking)**: ігнорування доповнених елементів при обчисленні втрат

Навчання моделей:

- **Регуляризація**: dropout, L2-регуляризація, обрізання градієнтів
- **Планування швидкості навчання**: поступове зменшення швидкості навчання
- **Раннє зупинення**: запобігання перенавчанню
- **Пакетна нормалізація**: стабілізація навчання

Оптимізація інференсу:

- **Променевий пошук (Beam search)**: пошук найбільш ймовірних послідовностей
- **Квантизація моделей**: зменшення точності параметрів для прискорення
- **Дистиляція знань**: передача знань від великої моделі до меншої
- **Кешування**: повторне використання обчислень для спільних підпослідовностей

Висновки

Моделі обробки послідовностей є ключовим компонентом сучасного глибокого навчання, що дозволяють вирішувати широкий спектр задач, від обробки природної мови до аналізу часових рядів та біоінформатики.

Ключові моменти:

1. **Еволюція архітектур**: від простих RNN до складних Трансформерів
2. **Вирішення проблеми зникаючого градієнта**: LSTM, GRU, механізм уваги
3. **Механізм уваги**: революційний підхід, що дозволяє моделям фокусуватися на релевантних частинах входу
4. **Трансформери**: архітектура, що домінує в сучасних моделях NLP
5. **Ембедінги**: потужні представлення слів, що захоплюють семантичні відношення

Майбутні напрямки:

- **Ефективні архітектури**: зменшення обчислювальної складності та вимог до пам'яті
- **Мультимодальні моделі**: інтеграція тексту, зображень, аудіо та відео
- **Інтерпретовність**: розуміння внутрішніх механізмів моделей

- **Навчання з меншою кількістю даних**: ефективне використання обмежених ресурсів
- **Етичні аспекти**: зменшення упереджень та забезпечення справедливості моделей

Моделі обробки послідовностей продовжують розвиватися, відкриваючи нові можливості для розуміння та генерації складних послідовних даних у різних доменах.

Рекомендована література

1. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
2. Jurafsky, D., & Martin, J. H. (2020). Speech and Language Processing. 3rd Edition draft.
3. Vaswani, A., et al. (2017). Attention Is All You Need. Advances in Neural Information Processing Systems.