▣ **강의명:** CSCI E-89B: 자연어 처리 입문

▣ **주차:** Lecture 12

▣ **교수명:** Dmitry Kurochkin

▣ **목적:** Lecture 12의 핵심 개념 학습

Contents

Lecture 12: Attention Mechanism & Transformers

자연어 처리의 혁명: 어텐션 메커니즘과 트랜스포머 아키텍처 완전 정복

개요 (Overview)

이번 강의는 현대 자연어 처리(NLP)의 가장 중요한 분기점인 **Attention Mechanism(어텐션 메커니즘)**과 이를 기반으로 한 **Transformer(트랜스포머)** 모델을 다룹니다.

기존 순환 신경망(RNN/LSTM)이 가진 한계점(병목 현상, 기울기 소실, 병렬화 불가)을 극복하기 위해 어텐션이 어떻게 등장했는지 살펴보고, 나아가 "순환(Recurrence)을 완전히 제거한" 트랜스포머 아키텍처의 작동 원리(Self-Attention, Multi-Head Attention, Positional Encoding)를 학습합니다. 마지막으로 이 구조가 어떻게 BERT와 GPT로 발전했는지 이해합니다.

□ 핵심 요약

핵심 목표

- RNN 기반 Seq2Seq 모델의 한계점(정보 병목, 장기 의존성 문제) 이해
- 어텐션(Attention)의 기본 원리: "중요한 부분에 집중한다"는 개념
- 트랜스포머(Transformer)의 구조: Encoder-Decoder, Self-Attention (Q, K, V)
- BERT(인코더 기반)와 GPT(디코더 기반)의 차이점

1 필수 용어 정리

본격적인 학습에 앞서, 낯설 수 있는 핵심 용어를 미리 정리합니다.

Table 1: *Lecture 12* 핵심 용어 사전

용어 (Term)	한국어	쉬운 설명
Seq2Seq	시퀀스 투 시퀀스	입력 문장을 받아 출력 문장을 만드는 구조 (예: 번역). 인코더와 디코더로 구성됨.
Bottleneck	병목 현상	긴 문장의 모든 정보를 하나의 고정된 크기 벡터에 억지로 구겨 넣을 때 발생하는 정보 손실.
Context Vector	문맥 벡터	입력 문장의 정보를 요약한 벡터. 어텐션에서는 매 시점마다 동적으로 변함.
Attention	어텐션(주목)	출력 단어를 만들 때, 입력 문장의 어느 단어를 더 '주의 깊게' 볼지 결정하는 기술.
Self-Attention	셀프 어텐션	문장 내의 단어들이 서로 어떤 관계가 있는지 파악하는 것 (예: '그것'이 가리키는 단어 찾기).
Query (Q)	쿼리	"내가 지금 찾고자 하는 정보는?" (질문자 역할)
Key (K)	키	"나는 누구인가?" (정보의 식별자 역할)
Value (V)	밸류	"내가 가진 실제 내용은 무엇인가?" (정보의 내용물 역할)
Transformer	트랜스포머	RNN 없이 오직 어텐션만으로 구성된 딥러닝 모델 아키텍처.

2 RNN의 한계와 어텐션의 등장

2.1 기존 RNN/LSTM의 문제점

과거 번역 모델(Encoder-Decoder)은 RNN을 사용했습니다. 입력 문장을 인코더가 읽어서 하나의 **고정된 크기의 벡터(Context Vector)**로 압축하고, 디코더가 이를 풀어서 번역문을 만들었습니다.

- **정보 병목(Information Bottleneck):** 100단어짜리 긴 문장을 겨우 128차원 벡터 하나에 압축해야 합니다. 정보 손실이 발생할 수밖에 없습니다.
- **장기 의존성(Long-Term Dependency):** 문장이 길어지면 앞부분의 내용이 뒷부분까지 전달되지 못하고 희석됩니다(Vanishing Gradient).
- **순차적 처리(Sequential Processing):** 단어를 하나씩 순서대로 처리해야 하므로 병렬 계산(GPU 활용)이 어렵고 속도가 느립니다.

2.2 어텐션(Attention)의 아이디어

직관적 이해: 통역사의 비유 RNN이 "문장 전체를 외운 뒤 한 번에 번역하는 통역사"라면,
어텐션은 **"번역할 때마다 원문을 다시 힐끔힐끔 쳐다보는 통역사"**입니다.

어텐션 메커니즘은 디코더가 단어를 출력할 때마다 인코더의 **모든 입력 단어**를 다시 참고합니다. 단, 그냥 보는 것이 아니라 **"지금 번역할 단어와 관련된 부분"**에 가중치(Attention Weight)를 두어 봅니다.

2.3 수식 없는 어텐션 작동 원리

1. **스코어 계산(Alignment Score):** 현재 디코더의 상태와 인코더의 각 단어가 얼마나 관련 있는지 계산합니다. 2. **확률 변환(Softmax):** 스코어를 0 1 사이의 확률값(합이 1)으로 바꿉니다. 이것이 '어텐션 가중치'입니다.

- 예: "Zone(프랑스어)"을 번역할 때, "Area(영어)"에 0.7, "Economic"에 0.2의 가중치를 둠.

3. **가중합(Weighted Sum):** 인코더의 정보들에 가중치를 곱해 더합니다. 이것이 새로운 동적 문맥 벡터(Dynamic Context Vector)가 됩니다.

[시각화: 어텐션 맵]
프랑스어 "Zone"이 출력될 때 → 영어 "Area" 부분의 픽셀이 밝게 빛남.
즉, 모델이 번역 시점에 'Area'라는 단어에 집중(Attend)하고 있음을 시각적으로 확인 가능.

Figure 1: 어텐션 가중치를 시각화하면 단어 간의 연관성을 알 수 있습니다.

3 트랜스포머 (Transformer)

2017년 구글은 *"Attention Is All You Need"*라는 논문을 통해 **RNN을 완전히 제거한** 모델인 트랜스포머를 제안합니다.

3.1 왜 RNN을 버렸는가?

RNN은 순서대로 계산해야 하므로 느립니다. 트랜스포머는 문장 전체를 한 번에 행렬로 입력받아 병렬 처리가 가능합니다. 이를 통해 학습 속도를 비약적으로 높이고, 더 많은 데이터를 학습할 수 있게 되었습니다(GPT, BERT의 탄생 배경).

3.2 핵심 1: 포지셔널 인코딩 (Positional Encoding)

RNN이 없으면 단어의 **순서 정보**가 사라집니다. "I love you"와 "You love I"를 구별할 수 없게 됩니다. 따라서, 단어 벡터에 **위치 정보(Positional Encoding)**를 더해줍니다.

- 사인(sin)과 코사인(cos) 함수를 이용해 각 위치마다 고유한 패턴의 값을 더해줍니다.
- 이를 통해 모델은 단어의 상대적/절대적 위치를 파악할 수 있습니다.

3.3 핵심 2: 셀프 어텐션 (Self-Attention)과 Q, K, V

트랜스포머의 심장입니다. 문장 내의 단어들이 서로 어떤 관계인지 파악합니다. 이를 위해 각 단어를 세 가지 역할로 나눕니다.

Q, K, V 비유: 파일 검색 시스템

- **Query (Q):** 검색창에 입력한 검색어 ("Sky에 대해 알고 싶어")
- **Key (K):** 파일의 제목/태그 ("이 파일은 Cloud에 관한 것임")
- **Value (V):** 파일의 실제 내용 ("구름은 흰색이고 하늘에 떠 있다...")

작동 과정: 1. 나의 Q와 상대방들의 K를 내적(Dot Product)하여 유사도를 구합니다. (Sky와 Cloud는 관련성이 높으므로 점수가 높음) 2. 점수를 $\sqrt{d_k}$로 나누고(스케일링), Softmax를 취해 확률로 만듭니다. 3. 이 확률을 상대방들의 V에 곱해서 더합니다. 4. 결과: "Sky"라는 단어 벡터는 "Cloud", "Blue" 등의 문맥 정보를 흡수하여 더 풍부한 의미를 갖게 됩니다.

Scaled Dot-Product Attention 공식

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

- $\sqrt{d_k}$로 나누는 이유: 벡터 차원(d_k)이 커지면 내적 값이 너무 커져서, Softmax의 기울기(Gradient)가 소실되는 것을 방지하기 위함(학습 안정화).

3.4 핵심 3: 멀티 헤드 어텐션 (Multi-Head Attention)

한 번만 어텐션을 수행하는 것이 아니라, 여러 개의 헤드(Head)로 나누어 병렬로 수행합니다.

- **비유:** 같은 문장을 읽을 때, 한 명은 문법을 보고, 한 명은 의미를 보고, 한 명은 대명사 지칭을

봅니다. 나중에 이들의 통찰을 합칩니다.

- 이를 통해 모델은 다양한 관점(Representation Subspaces)에서 문장을 이해할 수 있습니다.

3.5 핵심 4: 마스킹 (Masking)

트랜스포머의 디코더(Decoder) 학습 시 사용됩니다.

- **문제:** 디코더는 정답 문장을 생성해야 하는데, 학습 시 정답 전체를 한 번에 입력받습니다. 모델이 뒤에 올 정답 단어를 미리 "컨닝(Cheating)"하면 안 됩니다.
- **해결:** 현재 위치보다 뒤에 있는 단어들의 점수를 $-\infty$(음의 무한대)로 만들어 Softmax 결과가 0이 되게 합니다. 이를 **Look-ahead Mask**라고 합니다.

4 실습 코드 분석 (Keras)

다음은 Keras를 이용한 어텐션 레이어 구현의 핵심 로직입니다.

```
class AttentionLayer(Layer):
    def call(self, inputs):
        # 1. 스코어계산 (tanh 사용)
        # 와inputs 가중치를 W 내적하고를 bias 더한뒤 tanh 통과
        v = tf.tanh(tf.tensordot(inputs, self.W, axes=1) + self.b)

        # 2. Alignment Score 계산
        # 벡터와 u 내적하여스칼라점수 (vu) 생성
        vu = tf.tensordot(v, self.u, axes=1, name='vu')

        # 3. 어텐션가중치 (alpha) 계산 (Softmax)
        alphas = tf.nn.softmax(vu, axis=1)

        # 4. 문맥벡터 (Context Vector) 생성
        # 입력(inputs)에 가중치(alphas)를 곱해서합침 (reduce_sum)
        output = tf.reduce_sum(inputs * tf.expand_dims(alphas, -1), axis=1)

        return output
```

Listing 1: 사용자 정의 Attention Layer 핵심 로직

- 위 코드는 RNN 기반 어텐션 구조입니다. 트랜스포머의 QK^T 방식과는 약간 다르지만(여기선 $tanh$ 사용), "가중치를 계산해서 입력의 가중합을 구한다"는 핵심 원리는 같습니다.

5 BERT와 GPT: 트랜스포머의 자식들

트랜스포머 구조를 반으로 쪼개서 각각 발전시킨 것이 현재의 최신 모델들입니다.

- **BERT (Bidirectional Encoder Representations from Transformers):** 문맥을 양쪽에서 파악하므로 "이해(Understanding)" 능력이 뛰어납니다.
- **GPT (Generative Pre-trained Transformer):** 다음 단어를 예측하는 방식으로 학습하므로 "생성(Generation)" 능력이 뛰어납니다.

Table 2: *BERT*와 *GPT*의 비교

구분	BERT (Encoder 기반)	GPT (Decoder 기반)
기반 구조	트랜스포머의 **인코더(Encoder)**	트랜스포머의 **디코더(Decoder)**
방향성	양방향 (Bidirectional)	단방향 (Unidirectional, 왼쪽→오른쪽)
주특기	문장의 의미 파악, 빈칸 채우기, 분류	문장 생성, 다음 단어 예측
비유	문장 전체를 보고 해석하는 독해 전문가	앞 단어만 보고 뒷말 잇는 작가
활용 예	감성 분석, 질의응답(QA), 개체명 인식	챗봇, 소설 쓰기, 코드 생성

6 학습 점검 체크리스트 & FAQ

6.1 체크리스트

[ ] RNN의 장기 의존성 문제와 병목 현상이 무엇인지 설명할 수 있는가?

[ ] 어텐션이 입력 데이터의 가중합(Weighted Sum)을 구하는 과정임을 이해했는가?

[ ] 트랜스포머가 RNN을 사용하지 않고도 순서 정보를 아는 방법(Positional Encoding)을 아는가?

[ ] Self-Attention에서 Q, K, V가 각각 어떤 역할을 하는지 비유를 들어 설명할 수 있는가?

[ ] 트랜스포머의 인코더 부분(BERT)과 디코더 부분(GPT)의 차이를 구분할 수 있는가?

6.2 자주 묻는 질문 (FAQ)

Q1. 어텐션(Attention)과 셀프 어텐션(Self-Attention)은 다른 건가요?
A. 원리는 같지만 **대상**이 다릅니다. 일반 어텐션(Seq2Seq)은 디코더가 인코더를 보는 것이고(서로 다른 문장 간 관계), 셀프 어텐션은 문장 내에서 자기 자신 안의 단어들끼리의 관계를 보는 것입니다(예: 'I'와 'am'의 관계).

Q2. 왜 Q, K, V를 따로 만드나요? 그냥 단어 벡터 그대로 쓰면 안 되나요?
A. 가능은 하지만, 유연성(Flexibility)이 떨어집니다. 같은 단어라도 "질문할 때의 나(Query)", "비교 대상으로서의 나(Key)", "정보 제공자로서의 나(Value)"의 역할에 따라 다르게 표현될 수 있도록 별도의 가중치 행렬(W^Q, W^K, W^V)을 학습시키는 것이 성능이 훨씬 좋습니다.

Q3. 마스킹(Masking)은 왜 디코더에만 있나요?
A. 인코더는 이미 주어진 문장을 분석하는 것이라 미래 단어를 봐도 상관없습니다(오히려 다 봐야 문맥을 압니다). 하지만 디코더는 문장을 생성하는 과정이므로, 아직 생성하지 않은 미래의 단어를 미리 보고 예측하면 학습이 제대로 되지 않기 때문입니다.

Q4. 강의 초반 퀴즈 내용 중 CRF가 HMM보다 좋은 점은?
A. HMM은 바로 이전 상태(State)에만 의존한다고 가정하지만, CRF(Conditional Random Field)는 문맥 전체(과거와 미래)의 특성을 동시에 고려할 수 있어 더 복잡한 의존성을 모델링할 수 있습니다.

빠르게 훑어보기 (1분 요약)

Summary

- **문제:** RNN은 느리고, 긴 문장을 까먹음.
- **해결 1 (Attention):** 문장을 요약하지 말고, 필요할 때마다 원문을 다시 보자.
- **해결 2 (Transformer):** RNN을 버리고 어텐션만 쓰자. 병렬 처리로 속도 UP.
- **Self-Attention:** $Attention(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$
- **BERT:** 인코더 사용, 문맥 이해(양방향).
- **GPT:** 디코더 사용, 문장 생성(단방향).