

생성형 AI

Day 15

딥러닝 III



목차

1. NLP?
2. 텍스트 전처리
3. Word Embedding
4. 딥러닝과 자연어 처리
5. 실습과제



NLP(Natural Language Processing)

NLP(Natural Language Processing)?

- 자연어 처리는 컴퓨터가 인간의 언어(자연어)를 이해하고 생성할 수 있도록 하는 기술
- 텍스트나 음성 데이터를 처리하여 유의미한 정보를 추출하거나 인간과의 상호작용에 사용
- 기계 번역, 감성 분석, 챗봇, 음성 인식 등 다양한 응용 분야에서 사용

자연어 처리의 주요 응용 분야

기계 번역

- 두 개 이상의 언어 사이의 번역을 수행하는 기술, 영어 문장을 한국어 문장으로 번역

감성 분석

- 텍스트 데이터에서 감정(긍정, 부정, 중립)을 추출하는 기술, 영화 리뷰가 긍정적인지 부정적인지를 분석

챗봇

- 사용자의 질문에 자동으로 응답하는 대화형 AI 시스템

음성 인식

- 음성 데이터를 텍스트로 변환하는 기술, 음성 명령을 인식하여 수행

정보 검색

- 대규모 데이터베이스에서 사용자가 원하는 정보를 검색, 검색 엔진

NLP(Natural Language Processing)

코퍼스(Corpus)

- 코퍼스는 자연어 처리 연구와 모델 학습을 위해 수집된 대규모 텍스트 데이터셋
- 코퍼스는 언어의 실제 사용을 반영하여 다양한 언어적 현상을 분석 가능

코퍼스의 형태

- 일반 텍스트 코퍼스: Wikipedia, 뉴스 기사, 소설 등 다양한 주제와 형식의 텍스트 데이터
- 도메인 특화 코퍼스: 특정 분야에 관련된 텍스트 데이터 (예: 의학 논문, 법률 문서)
- 주석이 포함된 코퍼스: 텍스트와 함께 레이블(태그)이 포함된 데이터 (예: 감성 레이블, 개체명 레이블)

텍스트 전처리

토큰화(Tokenization)

- 문장을 단어, 부분 단어 또는 문자 단위로 분리하는 과정, "Hello, world!" → ["Hello", ",", "world", "!"]

정규화 (Normalization)

- 텍스트의 일관성을 유지하기 위해 변환하는 과정, 대문자를 소문자로 변환, 구두점 제거 등

불용어 제거 (Stop Words Removal)

- 분석에 불필요한 일반적인 단어를 제거하는 과정, "is", "the", "and"
- 불용어는 분석에 큰 기여를 하지 않기 때문에 제거하여 모델의 성능을 향상시킬 수 있음

어간 추출(Stemming)

- 단어의 어간을 추출하여 변형된 형태를 제거하는 과정, "running" → "run"

표제어 추출

- 단어의 표제어(기본 형태)를 추출하는 과정, "better" → "good"

Word Embedding

단어 임베딩

- 단어 임베딩은 단어를 고차원 벡터로 표현하여, 단어 간의 의미적 유사성을 벡터 공간에서 반영하는 기법
- 단어 간의 관계를 효과적으로 학습하고 표현할 수 있게 함
- 단어는 단어의 의미적 정보를 갖는 고정된 크기의 실수 벡터로 변환

왜 단어 임베딩이 중요할까?

차원 축소

- 원-핫 인코딩과 같은 방법에서는 단어의 개수만큼 차원이 필요, but 단어 임베딩은 상대적으로 낮은 차원에서 단어를 표현
- 의미적 유사성
- 단어 임베딩은 단어 간의 의미적 유사성을 벡터 공간에서 나타냄, "king" - "man" + "woman" \approx "queen"

학습 효율성

- 단어 임베딩은 모델 학습의 효율성을 높이고, 다양한 NLP 작업의 성능을 향상

주요 단어 임베딩 기법

- Word2Vec, GloVe, ...

Word Embedding

Word2Vec

- Word2Vec은 자연어 처리(NLP)에서 단어의 의미를 벡터 형태로 표현하기 위해 사용되는 임베딩 기법
- 단어를 고차원 벡터로 변환하여 단어 간의 의미적 유사성을 벡터 공간에 투영함
- 특히 대규모 코퍼스에서 효과적으로 학습, CBOW(Continuous Bag of Words)와 Skip-gram 두 가지 형태를 가짐

CBOW (Continuous Bag of Words) 모델

- CBOW 모델은 주어진 문맥 단어(주변 단어)를 사용하여 중심 단어를 예측하는 방식
- 문맥 단어들이 중심 단어를 결정하는 데 중요한 정보를 제공한다는 가정에 기반

Skip-gram 모델

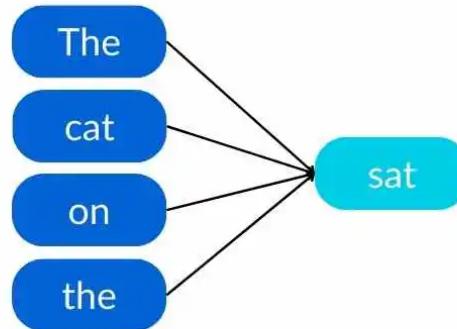
- Skip-gram 모델은 중심 단어로부터 주변 단어들을 예측하는 방식
- 중심 단어가 주어졌을 때, 해당 단어의 문맥에 있는 단어들을 예측

Word Embedding

Example Sentence: The cat sat on the mat.

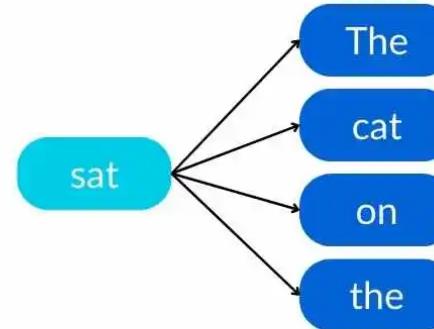
Continuous Bag-of-Words (CBOW)

Goal: Given context words,
predict the target word.



Skip-gram Model

Goal: Given a word,
predict the surrounding context words.



Word Embedding

예시 문장: “The quick brown fox jumps over the lazy dog”

- 중심 단어: "quick"
- 문맥 단어: ["The", "brown"]

CBOW

- 입력: ["The", "brown"] → 원-핫 인코딩 → 은닉층에서 평균 계산 → 소프트맥스 회귀 → 출력: “quick”

Skip-gram

- 입력: "quick" → 원-핫 인코딩 → 은닉층으로 변환 → 소프트맥스 회귀 → 출력: ["The", "brown"]

한계점

- 문맥에 따라 달라지는 단어의 의미를 파악하지 못함
- 학습 데이터에 없는 단어(Out-Of-Vocabulary)에 대해 벡터를 생성 불가

Word Embedding

단어 임베딩 활용

텍스트 분류

- 단어 임베딩을 사용하여 문서의 의미를 벡터로 변환한 후, 이를 입력으로 사용하여 텍스트 분류 모델을 학습

유사도 계산

- 두 단어 또는 문장의 임베딩 벡터 간의 코사인 유사도 등을 계산하여 의미적 유사성을 평가

기계 번역

- 단어 임베딩을 사용하여 소스 언어와 대상 언어 간의 의미적 맵핑을 학습

딥러닝과 자연어 처리 – RNN

순환신경망 (Recurrent Neural Network, RNN)?

- 순환 신경망(Recurrent Neural Network, RNN)은 순차 데이터를 처리하는 데 특화된 인공 신경망의 한 종류
- 시계열 데이터, 자연어 처리(NLP), 음성 인식 등 순차적 특성이 중요한 문제에서 널리 사용
- 이전 단계의 출력을 다음 단계의 입력으로 사용하는 순환 구조

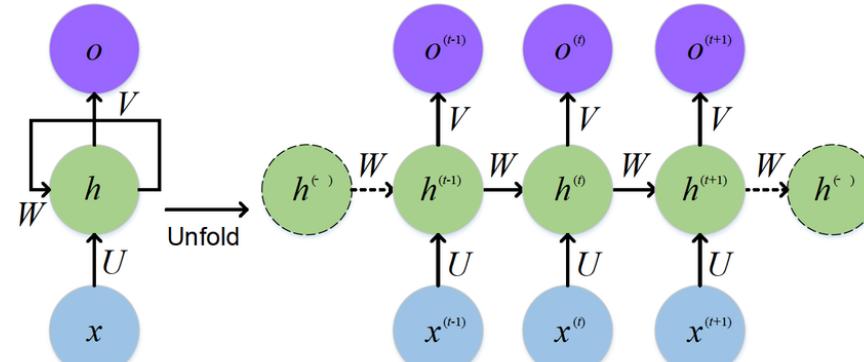
RNN의 구조

- 입력층: 입력 시퀀스의 각 요소 적용
- 은닉층: ‘이전 상태의 은닉 상태’와 ‘현재 입력’을 사용하여 새로운 은닉 상태를 계산
- 은닉 상태 계산: $h_t = \sigma(W_h h_{t-1} + W_x x_t + b_h)$
- 출력층: 은닉 상태를 사용하여 출력을 계산
- 출력 계산: $y_t = \phi(W_y h_t + b_y)$
- W_h, W_x, W_y : 가중치 행렬, b_h, b_y : 바이어스 벡터, σ : 활성화 함수, ϕ : 출력 활성화 함수

딥러닝과 자연어 처리 – RNN

RNN의 구조

- 입력층: 입력 시퀀스의 각 요소 적용
- 은닉층: ‘이전 상태의 은닉 상태’와 ‘현재 입력’을 사용하여 새로운 은닉 상태를 계산
- 은닉 상태 계산: $h_t = \sigma(W_h h_{t-1} + W_x x_t + b_h)$
- 출력층: 은닉 상태를 사용하여 출력을 계산
- 출력 계산: $y_t = \phi(W_y h_t + b_y)$
- W_h, W_x, W_y : 가중치 행렬, b_h, b_y : 바이어스 벡터, σ : 활성화 함수, ϕ : 출력 활성화 함수



딥러닝과 자연어 처리 – RNN

RNN의 학습

- 순환 신경망(Recurrent Neural Network, RNN)은 순차 데이터를 처리하는 데 특화된 인공 신경망의 한 종류
- RNN의 학습은 일반적인 신경망과 마찬가지로 역전파 알고리즘을 사용하여 가중치를 업데이트

RNN의 장점

- RNN은 시퀀스 데이터의 순차적 특성을 잘 반영한 모델
- RNN은 시간에 따른 의존성을 학습하여 이전 입력의 영향을 현재 출력에 반영 가능

RNN의 한계

- RNN이 긴 시퀀스를 학습할 때, 기울기가 소실되어 학습이 어려워지는 문제 발생 가능
- RNN은 기울기가 너무 커져서 수치적으로 불안정해지는 문제가 발생 가능
- RNN은 긴 시퀀스에서 장기 의존성을 학습하는 데 어려움을 겪을 수 있음

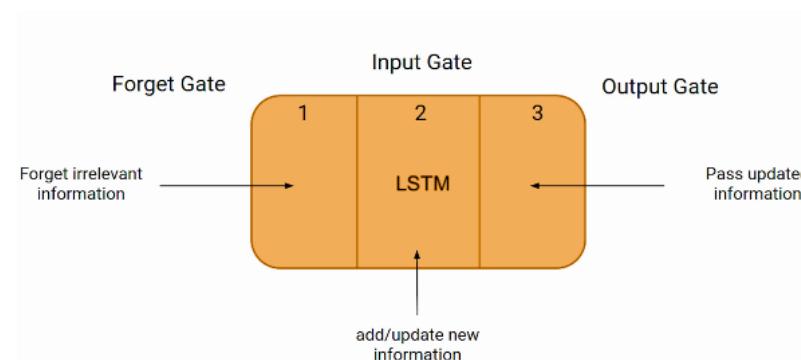
딥러닝과 자연어 처리

장단기 메모리 네트워크 (Long Short-Term Memory, LSTM)?

- LSTM은 RNN의 한계를 극복하기 위해 고안된 모델
- 장기 의존성을 효과적으로 학습
- 셀 상태(cell state)와 게이트 메커니즘을 통해 정보를 선택적으로 유지하거나 삭제

LSTM의 구조

1. 입력 게이트(Input Gate): 새로운 입력 정보를 얼마나 반영할지 결정
2. 망각 게이트(Forget Gate): 이전 셀 상태 정보를 얼마나 유지할지 결정
3. 출력 게이트(Output Gate): 현재 셀 상태 정보를 출력으로 얼마나 반영할지 결정



딥러닝과 자연어 처리

LSTM의 동작 원리

1. 망각 게이트

이전 셀 상태에서 어떤 정보를 버릴지 결정

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

f_t 는 망각 게이트의 활성화 값, σ 는 시그모이드 함수, W_f 는 가중치 행렬, b_f 바이어스 벡터

2. 입력 게이트

현재 입력 x_t 와 이전 은닉 상태 h_{t-1} 를 사용하여 새로운 후보 셀 상태 \tilde{C}_t 를 생성하고 얼마나 반영할지 결정

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

i_t 는 입력 게이트의 활성화 값, \tilde{C}_t 는 후보 셀 상태, \tanh 은 하이퍼볼릭 탄젠트 함수

3. 셀 상태 업데이트

망각 게이트와 입력 게이트를 통해 이전 셀 상태를 업데이트

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

C_t 는 현재 셀 상태

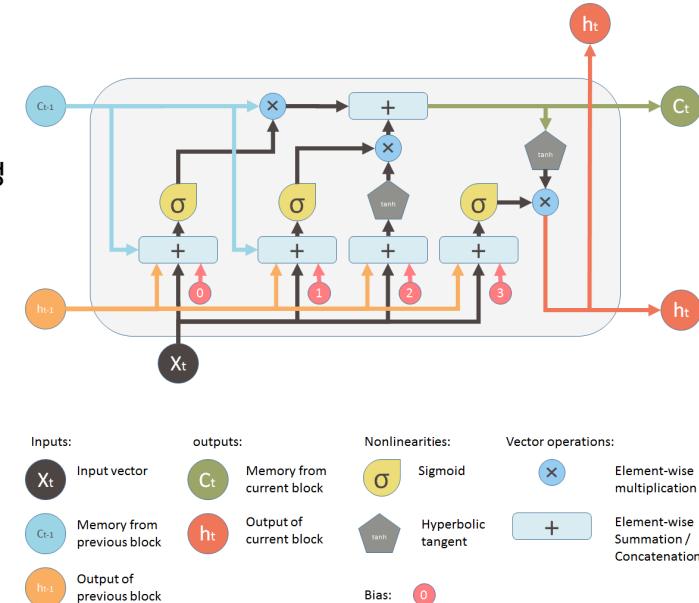
4. 출력 게이트

현재 셀 상태 C_t 를 기반으로 출력 값을 결정하고, 은닉 상태 h_t 를 업데이트

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

o_t 는 출력 게이트의 활성화값, h_t 는 현재 은닉 상태



딥러닝과 자연어 처리

LSTM의 장점

- 기울기 소실 문제 해결: 셀 상태를 통해 기울기 소실 문제를 완화하여 긴 시퀀스에서도 효과적으로 학습
- 장기 의존성 학습: 장기 의존성을 효과적으로 학습하여, 긴 시퀀스 데이터에서 중요한 정보를 유지
- 정보의 선택적 업데이트: 게이트 메커니즘을 통해 정보의 유입과 유출을 조절하여 중요한 정보를 선택적으로 유지

LSTM의 단점

- 복잡성: RNN에 비해 구조가 복잡하여 계산 비용이 높음
- 학습 시간: 더 많은 파라미터와 복잡한 구조로 인해 학습 시간이 길어짐

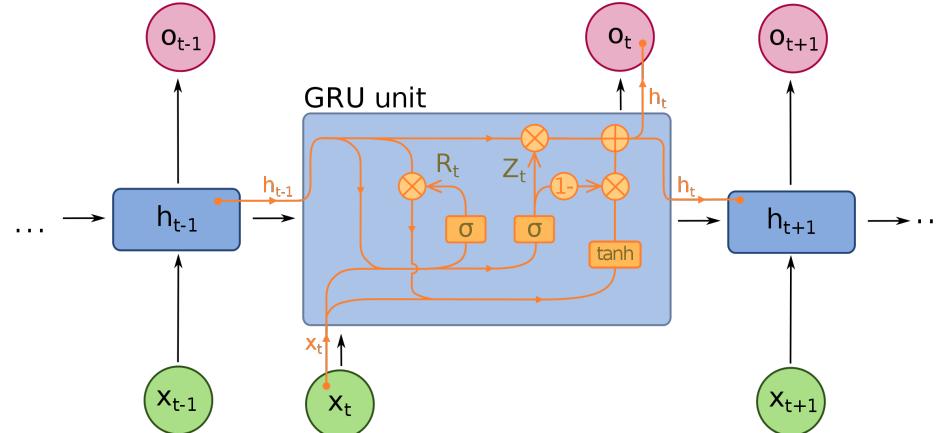
LSTM 응용

- 텍스트 생성: 문학 작품을 학습하여 새로운 문장을 생성
- 기계 번역: 입력 언어를 출력 언어로 번역

딥러닝과 자연어 처리

게이트 순환 유닛 (Gated Recurrent Unit, GRU)

- 게이트 순환 유닛은 순환 신경망(RNN)의 변형 중 하나로, 장단기 메모리 네트워크(LSTM)와 유사하게 RNN의 기울기 소실 문제를 해결하기 위해 개발
- LSTM보다 단순한 구조
- 비슷한 성능을 유지하면서도 계산 효율성을 높임
- 입력 시퀀스가 길거나 복잡한 경우에 효과적



딥러닝과 자연어 처리

게이트 순환 유닛 (Gated Recurrent Unit, GRU)

- 게이트 순환 유닛은 순환 신경망(RNN)의 변형 중 하나로, 장단기 메모리 네트워크(LSTM)와 유사하게 RNN의 기울기 소실 문제를 해결하기 위해 개발
- LSTM보다 단순한 구조
- 비슷한 성능을 유지하면서도 계산 효율성을 높임
- 입력 시퀀스가 길거나 복잡한 경우에 효과적

GRU 구조

- LSTM과 달리 셀 상태를 사용하지 않고, 은닉 상태만을 사용하여 정보를 전달
- 업데이트 게이트(Update Gate): 새로운 입력 정보를 얼마나 현재 은닉 상태에 반영할지 결정
- 리셋 게이트(Reset Gate): 이전 은닉 상태 정보를 얼마나 반영할지 결정

딥러닝과 자연어 처리

GRU 동작 원리

1. 업데이트 게이트

- 현재값과 이전 은닉 상태로 업데이트 게이트 값 계산

2. 리셋 게이트

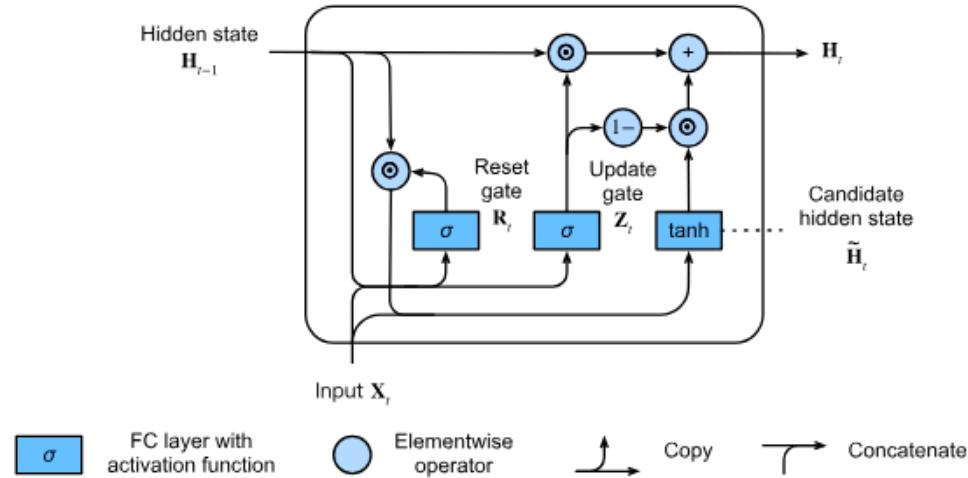
- 현재 입력, 이전 은닉 상태로 리셋 게이트 값 계산

3. 새로운 은닉 상태

- 리셋 게이트로 이전 은닉상태를 저정하고, 현재입력과 결합하여 새로운 후보 은닉상태 계산

4. 최종 은닉 상태

- 업데이트 게이트를 사용하여 이전 은닉상태와 새로운 후보 은닉상태를 결합하여 최종 은닉상태 계산



딥러닝과 자연어 처리

GRU의 장점

- GRU는 LSTM보다 구조가 단순하여 계산 효율성 높음
- 더 적은 파라미터로 LSTM과 유사한 성능 제공
- 게이트 메커니즘을 통해 기울기 소실 문제 완화

GRU의 단점

- LSTM에 비해 구조가 단순하여, 일부 복잡한 패턴을 학습하는 데는 LSTM보다 효과적이지 않을 수 있음

이론 실습

colab: https://colab.research.google.com/drive/1tFMBQsIMH_gi0qgPfdcQVHngsrt9dzmu?usp=sharing

실습 과제 (LSTM)

1. LSTM의 기본 구조 이해: LSTM의 기본 원리와 구조를 이해하기
2. 모델 사용: LSTM 모델을 사용하여 새로운 텍스트 생성
3. 실제 데이터셋 사용: 실제 데이터셋을 사용해보기(<https://www.gutenberg.org>, <https://storage.googleapis.com>)
4. 텍스트 생성: 학습된 모델을 사용하여 새로운 텍스트를 생성

팁

- 학습 시간을 단축하기 위해 데이터셋 크기를 적절히 조정
- 모델의 하이퍼파라미터(예: embedding_dim, rnn_units, batch_size 등)를 조정하여 모델 성능을 비교
- SimpleRNN 대신 LSTM이나 GRU 레이어를 사용해 모델을 확장

실습 진행