□ 강의 개요

학습 목표:

- 이 강의의 핵심 개념을 이해합니다
- 실전에 적용할 수 있는 지식을 습득합니다

주요 키워드: [자동으로 채워질 예정]

선행 지식: 기본적인 컴퓨터 사용 능력

자연어 처리(NLP) 핵심 정리 노트

Harvard Extension School - CSCI E-89B (Fall 2025)

정리: Gemini

October 26, 2025

▣ **강의명:** CSCI E-89B: 자연어 처리 입문

▣ **주차:** Lecture 03

▣ **교수명:** Dmitry Kurochkin

▣ **목적:** Lecture 03의 핵심 개념 학습

Contents

1 개요: 자연어 처리란 무엇인가?

□ 핵심 요약

자연어 처리(Natural Language Processing, NLP)는 컴퓨터가 인간의 언어를 이해하고, 해석하며, 생성하게 만드는 기술 분야입니다. 이 분야는 언어학(Linguistics)과 인공지능(Artificial Intelligence, AI)이 만나는 지점에 있습니다. 궁극적인 목표는 인간과 컴퓨터가 보다 자연스럽게 소통하는 것입니다. NLP는 텍스트 분류, 기계 번역, 챗봇 등 다양한 응용 분야의 핵심 기술입니다. 성공적인 NLP 모델을 구축하려면 텍스트를 컴퓨터가 이해할 수 있는 형태로 가공하는 전처리 과정이 매우 중요합니다.

1.1 인공지능, 머신러닝, 딥러닝, NLP의 관계

이 네 가지 개념은 종종 혼용되지만, 포함 관계를 이해하는 것이 중요합니다.

- **인공지능(AI)**: 가장 넓은 개념으로, 인간의 지능을 모방하는 모든 기술을 포함합니다. 여기에는 규칙 기반 시스템(rule-based system)처럼 전문가가 직접 규칙을 코딩하는 방식도 포함됩니다.
- **머신러닝(ML)**: AI의 한 분야로, 데이터로부터 컴퓨터가 스스로 '규칙'을 학습하게 하는 접근 방식입니다. 데이터를 입력하면 모델이 패턴을 찾아내 예측이나 분류를 수행합니다.
- **딥러닝(DL)**: 머신러닝의 한 분야로, 인간의 뇌 신경망을 모방한 심층 신경망(Deep Neural Network)을 사용합니다. 특히 이미지, 음성, 텍스트와 같은 비정형 데이터 처리에서 뛰어난 성능을 보입니다.
- **자연어 처리(NLP)**: AI의 한 분야로, 인간 언어에 특화된 기술입니다. NLP 문제를 해결하기 위해 규칙 기반 접근법, 전통적인 머신러닝, 그리고 최근에는 딥러닝 방법론이 활발히 사용됩니다.

개념 간의 관계 (벤 다이어그램 설명)

가장 큰 원인 **인공지능(AI)** 안에 **머신러닝(ML)**이 포함됩니다.
머신러닝 원 안에 **딥러닝(DL)**이 더 작은 원으로 포함됩니다.
자연어 처리(NLP)는 AI의 큰 원 안에 있으면서, 머신러닝과 딥러닝 영역과 상당 부분 겹칩니다. 이는 NLP가 AI의 다양한 기술을 활용한다는 것을 의미합니다.

2 핵심 용어 정리

용어	쉬운 설명	원어	비고
토큰화	문장을 단어나 문장 같은 의미 있는 작은 단위(토큰)로 쪼개는 과정	Tokenization	NLP 전처리의 가장 첫 단계
어간 추출	단어의 접사(접두사, 접미사)를 제거하여 어간(stem)을 추출하는 과정	Stemming	빠르지만, 결과가 사전에 없는 단어일 수 있음 ('running' → 'run')
표제어 추출	단어의 문법적 형태를 고려하여 기본형, 즉 표제어(lemma)를 찾는 과정	Lemmatization	문맥을 파악해야 하므로 느리지만, 결과가 사전에 있는 단어임 ('driving' → 'drive')
임베딩	단어를 저차원의 밀집된 숫자 벡터(dense vector)로 변환하는 기술	Embedding	단어 간의 의미적 관계를 벡터 공간에 표현. 원-핫 인코딩의 단점을 보완.
원-핫 인코딩	단어 사전에 있는 단어 중 하나만 1이고 나머지는 모두 0인 희소 벡터(sparse vector)로 표현하는 방식	One-Hot Encoding	단어 수가 많아지면 벡터 차원이 커지고, 단어 간 유사성을 표현하지 못함.
불용어	문장에서 큰 의미를 갖지 않아 분석에 불필요한 단어 (조사, 관사 등)	Stop Words	전처리 과정에서 제거하여 계산 효율성을 높임. (예: a, the, is, 의, 는)
OOV	훈련 데이터의 단어 사전에 없어 알 수 없는 단어	Out-of-Vocabulary	테스트 시 새로운 단어가 나타날 때를 대비해 특별 토큰으로 처리.

3 핵심 개념 및 원리

3.1 텍스트 전처리(Text Preprocessing)의 중요성

컴퓨터는 텍스트를 그대로 이해할 수 없습니다. 따라서 모델에 입력하기 전에 텍스트를 숫자 형태의 데이터로 변환하고 정제하는 과정이 필수적입니다. 이 과정을 '전처리'라고 부릅니다.

- **목표**: 분석에 불필요한 노이즈를 제거하고, 텍스트의 핵심 정보를 보존하며, 모델이 학습하기 좋은 형태로 데이터를 가공하는 것.
- **주요 단계**: 토큰화(Tokenization) → 정제(Cleaning, 예: 불용어 제거) → 정규화(Normalization, 예: 어간/표제어 추출) → 벡터화(Vectorization).

3.1.1 1. 토큰화 (Tokenization)

텍스트를 최소 단위인 '토큰(token)'으로 분할하는 과정입니다. 어떤 단위를 토큰으로 삼을지에 따라 여러 기법이 있습니다.

- **단어 토큰화 (Word Tokenization)**: 띄어쓰기나 구두점을 기준으로 텍스트를 단어 단위로 나눕니다.

  □ **예제:**

  입력: "Henry Ford's innovation, the assembly line."
  결 과 : `['Henry', 'Ford', "'s", 'innovation', ',', 'the', 'assembly', 'line', '.']`

- **문장 토큰화 (Sentence Tokenization)**: 마침표(.), 물음표(?) 등 문장 끝을 나타내는 기호를 기준으로 텍스트를 문장 단위로 나눕니다.

  □ **예제:**

  입력: "He created assembly lines. This revolutionized production."
  결과: `['He created assembly lines.', 'This revolutionized production.']`

- **서브워드 토큰화 (Subword Tokenization)**: 단어를 의미 있는 더 작은 단위(subword)로 나눕니다. OOV(Out-of-Vocabulary) 문제를 완화하는 데 효과적입니다.

3.1.2 2. 어간 추출 (Stemming)과 표제어 추출 (Lemmatization)

'running', 'runs', 'ran'과 같은 단어들은 형태는 다르지만 '달리다'라는 동일한 의미를 갖습니다. 이렇게 다양한 형태의 단어들을 하나의 기본 형태로 통일하여 단어 사전의 크기를 줄이고 모델의 일반화 성능을 높일 수 있습니다.

어간 추출 vs. 표제어 추출 비교

특징	어간 추출 (Stemming)	표제어 추출 (Lemmatization)
목표	단어의 어미를 잘라내어 어간(기본 형태)을 찾음	단어의 사전적 기본형(표제어)을 찾음
방식	규칙 기반으로 접사를 기계적으로 제거	품사 등 문맥 정보를 활용하여 사전을 참조
속도	빠름	느림
결과	사전에 없는 단어가 될 수 있음	항상 사전에 있는 단어
예시	"driving" → "driv"	"driving" → "drive"
	"transportation" → "transport"	"was" → "be"
	"electric" → "electr"	"cars" → "car"

언제 무엇을 쓸까?

- **어간 추출**: 검색 엔진의 인덱싱처럼 속도가 중요하고, 결과 단어의 언어적 정확성이 덜 중요한 경우.
- **표제어 추출**: 챗봇이나 기계 번역처럼 결과의 의미적 정확성이 매우 중요한 경우.

3.1.3 3. 임베딩 (Embedding)

텍스트를 벡터로 만드는 과정에서, 원-핫 인코딩은 단어 수만큼 차원이 커지고 단어 간 유사성을 표현하지 못하는 한계가 있습니다. 임베딩은 이러한 문제를 해결하기 위해 등장했습니다.

- **핵심 아이디어**: 단어를 고차원의 희소 벡터(sparse vector)에서 저차원의 밀집 벡터(dense vector)로 변환하는 것.
- **과정**:
  1. 단어 사전에 있는 각 단어에 대해 고유한 인덱스(정수)를 부여합니다.
  2. 신경망에 '임베딩 레이어'를 추가합니다. 이 레이어는 각 인덱스를 입력받아 특정 차원(예: 128차원)의 벡터로 매핑합니다.
  3. 이 매핑 가중치(lookup table)는 모델이 훈련하는 과정에서 다른 가중치들과 함께 학습됩니다. 결과적으로 의미가 비슷한 단어들은 벡터 공간에서 서로 가까운 위치에 배치됩니다.
- **장점**:
  - **차원 축소**: 수만 개의 차원을 수백 개의 차원으로 줄여 계산 효율성을 높입니다.
  - **의미 학습**: 단어의 문맥적 의미를 벡터에 담을 수 있습니다. (예: `king - man + woman ≈ queen`)
  - **훈련 가능**: 임베딩 벡터 자체가 모델의 파라미터로서 특정 과제에 맞게 최적화됩니다.

□ **예제: title**

'female'과 'male'이라는 두 단어가 있다고 가정해봅시다.

- **원-핫 인코딩**: female: `[1, 0]`, male: `[0, 1]` (2차원)
- **임베딩 (1차원으로 축소)**: female: `[1]`, male: `[0]`

수만 개의 단어가 있는 실제 문제에서는, 10000차원의 원-핫 벡터를 128차원의 임베딩 벡터로 변환하는 것과 같습니다. 이 변환 과정은 훈련을 통해 최적의 값을 찾습니다.

4 NLP 텍스트 분류 절차

다음은 20 newsgroups 데이터셋을 사용하여 '하키(hockey)'와 '판매(for sale)' 관련 게시물을 분류하는 모델을 구축하는 일반적인 절차입니다.

1. **데이터 로드 및 준비**
   - 훈련(train) 데이터와 테스트(test) 데이터를 로드합니다.
   - 분류할 카테고리(예: `rec.sport.hockey`, `misc.forsale`)를 지정합니다.
2. **텍스트 전처리 및 단어 사전 구축**
   - **토큰화**: 모든 훈련 텍스트를 단어 토큰으로 분할합니다.
   - **정규화 (선택)**: 어간 추출(stemming)이나 표제어 추출(lemmatization)을 적용하여 단어를 기본 형태로 통일합니다.
   - **빈도 계산**: 각 단어의 등장 빈도를 계산합니다.
   - **단어 사전 생성**: 가장 빈도가 높은 N개(예: 10,000개)의 단어만 선택하여 단어 사전을 만듭니다.
   - **OOV 처리**: 사전에 없는 단어(Out-of-Vocabulary)를 처리하기 위해 특별 토큰(예: 인덱스 0)을 예약합니다.
3. **텍스트를 시퀀스로 변환**
   - 각 텍스트(게시물)를 단어 사전에 따라 정수 인덱스의 시퀀스로 변환합니다.
   - 예: "the game was fun" → `[5, 120, 25, 8]`
   - OOV 단어는 예약된 인덱스(예: 0)로 변환합니다.
4. **패딩 (Padding)**
   - 신경망 모델에 입력하려면 모든 시퀀스의 길이가 동일해야 합니다.
   - 최대 길이(예: 500)를 정하고, 이보다 짧은 시퀀스는 뒤쪽에 특정 값(보통 0)을 채워 길이를 맞춥니다.
5. **신경망 모델 구축**
   - **임베딩 레이어**: 정수 인덱스를 입력받아 밀집 벡터로 변환합니다. (입력 차원: 단어 사전 크기, 출력 차원: 임베딩 차원)
   - **순환 신경망 (RNN/LSTM) 레이어**: 시퀀스 데이터의 시간적 패턴을 학습합니다.
   - **드롭아웃 (Dropout) 레이어**: 과적합(overfitting)을 방지하기 위해 훈련 중 일부 뉴런을 무작위로 비활성화합니다.
   - **출력 레이어**: 최종적으로 분류 결과를 출력합니다. (이진 분류의 경우, Sigmoid 활성화 함수를 사용하는 하나의 뉴런)
6. **모델 훈련 및 평가**
   - 모델을 컴파일합니다. (손실 함수: `binary`$_c$$rossentropy$, : `adam`)
   - 준비된 훈련 데이터로 모델을 훈련시킵니다.
   - 훈련이 끝난 후, 테스트 데이터로 모델의 성능(예: 정확도)을 평가합니다.

5 실습 코드 및 결과 분석

5.1 1. NLTK를 이용한 토큰화 및 어간 추출

5.1.1 토큰화 (Tokenization)

NLTK(Natural Language Toolkit) 라이브러리를 사용하면 단어 및 문장 토큰화를 쉽게 수행할 수 있습니다. [title=NLTK 단어 토큰화 코드]

```
import nltk
from nltk.tokenize import word_tokenize

# NLTK 데이터다운로드최초 ( 회1 필요)
nltk.download('punkt')

text = "Henry Ford's innovation, the assembly line process, changed the car industry's dynamics profoundly."

# 단어로토큰화
word_tokens = word_tokenize(text)
print(word_tokens)

# 결과:
# ['Henry', 'Ford', "'s", 'innovation', ',', 'the', 'assembly', 'line', 'process', ',', 'changed', 'the', 'car', 'industry', "'s", 'dynamics', 'profoundly', '.']
```

Listing 1: NLTK를 사용한 단어 토큰화

5.1.2 어간 추출 (Stemming)

가장 널리 쓰이는 Porter Stemmer와 Snowball Stemmer를 사용하여 토큰화된 단어들에 어간 추출을 적용할 수 있습니다. [title=NLTK 어간 추출 코드]

```
from nltk.stem import PorterStemmer, SnowballStemmer

words = ['Henry', 'Ford', "'s", 'innovation', 'changed', 'industry', 'dynamics', 'profoundly']

# Porter Stemmer 적용
porter = PorterStemmer()
porter_stems = [porter.stem(word) for word in words]
print("Porter Stemmer:", porter_stems)
# 결과: ['henri', 'ford', "'s", 'innov', 'chang', 'industri', 'dynam', 'profoundli']

# Snowball Stemmer 적용 (보다Porter 개선됨)
snowball = SnowballStemmer("english")
```

```
snowball_stems = [snowball.stem(word) for word in words]
print("Snowball Stemmer:", snowball_stems)
# 결과: ['henri', 'ford', "'s", 'innov', 'chang', 'industri', 'dynam', '
    profound']
```

Listing 2: Porter 및 Snowball Stemmer 적용

주의사항

결과에서 볼 수 있듯이, 어간 추출은 단어를 `industri`나 `profoundli`처럼 사전에 존재하지 않는 형태로 만들 수 있습니다. 이는 단순히 규칙에 따라 접미사를 제거하기 때문입니다. `Henry`가 `Henri`로 변하는 등 고유명사에도 적용될 수 있습니다.

5.2 2. SpaCy를 이용한 표제어 추출

SpaCy는 산업 수준의 성능을 제공하는 NLP 라이브러리로, 정확한 표제어 추출 기능을 제공합니다. [title=SpaCy 표제어 추출 코드]

```
import spacy

# SpaCy 모델로드최초 ( 회1 다운로드필요 )
# python -m spacy download en_core_web_sm
nlp = spacy.load("en_core_web_sm")

text = "Henry Ford's innovation changed the car industry's dynamics
    profoundly."
doc = nlp(text)

# 표제어추출
spacy_lemmas = [token.lemma_ for token in doc]
print(spacy_lemmas)

# 결과:
# ['Henry', 'Ford', "'s", 'innovation', 'change', 'the', 'car', '
    industry', "'s", 'dynamic', 'profoundly', '.']
```

Listing 3: SpaCy를 사용한 표제어 추출

결과 비교: Stemming vs. Lemmatization

- `changed` → `chang` (Porter Stemmer) vs. `change` (SpaCy Lemmatizer) - `dynamics` → `dynam` (Porter Stemmer) vs. `dynamic` (SpaCy Lemmatizer)
표제어 추출이 문법적으로 더 올바르고 해석 가능한 결과를 제공함을 알 수 있습니다.

5.3 3. 텍스트 분류 모델 결과 분석

20 newsgroups 데이터셋으로 훈련한 LSTM 모델의 테스트 정확도는 전처리 방식에 따라 미세한 차이를 보였습니다.

- **단순 토큰화만 적용**: 약 93.5%의 테스트 정확도
- **어간 추출(Stemming) 적용**: 약 97%의 테스트 정확도
- **표제어 추출(Lemmatization) 적용**: 약 95-96%의 테스트 정확도

결과 해석

이 특정 과제에서는 어간 추출을 적용했을 때 성능이 가장 좋았습니다. 이는 단어의 다양한 변형을 하나의 형태로 통일함으로써 모델이 더 적은 수의 특징으로 핵심 패턴을 학습할 수 있었기 때문일 수 있습니다. 표제어 추출은 어간 추출보다 더 정교하지만, 이로 인한 계산 비용 증가나 미미한 성능 차이로 인해 특정 상황에서는 어간 추출이 더 효율적일 수 있습니다.
결론: 어떤 전처리 기법이 최적인지는 데이터와 과제에 따라 다르므로, 여러 방법을 실험하고 검증 데이터셋에서의 성능을 비교하여 결정하는 것이 좋습니다.

6 자주 묻는 질문 (FAQ)

Q1: 어간 추출과 표제어 추출 중 항상 더 좋은 방법이 있나요?

A: 아니요, 항상 더 좋은 방법은 없습니다. 작업의 목표에 따라 선택이 달라집니다.

- **속도가 중요하다면** (예: 대규모 문서 인덱싱) 어간 추출이 더 나은 선택일 수 있습니다.
- **의미의 정확성이 중요하다면** (예: 챗봇, 기계 번역) 표제어 추출이 필수적입니다.

실제 프로젝트에서는 두 가지 방법을 모두 시도해보고 성능이 더 잘 나오는 쪽을 선택하는 경우가 많습니다.

Q2: 왜 원-핫 인코딩 대신 임베딩을 사용해야 하나요?

A: 두 가지 주된 이유가 있습니다.

1. **차원의 저주 회피**: 원-핫 인코딩은 단어 수가 많아지면 벡터의 차원이 수만, 수십만으로 커져 계산이 비효율적입니다. 임베딩은 이를 수백 차원의 밀집 벡터로 압축합니다.
2. **의미 관계 학습**: 원-핫 벡터들은 모두 서로 직교하므로 단어 간 유사도를 계산할 수 없습니다. 임베딩은 훈련 과정에서 비슷한 의미의 단어들을 벡터 공간상에 가깝게 배치하여 의미적 관계를 학습합니다.

Q3: OOV(사전에 없는 단어)는 왜 중요하고 어떻게 처리하나요?

A: 훈련 데이터에 없던 단어가 테스트 데이터에 나타나면 모델은 이를 처리할 수 없어 오류가 발생하거나 성능이 저하됩니다. 이것이 OOV 문제입니다.

- **처리 방법**: 단어 사전을 만들 때 '알 수 없는 단어'를 의미하는 특별 토큰(예: `<UNK>` 또는 `<OOV>`)을 추가합니다. 그리고 훈련 시에도 일부러 드물게 나타나는 단어들을 이 토큰으로 대체하여 모델이 OOV 상황에 대처하도록 학습시킬 수 있습니다. 테스트 시 사전에 없는 단어가 나오면 이 특별 토큰으로 매핑하여 처리합니다.

Q4: 드롭아웃(Dropout)은 왜 사용하나요?

A: 드롭아웃은 모델의 **과적합(Overfitting)**을 방지하기 위한 정규화(regularization) 기법입니다. 과적합은 모델이 훈련 데이터에만 너무 과도하게 맞춰져서, 새로운 데이터(테스트 데이터)에 대해서는 성능이 떨어지는 현상을 말합니다.

- **작동 원리**: 훈련 과정에서 각 미니배치마다 신경망의 뉴런 중 일부를 확률적으로 비활성화(출력을 0으로 만듦)합니다. 이를 통해 모델이 특정 뉴런에 과도하게 의존하는 것을 막고, 여러 뉴런이 협력하여 더 강건한(robust) 특징을 학습하도록 유도합니다.

7 한눈에 보는 핵심 요약

NLP 기본 개념

자연어 처리 (NLP)
컴퓨터가 인간의 언어를 다루게 하는 AI의 한 분야. 언어학과 컴퓨터 과학의 교차점.

텍스트 전처리 3대장

1. **토큰화 (Tokenization)** 문장을 단어/문장 등 작은 단위로 쪼개기.
2. **어간 추출 (Stemming)** 단어 끝을 잘라 기본형 찾기. 빠르지만 부정확할 수 있음.
3. **표제어 추출 (Lemmatization)** 사전을 이용해 진짜 기본형(표제어) 찾기. 정확하지만 느림.

단어의 벡터화: 원-핫 인코딩 vs. 임베딩

	원-핫 인코딩	**임베딩**
형태	희소 벡터 (대부분 0)	밀집 벡터 (의미 있는 실수값)
차원	고차원 (단어 수만큼)	저차원 (사용자 지정, 예: 128)
의미 표현	불가능 (모든 단어 독립적)	가능 (비슷한 단어는 벡터 공간에서 가까움)
결론	간단하지만 한계 명확	차원 축소 및 의미 학습에 효과적

NLP 텍스트 분류 파이프라인

데이터 로드 → 토큰화 → 정규화 (Stem/Lemma) → 정수 인코딩 → 패딩 → 모델 훈련 (Embedding+RNN) → 평가

핵심 모델 구성 요소

- **LSTM (Long Short-Term Memory)**: 순서가 중요한 시퀀스 데이터(문장 등)를 잘 처리하는 RNN의 한 종류.
- **Dropout**: 훈련 데이터에 모델이 과하게 맞춰지는 과적합을 방지하는 기술.
- **이진 교차 엔트로피 (Binary Cross-Entropy)**: 두 개 중 하나를 맞추는 이진 분류 문제에서 주로 사용하는 손실 함수.