▣ **강의명:** CSCI E-89B: 자연어 처리 입문

▣ **주차:** Lecture 04

▣ **교수명:** Dmitry Kurochkin

▣ **목적:** Lecture 04의 핵심 개념 학습

▣ **강의명:** CSCI E-89B: 자연어 처리 입문

▣ **주차:** Lecture 04

▣ **교수명:** Dmitry Kurochkin

▣ **목적:** Lecture 04의 핵심 개념 학습

▣ **강의명:** CSCI E-89B: 자연어 처리 입문

▣ **주차:** Lecture 04

▣ **교수명:** Dmitry Kurochkin

▣ **목적:** Lecture 04의 핵심 개념 학습

▣ **강의명:** CSCI E-89B: 자연어 처리 입문

▣ **주차:** Lecture 04

▣ **교수명:** Dmitry Kurochkin

▣ **목적:** Lecture 04의 핵심 개념 학습

□ 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

CSCI E-89B 자연어 처리 4주차 노트

Bag of Words, n-grams, and CNN

Contents

1 개요: 텍스트를 숫자로 바꾸는 여정

□ 핵심 요약

핵심 요약 이번 강의에서는 텍스트를 컴퓨터가 이해할 수 있는 숫자 벡터로 변환하는 방법을 다룹니다. 가장 기본적인 **Bag of Words**부터 시작하여, 단어의 순서를 일부 고려하는 **n-grams**를 배웁니다. 마지막으로, 이미지 처리에 주로 쓰이는 **Convolutional Neural Networks (CNN)**를 텍스트에 적용하는 원리를 탐구합니다. 이러한 기법들은 텍스트 분류, 감성 분석 등 다양한 자연어 처리(NLP) 문제의 기초가 됩니다. 궁극적으로는 텍스트의 의미적, 구조적 정보를 어떻게 효과적으로 포착할 것인가에 대한 고민이 담겨 있습니다.

□ 예제:

학습 로드맵

1. **기초 다지기**: Bag of Words (BoW)의 개념과 한계를 명확히 이해합니다.
2. **문맥 추가하기**: n-grams가 BoW의 어떤 단점을 보완하는지 파악합니다.
3. **고급 모델 맛보기**: 텍스트를 이미지처럼 다루는 CNN의 아이디어를 이해합니다.
4. **실습으로 체득하기**: Python 라이브러리(`sklearn`, `NLTK`, `spaCy`)를 사용해 BoW와 n-grams를 직접 구현해봅니다.
5. **개념 연결하기**: BoW의 희소성(sparsity) 문제를 해결하기 위한 대안으로 임베딩(embedding)의 필요성을 인식합니다.

2 핵심 용어 정리

자주 사용되는 전문 용어를 미리 익혀두면 학습에 도움이 됩니다.

Table 1: *4*주차 핵심 용어

용어	쉬운 설명	원어	비고
Bag of Words	문장의 단어 순서를 무시하고, 단어의 출현 빈도수만 가방에 담듯이 세는 방법	Bag of Words (BoW)	간단하지만 문맥 정보를 잃어버림
토큰화	문장을 의미 있는 단위(토큰)로 쪼개는 과정	Tokenization	단어, 글자, 서브워드(subword) 등이 토큰이 될 수 있음
n-gram	텍스트에서 연속적으로 나타나는 n개의 단어 묶음	n-gram	2-gram(bigram), 3-gram(trigram) 등이 있음. 지역적 문맥 포착
어간 추출	단어에서 접사(prefix, suffix)를 제거하여 기본형(어간)을 찾는 과정	Stemming	빠르지만, 결과가 실제 단어가 아닐 수 있음 (예: octopi → octop)
표제어 추출	단어의 사전적 기본형(표제어)을 찾는 과정	Lemmatization	문법적 품사를 고려하여 더 정확하지만, 어간 추출보다 느림
임베딩	단어를 의미를 담은 저차원의 조밀한(dense) 벡터로 표현하는 기법	Embedding	단어 간의 의미적 유사성을 벡터 공간의 거리로 표현 가능
CNN	이미지의 지역적 패턴을 추출하는 데 특화된 딥러닝 모델	Convolutional Neural Network	텍스트에 적용 시, n-gram처럼 지역적 단어 패턴을 학습
필터 (커널)	CNN에서 특정 특징(예: 수직선, 특정 단어 조합)을 감지하는 가중치 행렬	Filter (Kernel)	필터를 입력 데이터에 슬라이딩하며 특징 맵(feature map)을 생성
패딩	필터 연산 시 출력 크기가 줄어드는 것을 막기 위해 입력 데이터 주변을 특정 값(주로 0)으로 채우는 것	Padding	`VALID`(패딩 없음) vs `SAME`(출력 크기 유지)
스트라이드	필터가 입력 데이터 위를 한 번에 이동하는 칸의 수	Strides	스트라이드가 크면 출력 크기가 더 많이 줄어듦
풀링	특징 맵의 크기를 줄여(downsampling) 계산량을 감소시키고, 주요 특징을 강조하는 과정	Pooling	Max Pooling은 특정 영역에서 가장 큰 값만 남김

3 Bag of Words (BoW)

3.1 BoW의 정의와 직관

Bag of Words (BoW)는 텍스트를 숫자 벡터로 표현하는 가장 직관적이고 기본적인 방법입니다. 이름 그대로, 단어들을 순서 없이 '가방'에 넣고 각 단어가 몇 번 등장했는지만 세는 방식입니다.

□ 핵심 요약

핵심 아이디어 문법이나 단어의 순서는 완전히 무시하고, 오직 문서 내 각 단어의 출현 빈도에만 집중합니다. "John likes to watch movies. Mary likes movies too." 라는 문장이 있다면, BoW는 단순히 'John:1, likes:2, to:1, watch:1, movies:2, Mary:1, too:1' 와 같이 빈도를 기록합니다.

이 방법은 텍스트 분류, 감성 분석, 주제 모델링 등에서 간단하면서도 강력한 특징 벡터(feature vector)를 생성하는 데 사용됩니다.

3.2 BoW 생성 절차

BoW 표현을 만드는 과정은 다음 3단계로 나눌 수 있습니다.

1. **1단계: 토큰화 (Tokenization)**
   분석할 전체 텍스트(말뭉치, Corpus)를 의미 있는 최소 단위인 **토큰(token)**으로 분리합니다. 보통 단어 단위로 쪼갭니다.
   예시 문장: "Henry Ford introduced the Model T. Ford Model T was revolutionary."
   토큰화 결과: `['Henry', 'Ford', 'introduced', 'the', 'Model', 'T', '.', 'Ford', 'Model', 'T', 'was', 'revolutionary']`
2. **2단계: 어휘 집합(Vocabulary) 생성**
   전체 말뭉치에서 등장한 모든 고유한 토큰들을 모아 하나의 집합, 즉 어휘 집합을 만듭니다. 이 어휘 집합이 벡터의 차원이자 기준이 됩니다. 보통 텍스트에 등장하는 순서대로 추가합니다.
   예시 어휘 집합: `{"Henry", "Ford", "introduced", "the", "Model", "T", "was", "revolutionary"}`
3. **3단계: 벡터화 (Vectorization)**
   각 문장(또는 문서)을 어휘 집합을 기준으로 벡터로 변환합니다. 벡터의 각 차원은 어휘 집합의 특정 단어에 해당하며, 값은 해당 단어가 문장에서 나타난 빈도수입니다.
   예시 문장 벡터: 어휘 집합 순서에 따라 빈도를 세면 다음과 같습니다.
   - Henry: 1
   - Ford: 2
   - introduced: 1
   - the: 1
   - Model: 2
   - T: 2
   - was: 1