자연어 처리 시스템 비교 연구

Natural Language Processing Research

저자
(Authors)

이동영
Dongyoung Lee

출처
(Source)

한국정보과학회 학술발표논문집 , 2018.12, 1771-1773(3 pages)

발행처
(Publisher)

한국정보과학회
KOREA INFORMATION SCIENCE SOCIETY

URL

http://www.dbpia.co.kr/journal/articleDetail?nodeId=NODE07614088

APA Style

이동영 (2018). 자연어 처리 시스템 비교 연구. 한국정보과학회 학술발표논문집, 1771-1773

자연어 처리 시스템 비교 연구

이동영
성균관대학교 컴퓨터공학과
ldy241@naver.com

Natural Language Processing Research

Dongyoung Lee
Computer Engineering, Sungkyunkwan University

요 약

자연어 처리(NPL, Natural Language Processing) 분야는 컴퓨터가 사람처럼 인간의 언어를 이해하고 표현하게 해주는 분야이다. 현재 자연어 처리 시스템은 인공지능 분야의 한 갈래로 자리 잡고 있으며 수요가 늘어나며 급격한 기술적 발전이 이루어지고 있다. 본 논문에서는 자연어 처리 시스템을 중점적으로 조사한다. 이를 위하여 자연어 처리 시스템의 구성 요소들을 논의하며 다양한 방법론을 조사하고 분석한다. 마지막으로 여러 가지 자연어 처리 시스템을 비교하여 분석하고 앞으로 발전 방향과 응용 방법을 다룬다.

1. 서 론

자연어 처리는 사람이 말하는 언어를 기계언어로 분석하여 컴퓨터가 읽어 들어 작동할 수 있는 형태로 만드는 자연어의 이해나 그러한 형태를 반대로 인간이 이해할 수 있는 자연어로 표현하는 기술을 의미한다.

우리가 인간과 인간 사이의 상호 작용을 말할 때 우리는 자연어를 사용하여 인간이 서로 의사소통하는 방법에 대하여 이야기한다. 자연어는 사람들이 원어로 사용하는 언어이다. 한국어, 영어 및 일본어는 모두 자연어의 예이다. 반면에 컴퓨터는 항상 인공 언어(SQL, Java, C++ 등의 컴퓨터 프로그래밍 언어)로 작동한다. 이 언어는 기계에 명령을 전달하기 위하여 만들어졌다.

컴퓨터는 인공 언어로 작동하기 때문에 자연어를 이해할 수 없다. 이것은 자연어 처리 시스템이 해결해야하는 문제이다. 자연어 처리를 하면 컴퓨터는 사람이 말하는 자연어를 듣고 그 의미를 이해 한 다음 필요할 경우 자연어를 생성하여 응답하고 사람과 다시 통신할 수 있다. 그렇다면 자연어 처리 기술은 왜 필요한 것 일까? 자연어 처리의 목적[1]을 세 가지로 요약할 수 있다.

첫 번째는 대량의 자연어 데이터 처리이다. 많은 데이터 처리를 행하는 것보다 그 언어 데이터 속에 포함되어 있는 다른 언어들의 수, 다른 단어나 문장과 함께 사용되는 빈도분포 그리고 그 언어 문자 종류의 반도분포 등 언어 데이터가 가지는 각종의 통계 데이터를 수집 분석할 수가 있다. 시대와 함께 언어들이 어떻게 변화하고 발전했는지도 조사할 수 있다. 따라서 대량의 자연어 데이터의 통계적 분석과 컴퓨터를 수반한 자연어 처리기술이 달성한 역할은 큰 공헌을 하였다.

두 번째는 컴퓨터에 자연어를 읽고 이해시키는 자연어 이해 시스템(Natural Language Understanding System)의 연구이다. 이 응용기술들은 질문 응답 시스템(Machine Translation System)과 요약 시스템 등이 있다. 자연어에 대한 컴퓨터의 이해는 컴퓨터와 인간 사이에 존재하는 사용 언어가 서로 다르기 때문에 발생하는 불편함을 해소하기 위한 목적으로 하는 것이다.

세 번째는 컴퓨터상에서 자연어 처리의 모델을 작성하는 것보다 인간이 사용하는 언어 이해의 과정을 설명하는 것이다. 컴퓨터상의 언어 이해의 모델을 작성하고 그 움직임의 모양 때문에 인간의 언어 이해 과정을 미루어 추측하려고 하는 것으로서, 최근의 인지과학의 큰 연구과제가 되고 있다.

그렇다면 자연어 처리 시스템은 어떻게 개발해야 하는 것인가? 자연어 처리 시스템의 개발은 쉽지 않다. 왜냐하면 컴퓨터는 인간이 컴퓨터에게 정확하고 모호하지 않으며 구조화된 음성 명령을 통하여 말하는 것을 요구하기 때문이다. 하지만 인간의 말이 항상 정확한 것이 아니다. 인간의 말은 종종 모호하고 언어 구조는 속어, 방언 그리고 사회적인 맥락을 포함하여, 많은 복잡한 변수들에 의존할 수 있기 때문이다. 이를 수행하기 위하여 자연어를 입력하는 기술, 입력된 언어를 해석하는 기술, 해석된 언어를 응용하는 기술이 요구된다.

전통적인 머신러닝에 의존한 자연어 처리 시스템은 사람이 직접 추출한 피처에 강하게 의존한다. 이러한 피처들은 추출하는 데 시간이 많이 소요되고 불완전하다. 하지만 최근 Dense Vector Representation에 기반의 Neural

Network가 다양한 자연어 처리 시스템 Task에서 우수한 성능을 보여주었다. 이러한 트렌드는 워드 임베딩(Word Embedding)과 딥러닝(Deep Learning)기법의 성공으로써 도달한 것이다.

본 논문에서는 수많은 복잡한 딥러닝 기반의 알고리즘이 어려운 NPL 문제를 풀기위하여 간단한 딥러닝 프레임워크를 제시한다. 먼저 텍스트 분류(Text Classification)에 대한 방법과 이론을 알아본다. 그리고 이를 위하여 중점으로 둘 것은 단어 임베딩(Word Embedding)이다. 최근 초기 모델인 Continuous Bag-of-Words(CBoW), Relation Network(RN)을 조사해보고 Convolutional Neural Network(CNN)과 Self Attention 그리고 Recurrent Neural Network(RNN)에 대해서 조사한다. 마지막으로 위 방법론들을 바탕으로 향후 트렌드에 대하여 제시한다.

2. 텍스트 분류(Text Classification)

자연어 처리에서 텍스트 분류는 문장, 문단 또는 글을 어떤 카테고리에 분류하는 작업을 뜻한다. 본 논문에서는 텍스트 분류를 분석 방법론으로 정한다. 텍스트 분류의 입력은 Natural Language Sentence이다. 즉 문장이나 문단 혹은 문서가 들어오게 된다. 입력이 들어오면서 출력은 이 문장이 어떤 카테고리에 속하는지를 판별한다. 그렇다면 문장을 어떻게 컴퓨터 언어로 표현할지를 알아보아야 한다.

문장은 일련의 토큰(Token)으로 구성되어 있다. 텍스트 토큰은 주관적, 임의적인 성격을 갖고 있다. 이 토큰을 나누는 기준은 다양하다. 공백, 형태소, 어절, 비트숫자 등이 있을 수 있다. 컴퓨터에게 단어를 숫자로 표현하기 위해서, 단어장을 만들고, 중복되지 않는 인덱스(Index)로 바꾼다. 궁극적으로 모든 문장을 일련의 정수로 바꿔준다. 이를 인코딩(Encoding)이라고 한다. 하지만 관계없는 숫자의 나열로 인코딩하는 것이 우리가 원하는 것이 아니다. 그렇다면 어떻게 관계를 만들어야 할까.

한 가지 방법으로 “One hot encoding”이 있을 수 있다. 이 방법은 신경망이 토큰의 의미를 잡아내는데 적합하지 않다. 이에 대한 대안은 각 토큰을 연속 벡터 공간(Continuous Vector Space)에 투영하는 방법이다. 이를 임베딩(Embedding)이라고도 한다. 이는 Table Look Up 과정이 필요하다. 각 One hot encoding된 토큰에게 벡터를 부여하는 과정이다. 실질적으로는 One hot encoding 벡터 x와 연속 벡터 공간 w를 내적 한 것이다. Table Look Up 과정을 거친 후 모든 문장 토큰은 연속적이고 높은 차원의 벡터로 변한다.[2]

3. 워드 임베딩(Word Embedding)

워드 임베딩은 텍스트를 구성하는 단어를 수치화 하는 방법이다. 이는 자연어 처리에서 필수적인 개념이며 Word를 R차원의 Vector로 Mapping하는 것을 말한다. 본 논문에서는 총 다섯 가지의 방법에 대하여 비교 분석한다.

3.1. Continuous Bag-of-Words(CBoW)

CBoW는 단어장을 단어 주머니로 보게 되고, 이에 따라 단어의 순서는 무시한다. 즉, 토큰 순서가 어떻든 상관없고 그냥 벡터로 보겠다는 것이다. 그리고 문장에 대한 표현은 단어 벡터들을 평균시킨 벡터로 구한다. 이 결과는 3차원 Space라면 하나의 Point로 나타내어지며 이 Point가 문장의 의미를 결정하게 된다. 공간상에서 가까우면 비슷한 의미, 아니면 멀리 떨어져 있을 것이다. CBoW는 효과가 좋기 때문에 제일 먼저 시도해 보아야 한다. 즉, Baseline Model이다.[2]

3.2. Relation Network(RN, Skip-bigram)

CBoW의 결과는 좋은 성능을 내지만 단어가 뒤죽박죽 섞이는 것 대신에 단어 순서도 보고 각 단어들의 관계를 좀 더 자세히 보기 위하여 RN을 사용한다.[3] RN은 문장 안에 있는 모든 토큰(Pairs)을 보고, 각 쌍에 대해서 신경망을 만들어서 문장 표현을 찾는다. 즉 Pair의 Representation을 찾는다는 것이다. 이 과정을 거치고 난 후 벡터가 나오게 되는데 이들의 평균값을 찾는다. 그 이후는 신경망에서 사용하는 학습 방법과 동일하다. 즉 CBoW와의 차이는 순서를 무시하는 것에서 각 토큰들이 Pair를 갖는다는 것이다. 이를 통한 장점은 CBoW가 찾을 수 없는 여러 단어로 된 표현을 탐지할 수 있다. 하지만 RN도 만족스럽지 않은 점이 있는데 굳이 모든 단어의 Pair를 보아야 하는가에 대한 의문이다. 그래서 RN의 대체로 CNN을 생각해 볼 수 있다.

3.3. Convolutional Neural Network(CNN)

CNN으로 자연어 처리를 하게 되면 k-gram을 계층적으로(hierarchically) 보게 된다. Layer를 쌓을 때마다, 점진적으로 넓은 범위를 보기 때문에, “단어 > 다중 단어 표현 > 구절 > 문장” 순으로 보는 인간의 인식과도 같다.[2] Convolution Layer는 Max Pooling 계층이 잇따라 수행된다. 이러한 전략을 쓰는 데는 Input을 항상 고정된 차원의 Output으로 Mapping하기 때문이고, 전체 문장에서 가장 핵심적인 N-gram Feature를 유지하면서 출력의 차원을 줄일 수 있기 때문이다. 따라서 이런 조합은 Deep CNN을 만들기 위하여 겹쳐 쌓게 되며 문장의 분석을 개선할 수 있도록 풍부한 의미 정보를 포함하는 추상화된 표현을 잡아낸다.[4] 또한 좁은 지역 간의 단어의 관계도 볼 수 있다.

3.4. Self Attention

CNN의 단점은 Short Range Dependency이며 아주 긴 거리의 중요한 관계가 있을 때 Layer를 많이 쌓아야 하는 점이 있다. 이 네트워크는 먼 거리 간의 단어가 어떤 패턴을 이루고 있는지 찾기 힘들다. 하지만 CNN의

방식을 가중치가 부여된 RN의 일종으로 볼 수도 있다. 바로 이러한 관점을 적용한 것이 Self Attention이다.
Self Attention은 각 Pair를 보는데 그 관계의 중요성을 결정하는 함수가 있다. 이 함수가 Weight를 크거나 작게 결정할 수 있는 역할을 한다. 이렇게 하면 Long Range & Short Range Dependency를 극복할 수 있다. 따라서 관계의 중요성을 평가하는 기능이 추가되었다.

3.5. Recurrent Neural Network(RNN)

Self Attention이 좋은 방식이지만 복잡도와 특정 연산의 문제를 해결하기가 쉽지 않다. 그래서 그 대체로 나온 것이 RNN이다. RNN은 메모리를 갖고 있어 현재까지 읽은 정보를 저장할 수 있다. 이 문장의 정보를 시간의 순서에 따라 압축할 수 있다는 장점이 있다. 하지만 문장이 많이 길어질 수록 고정된 메모리에 압축된 정보를 담아야 하기 때문에, 앞에서 학습한 정보를 잊는다. 이는 곧 정보의 손실을 뜻하게 된다. 또한 토큰을 순차적으로 하나씩 읽어야 하기 때문에, 훈련할 때 속도가 기타 네트워크보다 느리다는 단점이 있다.

4. 결 론

본 논문에서 워드 임베딩(Word Embedding)을 하는 방법을 다섯 가지를 조사해보았다. 기본적으로 CBoW, RN(Skip-bigram) 그리고 CNN을 알아보았다. 또한 CNN과 RN을 같이 사용하면서 일반화 할 것인가에 대한 결과로 Self Attention이 나오게 되었다. 마지막으로 그 보완으로 RNN도 조사해보았다. 여기서 CBoW를 뺀 나머지 네 가지 같은 경우는 문장이 주어졌을 때, Vector 하나만 주어지는 것이 아니라 각 Token 위치별로 Vector들이 나오게 되는 것이다. 이 것을 계속 쌓는다고 해도 똑같이 계속 이어보면 Vector가 계속 나오는 것이다.

그리고 본 논문에서 조사한 워드 임베딩 다섯 가지 방법들이 배타적인 것이 아니라 혼합해서 사용할 수 있다. 예를 들면 최근에 Google에서 Self Attention과 RNN을 결합하여 RNN한번 Processing을 하고 그 위에 Self Attention을 하면 퍼포먼스 향상에 도움이 된다는 연구도 있다. 원리는 전부다 구별 가능하고 연속적인 Node들 이므로 어떻게 합치는가, 어떤 문제를 푸는가에 따라서 결정을 하고 특정 결합이 더 좋고 나쁨을 알 수 있다. 최종적으로 Classification을 위한다면 대부분 Averaging을 하게 된다 또는 RNN을 Set Representation위에다가 적용하게 된다.

결과적으로 본 논문에서 자연어 처리 시스템을 보다 효율적으로 사용하기 위하여 워드 임베딩 분야를 조사하였고 최근 이러한 방법론들을 결합하여 퍼포먼스를 높이는 연구가 진행되고 있다. 따라서 현실의 다양한 분야에 적용하여 효율을 낼 수 있기를 기대한다.

참 고 문 헌

[1] http://www.aistudy.co.kr/linguistics/natural/natural_language_processing.htm
[2] Kyunghyun Cho. (2018). Courant Institute (Computer Science) and Center for Data Science.
[3] Santoro et al. (2017). A simple neural network module for relational reasoning.
[4] Young T, Hazarika D, Poria S, Cambria E. (2017). Recent Trends in Deep Learning Based Natural Language Processing.