비전문가의 인공지능 분야 진입 실험: Tensor Flow를 기반으로

송윤섭* · 권영미

1. 서 론

인공지능[1]은 철학적으로 인간성이나 지성을 갖춘 존재, 혹은 시스템에 의해 만들어진 지능, 즉 인공적인 지능을 뜻한다. 인공지능은 컴퓨터 과학과 정보통신 기술의 발달로 많은 발전을 이루었는데, 자연언어처리(Natural Language Processing), 신경망(Neural Network) 구성, 복잡한 수학적 이론 증명 등 여러 방면에서 활발히 활용되고 있으며, 대학교나 IT기업들에 의해서 연구되고 있다. 또한 인공지능에 관련된 많은 오픈소스 소프트웨어들이 공개되면서 인공지능이 더 이상 전문가들만 다루는 영역에서 벗어나 일반인들도 조금만 학습하면 인공지능에 다가설 수 있는 환경이 되었다. 마이크로소프트의 CNTK(Computation Network Tool Kit)[2]와 DMTK(Distributed Machine learning Took Kit)[3], 구글의 텐서플로우(TensorFlow)[4], 페이스북의 토치(Torch)[5] 그리고 삼성전자의 딥러닝 플랫폼인 베레서(Veles)[6] 등이 그 예라 하겠다. 이 중 텐서플로우는 이미 많은 선행 연구와 사례를 통해 학습이 용이하고, 데스크탑, 노트북 그리고 모바일 디바이스 등 다양한 환경에서 사용이 가능하다는 점에서 처음 인공지능을 사용해 보려는 사람들에게 유용한 오픈소스 라이브러리이다. 또한 구글이라는 유력한 IT 기업이 만들어 공개하였다는 점에서 텐서플로우 프로젝트가 중도에 폐기되지 않을 것으로 예상되는 것도, 텐서플로우를 이용하는 또 하나의 장점이라 하겠다.

알파고를 기점으로 전국에서 인공지능 열풍이 불었고, 각종 학술모임에서 인공지능 강좌가 열렸으며, 정부에서도 인공지능 육성에 높은 관심을 보여 그간 상대적으로 소홀했던 인공지능 분야를 정책적으로 육성하고 지원 예산도 대폭 늘리겠다고 발표하였다[7]. 우리는 이러한 높은 관심을 받는 인공지능 영역으로 IT 분야를 아는 인공지능 비전문가가 뛰어 들었을 때, 어느 정도로 인공지능 기술을 활용할 수 있게 될지에 대해 관심을 가지게 되었다. 정말 일반인도 그 분야의 리소스들을 활용해 자신이 원하는 인공지능적 일들을 해 낼 수 있을지에 대한 호기심이었다. 우리는 해 볼 만한 일들을 여러 가지 생각해 보았으나, 너무 거창한 주제를 잡아 뛰어들기에는 활용 방안을 찾아내는 것도 힘들 수 있고, 결과적으로 실패할 수 있다고 보고, 한 달이라는 기간을 두고, 많은 선행 예제들을 구할 수 있는 텐서플로우를

※ 교신저자(Corresponding Author) : 권영미, 주소 : 대전광역시 유성구 대학로 79 충남대학교 공과대학 전파정보통신공학과, 전화: 042-821-6890, FAX: 042-823-5586, E-mail : ymkwon@cnu.ac.kr
* 충남대학교 공과대학 전파정보통신공학부
(E-mail : sys145@naver.com)

이용하여, 가장 쉽게 접근해 인공지능의 맛보기를 할 수 있는 영역의 어플리케이션에 도전해 보기로 하였다. 본고는 이와 같이 인공지능에 관심을 갖고 활용해 보려는 사람들에게 경험을 나누기 위해 작성되었다. 실험에 사용한 응용은 Word Embedding 이라는 것인데, 신문 기사 등을 학습시켜, 기사에 나왔던 단어들 간의 연관성 정도를 예측하게 하는 것이다.

이 실험에는 프로그래밍을 좋아하고, Python을 알며, 회사에서의 프로그래밍 경험도 있지만, 인공지능에 관해서는 수업도 들은 적 없는 IT 계열의 학부 4학년생이 참여하였다. 어떤 백그라운드의 사람인가에 따라 인공지능 진입에 대해 느끼는 어려움의 정도나 성공 확률도 달라질 것이다.

본고의 구성은 다음과 같다. 2장에서는 인공지능의 핵심이라고 할 수 있는 기계학습의 일반적 이론을 깊지 않은 수준에서 소개하고, 3장에서는 우리가 설정한 목표를 이루기 위해 텐서플로우를 사용한 과정을 소개한다. 그리고 4장에서 본 실험의 결론을 맺는다.

2. 기계학습(머신 러닝)의 과정: 기초

본 실험에 어떤 과정들이 필요한지 이해하기 위해 기계학습이 어떻게 동작하는지 먼저 간단하게 학습하였다. 그 내용은 다음과 같다.

2.1 데이터 수집과 데이터 정리

머신 러닝의 첫 번째 단계는 데이터 준비이다. 이는 기계가 어떤 데이터를 어떻게 학습할지를 결정하는 단계인데 이때는 방대한 양의 데이터를 수집하고 학습에 용이한 상태로 변경 및 정제하는 과정이 필요하다. 수집한 데이터를 어떻게 정리하는 가에 따라서 기계학습 알고리즘을 선택하는데 영향을 미치기 때문에 어떤 방법으로 기계학습을 진행할지를 생각하고 그에 맞게 데이터를 정리해야 한다.

2.2 기계학습 알고리즘 선택

기계학습의 방법은 크게 지도 학습(Supervised Learning)과 비지도 학습(Unsupervised Learning) 두 종류로 나눌 수 있다.

지도 학습은 데이터와 그에 해당하는 값인 레이블(label)로 구성된 데이터 셋을 활용하여 학습을 진행하는 방법이다. 각 데이터에 대응되는 레이블 값은 사람이 지정해야 한다는 비용이 들지만 정확도가 높은 데이터를 사용할 수 있는 장점이 있다. 지도학습을 사용하는 분야로는 분류(Classification)와 회귀(Regression)가 있다.

비지도 학습은 레이블이 되어 있지 않은 데이터를 컴퓨터가 스스로 학습하게 하는 방법이다. 비지도 학습에서는 데이터가 주어졌을 때 각 데이터들의 연관성을 비교해 몇 개의 묶음으로 나누는 군집화(Clustering)와 이 데이터들이 어떤 확률 분포를 가지고 있는지 추측하는 분포 추정(Underlying Probability Density Estimation)을 거쳐 스스로 학습해 나간다.

준비한 데이터 셋이 레이블이 있는 데이터 셋이면 지도 학습의 알고리즘을 선택하고 아니면 비지도 학습의 알고리즘을 선택하면 된다.

2.3 훈련과 테스트

데이터가 준비되고 기계학습 알고리즘을 선택했으면 그 알고리즘을 따라 훈련이 잘 되고 학습이 잘 되었는지 확인하는 작업이 필요하다.

지도 학습의 경우 데이터를 준비할 때, 훈련 데

이터(Traning Data)와 테스트 데이터(Test Data)로 나누어 준비한 후, 훈련 데이터로 알고리즘을 학습시키고 테스트 데이터들을 입력으로 넣었을 때 그 데이터에 대응되는 레이블 값을 얼마나 정확하게 예측하는지를 측정하여 학습의 정도를 판단한다.

비지도 학습인 경우 데이터에 레이블이 없기 때문에 테스트를 기계가 진행할 수 없어 테스트 데이터를 준비하는 것은 의미가 없다. 대신 비지도 학습의 결과인 클러스터링 결과 등을 보고, 실험자가 학습의 유효성을 판단해야 한다.

2.4 기계학습 심화

기계학습을 진행하여 얻은 결과를 확인하고 그 상태의 기계학습에 만족할지, 아니면 기계학습을 좀 더 심화시킬 것인지를 결정해야 한다. 기계학습이 응용에 적용되어 사람들에게 주어지는 결과의 형태는 수치화된 데이터일 수도 있고 보고서나 시각화한 이미지 등일 수 있다.

기계학습 심화 과정이 필요한 경우, 훈련 데이터와 테스트 데이터의 비율을 조정하거나 학습 단계의 횟수를 조정하는 등의 방법으로 데이터와 적용 알고리즘을 수정해서 정확도를 높이고, 이 과정을 반복하여 결과의 질을 높여 나간다.

3. 인공지능 적용: Word Embedding

서론에서 소개한 것처럼, 인공지능 영역 진입을 위한 도전 과제로 Word Embedding을 해결해 나갈 것이다. 본 장에서는 2장에 소개한 절차들을 실제 문제에 어떻게 적용해 나가는지를 기술한다.

Word Embedding의 목적은 유사한 단어일수록 가까운 거리에 위치하도록 각 단어에 해당하는 벡터 값을 찾는 것이다[8]. 사람이 간여하지 않는 비지도 학습을 통해 Word Embedding을 해결할 것이며, 이를 쉽게 해 나가기 위해 word2vec툴을 사용하였다. Word2vec은 Tomas Mikolov 외 구글 연구원들이 발표한 “Efficient Estimation of Word Representations in Vector Space” 논문[9]에서 제안한 Word Embedding방법으로, 비지도 학습으로 단어들의 위치에 따라 각 단어의 의미를 벡터형태로 표현하는 기법이다. Word2vec은 단어들을 수십~수백 차원의 벡터(vecotr)로 변환해 단어의 의미를 추론한다. 내부적으로 두 개의 신경망 모델을 이용해 단어를 학습하는데[8, 10], 큰 말뭉치를 입력으로 받아 각 단어들을 벡터공간에 위치시켜 거대한 다차원 공간을 생성하며, 같은 말뭉치를 공유하는 단어들끼리 가까운 벡터공간에 위치하게 한다.

텐서플로우를 사용한 word2vec 실습은 자료를 쉽게 구할 수 있어서 실제로 알고리즘 부분은 거의 수정하지 않고 사용할 수 있다. 본 실험에서는 word2vec에서 필요한 데이터 셋을 준비하여 기계학습을 시키고 그에 따른 Word Embedding의 결과를 도출하였다. 2장에서 설명한 기계학습의 과정을 본 실험에서 어떻게 적용하였는지 도식화하면 [그림 1]과 같다. 우선 분석에 관련되는 뉴스나 자료들을 수집하고, Python으로 작성된 형태소 분석기인 KoNLPy (“코엔엘파이”라 읽음)[11]를 사용해 불용어들을 제거하여 두 번째 단계의 데이터로 사용될 수 있도록 준비한다(단계 1). 그 후에 word2vec 프로그램을 이용해 Word Embedding 결과를 얻어낸다(단계 2). 그리고 마지막으로 얻어진 결과를 Python 그리기 툴 라이브러리인 metplotlib[12]를 이용해 시각화한다(단계 3).

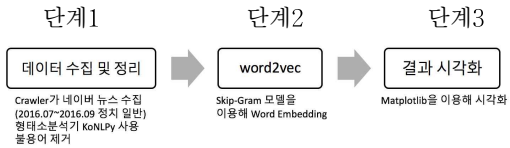

그림 1. Word Embedding 실험 절차

3.1 단계 0: 텐서플로우 설치

본 실험을 진행하기 전에 텐서플로우와 실험에 필요한 환경을 구축한다. 텐서플로우는 Unix계열의 운영체제인 Linux와 Mac OS X만 지원한다. 텐서플로우는 여러 가지 방법으로 설치가 가능한데 다음은 Python 패키지 소프트웨어인 pip를 이용해서 Mac OX S와 Python 2.7 버전을 사용하는 환경을 기준으로 설치하는 방법이다.

[그림 2]의 명령어를 터미널에서 입력한다. pip를 설치하고 Python 유틸리티 패키지인 six를 설치한다.

```
$ sudo easy_install pip
$ sudo easy_install --upgrade six
```

그림 2. pip 설치

텐서플로우를 설치할 환경에 따라 바이너리 주소를 [그림 3]과 같이 선택한다.

```
# Mac OS X, CPU only, Python 2.7:
$ export TF_BINARY_URL=https://storage.googleapis.com/tensorflow/mac/cpu/tensorflow-0.10.0-py2-none-any.whl

# Mac OS X, GPU enabled, Python 2.7:
$ export TF_BINARY_URL=https://storage.googleapis.com/tensorflow/mac/gpu/tensorflow-0.10.0-py2-none-any.whl
```

그림 3. 텐서플로우 바이너리 선택

다음은 pip를 이용해 텐서플로우를 설치한다.

```
# Python 2
$ sudo pip install --upgrade $TF_BINARY_URL
```

그림 4. 텐서플로우 설치

텐서플로우 설치가 완료 되었다면 [그림 5]와 같이 Python에서 텐서플로우를 import하여 하위 메소드들이 잘 실행되는지 확인한다. [그림 5]는 설치가 잘 되어 동작하는 결과의 화면이다.

```
$ python
>>> import tensorflow as tf
>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()
>>> print(sess.run(hello))
Hello, TensorFlow!
>>> a = tf.constant(10)
>>> b = tf.constant(32)
>>> print(sess.run(a + b))
42
```

그림 5. 텐서플로우 설치 확인

마지막으로 실험에 필요한 형태소 분석기인 KoNLPy와 시각화 라이브러리인 matplotlib를 설치한다.

```
$ pip install konlpy matplotlib
```

그림 6. KoNLPy, matploblib 설치

3.2 단계 1: 데이터 수집 및 정리

Word Embedding에 사용할 기본 데이터를 본 실험에서는 인터넷에서 쉽게 구할 수 있는 뉴스 기사들로 사용했다. Crawler가 2016년 7월 ~ 9월 네이버 정치 일반 부분의 62251건의 뉴스 본문 내용들을 수집하였고 그 후에 [그림 7]와 같이 Python 기반의 한글 형태소 분석기인 KoNLPy를 통해 텍스트로 저장된 뉴스에서 필요 없는 단어나 특수문자 등 불용어를 제거하였다. KoNLPy를 사용하는 방법은 [그림 8]에 코드와 함께 간단하게 설명하였다. text에 데이터 소스 문장을 입력하고 필터(filter)로 나열한 형태소들을 제외한 단어들을 리스트로 만들게 하는 예제이다. 본 예제에서 필터는 '~가', '~는'과 같은 조사나 마침표 등을 없애도록 하였다.

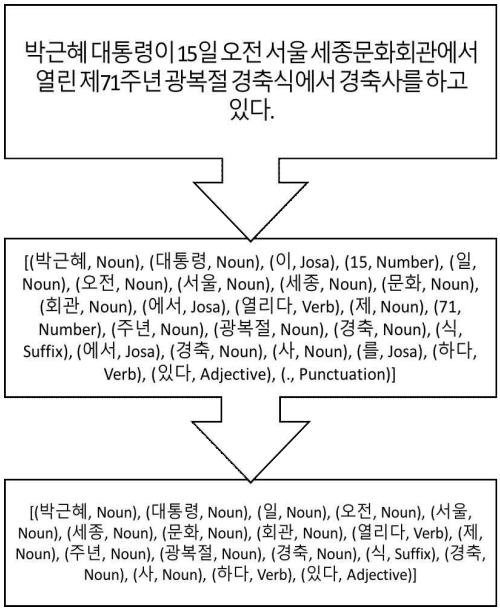

그림 7. 데이터 정리

```
# 형태소 분석기를 불러온다.
from konlpy.tag import Twitter
twitter = Twitter()

filter_list = ['Josa', 'Koreanparticle', 'Punctuation', 'Number', 'Foreign']
text = "박근혜 대통령이 15일 오전 서울 세종 문화회관에서 열린 제71주년 광복절 경축식에서 경축사를 하고 있다."
word_list = twitter.pos(text.decode('utf-8'), norm=True, stem=True)
result = [word[0] for word in word_list if word[1] not in filter_list]
#  ["박근혜","대통령","일","오전","서울","세종","문화","회관","열리다","제","주년","광복절","경축","식","경축","사","하다","있다"]
```

그림 8. KoNLPy

3.3 단계 2: word2vec을 이용한 Word Embedding

word2vec에서 사용하는 모델에는 CBOW (Continuous Bag-Of-Words)와 Skip-Gram 두 가지가 있는데, 본 실험에서는 Skip-Gram 모델을 사용하였다. Skip-Gram 모델은 주제 단어의 주변 단어들을 데이터 셋으로 구성한 후 새로운 단어가 주어질 때 그 주변 단어들을 예측하는 학습을 진행한다. 그리고 학습한 결과를 Word Embedding에 이용한다.

실험은 텐서플로우 프로젝트 홈페이지[13]에 오픈되어 있는 word2vec을 위한 튜토리얼 소스 코드를 활용하였다. 이 코드를 기반으로 이전 과정에서 준비한 데이터를 입력하였다. 본 실험을 위해 [그림 9]와 같이 한글을 위한 설정을 해주고

입력 데이터를 준비한 뉴스 데이터로 바꿔준다.

```
# 한글을 위한 폰트를 추가한다.
from matplotlib import rc
rc('font',family='AppleGothic')

# 51 번째 줄에 read_data 함수를 수정한다.
def read_data(filename):
  with open('input_data','rb') as f:
    data = f.read().split()
  return data
```

그림 9. 코드 수정

터미널에서 word2vec Python 코드를 실행하는 명령어를 입력한다.

```
$ python word2vec_basic.py
```

그림 10. word2vec 실행

3.4 결과 시각화

3.3절 word2vec의 실행 결과로 단어들의 벡터 값을 담은 배열 형태의 데이터를 얻을 수 있는데 이 배열 값들을 눈으로 들여다보면서 유의미한 정보를 찾아내는 것은 매우 힘든 일이다. 그래서 word2vec의 결과로 얻어진 단어 벡터들을 이미지로 형상화한다. 각 단어들의 연관성 정도에 따라 가깝거나 멀리 배치된 벡터공간의 이미지를 얻으면 결과를 분석하기에 훨씬 용이하다. 이를 위해 Python 시각화 라이브러리 matplotlib를 이용하는데, [그림 11]의 코드가 이 라이브러리를 이용하는 코드 부분이다. 이 과정을 진행하고 나면 tsne.png 라는 이름의 이미지 파일이 생성되는데 [그림 12]과 같은 형태의 이미지이다. [그림 12]를 보면 '한국' 이라는 단어 근처에 대한민국, 중국, 미국, 러시아 등 연관 있는 단어들이 모여 있어 이들 국가들이 정치 일반 분야에서 연관성이 서로 높다고 텐서플로우가 분석했음을 확인할 수 있다.

```
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
from matplotlib import rc
rc('font',family='AppleGothic')
# from matplotlib import fo
tsne = TSNE(perplexity=30,
n_components=2, init='pca', n_iter=5000)
plot_only = 500
low_dim_embs =
tsne.fit_transform(final_embeddings[:plot_o
nly,:])
labels = [reverse_dictionary[i] for i in
xrange(plot_only)]

plot_with_labels(low_dim_embs, labels)
```

그림 11. 시각화 코드

4. 결론

구글의 알파고, 마이크로소프트의 코타나, 아이폰의 시리 등 인공지능 관련 기술은 이제 주변을 조금만 둘러보면 쉽게 발견할 수 있게 되었다. 그만큼 인공지능이 생활 속의 일부분으로 점점 더 자리 잡아 가고 있음을 알 수 있다. 그에 따라 인공지능에 관련된 기술들도 발전해 비전문가들도 인공지능분야에 쉽게 접근할 수 있는 환경이 주어졌다고 애기되어진다. 본고에서는 IT 계열의 사람들이 인공지능 분야의 결과물을 활용하려고 할 때 그 진입장벽이 얼마나 쉬운지 또는 어려운지를 실험해 보았다.

짧은 시간 동안 쉽게 결과를 얻어 이 분야를 체험하기 위해 우선 많은 자료를 구할 수 있는

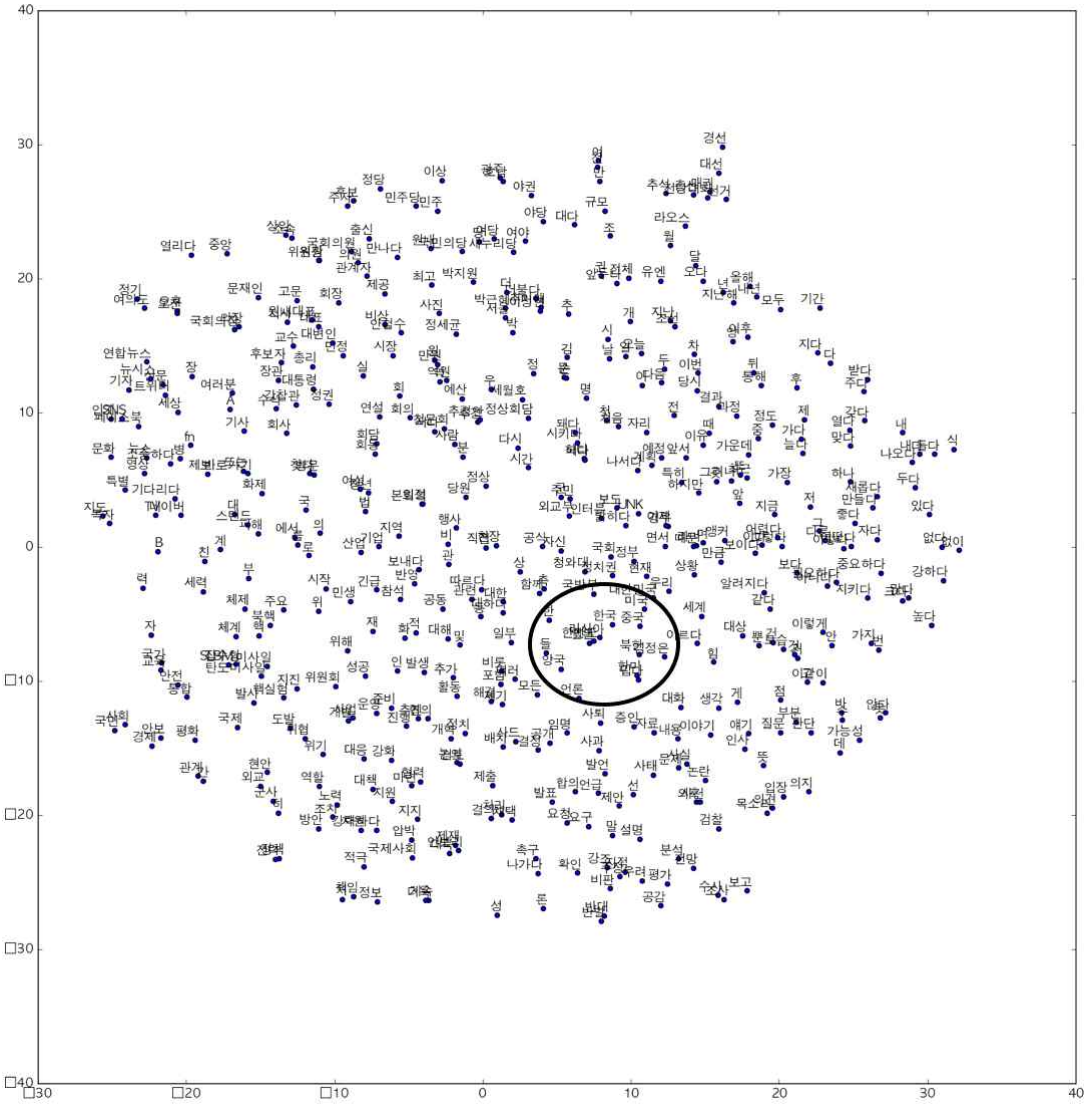

그림 12. Word Embedding 시각화

분야를 시도해 보았는데, 어떤 알고리즘을 사용해야 하는지 판단할 수 있어야 했고, 데이터를 어떻게 만들어내야 하는지에 대한 지식을 쌓은 후 필요한 데이터를 수집해 내야 했다.

텐서플로우를 사용하여 기계학습을 진행해본 결과 이미 충분히 연구되고 개발된 분야의 인공지능은 사용자가 데이터를 준비하고 환경만 구성한다면 손쉽게 인공지능을 다룰 수 있다는 것을 알 수 있었다. 수많은 오픈소스 소프트웨어들과 그에 관련된 데이터들이 공공데이터로 오픈되어 있기 때문이다. 하지만 기계학습을 이해하고 인공지능을 구축하려면 수학적 지식과 프로그래밍 지식이 요구된다. 기계학습은 수학적으로 증명된 공식들을 컴퓨터 과학에 접목시켜 사용하는 것이

기 때문이다.

본 실험에 IT 계열의 학부생이 참여해본 결과 기계학습에 대한 개념을 깊이 있게 학습하는 데에는 어려움이 있었으며, 필요한 부분만 학습을 진행하는 데에도 시간이 소요되었다. 또한 데이터를 수집하고 정리하는 과정은 직접 해야 하기 때문에 이 과정을 어떤 방법으로 어떻게 처리해야 하는지 고민하는 시간이 필요했다.

IT 계열의 사람에 비해 비IT 계열의 일반인들이라면 인공지능을 활용하기에 더욱 시간이 많이 걸릴 것이라 생각되며, IT 계열의 사람이라 하더라도 원하는 분야의 인공지능적 실험을 위해서는 많은 고민과 연구가 필요한 것으로 보인다. 보다 많은 사람들이 이 인공지능 분야에 전문적이든 비전문적이든 계속 관심을 갖고 활용하며, 자신만의 코드와 라이브러리들을 풍성하게 생성하고 공유해 나갈 때 전반적인 인공지능 생활화는 더욱 가속화 될 것이다.

참 고 문 헌

[1] 인공지능 - 위키백과, 우리 모두의 백과사전. https://ko.wikipedia.org/wiki/인공지능 (accessed Sep., 18, 2016).

[2] CNTK - Computational Network Toolkit. https://cntk.ai/ (accessed Sep., 18, 2016).

[3] Distributed Machine Learning Toolkit. http://www.dmtk.io/ (accessed Sep., 18, 2016).

[4] TensorFlow – an Open Source Software Library for Machine Intelligence. https://www.tensorflow.org/ (accessed Sep., 18, 2016).

[5] Torch - Scientific computing for LuaJIT. http://torch.ch/ (accessed Sep., 18, 2016).

[6] VELES - Distributed platform for rapid Deep learning application development. https://velesnet.ml/ (accessed Sep., 18, 2016).

[7] MTN 2016년 3월 16일 인터넷 뉴스, http://news.mtn.co.kr/issue/viewer.mtn?gidx=2016031615445183333 (accessed Sep., 30, 2016).

[8] Word2Vec 소개, http://deeplearning4j.org/kr/kr-word2vec (accessed Sep., 18, 2016).

[9] Mikolov, T., Chen, K., Corrado, G., & Dean, J. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781. (2013).

[10] 빅데이터 실전 튜토리얼, https://jiafei427.wordpress.com/2015/08/강좌-빅데이터-실전-튜토리얼

[11] Python 한국어 NLP, http://konlpy-ko.readthedocs.io/ko/v0.4.3/ (accessed Sep., 30, 2016).

[12] metplotlib, http://matplotlib.org/ (accessed Sep.18, 2016)

[13] word2vec_basic.py 소스 코드, https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/tutorials/word2vec/word2vec_basic.py (accessed Sep., 18, 2016).

송 윤 섭

• 현 재 충남대학교 정보통신공학과 재학 중
• 관심분야: 빅 데이터 분석, 분산처리 시스템

권 영 미

• 1986년 서울대학교, 컴퓨터공학과 학사
• 1988년 서울대학교, 컴퓨터공학과 석사
• 1996년 서울대학교, 컴퓨터공학과 박사
• 1993년~1995년 ETRI 연구원
• 1996년~2002년 목원대학교 컴퓨터공학과 조교수
• 2006년~2007년 Indian Statistical Institute 객원연구원
• 2002년~현재 충남대학교 전파정보통신공학과 교수
• 관심분야 : Internet Protocols, WSN, Embedded System, Cloud computing