이미지 기반의 한국어 텍스트 이해하기

Korean text Understanding from Scratch

박건후, 심재빈
(Guenhoo Park, Jaebeen Sim)
지도교수 : 최원익(Wonik Choi)

Abstract: This paper proposes a method of applying deep - learning method after converting Korean text into image to understand and classify Korean text. The conventional method requires a predefined word dictionary and a parser in order to understand a specific language of the machine, so that unnecessary expense is increased and thus it is limited to be used universally. In order to solve these problems, this paper suggests a way to understand Korean without help of word dictionary and parser. The proposed method is to quantize Korean text to image and train it in network to understand and classify meaning. Specifically, Korean text is imaged by quantization in three different ways and then learned by CNN (Convolutional Neural Network) network. As a result, we show that Internet articles can be classified to five categories and show the performance difference according to the quantization methods and the networks.

Keywords: classify Korean text., Quantization, CNN

Ⅰ. 서 론

기존의 텍스트 딥러닝(deep learning)은 대부분 Bag of Words[1]와 word2vec[2]을 기반하고 있다. 두 기술은 높은 정확도를 제공하지만 미리 생성해 놓은 단어사전과 구문분석기 등과 같이 텍스트를 단어, 문장, 문단 단위로 분해하고 그 의미를 해석하는 단계가 필요로 했다. 하지만, 이러한 접근 방법은 단어 사전에 없는 단어나 해석할 수 없는 구문들이 텍스트에 존재하면 정확도가 크게 떨어지는 단점이 있다. 더욱이 단어사전과 구문분석기의 생성 비용과 그 이후의 유지비용은 막대하다. 따라서, 본 논문에서는 단어의 의미와 구문의 구조 등을 파악하는 것이 아니라 문자 자체를 양자화(quantization)하여 문장을 이미지화한다. 그리고 이렇게 생성된 이미지 셋을 CNN을 사용하여 텍스트의 의미를 이해하고자 한다. [3]에서는 영문과 중문을 대상으로 이러한 방식을 실험하였고, 구체적으로는 Yahoo!의 Answers Topic Classification을 한 Train의 결과가 기존의 Bag of words보다 약 10% 정도, word2vec보다 약 20% 정도 높은 정확도를 보여준다. 이 기법의 동작하는 방식은 그림 1과 같다.[3]

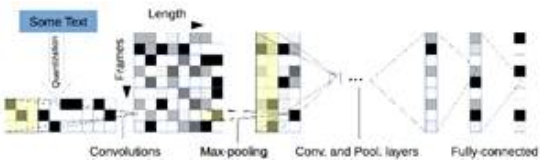

그림 1. “Text Understanding from Scratch”[3]의 개요

하지만 [3]에서 텍스트를 양자화하기 위해 로마자로 변환하는 기법을 제시하고 있지만 이 양자화 기법은 한글의 경우는 적용되기 어렵다. 이는 한국어가 초성, 중성, 종성로 구성된 언어라는 특성 때문이다. 따라서, 본 논문에서는 한국어 텍스트를 양자화하는 방법 CNN(Convolutional Neural Network) 적용한 3가지를 제안한다.

CNN은 합성곱 신경망(Convolutional Neural Network)으로 최소한의 전처리를 사용하도록 설계된 다 계층 퍼셉트론의 한 종류이다. 여러 개의 합성곱 계층과 일반적인 인공신경망 계층들로 이루어져 있으며, 가중치와 통합 계층들을 추가로 활용한다. CNN은 2차원 구조의 입력 데이터를 충분히 활용 할 수 있게 되어 영상, 음성 분야 모두에서 좋은 성능을 보여준다. [4]

양자화 대상은 그림 2와 같이 한글 49자와 더불어 자주 쓰이는 숫자, 특수문자, 공백을 포함한 46자의 다른 문자를 포함하여 총 95자로 구성한다. 여기에 [3]과 달리 공백을 포함한 이유는 한글에서 띄워 쓰기에 따라 뜻이 완전히 달라지는 경우가 있기 때문에 이를 방지하기 위함이다.

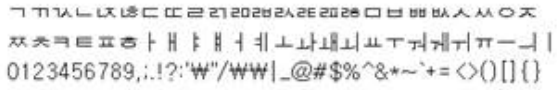

그림 2. 한글 양자화 대상

러닝을 하였을 때 정확도에 대하여 충분한 신뢰성을 확보하기 위해서 k-fold cross validation[5]을 해준다. 본 논문에서는 6-fold cross validation을 적용한다.

II. 시스템 모델 및 본문

2.1 한글 분류 방법

본 논문에서는 학습을 시키기 위해서 torch7을 사용한다. 한글을 양자화하기 위해서 먼저 초성과 중성, 종성을 분류할 필요가 있기 때문에 유니코드를 이용해서 분류한다.

분류를 하게 된다면 “한국어”는 “ㅎㅏㄴㄱㅜㄱㅇㅓ” 이라는 모양으로 길이 8로 나오게 된다.

```
-- 유니코드 표에 따른 초성 중성 종성 구분법
function HangulSeparator (str)
   local output = ""
   for i=1, utf8.len(str) do
      if utf8.byte(str, i) >= 0xAC00 and utf8.byte(str, i) <= 0xD7A3 then
         local cHex = utf8.byte(str, i) - 0xAC00
         local jong = cHex % 28
         local jung = ((cHex - jong) / 28) % 21
         local cho = (((cHex - jong) / 28) - jung) / 21

         output = output .. rCho[cho+1] .. rJung[jung+1] .. rJong[jong+1]
      else
         output = output .. utf8.sub(str,i,i)
      end
   end

   return output
end
```

그림 3. 유니코드 표에 따른 분류

2.2 한글 양자화 방법

본 논문에서는 다음과 같은 3가지의 다른 양자화 방법을 제안한다. 앞서 초성 중성 종성 분류 방법을 적용한 한 character당 한 length를 할당하는데 그 이유는 한글의 경우 한 글자당 한 length를 할당하면 초성과 종성이 겹치는 경우가 발생하기 때문이다.

1) 첫 번째 양자화 방법은 한글 49자와 숫자, 특수문자, 공백을 포함한 46자를 1열로 나열하여 대응되는 부분을 검게 표시(mark)하는 방법이다. 이를 모든 문자에 적용하여 이어 붙인다. 이 방법을 통해 95의 index에 대응되는 이미지를 얻게 된다.

예) “글 자”라고 입력하게 되면 초성 중성 종성 분리에 의하여 length 6의 “ㄱㅡㄹ ㅈㅏ”으로 바뀌어지게 되고 방법 1에 의해 대응되는 Index는 표 1과 같다.

표 1. “글자”를 방법 1에 의해 양자화 할 경우 대응되는 Index

해당 문자	Index
ㄱ	1
ㅡ	47
ㄹ	9
	95
ㅈ	24
ㅏ	31

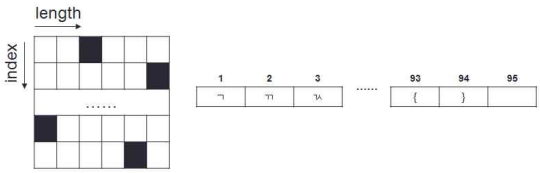

그림 4. 방법 1 : 직관적으로 양자화하는 방법

```
function Data:stringToTensor(str, l, input, p)
   local s = utf8.lower(str)
   local l = l or utf8.len(s)
   local t = input or torch.Tensor(utf8.len(self.hangul), l)
   t:zero()
   local j = 1
   for i = utf8.len(s), math.max(utf8.len(s) - l + 1, 1), -1 do
      if self.dict[utf8.sub(s,j,j)] then
         t[self.dict[utf8.sub(s,j,j)]][utf8.len(s) - i + 1] = 1
      end
      j = j + 1
   end
   return t
end
```

그림 5. 방법 1로 양자화하는 코드

2) 두 번째 양자화 방법은 맨 앞 구분 인자를 두고, 양자화 대상인 95자를 한글 49자와 특수문자 및 숫자 46자로 분리하여 양자화를 한다. 양자화하는 문자가 한글이라면 구분 인자를 공백으로 두고 한글 49자 중 대응되는 부분을 검게 표시한다. 특수문자 및 숫자라면 구분 인자를 검게 표시하고 추가로 46자 중 대응되는 부분을 검게 표시한다. 이 방법을 통해 최대 50의 길이를 가지게 된다.

예) “글 자”를 방법 2로 양자화하게 되면 대응되는 Index는 표 2와 같다.

표 2. “글자”를 방법 2에 의해 양자화 할 경우 대응되는 Index

해당 글자	Index
ㄱ	2
ㅡ	48
ㄹ	10
	1, 47
ㅈ	25
ㅏ	32

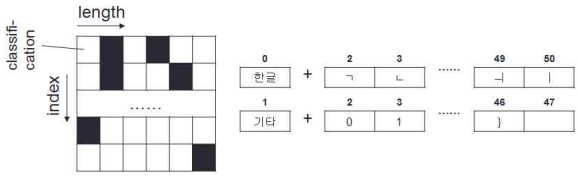

그림 6. 방법 2 : 구분자를 두어 양자화하는 방법

```
function Data:stringToTensor(str, l, input, p)
   local s = utf8.lower(str)
   local l = l or utf8.len(s)
   local t = input or torch.Tensor(self.hangul_length + 1, l)
   t:zero()
   local j = 1
   for i = utf8.len(s), math.max(utf8.len(s) - l + 1, 1), -1 do
      if self.dict[utf8.sub(s,j,j)] then
         local charLocation = self.dict[utf8.sub(s,j,j)]
         local realLocation = charLocation % self.hangul_length
         if(realLocation == 0) then
            realLocation = self.hangul_length
         end
         t[1][utf8.len(s) - i + 1] = math.floor(charLocation / self.hangul_length)
         t[realLocation + 1][utf8.len(s) - i + 1] = 1
      end
      j = j + 1
   end
   return t
end
```

그림 7. 방법 2로 양자화하는 코드

3) 세 번째 양자화 방법은 첫 번째 양자화 방법과 동일한 상태에서 대응되는 인덱스(index)를 이진법으로 변환하여 1에 대응되는 부분을 검게 표시한다. 이 방법을 통해 7의 길이를 가지는 이미지를 얻게 된다.

예) “글 자”를 방법 3로 양자화하게 되면 대응되는 Index는 표 3와 같다.

표 3. “글자”를 방법 3에 의해 양자화 할 경우 대응되는 Index

해당 글자	Index	해당 2진수
ㄱ	7	0000001
ㅡ	2, 4, 5, 6, 7	0101111
ㄹ	4, 7	0001001
	1, 3, 4, 5, 6, 7	1011111
ㅈ	3, 4	0011000
ㅏ	3, 4, 5, 6, 7	0011111

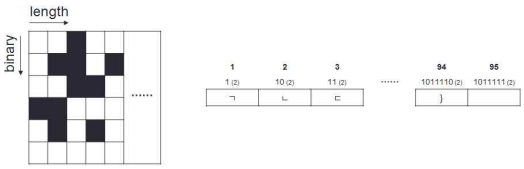

그림 8. 방법 3 : 순서를 2진법화 하여 index를 기준으로 양자화하는 방법

```
function Data:stringToTensor(str, l, input, p)
   local s = utf8.lower(str)
   local l = l or utf8.len(s)
   local t = input or torch.Tensor(self.binary_length, l)
   t:zero()
   local j = 1
   for i = utf8.len(s), math.max(utf8.len(s) - l + 1, 1), -1 do
      if self.dict[utf8.sub(s,j,j)] then
         local p = 1
         local num = self.dict[utf8.sub(s,j,j)]
         while num > 0 do
            t[p][utf8.len(s) - i + 1] = num % 2
            num = math.floor(num / 2)
            p = p + 1
         end
      end
      j = j + 1
   end
   return t
end
```

그림 9. 방법 3로 양자화하는 코드

예를 들어, “한글”이란 문자가 3가지 양자화 방법을 거치게 되면 그림 6과 같은 이미지가 만들어진다. 이러한 과정을 거친 이미지 셋을 학습에 사용하게 되는데, 각 방법에 따라 나온 결과에서 얻어진 이미지 크기가 상대적으로 균일한 차이를 이룬다. 각 방법에 따라 양자화 된 이미지를 CNN에서 가중치를 인식하게 되는데 가중치를 시각화한 모습이 그림 10.과 같다.

그림 10. 가중치를 시각화한 모습

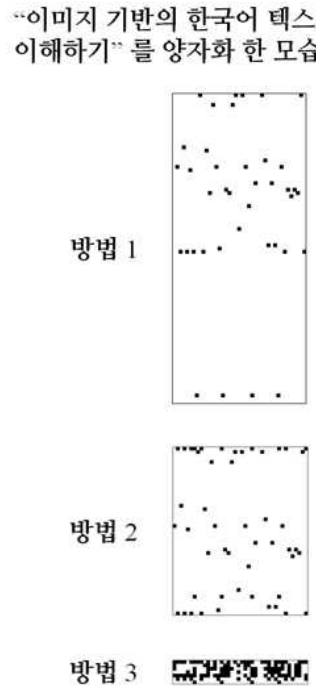

그림 11. "이미지 기반의 한국어 텍스트 이해하기"를 양자화한 이미지

2.3 데이터 셋

본 논문에서는 제안 방법의 성능을 보이기 위한 데이터 셋으로서 조선일보 2006년 1월부터 2017년 02월까지 기사의 첫 문단으로 구성하였다. 5가지의 카테고리(정치, 사회, 경제, 스포츠, 엔터테이먼트)를 대상으로 Crawling을 하였다. Crawling은 Node.js와 Cheerio library를 사용하였다.

표 4. 실험을 위해 Crawling된 데이터

카테고리	Total
정치	383,428
사회	87,446
경제	277,663
스포츠	198,352
엔터테이먼트	90,161

2.4 데이터 필터링

첫 문단의 데이터를 Crawling하였을 때 다양한 자료가 모이게 되는데 기사의 내용과는 상관없는 기자의 이름과 신문사의 이름은 학습의 정확도를 위해서 삭제하였다. 처음에 걸러지지 않았을 때 높은 정확도가 나왔지만 기자의 이름을 특징점으로 잡는 경우가 있어서 정확한 트레이닝이 되지 않았다. 따라서 이를 방지하기 위해서 기사의 내용과는 상관없는 내용을 모두 Query를 통하여 제거하였다.

2.5 사용한 네트워크

사용한 CNN 네트워크는 총 7 layers로 구성하였다. 입력되는 Input은 데이터의 첫 문단으로 너무 큰 길이를 방지하기 위해서 1014자의 길이로 제한하였다. 만약 정해진 길이를 넘는다면 문단의 뒤부터 1014자를 읽어 오게 구성하였다. 그 이유는 문장의 뒤가 앞보다 상대적으로 중요한 문장이 나오는 문장 형식을 취하고 있기 때문이다. 그리고 너무 짧은 경우에도 학습이 제대로 되지 않기 때문에 최소한 20글자는 넘은 것을 사용하였다.

III. 결과

첫 번째에는 트레이닝 데이터 셋을 각 카테고리 별로 3만개, 테스트 데이터 셋을 6천개 총 15만개, 3만개를 하였다. 두 번째에는 트레이닝 데이터 셋을 각 카테고리 별로 7만개, 테스트 데이터 셋은 1만 4천개 총 35만개, 7만 5천개를 하였다. 각 데이터는 랜덤으로 카테고리 별 개수를 맞췄으며 학습은 신뢰성을 위해서 6-fold cross validation을 실행하였다.

1) 첫 번째 실험

두 번째 실험에서는 카테고리 별 각 3만개, 총 15만 개의 트레이닝 데이터 셋을 사용하였고 테스트 데이터 셋으로는 각 카테고리 별 6천개, 총 3만개로 테스트를 수행하였으며, 그 결과는 표 5와 같다. 신뢰성을 위해서 6-fold cross validation을 하였다.

표 5. 첫 번째 데이터를 대상으로 카테고리화 한 결과

	Train	Test
방법 1	99.2%	86.8%
방법 2	99.5%	87.5%
방법 3	99.4%	85.8%

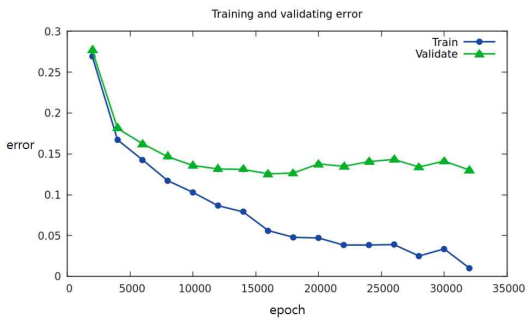

그림 12. 방법 1 - 각 카테고리 별 3만개 훈련시킨 결과

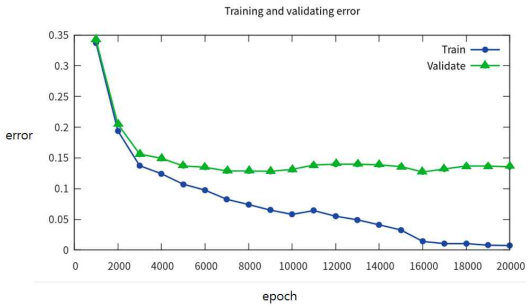

그림 13. 방법 2 - 각 카테고리 별 3만개 훈련시킨 결과

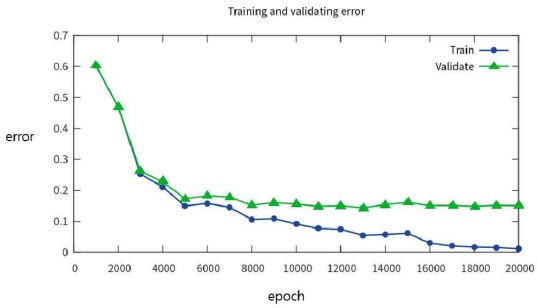

그림 14. 방법 3 - 각 카테고리 별 3만개 훈련시킨 결과

2) 두 번째 실험

두 번째 실험에서는 카테고리 별 각 7만개, 총 35만 개의 트레이닝 데이터 셋을 사용하였고 테스트 데이터 셋으로는 각 카테고리 별 1만 4천개, 총 7만개로 테스트를 수행하였으며, 그 결과는 표 6와 같다. 신뢰성을 위해서 6-fold cross validation을 하였다.

표 6. 두 번째 데이터를 대상으로 카테고리화 한 결과

	Train	Test
방법 1	96.5%	88.7%
방법 2	96.3%	89.5%
방법 3	94.5%	88.5%

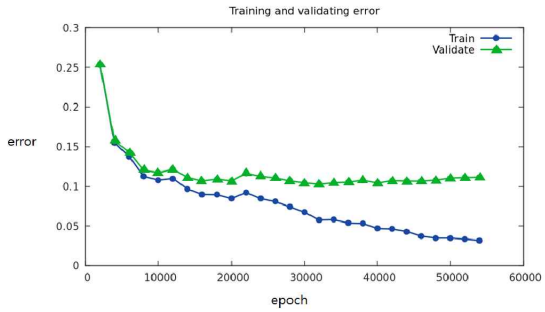

그림 15. 방법 1 - 각 카테고리 별 7만개 훈련시킨 결과

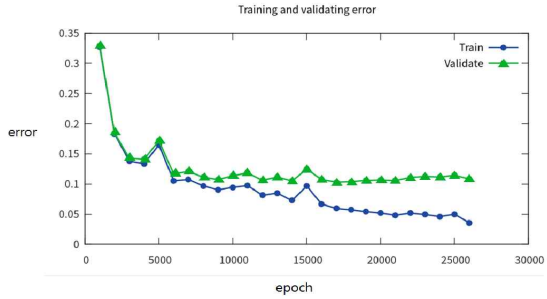

그림 16. 방법 2 - 각 카테고리 별 7만개 훈련시킨 결과

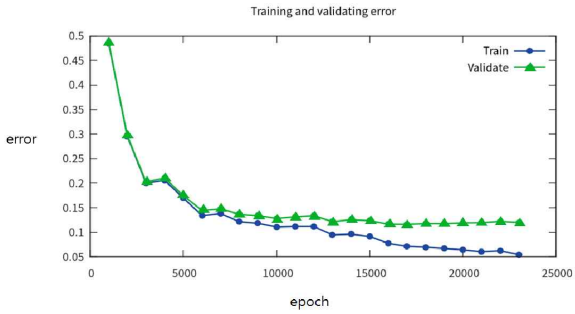

그림 17. 방법 3 - 각 카테고리 별 7만개 훈련시킨 결과

실험 1, 실험 2 모두 6-fold cross validation을 통하여 데이터의 신뢰성을 확보하였습니다. 두 실험 모두 정확도를 기준으로 높은 순서대로 방법2, 방법1, 방법3임을 보이고 있다.

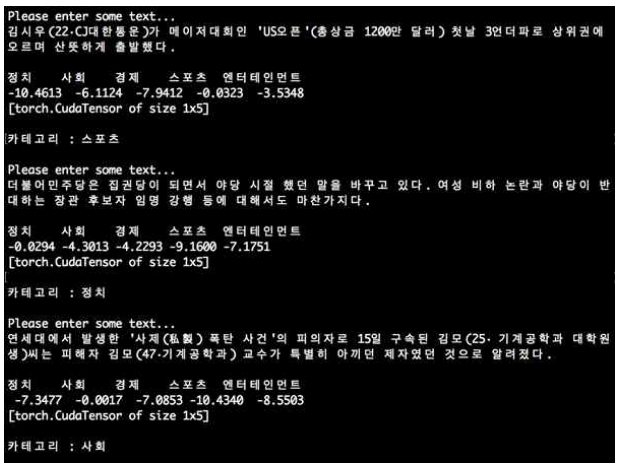

그림 18. 학습 시킨 데이터를 기반으로 한 테스트

그림 18은 7만개를 학습 시킨 방법 2의 트레이닝 된 데이터 셋에 2017.06.16 데이터를 입력시킨 모습이다. 학습시킨 데이터의 기간과는 4달정도 차이가 있지만 상당히 높은 정확도를 보여주고 있다. 기존의 데이터가 있다면 새로운 데이터를 구분해내는데 있어서도 문제없다는 것을 보여준다.

IV. 결론 및 향후 제시 방향

양자화 방식을 통해서 각 양자화 방식마다 학습 중 특징적인 모습을 보여 주었다. 그리고 최종 결과에서 방법에 따라서 정확도의 차이를 보이는데 이 중 방법 2가 가장 높은 정확도를 보여주므로 이를 가장 훌륭한 방법이라고 본 논문은 제시한다.
글자를 양자화할 때 길이가 직관적인 것 보다 일정한 방법을 통해서 줄여주는 것이 학습 중 특징점을 잡기에 유리하지만 방법3와 같이 너무 길이가 짧게 된다면 오히려 특징점을 잡기 어렵다는 것을 보여준다.
아울러, 본 논문에서는 인터넷 기사를 대상으로 하였으나 [3]에서와 같이 분류가 가능한 사전, 댓글 등에 대해서도 적용이 가능할 것이며 높은 정확도를 얻을 수 있을 것이다. 이를 통해서 기존의 막대한 생성비용과 유지비용이 소모되는 단어사전이나 구문분석기가 필요하지 않고 한글을 높은 정확도로 분류할 수 있다.
또한 본 실험에서 데이터 셋을 늘려감에 따라서 전체적인 학습의 정확도가 올라가는 모습을 보이는데 충분한 데이터가 있다면 더욱 올라갈 것으로 기대된다.

[참고문헌]

[1] Sivic, Josef, "Efficient visual search of videos cast as text retrieval", IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 31, NO. 4. IEEE. pp. 591-605, 2009.
[2] Mikolov, Tomas, Le, Quoc V, and Sutskever, Ilya., “Exploiting similarities among languages for machine translation,” arXiv preprint arXiv:1309.4168, 2013
[3] Xiang Zhang, Yann LeCun “Text Understanding from Scratch,” arXiv:1502.01710, Feb 2015.
[4] 김지원, 표현아, 하정우, 이찬규, 김정희, “다양한 딥러닝 알고리즘과 활용,” 『정보과학회지 제33권 제8호』, 한국정보과학회, 2015.
[5] Juan Diego Rodriguez, Aritz Pe´rez, and Jose Antonio Lozano, “Sensitivity Analysis of k-Fold Cross Validation in Prediction Error Estimation, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 32, NO. 3, MARCH 2010