TEXT 2
SCORE

SHIM JONG HWA
HAN SEOK HEE
KIM GUN HO

Overview

쎄쎄쎄 님의 의견

STUDENT's HONEST REVIEWS

작성일 2019-02-18 19:05:53

희곡만 읽고 가서 조별로 이야기하면 끝... 조별발표 하나 있는데 그냥 별로 신경은 안쓰시는 느낌...? 이번에 평가가 과제 2개에 발표 하나, 출석이어서 굉장히 편했습니다

Crawling

Data Preprocessing

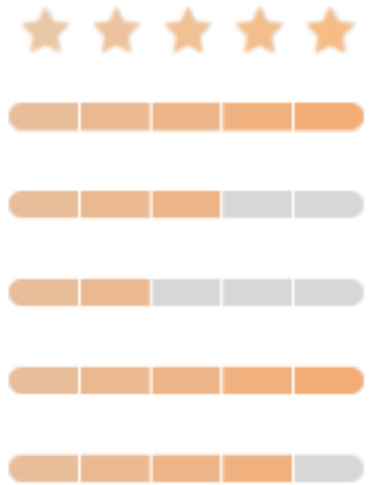
Word Embedding

Model

Result Analysis

Feedback

Text
Embedding vector
Score
"Feedback"



[Presentation Procedure]

0. Dataset

1. Crawling

2. Data Preprocessing

3. Word Embedding

4. Classification Model

5. Result Analysis

0. Dataset –kuklue-

Why?

1). Anonymous

2). No access allowed to professors

3). Review texts with 5 scores

1. Crawling - (2)

2014 ~ 2018 "5"years of
Kuklue Review Data

136,133 pairs

58.8MB of CSV file

Size	Kind
9.3 MB	CSV Document
21.6 MB	CSV Document
27.9 MB	CSV Document

Distribution

90% => "Training Set"

10% => "Testing Set"

2. Data Preprocessing - (1)

[1]. Special Symbols, English Omitted

ex). ! ? , . -, = / a,b,c,d ..

```
hangul = re.compile('[^ ㄱ-ㅣ가-힣]+')
sentence = hangul.sub('',r1[2].value)
```

2. Data Preprocessing - (2)

[2]. Omitted Korean's meaningless
=> self-standing Ja-eums & Mo-eums

ex). ㄴㅈ, ㅈㄱㄴ, ㅋㅋㅋ, ㅎ, ㅗㅜㅑ, ㅇㅅㅇ => Omitted

```
sentence = hangul.sub('',r1[2].value)
for i in sentence:
    if i in test.CHOSUNG_LIST or i in test.JUNGSUNG_LIST or i in test.a:
        sentence = sentence.replace(i, '')
```

2. Data Preprocessing - (3) overview

과제는 "항상 꼭!!!" 제 시간에 내야 됩니다. ㅋㅋ(하나라도 안내면 ㅎㅎ) wow! 바이

Original text

Filtering English & Special Symbols

과제는 항상 꼭 제 시간에 내야 됩니다 ㅋㅋ하나라도 안내면 ㅎㅎ 바이

First-Filtered text

Filtering independent Ja-eums & Mo-eums

과제는 항상 꼭 제 시간에 내야 됩니다. 하나라도 안내면 바이

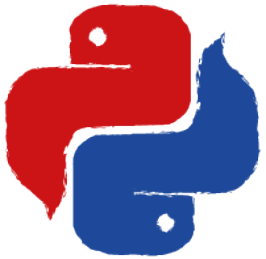
Final text

3. Word Embedding - (1) Classical method - 1

[1]. Tokenizing to Morphemes (Konlpy)

아버지가방에들어가신다

Hannanum	Kkma	Komoran	Mecab	Twitter
아버지가방에 들어가 / N	아버지 / NNG	아버지가방에 들어가신다 / NNP	아버지 / NNG	아버지 / Noun
이 / J	가방 / NNG		가 / JKS	가방 / Noun
시ㄴ다 / E	에 / JKM		방 / NNG	에 / Josa
	들어가 / VV		에 / JKB	들어가신 / Verb
	시 / EPH		들어가 / VV	다 / Eomi
	ㄴ다 / EFN		신다 / EP+EC	



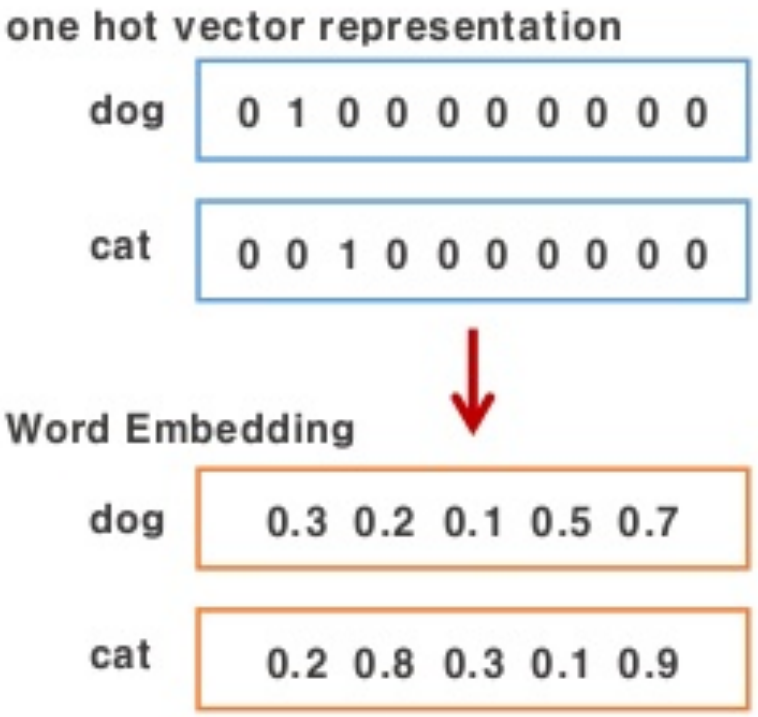
KoNLPy

3. Word Embedding - (1) Classical method - 2

[2]. POS Tagging

```
In [4]: kkma.pos('한국어 분석을 시작합니다 재미있어요~~')

Out[4]: [('한국어', 'NNG'),
         ('분석', 'NNG'),
         ('을', 'JKO'),
         ('시작하', 'VV'),
         ('ㅂ니다', 'EFN'),
         ('재미있', 'VA'),
         ('어요', 'EFN'),
```

[3]. Word2Vec



3. Word Embedding - (2) FastText -1-

Integrating a few
Ja-eums & Mo-eums units
into an N-grams
for training the model,
in consideration of “peculiarity of Korean language”.

*Reference : Subword-level Word Vector Representations for Korean, ACL 2018, Sungjoon Park, Jeongmin Byun, Sion Baek, Yongseok Cho, Alice Oh(4) KAIST, Republic of Korea

3. Word Embedding - (2) FastText -2-

WHY DO WE USE fastText Ja/Mo-eum unit N-grams?

*All words are basically
⇒ ROOT(어근) + AFFIX(접사)
⇒ In Korean, affix(접사) changes *dynamically!!* changes according to "the context" & "last Ja/mo-eum of oreceding ROOT".

먹다 (EAT)=> 먹이다(EAT)/ 먹을까?(EAT?)
자다 (SLEEP) => 자라!(SLEEP!), 잘까?(SLEEP?)
준형이가/철수가?=> JAMES/TOM

But !! Meaning of the 먹다, 먹이다, 자라! 잘까? is quite similar!!
=> Information loss !

3. Word Embedding - (2) FastText -3-

Example

Target word : '먹었다' / n=3

{ㅇ, ㅁ, ㅓ}, {ㅓ, ㄱ, ㅇ}, {ㄱ, ㅇ, ㅓ}, {ㅓ, ㅆ, ㄷ}, {ㅆ, ㄷ, ㅏ}, {ㅏ, e, >}

- This project used [3-6]Ja/Mo-eums per One N-gram

- A unique vector of a certain dimension is designated per one N-gram
  (In our case, 200d vector per a N-gram)
- The whole word '먹었다' 's vector is
  => " the sum of all those N-grams"

3. Word Embedding - (2) FastText -4-

FastText: word embedding

•Character n-gram based word embedding

ex) 먹었다

{<, ㅁ, ㅓ}, {ㅁ, ㅓ, ㄱ}, { ㅓ, ㄱ, ㅇ}, {ㄱ, ㅇ, ㅓ}, {ㅇ, ㅓ, ㅆ}, { ㅓ, ㅆ, ㄷ}, {ㅆ, ㄷ, ㅏ}, {ㄷ, ㅏ, e}, { ㅏ, e, >}
3-gram

{<, ㅁ, ㅓ, ㄱ}, {ㅁ, ㅓ, ㄱ, ㅇ}, { ㅓ, ㄱ, ㅇ, ㅓ}, {ㄱ, ㅇ, ㅓ, ㅆ}, {ㅇ, ㅓ, ㅆ, ㄷ}, { ㅓ, ㅆ, ㄷ, ㅏ}, {ㅆ, ㄷ, ㅏ, e}, {ㄷ, ㅏ, e, >}
4-gram

{<, ㅁ, ㅓ, ㄱ, ㅇ}, {ㅁ, ㅓ, ㄱ, ㅇ, ㅓ}, { ㅓ, ㄱ, ㅇ, ㅓ, ㅆ}, {ㄱ, ㅇ, ㅓ, ㅆ, ㄷ}, {ㅇ, ㅓ, ㅆ, ㄷ, ㅏ}, { ㅓ, ㅆ, ㄷ, ㅏ, e}, {ㅆ, ㄷ, ㅏ, e, >}
5-gram

{<, ㅁ, ㅓ, ㄱ, ㅇ, ㅓ}, {ㅁ, ㅓ, ㄱ, ㅇ, ㅓ, ㅆ}, { ㅓ, ㄱ, ㅇ, ㅓ, ㅆ, ㄷ}, {ㄱ, ㅇ, ㅓ, ㅆ, ㄷ, ㅏ}, {ㅇ, ㅓ, ㅆ, ㄷ, ㅏ, e}, { ㅓ, ㅆ, ㄷ, ㅏ, e, >}
6-gram

3. Word Embedding - (2) FastText -5-

{ㅁ, ㅓ, ㄱ} = [-0.4 -0.2 0.1 … 0.1 0.4]
{ㅇ, ㅓ, ㅆ} = [0.3 -0.1 0.3 … -0.2 0.6]
{ㅆ, ㄷ, ㅏ, e} = [-0.1 -0.3 -0.2 … 0.3 0.1]

A word vector

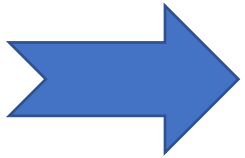
먹었다 = [-0.1, -0.3, 0.1, … 0.1, 0.55] element-wise sum of !!subwords!!

A lot of word vectors for per one text review

수업= [-0.2 -0.5 0.1 … -0.1 0.3]
자체는= [-0.3 -0.8 0.2 … -0.7 0.8]
재미있고= [-0.4 -0.9 0.1 … -0.9 0.2]
…
주지않는= [-0.1 -0.9 0.1 … -0.2 0.1]
듯= [-0.6 -0.5 0.3 … -0.6 0.2]

element-wise sum / the number of words

"수업자체는 재미있고 교수님도 좋으십니다.
하지만 매주 번역을 해야하는 부담감은 있고, 조별로 교재 발표를 한번 하는데 학점에 큰 영향은 주지않는 듯.."



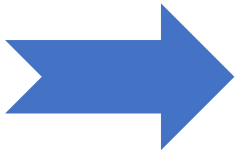
[-0.1 -0.3 -0.2 … 0.3 0.1]

200-D vector

```python
def make_subword_dict(word_list, n_gram_min, n_gram_max):
    subword_dict = {} # word - subword dictionary
    sub2i = {} # subwords to index
    subword_list = []
    for word in word_list:
        subwords = []
        jamo = test.convert(word)
        new_word = list('<' + jamo + '>')
        for n_gram in range(n_gram_min, n_gram_max + 1): #make n-gram subword token
            if n_gram > len(new_word):
                break
            for i in range(len(new_word)):
                if i >= (len(new_word) - (n_gram-1)):
                    break

                subword = ''.join(new_word[i:i+n_gram])
                subwords.append(subword)
                subword_list.append(subword)

        subword_dict[word] = subwords

    subword_list = list(set(subword_list))

    counter = 0
    for subword in subword_list:
        sub2i[subword] = counter
        counter += 1
```

To put it simply :
One text review becomes !! One 200D vector in our model.

"수업자체는 재미있고 교수님도 좋으십니다. 하지만 매주 번역을 해야하는 부담감은 있고, 조별로 교재 발표를 한번 하는데 학점에 큰 영향은 주지않는 듯.."



[-0.1 -0.3 -0.2 ··· 0.3 0.1]

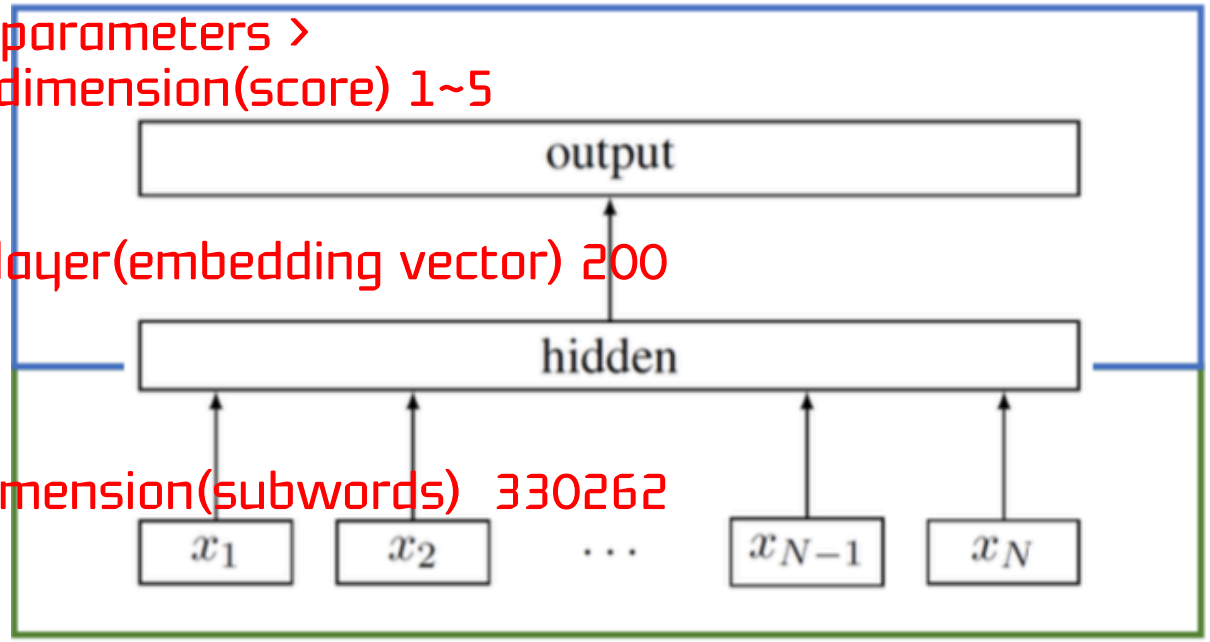
200-D vector

4. Fully Connected Model – (1)Architecture

< Hyperparameters >
output dimension(score) 1~5

hidden layer(embedding vector) 200

input dimension(subwords) 330262

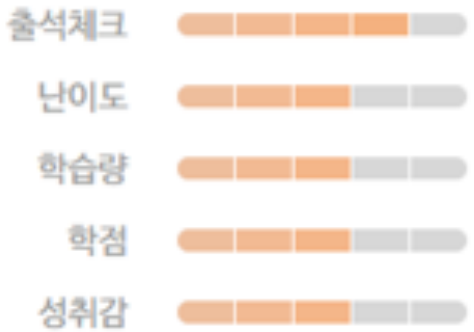


Classification scores to
5 levels of scores.

200-dimension vector input
- a review text vector

sub-words(0.33million)

Repeat that for 5 types of scrores!! =>



4. fully-Connected Model – (2)why this model?

1. RNN and LSTM are also possible to use

⇒BUT, It requires

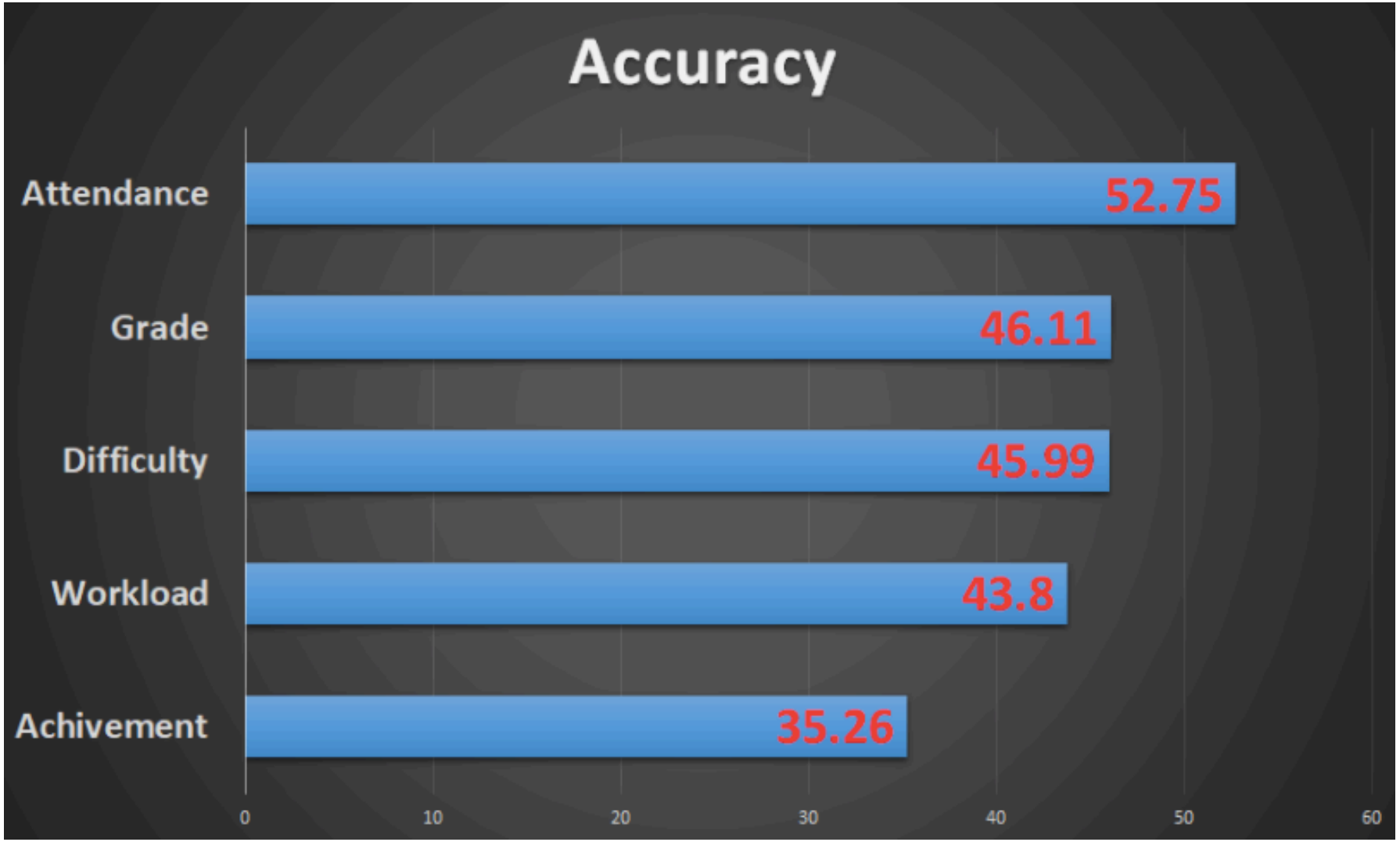
1). More memory,

2). Computing power & time

⇒If word embedding is properly done,

Fully-Connected layer will show good level of accuracy

5. Result Analysis (1)



Hyperparameters

output dimension(score) 1~5
hidden layer(embedding vector) 200
input dimension(subwords) 330262

5. Result Analysis (2) – analysis

1. Achievement seems to lack concrete and consistent signals to ease prediction from text to score.

2. People tend to give similar scores for Difficulty and Workload.

3. Attendance seems to have frequently and consistently occurring signal words such as ”출석체크”,“매일”

4. Difficulty !! People are required to give all 5 scores but there is no rule for writing the text.
   1). Not about all of the indicators.
   2). Tend to be emotion-heavy or expression-heavy.
   => Hard to extract features.

5. Result Analysis (3) – limitations and future research

1. Due to Time limitation, we couldn’t try many other options of learning model architecture.

2. In the future, despite information loss, doing a-few-words n-gram is worth experimenting.

Q & A