Subject Classification of Blue House Petitions

Jinwoo Oh

I. Introduction

In last 3 years, the number of the national petition has surged since president Moon opened a communication platform on the official website of Blue House, the presidential office. This project was conducted to classify the Blue House petition documents into three different subjects with dataset provided by **Dacon.io challenge** [1]. The 45k petition dataset was prepared for the challenge composed of 40k training-set and 5k test-set. The training-set has 3 different approximately uniformly distributed subject labels as illustrated in the Table 1 below.

TABLE I
Labeled Training Set[a]

Label	Subject	# of Data
0	Human rights / Gender equality	13299
1	Culture / Art / Sports / Media	13335
2	Parenting / Education	13299

II. Baseline performance Evaluation

The baseline model utilized a popular recurrent neural network architecture, **LSTM** [2] which captures the features of sequential data and a fully-connected layer. 0.8192 Accuracy was recorded by the baseline model on the challenge.

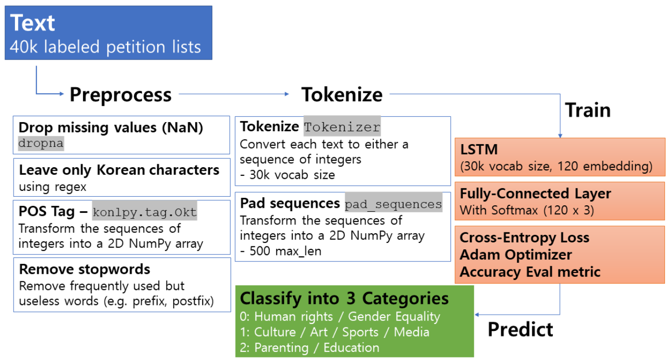

Text
40k labeled petition lists
Preprocess
Tokenize
Train
Drop missing values (NaN) dropna
Leave only Korean characters using regex
POS Tag – konlpy.tag.Okt Transform the sequences of integers into a 2D NumPy array
Remove stopwords Remove frequently used but useless words (e.g. prefix, postfix)
Tokenize Tokenizer Convert each text to either a sequence of integers - 30k vocab size
Pad sequences pad_sequences Transform the sequences of integers into a 2D NumPy array - 500 max_len
LSTM (30k vocab size, 120 embedding)
Fully-Connected Layer With Softmax (120 x 3)
Cross-Entropy Loss
Adam Optimizer
Accuracy Eval metric
Classify into 3 Categories
0: Human rights / Gender Equality
1: Culture / Art / Sports / Media
2: Parenting / Education
Predict

Fig. 1. Baseline Model

First of all, any data including missing values (ie. NaN values) were removed with dropna() function, since it is possible to raise a problem. Secondly, any data including not Korean language were removed to utilize Korean language processing. Okt(twitter) was used to analyze morphemes. To remove useless tokens from vocabulary, [’의’,’가’,’이’,’은’,’들’,’는’,’좀’,’잘’,’걍’,’과’,’도’,’를’,’으로’,’자’,’에’,’와’,’한’,’하다’,’을’] were used for stopwords. Pre-processed data were tokenized into 30k-sized vocabulary with Keras tokenizer library. The pre-processed vocabulary was embedded to 120-D vector for training with LSTM layer and 120x3 FC layer. Soft-max, Cross-entropy and Adam were used for activation function, loss and optimizer respectively. Accuracy was used for evaluate the model performance since it has been commonly used to evaluated classification models. It took 8 minutes for the whole training, 15 epoch and 30 seconds per epoch with a single NVIDIA Tesla T4 GPU on Colab. The environment set-ups are as follows.

- Python 3.6.9
- TensorFlow 2.2.0-rc3
- CUDA 10.1
- Pandas 1.0.3
- scikit-learn 0.22.2
- Konlpy 0.5.2

III. New methodology

A. Motivation

Even though a variety of tuning were conducted and validated with the baseline model (e.g. early-stopping, hyperparameter tuning and dividing train-set into train-valid set), the model no longer performed well over-fitted to the dataset. I decided to change the deep learning model to machine learning model which is more robust on small dataset. According to the comparison results of Korean PoS taggers [3] [4], Mecab [5] tagger outperformed any other taggers in terms of accuracy and speed. MeCab has taken instead of Okt(Twitter) tagger on this project. While it took more than 30minutes to parse training dataset with Okt, it only took 90seconds with MeCab in the experiment. Since it is important to embed POS tokens into vector capturing features of text, I adopted TF-IDF [6].

TABLE II
Performance of POS tagger

Tagger	Accuracy
Okt	0.8192
MeCab	0.8436

B. Model Architecture

Overall, I vectorized the pre-processed tokens with TF-IDF which converts them to a matrix of Term Frequency–Inverse Document Frequency(TF-IDF) features. I removed any disturbing words using regex and stopwords remove custom functions. Linear support vector classifier was taken for estimator followed by calibrated classifier with cross validation. Since it only took a second to fit train-set on the whole model and recorded 0.8808 accuracy for test-set, the proposed model

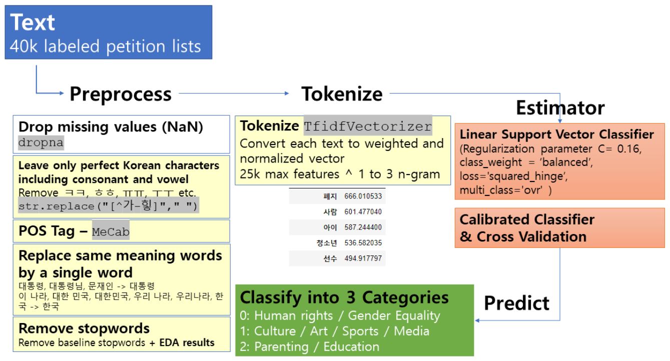
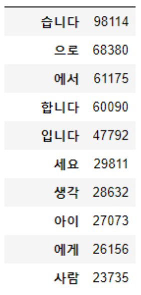

Text
40k labeled petition lists
Preprocess
Tokenize
Estimator
Drop missing values (NaN)
dropna
Leave only perfect Korean characters including consonant and vowel
Remove ㅋㅋ, ㅎㅎ, ㅠㅠ, ㅜㅜ etc.
str.replace("[^가-힣]"," ")
POS Tag – MeCab
Replace same meaning words by a single word
대통령, 대통령님, 문재인 -> 대통령
이 나라, 대한 민국, 대한민국, 우리 나라, 우리나라, 한국 -> 한국
Remove stopwords
Remove baseline stopwords + EDA results
Tokenize TfidfVectorizer
Convert each text to weighted and normalized vector
25k max features ^ 1 to 3 n-gram
폐지 666.010533
사람 601.477040
아이 587.244400
청소년 536.582035
선수 494.917797
Linear Support Vector Classifier
(Regularization parameter C= 0.16, class_weight = 'balanced', loss='squared_hinge', multi_class='ovr')
Calibrated Classifier & Cross Validation
Predict
Classify into 3 Categories
0: Human rights / Gender Equality
1: Culture / Art / Sports / Media
2: Parenting / Education

Fig. 2. Proposed Model

is more computationally efficient and outperformed beyond compare.

C. *Pre-processing*

With Exploratory Data Analysis(EDA), I discovered that English characters and single Korean characters (e.g. ㅋㅋ, ㅎㅎ, ㅜㅜ, ㅠㅠ) preclude capturing features on TF-IDF. Moreover, I replaced same meaning words such as 대한민국, 대한 민국, 우리 나라, 우리나라, 이 나라, 한국 by a single word 한국. In consequence of counting the whole words on dataset and creating a summed Document Term Matrix on Fig. 3, I observed other useless words which do not contribute to extract features and removed them.



습니다	98114
으로	68380
에서	61175
합니다	60090
입니다	47792
세요	29811
생각	28632
아이	27073
에게	26156
사람	23735

Fig. 3. Term frequency of train-set documents

D. *Vectorize*

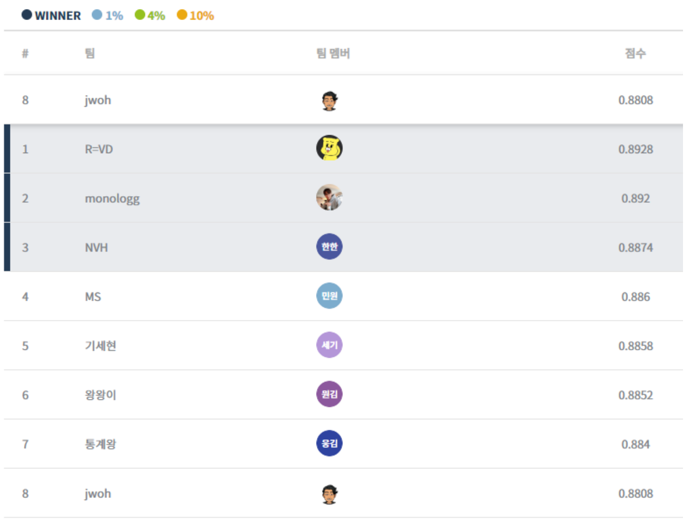
Since it is important to catch how many words were used for each label on classification task, I took Term Frequency–Inverse Document Frequency(TF-IDF). It reflects how important a word is to a document I employed 1 to 3 n-gram on TF-IDF to capture contexts and used 25k max feature size. 25k max feature size surpassed 10k, 30k and 40k in the experiments.

E. *Estimator*

I utilized simple linear support vector classifier which is robust on classification tasks [7]. I tried to use light gradient boosting machine and multinomial naive bayes model, however it yield poor performance. The lower Regularization parameter is, the better performed. Thus, I took 0.16 for the optimized parameter. To prevent over-fitting, I used Calibrated classifier on the very last estimator.

IV. Results

The proposed model has computational efficiency, 1 minute training-time and high-performance 0.8808 accuracy recording 8th on the challenge.



● WINNER ● 1% ● 4% ● 10%

#	팀	팀 멤버	점수
8	jwoh		0.8808
1	R=VD		0.8928
2	monologg		0.892
3	NVH	한한	0.8874
4	MS	민원	0.886
5	기세현	세기	0.8858
6	왕왕이	원김	0.8852
7	통계왕	웅김	0.884
8	jwoh		0.8808

Fig. 4. Leader board

V. Future Works

A. *Neural Network Approach*

Transformer based language models have outperformed any other models these days. Even though it is only 45k dataset, I am going to try to use pre-trained Transformer models and fine-tune the model with data augmentation of given dataset. Popular pre-trained models, Ko-BERT [8] and KoELECTRA [9], would be utilized for the future works.

References

[1] Dacon.io (https://dacon.io/competitions/open/235597/overview/)
[2] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.
[3] 한국어 형태소 분석기 성능 비교 (https://iostream.tistory.com/144)
[4] 이기창, 한국어 임베딩, 에이콘출판사
[5] 은전한닢 (https://bitbucket.org/eunjeon/mecab-ko-dic/src/master/)
[6] Mohit Iyyer, Varun Manjunatha, Jordan BoydGraber, and Hal Daum´e III. 2015. Deep unordered composition rivals syntactic methods for text classification. In Proc. ACL.
[7] https://www.kaggle.com/selener/multi-class-text-classification-tfidf
[8] https://github.com/SKTBrain/KoBERT
[9] https://github.com/monologg/KoELECTRA