

Kakao Shopping Classification

CoE 202. Term Project

Fall, 2019

TA: Kiwon Lee (kaiser5072@kaist.ac.kr)

Professor: Yong Hoon Lee

Overview

- » **Objective:** The goal of this project is to design a product classifier that predicts categories of each product. A product belongs to a maximum of **4 categories**, having each of which is a hierarchical structure.

Product name	Category 1	Category 2	Category 3	Category 4
"맛있는 제주차 3종세트 "	음료/생수/커피	차/티백	차 선물세트	X
"애플 맥북 "	노트북/태블릿PC	노트북	휴대용 (10~13인치형)	X
⋮		⋮		

- › You must predict a total of four categories for each product.
- › The final score is calculated by the weighted average of the accuracies for the four categories (Max. 1.225):

$$\text{Acc} = \frac{((\text{cat. 1 acc.}) * 1.0 + (\text{cat. 2 acc.}) * 1.2 + (\text{cat. 3 acc.}) * 1.3 + (\text{cat. 4 acc.}) * 1.4)}{4}$$

Dataset (I)

» Features

- › **'pid'**: Encrypted item Id (ex. L3652636882)
- › **'product'**: Original name (ex. 맥북 2017년형 애플 맥북 12형 MNYN2KH/A8G 512G J K)
- › **'brand'**: Brand name (ex. 맥북)
- › **'model'**: Refined product name (ex. 12형 레티나 2017년형 (MNYN2KH/A))
- › **'maker'**: Maker name (ex. 애플)
- › **'price'**: Price of product. Mark as -1 if there is no price. (ex. -1)
- › **'img_feat'**: Image features from ResNet50 (ex. 2,048×1 vector)

» Labels: There are 2,289 labels (combination of each categories).

- › bcateid: Label for category 1 range 1~52
- › scateid: Label for category 3 range 1~2070
- › mcateid: Label for category 2 range 1~463
- › dcateid: Label for category 4 range 1~183

Dataset (II)

» We provide **6 dataset** files (Download [#1](#), [#2](#)): The first 3 files contain all features with the exception of “img_feat”

- ① ‘train.chunk.csv’ : 800,000 sets of features for training
- ② ‘valid.chunk.csv’ : 178,830 sets of features for validation
- ③ ‘test.chunk.csv’ : 178,830 sets of features without labels for submission
- ④ ‘train.image.feats.pickle’ : 800,000 sets of features from ResNet for training
- ⑤ ‘valid.image.feats.pickle’ : 178,830 sets of features from ResNet for validation
- ⑥ ‘test.image.feats.pickle’ : 178,830 sets of features from ResNet for test

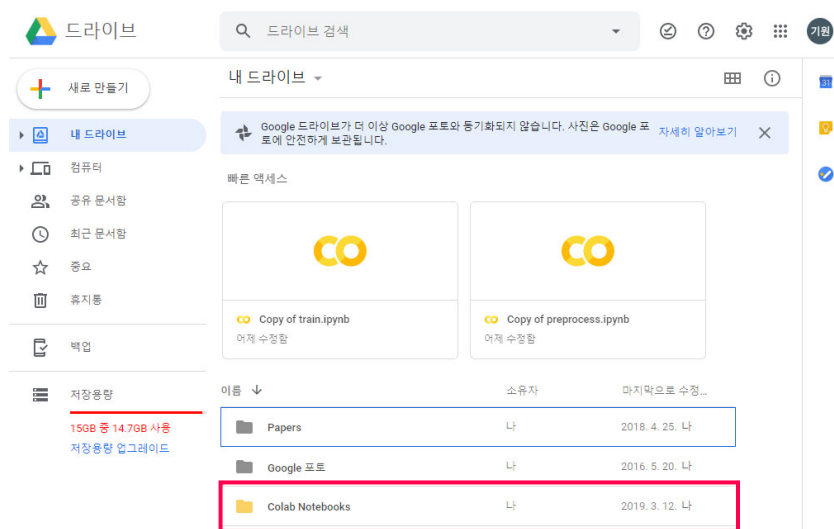
Prerequisite (I)

» Download the datasets

- › Find the datasets in KLMS, and download the datasets to your computer

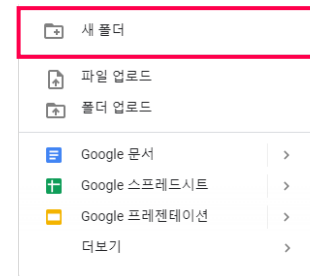
» Upload the datasets to Google drive

- › You need 10GB of free space in your Google drive



① Move to “Colab Notebooks” folder

② Create a new folder named “CoE202_KakaoArena”



Prerequisite (II)

내 드라이브 > Colab Notebooks

이름 ↓	소유자	마지막으로 수정...
CoE202_KakaoArena	나	2019. 11. 2. 나
Untitled3.ipynb	나	2019. 11. 2. 나
train.ipynb	나	오후 4:01 나
preprocess.ipynb	나	오후 4:01 나

③ Upload jupyter file to “Colab Notebooks” folder

④ Create a new folder named “models”
in “CoE202_KakaoArena” folder

내 드라이브 > Colab Notebooks > CoE202_KakaoArena

이름 ↓	소유자	마지막으로 수정...
models	나	2019. 11. 2. 나
.ipynb_checkpoints	나	2019. 11. 2. 나
valid.image.feats.pickle	나	2019. 11. 7. 나
valid.chunk.csv	나	2019. 11. 7. 나
train.image.feats.pickle	나	2019. 11. 7. 나
train.chunk.csv	나	2019. 11. 7. 나
test.image.feats.pickle	나	2019. 11. 7. 나
test.chunk.csv	나	2019. 11. 7. 나

⑤ Upload the datasets to “CoE202_KakaoArena” folder

» Link in colab

- › Run the following code in the jupyter file and enter the url that comes out
- › Log in your google Id, check the authorization code and enter it in colab

```
from google.colab import drive
drive.mount('/content/drive')
```

Baselines

» We provide a baseline solution to help you perform the project. This baseline solution uses only the *'product'* feature.

» **Preprocessing**


- › Preprocessing includes 3 steps (indexing labels, parsing text, building bag of words)
- › First, load the datasets using pandas framework

```
df = pd.read_csv('./drive/My Drive/Colab Notebooks/CoE202_KakaoArena/train.chunk.csv')
```

Preprocessing (I)

» Indexing labels:

- › The label of an item is a 4×1 vector. For simplicity, we use “item index” instead of the label

bcateid: 49 mcateid: 261 scateid: 1287 dcateid: -1	 Indexing the label vector	Index of this 4×1 vector is “1650”
---	---	--

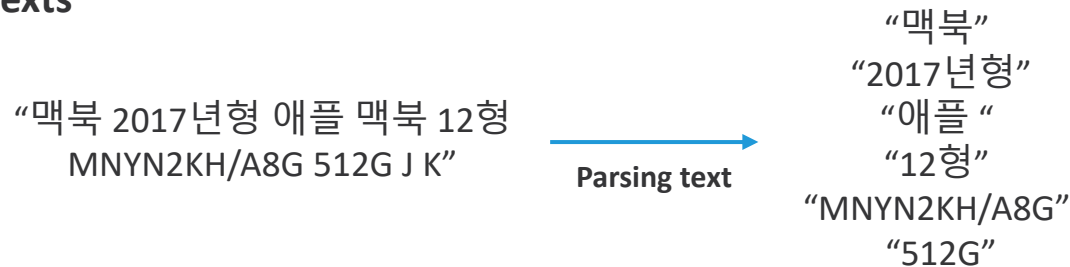
```
y_vocab = {} # Define dictionary for labels

sz = df['pid'].shape[0] # The number of data, 800,000 for training data
for i in tqdm.tqdm(range(sz), mininterval=1): # tqdm predict the remaining time during for loop
    b = df['bcateid'][i] # Get category 1
    m = df['mcateid'][i] # Get category 2
    s = df['scateid'][i] # Get category 3
    d = df['dcateid'][i] # Get category 4
    class_name = '{}>{}>{}>{}'.format(b, m, s, d) # Combine four categories
    if class_name not in y_vocab: # If the combination of categories isn't in dictionary, add it to dictionary.
        y_vocab[class_name] = len(y_vocab)

y_dict = {y: idx for idx, y in enumerate(y_vocab)}
pickle.dump(y_dict, open('./drive/My Drive/Colab Notebooks/CoE202_KakaoArena/y_vocab.pickle', 'wb'), 2)
```


Preprocessing (II)

» Parsing texts



```
feature = df['product'][i]
if isinstance(feature, str) and feature != 'nan':
    feature = re_sc.sub(' ', feature).strip().split()
```

```
words = [w.strip() for w in feature]
words = [w for w in words
         if len(w) >= 2 and len(w) < 31]
```

```
# Get the product name
# Check the product name is string or empty
# re_sc eliminate the special symbols (!@#$"... )
# strip() is a function to remove blank before and after of strings
# split() is the function of separating strings based on spaces.
# Make separated words to list

# If each word is less than 2 and more than 31 is discarded.
```

Preprocessing (III)

» Building the bag of words using a hash function

“맥북”	50239	2
“2017년형”	67893	1
“애플”	20399	1
“맥북”	50239	2
“12형”	9486	1
“MNYN2KH/A8G”	89705	1
“512G”	12309	1

Build bag of words

There are 100,000 words and the corresponding indices.

```
x = [hash(w) % 100000 + 1 for w in words]      # Mapping words to integer using hash function
xv = Counter(x).most_common(32)               # Count the number of words in a string

x = np.zeros(32, dtype=np.int32)
v = np.zeros(32, dtype=np.int8)

for j in range(len(xv)):
    x[j] = xv[j][0]                           # Mapped integers of words
    v[j] = xv[j][1]                           # Counts of words
```

Preprocessing (IV)

» Save the preprocessed data

```
train_dset = {'product': np.asarray(products), 'w_product': np.asarray(w_products), 'label': np.asarray(labels)}  
  
with gzip.open('./drive/My Drive/Colab Notebooks/CoE202_KakaoArena/train.chunk.pickle', 'wb') as f:  
    pickle.dump(train_dset, f)
```

» Continue preprocessing for validation and test datasets.

Training process (I)

» Load the training and validation data

```
# training data
with gzip.open('./drive/My Drive/Colab Notebooks/CoE202_KakaoArena/train.chunk.pickle', 'rb') as f:
    traindata = pickle.load(f)

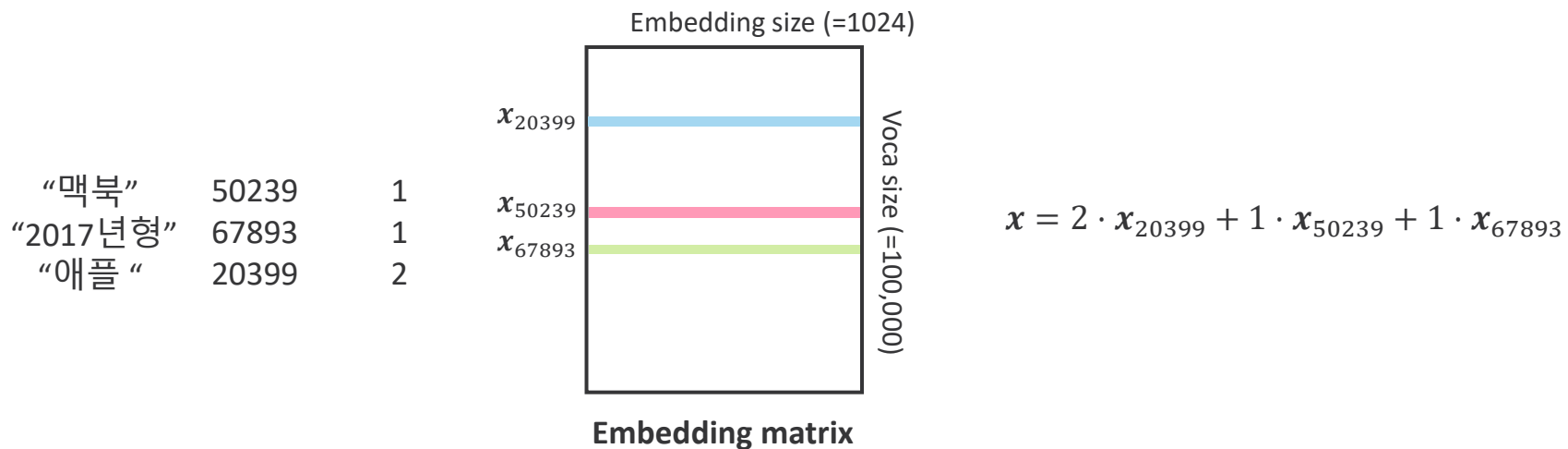
train_product = traindata['product'][:]
train_wproduct = traindata['w_product'][:]
train_label = traindata['label'][:]

# validation data
with gzip.open('./drive/My Drive/Colab Notebooks/CoE202_KakaoArena/valid.chunk.pickle', 'rb') as f:
    validdata = pickle.load(f)

valid_product = validdata['product'][:]
valid_wproduct = validdata['w_product'][:]
valid_label = validdata['label'][:]
```

Training process (II)

» Word embedding: We learn $1,024 \times 1$ vectors representing the word indices.



```
# Define embedding
embd = Embedding(voca_size, embd_size, name='product_embd')

# Product embedding
t_product_embd = embd(products)

# Multiply the weights of product vectors
w_product_mat = tf.reshape(w_products, [-1, max_len, 1])
product_embd_mat = tf.keras.layers.dot([t_product_embd, w_product_mat], axes=1)
product_embd = tf.reshape(product_embd_mat, [-1, embd_size])

# Define embedding function
# Get the embedding vector of products
# Multiply the counts of products to the corresponding embedding vectors
```

Training process (III)

» Fully-connected layers (1 hidden layer)

```
# Non-linear activations
embd_out = tf.layers.dropout(product_embd, training=is_train)
bn = tf.layers.batch_normalization(embd_out, training=is_train)
relu = tf.nn.relu(bn)

# logits
logits = tf.layers.dense(relu, n_classes)

# Softmax
preds = tf.nn.softmax(logits, name='predictions')
```

Dropout
Batch normalization
ReLU function
Fully-connected layer

» Loss function & Optimizer

```
# Softmax cross entropy loss
loss = tf.losses.softmax_cross_entropy(onehot_labels=tf.one_hot(labels, n_classes), logits=logits)

# Weight decay
reg_losses = tf.get_collection(tf.GraphKeys.REGULARIZATION_LOSSES)
loss = tf.add_n([loss] + reg_losses, name='total_loss')

# Optimizer
optm = tf.train.AdamOptimizer(lr)
train_op = optm.minimize(loss, global_step=tf.train.get_global_step(), name='step_update')
update_ops = tf.get_collection(tf.GraphKeys.UPDATE_OPS)
train_op = tf.group([train_op, update_ops])
```

Test process

» Load the test datasets

```
# test data
with gzip.open('./drive/My Drive/Colab Notebooks/CoE202_KakaoArena/test.chunk.pickle', 'rb') as f:
    testdata = pickle.load(f)

test_product   = testdata['product'][:]
test_wproduct  = testdata['w_product'][:]
pids           = testdata['pids'][:]
```

» Save the predictions for submission

```
# Indexing of predictions
argpreds = np.argmax(preds, axis=1)

# Load label dictionary
y_dict = cPickle.loads(open('./drive/My Drive/Colab Notebooks/CoE202_KakaoArena/y_vocab.cPickle').read())

# Inverse label dictionary
inv_y_dict = dict((y,x) for x,y in y_dict.iteritems())
submissions = [inv_y_dict[argpred] for argpred in argpreds]

# Write the results to 'submissions.csv'
f = open('./drive/My Drive/Colab Notebooks/CoE202_KakaoArena/submissions.csv', 'w')
for i, j in zip(pids, submissions):
    line = '{}{}\n'.format(i,j)
    f.write(line)
f.close()
```

Submissions (I)

» Final reports

1. Use 'brand', 'maker' and 'model' features in addition to 'product' features for training. Report your results.
2. Change the model to use 'image' features and 'price' features for training. Report your results.
3. Change the model and tune the hyperparameters to increase the score as much as possible.

» Submissions

- › Due date: 12/17
- › email to kaiser5072@kaist.ac.kr
- › Final submissions must include the 'submissions.csv' file and model save file in Colab ('checkpoint', 'dnn_models.data', 'dnn_models.index', 'dnn_models.meta').

» Reference (Do not copy the reference code)

- › https://github.com/yookyoungKoh/kakao_challenge
- › <https://github.com/ywkim/kakao-arena-shopping>
- › <https://github.com/tantara/kakao-arena-product-classification>

Submissions (II)

» Leader board

- › We provide a leader board to check your score in real time. (Access with Google Chrome)

<http://kalman.kaist.ac.kr/coe202>

- › Click the “Choose File” and submit your “submissions.csv” file. You can check your scores. If your score in top 10, your score will be uploaded in the leader board.

Leader board

We provide a leader board to check your score in real-time. Click the “Choose File”. You can check your scores. If your score in top 10, your score will be uploaded in the leader board below.

Rank	Id	Score
1	Baselines	0.99618
2		
3		
4		
5		
6		
7		
8		
9		
10		

Submission

파일 선택

선택된 파일 없음

Submission