Matching Job Description to Job Titles: Machine Learning Mechanism

Strayn Wang

The input dataset here contains two parts: a) cleaned job posting as X, which has only the extracted information of required technical skills, job responsibilities, job qualifications; b) cleaned job titles as Y, whose stopwords, irrelevant description and job-level components are all removed (but the descriptive component and objective component haven't been seperated).
X: Job descriptions are encoded to sqeuences, and post-padded with 0s, then serialize with time step of 15.
Y: The job labels are encoded with fastText (https://github.com/facebookresearch/fastText).
I randomly sampled 80% of original dataset to form training dataset and leave the left to be testing dataset.
training: I chose to minimize cosine proximity loss between vectorized job description (by a trainable LSTM) and vectorized job titles (by a un-trainable fastText).
I tested 21 LSTM variants to find out optimal structures to vectorize job description, among them I chose the LSTM-19, which has the following structures:

```
def create_LSTM(input_dim,output_dim,time_steps=10,embedding_matrix=[]):
    batch_size = 1
    # inputs.shape = (batch_size, time_steps, input_dim)
    inputs = Input(shape=(batch_size,time_steps, input_dim))
    if embedding_matrix != []:
        embedding_layer = Embedding(embedding_matrix.shape[0],
                                    embedding_matrix.shape[1],
                                    weights=[embedding_matrix],
                                    input_shape=(input_dim,),
                                    trainable=False)
        x = embedding_layer(inputs)
        x = Reshape([embedding_matrix.shape[1],input_dim])(x)
    else:
        x = Reshape([time_steps,input_dim])(inputs)

    x = Bidirectional(LSTM(100, return_sequences=True))(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    x = attention_3d_block(x,input_dim=200)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    #LSTM OUT
    x = Bidirectional(LSTM(150))(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    #NN OUT
    x = Dense(output_dim, activation='tanh')(x)
    model = Model(input=inputs, output=x)
    print(model.summary())
    return model
```

evaluation: Except for looking at the predicted titles by human intuition, I made a score called *Ranking Percentage Score*, which is the percentage of correct label located in the prediction sequence, which is ordered by predicted cosine proximity.

1. Load the packages

In [1]:

```
import numpy as np
from numpy.random import seed
seed(1)
from tensorflow import set_random_seed
set_random_seed(2)
from keras.preprocessing import sequence
from keras.preprocessing.text import Tokenizer
import pandas as pd
import json
import os
import datetime
try:
        CWDIR = os.path.abspath(os.path.dirname(__file__))
except:
        CWDIR = os.getcwd()

from keras import metrics
from keras.optimizers import SGD, Adam, RMSprop
pd.options.mode.chained_assignment = None  # default='warn'

def import_local_package(addr_pkg,function_list=[]):
        #import local package by address
        #it has to be imported directly in the file that contains functions
required the package, i.e. it cannot be imported by from .../utils import i
mport_local_package
        import importlib.util
        spec = importlib.util.spec_from_file_location('pkg', addr_pkg)
        myModule = importlib.util.module_from_spec(spec)
        spec.loader.exec_module(myModule)
        if len(function_list)==0:
                import re
                function_list = [re.search('^[a-zA-Z]*.*',x).group() for x
in dir(myModule) if re.search('^[a-zA-Z]',x) != None]

        for _f in function_list:
                try:
                        eval(_f)
                except NameError:
                        exec("global {}; {} = getattr(myModule,'{}')".forma
t(_f,_f,_f)) #exec in function has to use global in 1 line
                        print("{} imported".format(_f))

        return
import_local_package(os.path.join(CWDIR,'./lib/utils.py'),['train_model','l
oad_model','get_rank_df'])
import_local_package(os.path.join(CWDIR,'./data/lib/prepare_data.py'),[])
```

```
/home/nyartsgnaw/anaconda3/lib/python3.6/site-packages/h5py/__init__.py:36:
FutureWarning: Conversion of the second argument of issubdtype from `float`
to `np.floating` is deprecated. In future, it will be treated as `np.float6
4 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

```
train_model imported
load_model imported
get_rank_df imported
Word2Vec imported
WordNetLemmatizer imported
get_similar_words imported
get_time_series imported
nltk imported
prepare_data imported
remove_stopwords imported
sent_tokenize imported
stopwords imported
wn imported
word_tokenize imported
```

2. Setup the experiment

In [2]:

```
# inputs
path_exp = os.path.join(CWDIR,'./experiments/exp_logs.xlsx')
df_exp = pd.read_excel(path_exp)
with open(os.path.join(CWDIR,'./experiments/.idx'),'r') as f:
    idx = int(f.read())
idx =35

exp = df_exp.iloc[idx]

# setup model parameters
start_time = datetime.datetime.now()
EXP_ID = exp['EXP_ID'] #the name for this experiment
MODEL_ID = exp['MODEL_ID'] #model framework
OUTPUT_DIM = int(exp['OUTPUT_DIM']) # LSTM output vector dimension, should match that of Word2Vec of labels
INPUT_DIM = int(exp['INPUT_DIM']) # LSTM input vector dimension, length of tokens cut from original data texts for each record
TIME_STEPS = int(exp['TIME_STEPS']) #for LSTM sequential
N_EPOCH = int(exp['N_EPOCH']) #for LSTM
PATIENCE = int(exp['PATIENCE']) #for LSTM
TRAIN_MODEL = int(exp['IS_TRAIN'])
LOSS=exp['LOSS_FUNC']
print(exp)
import_local_package(os.path.join(CWDIR,'./experiments/models/{}.py'.format(MODEL_ID)),[])

# setup logging address
path_vectors = os.path.join(CWDIR,'./logs/models/vectors_JT-{}.csv'.format(INPUT_DIM))
path_model = os.path.join(CWDIR,'./logs/models/LSTM_{}.model'.format(EXP_ID))
path_eval = os.path.join(CWDIR,'./logs/eval/LSTM_eval_{}.csv'.format(EXP_ID))
path_training_model = os.path.join(CWDIR,'./logs/models/LSTM_train_{}.model'.format(EXP_ID))
path_training_log = os.path.join(CWDIR,'./logs/train_logs/LSTM_logs{}.csv'.format(EXP_ID))
#       if os.path.isfile(path_training_log):
#               os.remove(path_training_log)
os.system('mkdir -p {}'.format(os.path.join(CWDIR,'./logs/eval/')))
os.system('mkdir -p {}'.format(os.path.join(CWDIR,'./logs/models/')))
os.system('mkdir -p {}'.format(os.path.join(CWDIR,'./logs/train_logs/')))
# fix random seed for reproducibility
np.random.seed(7)
```

```
EXP_ID                         35
N_EPOCH                       200
PATIENCE                      100
IS_TRAIN                        1
LOSS_FUNC        cosine_proximity
OUTPUT_DIM                    200
INPUT_DIM                     200
TIME_STEPS                     15
MODEL_ID                  LSTM_19
IS_RUN                          0
RANK_SCORE               0.506717
QUIT_LOSS               -0.876086
QUIT_MSE               0.00585117
QUIT_EPOCH                    199
N_PARAMS                   765200
start_time    2018-08-06 20:03:05
end_time      2018-08-06 21:32:46
note                      Bi-LSTM
Name: 35, dtype: object
Activation imported
BatchNormalization imported
Bidirectional imported
Conv1D imported
Conv2D imported
Conv2DTranspose imported
Convolution3D imported
Dense imported
Dropout imported
Embedding imported
Flatten imported
GaussianDropout imported
GaussianNoise imported
Input imported
K imported
LSTM imported
Lambda imported
LeakyReLU imported
MaxPooling2D imported
Model imported
Permute imported
RepeatVector imported
Reshape imported
Sequential imported
TimeDistributed imported
UpSampling1D imported
UpSampling2D imported
UpSampling3D imported
attention_3d_block imported
create_LSTM imported
initializers imported
merge imported
regularizers imported
```

3. Load/prepare the data

In [3]:

```
# read the label Ys
path_vectors = os.path.join(CWDIR,'./logs/models/vectors_JT-{}.csv'.format(
INPUT_DIM))
if not os.path.isfile(path_vectors):
    os.system('python {}'.format(os.path.join(CWDIR,'./lib/train_fasttext.p
y')))

labels = pd.read_csv(path_vectors).values

# read the data Xs
path_data = os.path.join(CWDIR,'./data/df_all.csv')
df_all = pd.read_csv(path_data)

# encode the data into sequence
tokenizer = Tokenizer()
tokenizer.fit_on_texts([' '.join(df_all['texts'])])
data = tokenizer.texts_to_sequences(df_all['texts'])
data = sequence.pad_sequences(data, padding='post',truncating='post',maxlen
=INPUT_DIM) # truncate and pad input sequences

# prepare trainig/testing data/labels
judge = (df_all['split']=='train').values
train_data = data[judge]
test_data = data[~judge]
train_labels = labels[judge]
test_labels = labels[~judge]

# serialize the data/labels to model input/output format
X_train, Y_train = get_time_series(train_data,train_labels,TIME_STEPS,0)
X_test, Y_test = get_time_series(test_data,test_labels,TIME_STEPS,0)
```

4. Train the model

In [4]:

```
# create/load the model
from keras.models import load_model
if os.path.isfile(path_model):
    model = load_model(path_model)
else:
    embedding_matrix = []
#        embedding_matrix = load_embedding_fasttext(path_JD)
#       model = create_LSTM(input_dim=INPUT_DIM,output_dim=OUTPUT_DIM,embed
ding_matrix=embedding_matrix)
    model = create_LSTM(input_dim=INPUT_DIM,output_dim=OUTPUT_DIM,time_step
s=TIME_STEPS,embedding_matrix=embedding_matrix)
# train the model
N_EPOCH = 1
if TRAIN_MODEL == True:
    adam=Adam(lr=0.005, beta_1=0.9 ,decay=0.001)
    model.compile(loss=LOSS, optimizer=adam, metrics=['mse','cosine_proximi
ty'])
    model = train_model(model,X_train=X_train.reshape([-1,1,TIME_STEPS,INPU
T_DIM]),\
                        Y_train=Y_train,\
                        verbose=1,n_epoch=N_EPOCH,validation_split=0.1,pati
ence=PATIENCE,\
                        model_path=path_training_model,
                        log_path=path_training_log)
    model.save(path_model)
```

```
/home/nyartsgnaw/anaconda3/lib/python3.6/site-packages/keras/engine/topolog
y.py:1271: UserWarning: The `Merge` layer is deprecated and will be removed
after 08/2017. Use instead layers from `keras.layers.merge`, e.g. `add`, `c
oncatenate`, etc.
  return cls(**config)
```

```
WARNING:tensorflow:Variable *= will be deprecated. Use variable.assign_mul
if you want assignment to the variable value or 'x = x * y' if you want a n
ew python Tensor object.
1
Train on 12436 samples, validate on 1382 samples
Epoch 1/1
 - 28s - loss: -7.0686e-01 - mean_squared_error: 0.0069 - cosine_proximity:
-7.0686e-01 - val_loss: -6.5758e-01 - val_mean_squared_error: 0.4877 - val_
cosine_proximity: -6.5758e-01
```

5. Evaluate the model

In [9]:

```
# evaluate the model by ranking percentage score
yhat = model.predict(X_test.reshape([-1,1,TIME_STEPS,INPUT_DIM]))[:5]
# prepare the original testing labels
titles_all = df_all.titles.values
titles_test = titles_all[~judge][:5]
Y = np.concatenate([Y_test,Y_train])

df = get_rank_df(yhat,titles_test,Y,titles_all)
#       df = get_rank_df(yhat,titles_test,Y_test,titles_test)
df.to_csv(path_eval,index=False)
```

```
Job: full-time community connections intern paid internship
   software engineer
   data administrator 1 it division financial monitoring center
   radio optimization senior engineer
   head financial department
   safety manager
   procurement specialist
   sales senior specialist commercial directorate
   web developer
   domestic expert international exposure accounting
   human resources senior specialist
Percentage_Rank of 0: 0.7865214116010895

Job: bcc specialist
   software engineer
   data administrator 1 it division financial monitoring center
   radio optimization senior engineer
   head financial department
   safety manager
   procurement specialist
   sales senior specialist commercial directorate
   domestic expert international exposure accounting
   human resources senior specialist
   education officer
Percentage_Rank of 1: 0.7081184446891117

Job: chauffeur fsn-3 fp-bb*
   software engineer
   data administrator 1 it division financial monitoring center
   radio optimization senior engineer
   head financial department
   safety manager
   procurement specialist
   sales senior specialist commercial directorate
   domestic expert international exposure accounting
   human resources senior specialist
   education officer
Percentage_Rank of 2: 0.9626818102798864

Job: demographic analysis workshop
   software engineer
   data administrator 1 it division financial monitoring center
   radio optimization senior engineer
   head financial department
   safety manager
   procurement specialist
   sales senior specialist commercial directorate
   human resources senior specialist
   education officer
   administrative assistant
Percentage_Rank of 3: 0.8447586486643102

Job: receptionist
   software engineer
   data administrator 1 it division financial monitoring center
   radio optimization senior engineer
   head financial department
   safety manager
   procurement specialist
   sales senior specialist commercial directorate
   technical writer
   domestic expert international exposure accounting
   web developer
```

6. Log the experiment

In [11]:

```
# log the results
exp['start_time'] = str(start_time.replace(microsecond=0))
exp['end_time'] = str(datetime.datetime.now().replace(microsecond=0))

try:
    exp['RANK_SCORE'] = df['rank'].mean()
except Exception as e:
    print(e)
try:
    df_log = pd.read_csv(path_training_log)
    exp['QUIT_LOSS'] = df_log.iloc[-1]['loss']
    exp['QUIT_EPOCH'] = df_log.iloc[-1]['epoch']
    exp['QUIT_MSE'] = df_log.iloc[-1]['mean_squared_error']
except Exception as e:
    print(e)

try:
    exp['N_PARAMS'] = model.count_params()
except Exception as e:
    print(e)
```

```
Error tokenizing data. C error: Expected 5 fields in line 202, saw 7
```