ex

March 17, 2024

0.0.1 Loading the data

[2]:
```
import jsonlines

# Define a function to read a JSONL file
def read_jsonl(file_path):
    sentence_id = []; summary = []; sentence = []
    with jsonlines.open(file_path, 'r') as reader:
        for line in reader:
            sentence_id.append(line['id'])
            summary.append(line['summaries'][0])
            sentence.append(line['text'])
    return sentence_id,summary,sentence

# Example usage
file_path = 'newsroom.jsonl'
s_id,summary,sentence = read_jsonl(file_path)
```

0.0.2 Training Data

[12]:
```
columns = ['s_id','sentence']
data = df[columns]
data.head()
```

[12]:

	s_id	sentence
0	newsroom-val-title-0	Real Madrid have confirmed they have agreed to…
1	newsroom-val-title-1	American Pie singer Don McLean was arrested on…
2	newsroom-val-title-2	A candidate for governor of the northern Mexic…
3	newsroom-val-title-3	Bill Parcells, the two-time Super Bowl-winning…
4	newsroom-val-title-4	IBM’s data crunching service for the healthcar…

0.0.3 Preprocessing the Data

[13]:
```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
```

```
import string

# Download NLTK resources
nltk.download('punkt')
nltk.download('stopwords')

# Initialize stemmer and set of stopwords
stemmer = PorterStemmer()
stop_words = set(stopwords.words('english'))

def preprocess_text(text):
    # Convert text to lowercase
    text = text.lower()

    # Tokenization
    tokens = word_tokenize(text)

    # Join tokens back into a single string
    processed_text = ' '.join(tokens)

    return processed_text


# Apply the preprocessing function to the 'sentence' column of the DataFrame
data['processed_sentence'] = data['sentence'].apply(preprocess_text)

# Display the DataFrame with the processed sentences
data.head()
```

```
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[13]:

	s_id	sentence	processed_sentence
0	newsroom-val-title-0	Real Madrid have confirmed they have agreed to…	real madrid confirm agre sign mexican striker …
1	newsroom-val-title-1	American Pie singer Don McLean was arrested on…	american pie singer mclean arrest misdemeanor …
2	newsroom-val-title-2	A candidate for governor of the northern Mexic…	candid governor northern mexican state tamauli…
3	newsroom-val-title-3	Bill Parcells, the two-time Super Bowl-winning…	bill parcel two-tim super bowl-win coach rejoi…
4	newsroom-val-title-4	IBM’s data crunching service for the healthcar…	ibm ’ data crunch servic healthcar industri wa…

[14]:
```
print(data['sentence'][0])
```

```
Real Madrid have confirmed they have agreed to sign the Mexican striker Javier
Hernández on a season-long loan from Manchester United.
```

[15]:
```
print(data['processed_sentence'][0])
```

```
real madrid confirm agre sign mexican striker javier hernández season-long loan
manchest unit
```

0.0.4 Creating word embeddings

[16]:
```
import numpy as np

def load_glove_embeddings(file_path):
    embeddings_index = {}
    with open(file_path, 'r', encoding='utf-8') as f:
        for line in f:
            values = line.split()

            embeddings_index[word] = coefs
    return embeddings_index

word_embeddings = load_glove_embeddings(glove_path)
embedding_dim = len(next(iter(word_embeddings.values())))
print("Embedding Dimension:", embedding_dim)
```

```
Embedding Dimension: 100
```

[17]:
```
print(word_embeddings['the'])
```

```
[-0.038194 -0.24487   0.72812  -0.39961   0.083172  0.043953 -0.39141
  0.3344   -0.57545   0.087459  0.28787  -0.06731   0.30906  -0.26384
 -0.13231  -0.20757   0.33395  -0.33848  -0.31743  -0.48336   0.1464
 -0.37304   0.34577   0.052041  0.44946  -0.46971   0.02628  -0.54155
 -0.15518  -0.14107  -0.039722  0.28277   0.14393   0.23464  -0.31021
  0.086173  0.20397   0.52624   0.17164  -0.082378 -0.71787  -0.41531
  0.20335  -0.12763   0.41367   0.55187   0.57908  -0.33477  -0.36559
 -0.54857  -0.062892  0.26584   0.30205   0.99775  -0.80481  -3.0243
  0.01254  -0.36942   2.2167    0.72201  -0.24978   0.92136   0.034514
  0.46745   1.1079   -0.19358  -0.074575  0.23353  -0.052062 -0.22044
  0.057162 -0.15806  -0.30798  -0.41625   0.37972   0.15006  -0.53212
 -0.2055   -1.2526    0.071624  0.70565   0.49744  -0.42063   0.26148
 -1.538    -0.30223  -0.073438 -0.28312   0.37104  -0.25217   0.016215
 -0.017099 -0.38984   0.87424  -0.72569  -0.51058  -0.52028  -0.1459
  0.8278    0.27062 ]
```

[18]:
```
import numpy as np
from tensorflow.keras.preprocessing.sequence import pad_sequences
from gensim.models import KeyedVectors

word_vectors = word_embeddings

# Function to convert a sentence to word embeddings
def sentence_to_embeddings(sentence, word_vectors, embedding_dim):
    words = sentence.split()
    embeddings = []
    for word in words:
        if word in word_vectors:
            embeddings.append(word_vectors[word])
        else:
            # If word not in vocabulary, use zero vector
            embeddings.append(np.zeros(embedding_dim))
    return embeddings

# Display the preprocessed data
data.head()
```

[18]:

	s_id	sentence	processed_sentence	embeddings	padded_embeddings
0	newsroom-val-title-0	Real Madrid have confirmed they have agreed to…	real madrid confirm agre sign mexican striker …	[[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,…	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,…
1	newsroom-val-title-1	American Pie singer Don McLean was arrested on…	american pie singer mclean arrest misdemeanor …	[[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,…	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,…
2	newsroom-val-title-2	A candidate for governor of the northern Mexic…	candid governor northern mexican state tamauli…	[[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,…	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,…
3	newsroom-val-title-3	Bill Parcells, the two-time Super Bowl-winning…	bill parcel two-tim super bowl-win coach rejoi…	[[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,…	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,…
4	newsroom-val-title-4	IBM’s data crunching service for the healthcar…	ibm ’ data crunch servic healthcar industri wa…	[[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,…	

```
4  [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,…
```

[28]:
```
temp = df['summary']
temp = pd.DataFrame(temp)
temp.head()
```

[28]:
```
                                             summary
0  Real Madrid sign Javier Hernández on loan from…
1  American Pie singer Don Mclean arrested on dom…
2  Candidate for governor of Mexican state of Tam…
3        Bill Parcells rejoining ESPN for third time
4  IBM Watson Health now counts CVS Health as a p…
```

[33]:
```
word_vectors = word_embeddings

# Function to convert a sentence to word embeddings
def sentence_to_embeddings(sentence, word_vectors, embedding_dim):
    words = sentence.split()
    embeddings = []
    for word in words:
        if word in word_vectors:
            embeddings.append(word_vectors[word])
        else:
            # If word not in vocabulary, use zero vector
            embeddings.append(np.zeros(embedding_dim))
    return embeddings


max_seq_length = 50  # Maximum sequence length for padding/truncation
temp['s_padded_embeddings'] = pad_sequences(temp['s_embeddings'],␣
 ↪maxlen=max_seq_length, padding='post', truncating='post').tolist()

# Display the preprocessed data
temp.head()
```

[33]:
```
                                             summary  \
0  Real Madrid sign Javier Hernández on loan from…
1  American Pie singer Don Mclean arrested on dom…
2  Candidate for governor of Mexican state of Tam…
3        Bill Parcells rejoining ESPN for third time
4  IBM Watson Health now counts CVS Health as a p…

                                    processed_summary  \
0  real madrid sign javier hernández loan manches…
1  american pie singer mclean arrest domest viole…
2  candid governor mexican state tamaulipa kill s…
3                 bill parcel rejoin espn third time
```

```
4          ibm watson health count cv health partner

                                              s_embeddings  \
0  [[0.45006, 0.15098, 0.31014, -0.20369, -0.2210…
1  [[0.38666, 0.64827, 0.72807, -0.077056, 0.1545…
2  [[-0.33871, -0.37143, 0.4443, 0.72357, -0.3119…
3  [[-0.10535, -0.025048, 0.55525, -1.0371, 0.221…
4  [[0.4875, 0.4214, 0.013491, 0.71504, 0.3708, -…

                                  s_padded_embeddings
0  [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,…
1  [[0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0…
2  [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,…
3  [[0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
4  [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,…
```

[35]:
```
final.columns
```

[35]:
```
Index(['s_id', 'sentence', 'processed_sentence', 'embeddings',
       'padded_embeddings', 'summary', 'processed_summary', 's_embeddings',
       's_padded_embeddings'],
      dtype='object')
```

0.0.5 Running the Model & Evaluation Metrics

[121]:
```
import numpy as np
from tensorflow.keras.layers import Input, Embedding, LSTM, Dense

# Tokenization
max_words = 10000  # Maximum number of words to tokenize
max_seq_len = 100  # Maximum sequence length for padding

tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(final['processed_sentence'])
tokenizer.fit_on_texts(final['processed_summary'])

# Padding sequences to make them of uniform length
sentence_sequences = pad_sequences(sentence_sequences, maxlen=max_seq_len,␣
 ↪padding='post')
summary_sequences = pad_sequences(summary_sequences, maxlen=max_seq_len,␣
 ↪padding='post')

# Model architecture
embedding_dim = 100  # Dimension of word embeddings
hidden_units = 128   # Number of units in LSTM layer

# Embedding layers
```

```
embedding_layer = Embedding(input_dim=max_words, output_dim=embedding_dim)

# Embedding lookup for sentence and summary
sentence_embedding = embedding_layer(sentence_input)
summary_embedding = embedding_layer(summary_input)

# RNN for sentence and summary
sentence_rnn = lstm_layer(sentence_embedding)
summary_rnn = lstm_layer(summary_embedding)
```

[122]:
```
# Define and compile the model
from tensorflow.keras.utils import to_categorical

model = Model(inputs=[sentence_input, summary_input], outputs=summary_output)
model.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])

# One-hot encode the summary sequences
summary_sequences_one_hot = to_categorical(summary_sequences,␣
 ↪num_classes=max_words)

# Training the model
model.fit([sentence_sequences, summary_sequences], summary_sequences_one_hot,␣
 ↪epochs=10, batch_size=32, validation_split=0.2)
```

```
Epoch 1/10
7/7 [==============================] - 9s 1s/step - loss: 9.1619 - accuracy:
0.6582 - val_loss: 9.0024 - val_accuracy: 0.9184
Epoch 2/10
7/7 [==============================] - 8s 1s/step - loss: 8.3912 - accuracy:
0.9211 - val_loss: 7.4508 - val_accuracy: 0.9184
Epoch 3/10
7/7 [==============================] - 7s 1s/step - loss: 6.6197 - accuracy:
0.9211 - val_loss: 5.4238 - val_accuracy: 0.9184
Epoch 4/10
7/7 [==============================] - 7s 1s/step - loss: 4.3803 - accuracy:
0.9211 - val_loss: 3.0768 - val_accuracy: 0.9184
Epoch 5/10
7/7 [==============================] - 6s 950ms/step - loss: 2.2278 - accuracy:
0.9211 - val_loss: 1.3990 - val_accuracy: 0.9184
Epoch 6/10
7/7 [==============================] - 7s 1s/step - loss: 1.0837 - accuracy:
0.9211 - val_loss: 0.9076 - val_accuracy: 0.9184
Epoch 7/10
7/7 [==============================] - 6s 947ms/step - loss: 0.8279 - accuracy:
0.9211 - val_loss: 0.8369 - val_accuracy: 0.9184
Epoch 8/10
```

```
7/7 [==============================] - 7s 1s/step - loss: 0.7835 - accuracy:
0.9211 - val_loss: 0.8246 - val_accuracy: 0.9184
Epoch 9/10
7/7 [==============================] - 6s 866ms/step - loss: 0.7641 - accuracy:
0.9211 - val_loss: 0.8179 - val_accuracy: 0.9184
Epoch 10/10
7/7 [==============================] - 8s 1s/step - loss: 0.7473 - accuracy:
0.9211 - val_loss: 0.8114 - val_accuracy: 0.9184
```

[122]:
```
<keras.src.callbacks.History at 0x7d9fc3d53af0>
```

[143]:
```
from nltk.translate.bleu_score import sentence_bleu
from nltk.translate.bleu_score import SmoothingFunction
from rouge import Rouge
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Define a function to preprocess the input sentence and expected summary
def preprocess_input_and_summary(input_sentence, expected_summary):
    processed_input_sentence = preprocess_text(input_sentence)
    processed_expected_summary = preprocess_text(expected_summary)
    return processed_input_sentence, processed_expected_summary


# Define a function to decode the summary sequence
def decode_summary(summary_sequence, tokenizer):
    decoded_summary = tokenizer.sequences_to_texts(summary_sequence)
    decoded_summary = [sentence.split() for sentence in decoded_summary]
    decoded_summary = [' '.join(sentence) for sentence in decoded_summary]
    return decoded_summary[0]

def compress(input_sentence,expected_summary):
    # Preprocess input sentence and expected summary
    processed_input_sentence, processed_expected_summary =␣
 ↪preprocess_input_and_summary(input_sentence, expected_summary)

    # Compute evaluation metrics
    bleu_score = sentence_bleu([processed_expected_summary.split()],␣
 ↪decoded_summary.split(), smoothing_function=SmoothingFunction().method1)
    rouge = Rouge()
    rouge_scores = rouge.get_scores(decoded_summary, processed_expected_summary)

    print("Summary:", decoded_summary)

    # Print evaluation metrics
    print("BLEU Score:", bleu_score)
```

[ ]:
```
# Sample data
sentences = [
    "New jobless numbers are a bit of a mixed bag for President Obama and his␣
 ↪reelection bid."]

# Preprocessing
stop_words = set(stopwords.words('english'))

def preprocess_text(text):
    text = text.lower()  # Convert to lowercase
    text = re.sub(r'[^a-zA-Z\s]', '', text)  # Remove non-alphabetic characters
    tokens = word_tokenize(text)  # Tokenization
    filtered_tokens = [word for word in tokens if word not in stop_words]  #␣
 ↪Remove stopwords
    return ' '.join(filtered_tokens)

# Tokenization and preprocessing
sentences = [preprocess_text(sentence) for sentence in sentences]

# Tokenization
tokenizer = tf.keras.preprocessing.text.Tokenizer()
tokenizer.fit_on_texts(sentences)
sequences = tokenizer.texts_to_sequences(sentences)

# Padding sequences for uniform input length
max_seq_len = max([len(seq) for seq in sequences])
padded_sequences = pad_sequences(sequences, maxlen=max_seq_len, padding='post')

# Model parameters
vocab_size = len(tokenizer.word_index) + 1
embedding_dim = 64
hidden_units = 128

# Define models
models = {
    "LSTM": Sequential([
        Embedding(vocab_size, embedding_dim, input_length=max_seq_len),
        LSTM(hidden_units, return_sequences=True),
        Dense(vocab_size, activation='softmax')
    ])
}

# Compile and train models
for model_name, model in models.items():
    model.compile(loss='sparse_categorical_crossentropy', optimizer=Adam(lr=0.
 ↪001), metrics=['accuracy'])
    print(f"Training {model_name}...")
```

```
    model.fit(padded_sequences, np.expand_dims(padded_sequences, -1),␣
 ↪epochs=100, verbose=0)
    print(f"{model_name} trained.\n")

# Example compression
test_sentences = [
    "New jobless numbers are a bit of a mixed bag for President Obama and his␣
 ↪reelection bid."]

test_sentences = [preprocess_text(sentence) for sentence in test_sentences]
test_sequences = tokenizer.texts_to_sequences(test_sentences)
test_padded_sequences = pad_sequences(test_sequences, maxlen=max_seq_len,␣
 ↪padding='post')
```

[3]:
```
for model_name, model in models.items():
    print(f"\n{model_name} compressed sentences:")
    for i, test_sentence in enumerate(test_sentences):
        compressed_sequence = model.predict(test_padded_sequences[i:i+1])
        compressed_sequence = np.argmax(compressed_sequence, axis=-1)
        compressed_sentence = ' '.join([tokenizer.index_word[idx] for idx in␣
 ↪compressed_sequence[0] if idx != 0])
        print(f"{i+1}. {compressed_sentence}")
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate`
or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
Training LSTM…
LSTM trained.


LSTM compressed sentences:
1/1 [==============================] - 0s 484ms/step
1. New jobless numbers a mixed bag for Obama
```

[ ]: