Asdos PJ: Luthfi Balaka
CSCE604231: PENGOLAHAN BAHASA MANUSIA (NLP)
October 1, 2024

Tugas Lab 5 (TL 5)
Melatih Model *Embedding* Melalui *Task* Klasifikasi
***Deadline*: Jumat 4 Oktober 2024 jam 22:00**

Pada tugas ini, Anda diminta untuk bereksperimen dalam melatih model *embedding* untuk klasifikasi jenis berita yang sifatnya multikelas (ada > 2 kelas). Luarannya adalah laporan hasil eksperimen Anda yang mencakup data dan analisis. Tujuannya adalah agar Anda terbiasa dengan proses eksperimen, termasuk mengumpulkan data dan menganalisis hasil, yang akan Anda lakukan saat mengerjakan proyek akhir.

1. Apa yang Perlu Dipahami?

Klasifikasi multikelas merupakan suatu *task* yang bertujuan untuk memberikan label pada *data points*. Berbeda dengan klasifikasi biner, klasifikasi multikelas melibatkan ≥ 3 kelas. Contoh sederhananya adalah melabeli sentimen suatu teks sebagai teks yang positif, negatif, atau netral. Namun, pada tugas ini, *task* Anda adalah melabeli kumpulan berita,[1] masing-masing terdiri atas judul dan deskripsi. Terdapat empat label jenis berita, yakni berita dunia (0), olahraga (1), bisnis (2), dan teknologi (3).

Tantangan utama dalam *task* ini, termasuk banyak *task* lainnya pada NLP, adalah mengubah representasi tekstual menjadi numerik. Hal ini diperlukan karena model *Machine Learning* (ML), khususnya yang berbasiskan *deep learning*, melakukan perkalian vektor numerik, termasuk inputnya. Solusi konvensional yang sering diterapkan adalah dengan TF-IDF, sebagaimana yang sudah dijelaskan di kelas. Namun, solusi kontemporer yang digunakan adalah dengan *embedding*, yakni representasi vektor dari suatu teks yang menangkap makna dari teks tersebut.

Selanjutnya, akan dijelaskan secara garis besar langkah-langkah utama dalam melatih *embedding* pada tugas ini. Anda bisa mempelajari secara lebih matematis cara kerja arsitektur yang digunakan melalui mata kuliah seperti `Pemelajaran Mesin`.

1.1. Melakukan Tokenisasi dan Padding.

Hal pertama yang dilakukan, sebgaimana yang tertera pada Listing 1 adalah melakukan tokenisasi pada teks dan menambahkan *padding*. *Padding* diberikan pada sekuens token yang panjangnya kurang dari `maxlen`. Tujuannya adalah agar panjang sekuens dari semua teks seragam sehingga bisa diproses secara *batch*. Jika panjang sekuens melebihi `maxlen`, maka akan dipotong hingga `maxlen` saja.

```
# Separasi antara teks yang digunakan dan label jenis berita
texts = train_df['description'].values  # Bisa juga teks lain seperti judul dari berita
labels = train_df['news_type'].values

# Melatih dan melakukan tokenisasi pada teks
tokenizer = Tokenizer()
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts)

# Melakukan padding teks sepanjang maksimal maxlen
maxlen = 1000
X_train = pad_sequences(sequences, maxlen=maxlen)
```

Listing 1. Tokenisasi dan *Padding* Teks

[1] Diambil dari AG News Classification Dataset.

1.2. **Mendefinisikan Model.** Hal kedua yang dilakukan, sebagaimana yang tertera pada Listing 2, adalah mendefinisikan `model`. Dalam hal ini, terdapat tiga *layer* pada modelnya, yakni:

1) `Embedding`: Ini merupakan *layer word embedding* dengan jumlah kosakata yang dikenali (`input_dim`) sesuai dengan *tokenizer*-nya. Lalu, ukuran *embedding* yang dihasilkan untuk setiap token adalah sepanjang `output_dim`.
2) `GlobalAveragePooling1D`: Luaran dari *layer* sebelumnya adalah *embedding* dari setiap token pada sekuens. *Layer* ini berfungsi untuk merata-ratakan semua *embedding* tersebut. Dengan kata lain, tujuannya adalah menangkap makna keseluruhan dari semua token yang ada.
3) `Dense`: Ini merupakan *layer* yang menghasilkan prediksi jenis berita dari input yang diberikan.

Secara keseluruhan, ini merupakan suatu model klasifikasi. Namun, di dalamnya terdapat model *embedding* yang bertujuan menangkap makna dari teks. Model *embedding* ini akan sekaligus dilatih bersamaan dengan dilatihnya `model` keseluruhan. Sebagai catatan, ini merupakan salah satu dari sekian banyak cara yang bisa dilakukan untuk melatih model *embedding*. Nantinya, model *embedding* bisa saja digunakan secara terpisah, tanpa menggunakan *layer* lainnya.

```
# Mendefinisikan model dan beberapa komponen terkait
model = Sequential()
model.add(Embedding(input_dim=len(tokenizer.word_index) + 1, output_dim=8))
model.add(GlobalAveragePooling1D())
model.add(Dense(num_classes, activation='softmax'))

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
    metrics=['accuracy'])
```

Listing 2. Mendefinisikan Model

1.3. **Melatih dan Menguji Model.** Hal terakhir yang dilakukan, sebagaimana yang ditunjukkan pada Listing 3, adalah melatih dan menguji model. Saat melatih model, perhatikan bahwa ada beberapa *hyperparameter* yang bisa diatur. Salah satunya adalah `epochs`, yakni jumlah iterasi latihan pada data *training*. Secara umum, semakin banyak latihan, maka semakin tinggi kemungkinan untuk bisa menangkap pola pada data *training* dengan lebih baik. Namun, bisa juga terjadi *overfitting*, yakni model terlalu menyesuaikan dengan pola pada data *training*, termasuk *noise* yang ada. Selain diuji pada data *validation* saat dilatih, model juga akan diuji pada data *testing* untuk memberikan gambaran terkait kemampuan model pada data yang "baru".

```
# Melatih model
model.fit(X_train, y_train, epochs=5, batch_size=32, validation_data=(X_val
    , y_val))

# Menguji model pada data testing, khususnya dengan deskripsi berita
test_sequences = tokenizer.texts_to_sequences(test_df['description'].values
    )
X_test = pad_sequences(test_sequences, maxlen=maxlen)
y_test = test_df['news_type'].values

loss, accuracy = model.evaluate(X_test, y_test)
print(f'Test Accuracy: {accuracy}')
```

Listing 3. Melatih dan Menguji Model

2. Apa yang Harus Dikerjakan?

Setelah memahami informasi di atas, berikut adalah beberapa hal yang perlu dilakukan.

1) *Clone*/unduh repositori GitHub ini yang berisi:

- `classification.ipynb`: Kode untuk melatih dan menguji model klasifikasi berita dengan *default hyperparameters*.
- `train/test.csv`: Dataset untuk *training*/*testing*.

2) Jalankan kode tersebut di platform yang menyediakan GPU, misalnya Google Colab atau Kaggle.
3) Lakukan beberapa eksperimen berikut[2]:
   - **Eksperimen 1**: Dengan semua *hyperparameter* bernilai *default*, latih dan uji model dengan dua input berbeda: judul dan deskripsi berita.
   - **Eksperimen 2**: Dengan input deskripsi berita dan *hyperparameter* lain bernilai *default*, latih dan uji model dengan tiga nilai `maxlen` berbeda: 10, 100, dan 1000.
   - **Eksperimen 3**: Dengan input deskripsi berita dan *hyperparameter* lain bernilai *default*, latih dan uji model dengan dua nilai `epochs` berbeda: 1 dan 5.
4) Buat laporan berformat PDF yang berisikan data eksperimen dan analisis singkat dari setiap eksperimen. Untuk analisis, kaitkan dengan input model (teks berita) dan makna dari ubahan yang Anda lakukan.

3. Pengumpulan Tugas

Kumpulkan *file* laporan dengan aturan nama `NLP-TL5-[NamaAnda].pdf`.

- Contoh: `NLP-TL5-LuthfiBalaka.pdf`.

[2]Pastikan untuk selalu mencatat nilai akurasi pada dataset *training*, *validation*, dan *testing*. Untuk *training* dan *validation*, catat secara lengkap untuk setiap *epoch*.