수DA쟁이 1기 개인 프로젝트

Novel Wirter Classification

2019010718 수학과 강수연

Novel Writer Classification

소설 속 문장 뭉치 분석을 통한 저자 예측

Contents

- 주제 선정 이유 및 목표
- 자연어 처리 과정
- 현재 진행 과정
- 최종 발표까지의 계획

주제 선정 이유 및 목표

주제 선정 이유 : 다음 1기 교재 ‘딥러닝 이용한 자연어 처리 입문’의 주제인 자연어 처리에 대해 미리 한 번 접해보고 싶어 자연어 처리를 활용하는 주제를 선정

목표 : 자연어 처리에 대해 공부하고 주어진 데이터를 활용하여 소설 작가 예측해보기

자연어 처리

자연어 처리란?

텍스트 분류

특정 문장이나 문서를 카테고리로 분류
ex) 스팸 메일 분류

내용 요약

- 추출 요약 : 중요한 문장 뽑아냄
- 생성 요약 : 요약문 새롭게 생성

감성 분석

문서 내용의 긍정이나 부정을 파악하여 평가를 내림

기계 번역

서로 다른 나라의 언어로 번역

자연어 처리

자연어 처리 과정

1. Preprocessing
: 불용어 제거, 형태소 분석, 표제어 추출

2. vectorization
: One-hot encoding, Count vectorization, Tfidf, Padding

3. Embedding
: Word2vec, Doc2vec, Glove, Fasttext

4. Modeling
: GRU, LSTM, Attention

NLP Preprocessing

형태소 분석기(Pos Tagger)

- **Mecab** : 속도가 굉장히 빠르고 분석 결과가 좋음
- **Komoran** : 정제되지 않은 글에 대해 먼저 사용해보는 것이 좋음
- **KKma** : 분석 시간이 오래 걸림
- **Okt** : 품사 태깅 결과를 알아보기 쉽게 반환
- **Khaiii** : 카카오에서 가장 최근에 공개한 분석기

Vectorization

NLP를 수치로 바꿔주는 과정

- **One Hot Encoding** : 해당 단어가 존재하면 1, 그렇지 않으면 0으로 표기

- **Count vectorization** : 단어를 활용하여 각 문장이 갖고 있는 토큰의 개수를 기반으로 문장을 벡터화

- **Tfidf** : 등장 횟수가 많고 분서 분별력이 있는 단어들을 점수화 하여 벡터화

* **Padding** : 문장의 길이를 맞춰주기 위해 부족한 길이만큼 0을 채워넣는 과정

Embedding

단어나 문장들 사이의 관계(유사도) 파악

비슷한 의미를 내포하고 있는 토큰들은 서로 가깝게, 그렇지 않은 토큰들은 서로 멀리 뿌리도록 하는 것이 임베딩의 목적이다.

Embedding

- **Keras Embedding Layer** : 무작위로 특정 차원으로 입력 벡터들을 뿌린 후 학습을 통새 가중치들을 조정해 나가는 방식(단어 사이의 관계 반영 X)

- **word2vec** : 주변 단어와의 관계를 통해 해당 단어의 의미적 특성을 파악

- **glove** : 각 토큰들 간의 유사성은 그대로 가져가면서 데이터 전체에 대한 빈도 반영

- **Fasttext** : 단어 단위가 아닌 서브 단어를 단위로 사용

Modeling

- **RNN** : 순차적인 특징 때문에 연산에 많은 시간이 걸리고 역전파 소실 문제가 발생

- **LSTM** : RNN에 cell state 추가. 불필요한 정보들을 걸러내어 매끄러운 진행 가능 -> 역전파 소실 문제 줄여 성능 증가

- **GRU** : LSTM의 장점을 가져오면서 속도적인 부분 개선

- **Attention** : RNN 모델의 구조적 한계 극복. 어떤 토큰의 정보가 가장 큰 도움을 줬는지 파악 가능

현재 진행 상황

기본적인 keras 사용

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_length),
    tf.keras.layers.GlobalAveragePooling1D(),
    tf.keras.layers.Dense(24, activation='relu'),
    tf.keras.layers.Dense(5, activation='softmax')
])
```

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

```
print(model.summary())
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 500, 16)           320000
_________________________________________________________________
global_average_pooling1d_1 ( (None, 16)                0
_________________________________________________________________
dense_2 (Dense)              (None, 24)                408
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 125
=================================================================
Total params: 320,533
Trainable params: 320,533
Non-trainable params: 0
_________________________________________________________________
None
```

현재 진행 상황

```
num_epochs = 20
history = model.fit(train_padded, y_train,
                    epochs=num_epochs, verbose=2,
                    validation_split=0.2)
```

```
Epoch 1/20
1372/1372 - 14s - loss: 1.5641 - accuracy: 0.2785 - val_loss: 1.5315 - val_accuracy: 0.2753
Epoch 2/20
1372/1372 - 13s - loss: 1.3507 - accuracy: 0.4478 - val_loss: 1.2112 - val_accuracy: 0.5349
Epoch 3/20
1372/1372 - 14s - loss: 1.1183 - accuracy: 0.5542 - val_loss: 1.0887 - val_accuracy: 0.5559
Epoch 4/20
1372/1372 - 14s - loss: 1.0127 - accuracy: 0.5949 - val_loss: 1.0155 - val_accuracy: 0.5965
Epoch 5/20
                                    ⋮
Epoch 15/20
1372/1372 - 12s - loss: 0.5368 - accuracy: 0.8071 - val_loss: 0.7877 - val_accuracy: 0.7167
Epoch 16/20
1372/1372 - 12s - loss: 0.5163 - accuracy: 0.8147 - val_loss: 0.7933 - val_accuracy: 0.7167
Epoch 17/20
1372/1372 - 12s - loss: 0.4989 - accuracy: 0.8209 - val_loss: 0.7785 - val_accuracy: 0.7217
Epoch 18/20
1372/1372 - 12s - loss: 0.4801 - accuracy: 0.8273 - val_loss: 0.7832 - val_accuracy: 0.7206
Epoch 19/20
1372/1372 - 13s - loss: 0.4633 - accuracy: 0.8345 - val_loss: 0.7877 - val_accuracy: 0.7203
Epoch 20/20
1372/1372 - 13s - loss: 0.4500 - accuracy: 0.8387 - val_loss: 0.7942 - val_accuracy: 0.7198
```

최종 발표 까지의 계획

1. 자연어 처리 과정을 숙지하고 그에 따라 코드 작성 및 수정 작업하기
2. 여러 가지 모델을 사용하여 성능 비교해보기
3. 하나의 이미지 파일로 시각화 하여 최종 발표(2/23) 준비하기