

# 프랑스 영화 리뷰 사이트 데이터셋 구축과 감정분석

권성우 (KWON SUNGWOO) 2023-82030

서울대학교 언어학과 언어데이터 과학 2023F

2023/12/18

\* LATEX으로 작성하여 보고서를 .pdf로 제출하게된 점 양해 부탁드립니다.

## 1 연구개요

### 1.1 연구배경

오늘날 여러가지 머신러닝과 딥러닝 기법을 사용한 감정 분석은 그 성능면에서 각광 받고 있는 분야이고, 특히 그 예시로 많이 사용되는 주제 중 하나가 영화 감상평을 감정의 polarity에 따라 분류하는 작업입니다: 그에따라 한국어는 네이버, 영어는 IMDb 같이 대용량 데이터베이스를 가지고 있는 전문적인 영화 리뷰 사이트의 데이터를 가지고 연구가 진행된 바가 있습니다. 이번 연구에서는 프랑스 텔레비전 산업 전문 사이트 allociné.fr의 영화 감상평 데이터를 수집해 현존하는 머신러닝 기법을 사용한 감정분석을 시도해 볼 것입니다. (연구 계획서에는 SensCritique이라는 사이트에서 데이터를 얻을 것이라고 서술했지만, 해당 사이트에서 data scraping을 허용하지 않기에 다른 사이트를 사용하였습니다.)

### 1.2 연구목적과 달성 여부

이 프로젝트를 통하여 실행 해보려 하는 작업은 크게 두 가지입니다.

- Beautifulsoup, pandas, pickle과 같은 파이썬 패키지와 라이브러리들을 이용해 웹에서 정보 스크래핑 해온 다음 데이터셋 가공, 그 과정에서 나오는 수치들을 시각화해 살펴보는 것.
- TF-IDF Embedding 기법과 Grid Search를 적용한 로지스틱 회귀로 감정 분류기 구현과 튜닝, 간단한 분석 진행.

두 가지 모두 수업 시간에 배운 내용들의 연장선에 위치합니다. 기술적인 측면에서도 수업 시간에 사용된 코드를 최대한 활용하려 노력했습니다.

큰 틀의 연구 계획 두 가지 모두 달성했습니다. 연구 확장 방향에 대해서는 마지막 섹션에서 소개하고 있습니다.

## 2 연구 내용

데이터 수집과 모델 구현이라는 두 가지 큰 틀 안에서 각자 어떻게 문제에 접근했고 어떠한 분석을 수행하였는지 섹션을 2개로 나누어 연구 내용을 서술할 계획입니다.

### 2.1 데이터 수집

\* 이 섹션에서 사용된 코드는 [https://github.com/Sungwoo-Kwon/LDS\\_2023/blob/main/project/codes/scraping.py](https://github.com/Sungwoo-Kwon/LDS_2023/blob/main/project/codes/scraping.py)와 [https://github.com/Sungwoo-Kwon/LDS\\_2023/blob/main/project/codes/scrape\\_allocine.ipynb](https://github.com/Sungwoo-Kwon/LDS_2023/blob/main/project/codes/scrape_allocine.ipynb)에서 확인 할 수 있습니다.

<https://www.allocine.fr/films/>에 저장되어 있는 영화들을 개봉 시기를 기준으로 2006년부터 2020년 초기까지 필터링해 총 113,291의 영화들에게서 url을 스크랩 해 왔습니다. 그 후 데이터를 21개의 batch로 나누어 사전 정의된 함수를 이용해 순차적으로 리뷰, 평점, 추천/비추천 수, 리뷰 내용 등을 추출하고 pickle을 이용하여 저장했습니다. 아래는 추출된 데이터를 pandas 데이터 프레임으로 불러온 예시입니다.

|        | film-url  | rating | review  | date       | helpful | unhelpful |
|--------|---|--------|---|------------|---------|-----------|
| 53894  | <a href="http://www.allocine.fr/film/lichefilm-111632/critiques/spectateurs">http://www.allocine.fr/film/lichefilm-111632/critiques/spectateurs</a> | 1.0    | Sans grande surprise, le premier opus de la franchise (il en existe trois actuellement) est dépourvu d'originalité et d'un quelconque intérêt. Sexy Dance (2006) est ultra conventionnel, prévisible au bout de dix minutes et pour un film qui avoisine les 105 minutes, il y en a clairement 45 en trop, entre les répétitions inutiles, les séquences musicales et de danses à rallonge, les répliques mièvres, les disputes et les réconciliations, bref, une sortie en DTV aurait largement suffit ! Espérons que le second volet (2008) et le troisième (2010) seront relever le niveau.  | 2010-09-21 | 1       | 1         |
| 366314 | <a href="http://www.allocine.fr/film/lichefilm-209558/critiques/spectateurs">http://www.allocine.fr/film/lichefilm-209558/critiques/spectateurs</a> | 3.5    | enfin un téléfilm Gulli qui traite des vacances ! un téléfilm très sympathique, bien joué et divertissant pour les moins de 12 ans.   | 2014-11-14 | 1       | 2         |
| 14939  | <a href="http://www.allocine.fr/film/lichefilm-109788/critiques/spectateurs">http://www.allocine.fr/film/lichefilm-109788/critiques/spectateurs</a> | 5.0    | En voyant la multiplication de critiques négatives envers ce monument du cinéma moderne, j'ai regardé There Will Be Blood une troisième fois. Et le verdict est toujours le même, on est toujours face à une leçon magistrale de cinéma. Les acteurs sont plus qu'excellents (Day-Lewis comme à son habitude fait exploser le talentomètre, mais aussi le sous-estimé Paul Dano). Les dialogues sont écrits à la perfection. La réalisation est irréprochable, offre certains plans très beaux plans mais arrive à asseoir une ambiance angoissante et poisseuse, appuyé par la musique qui découte au début mais arrive à nous influencer inconsciemment et instaurer chez le spectateur un profond malaise à certains moments (les dangers du métier de mineur dans une exploitation pétrolière sont parfaitement retracés à l'écran). Cette ambiance sera une histoire pastoralement, celle d'un exploitant misanthrope (de plus en plus détestable) sombrant dans la folie et qui ne cherche qu'une chose: être assez riche pour pouvoir vivre coupé du monde. Tout est parfait, rien que la scène de fin (grand moment, interrogation sur la foi religieuse, le dévoilement de la vraie nature des personnages principaux, le milk-shake, etc) devrait suffire nous faire aimer There Will Be Blood, ou au moins lui reconnaître ses indéniables qualités. | 2017-01-11 | 3       | 0         |

### 2.2 데이터 전처리

\* 이 섹션에서 사용된 코드는 [https://github.com/Sungwoo-Kwon/LDS\\_2023/blob/main/project/codes/data\\_prep.ipynb](https://github.com/Sungwoo-Kwon/LDS_2023/blob/main/project/codes/data_prep.ipynb)에서 확인 할 수 있습니다.

다음으로 이 데이터를 저희의 목적에 맞게 가공해 보았습니다.

먼저 감정분류를 위해 리뷰들을 평점에 기반해 3개의 범주로 분류해 주었습니다(평점은 0-5점의 범위 내에서 0.5 단위로 줄 수 있고 0점은 줄 수 없습니다). 4점 이상 5점이하는 '긍정적' 리뷰로, 2점 초과 4점 미만은 '중립적' 리뷰로, 0.5점에서 2점은 '부정적' 리뷰로 간주하였습니다.

그 후, 평점당/ 범주당 분포를 시각화 해보았습니다.

전체적으로 긍정적인 리뷰가 중립적, 부정적 리뷰보다 많았다는 것을 관찰 할 수 있습니다.

그 다음으로, 각 년도 별 리뷰 개수를 살펴보았습니다(이것에 관해 특별한 전처리를 하지는 않았습니다).

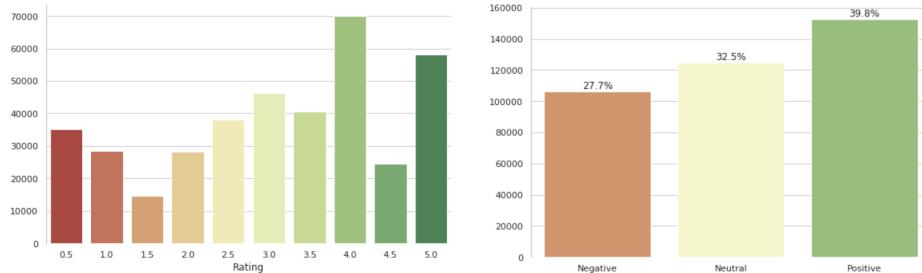
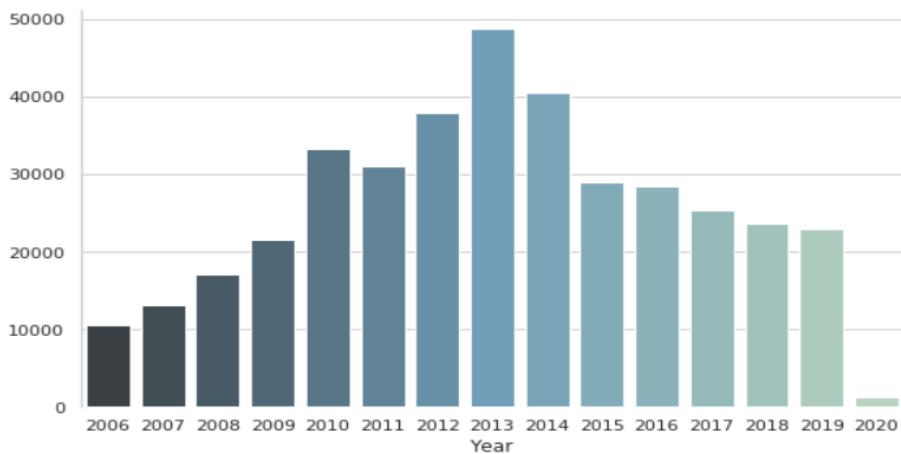
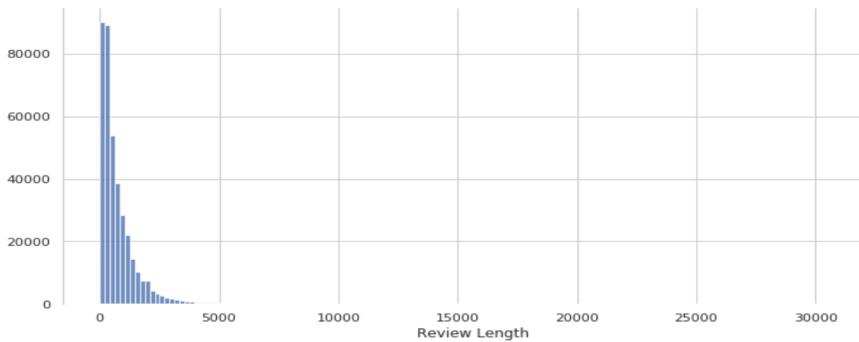


Figure 1: 평점별 리뷰 분포(좌)와 감정 polarity별 리뷰 분포(우)

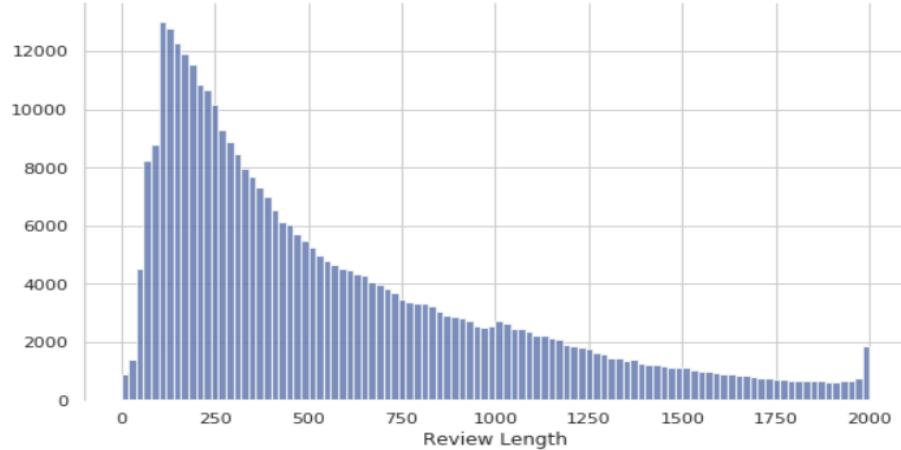


그래프 상으로는 2006년부터 점점 리뷰의 개수가 올라가다 2013년에 최고점을 달성하고 그 후로는 꾸준히 하락했습니다. 실제로 연구자가 느끼기에도 2013년에 가장 활발하게 리뷰가 작성이 되었다는것이 어느정도 이해가 되는 이유는, 2013년을 기점으로 프랑스 현지에 스마트폰 보급량이 빠른 속도로 증가해 사람들이 직접 리뷰 사이트에 접속해서 리뷰를 남기는 일이 줄어들었다는 점 때문입니다.

이후 리뷰의 길이 (글자수 기반) 별로 분포를 시각화 해 보았습니다.



이상치의 관찰과 더불어 데이터 양을 고려할때, 문자수 2000개를 threshold로 정해 2000자를 넘는 데이터는 제외하도록 하겠습니다 (전체 데이터의 6 퍼센트 정도).



150-200자로 이루어진 리뷰가 가장 많았고, 글자 수가 많아질수록 데이터 수가 적어지는 것이 설명히 보이는, 오른쪽으로 꼬리가 긴 분포를 형성하고 있습니다. 2000자 부근에서 갑자기 데이터가 상승하는 것을 주목할만 합니다.

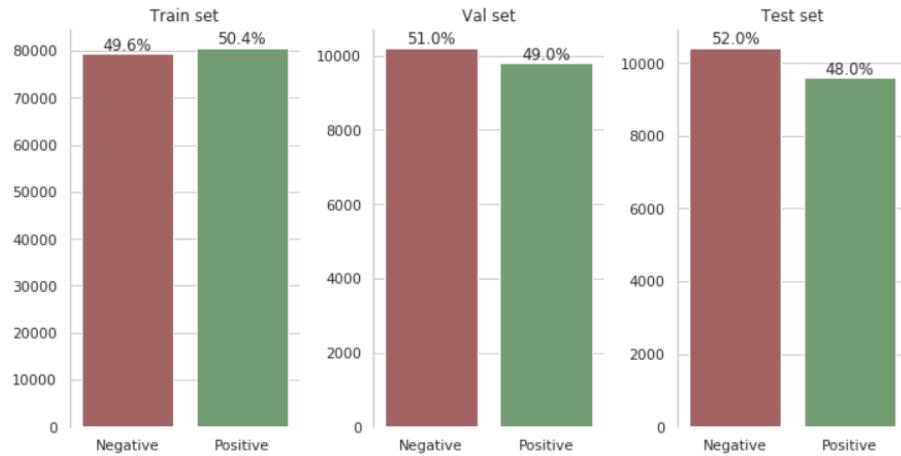
긴 리뷰를 제외한 또 한가지의 이유는, 긴 리뷰중에 완전히 영화에 대한 평가가 아닌, 다른 주제들과 관련된 내용들이 자주 있었기 때문에 모델 트레이닝에 영향을 줄 수 있을것이라 생각되었기 때문입니다.

그 밖의 전처리로는 다음을 실시하였습니다.

1. 불필요한 열 삭제: 별점, 날짜, 추천/비추천 수와 같이 감정분류에 필요하지 않은 열들은 삭제하였습니다.
2. 연구의 목적이 Multinomial이 아닌 binomial 로지스틱 회귀이기 때문에 'neutral'에 해당하는 레코드는 제외하였습니다. 따라서 부정적 리뷰들을 -1에서 0으로 재코딩 해주었습니다.
3. 최대 리뷰 개수를 영화당 30개로 제한하였습니다(스크랩 과정에서 제한을 했지만 다시 한번 필터링을 해주었습니다).
4. 리뷰 길이(글자수)가 3 미만은 모두 삭제하였습니다. 세 글자로 제한한 것은, 프랑스 어에서 어떤 것의 퀄리티에 대해서 사용할 수 있는(또한 자주 사용되는) 가장 짧은 단어들이 'bon(좋은)', 'nul(별로인, 시시한)', 'mal(나쁜)'등 세 글자로 이루어져 있기 때문입니다.

또한, 모델에 투입할 데이터셋들 내 긍정적 리뷰와 부정적 리뷰들의 비율을 맞추기위해 두 classes의 데이터 개수를 balance 해주었습니다. 그 결과, 각 class당 101,349개의 리뷰가 할당되었습니다.

마지막으로 데이터셋을 8:1:1의 비율로 train/validation/test으로 무작위 할당해주었습니다. 이때, df.sample() 함수의 'random\_state=' 옵션을 이용하여 seed를 고정 해주었습니다. 아래 그림은 분할이 완료된 데이터셋을 시각화한 것입니다.



세 데이터셋 모두 거의 50:50으로 비율이 맞춰져 있는 것을 확인 할 수 있습니다.

### 3 감정분류 모델 구현과 분석

\*이 섹션에 사용된 코드는 [https://github.com/Sungwoo-Kwon/LDS\\_2023/blob/main/project/codes/model.ipynb](https://github.com/Sungwoo-Kwon/LDS_2023/blob/main/project/codes/model.ipynb)에서 확인 가능합니다.

\*\*이 섹션은 모델 구현과 분석을 하위 섹션으로 구분짓지 않습니다.

첫번째로, 전 섹션에서 준비한 데이터셋을 불러옵니다.

```
import pickle

PICKLE_PATH = 'path_to_pickle_file.pkl'

with open(PICKLE_PATH, 'rb') as reader:
    data = pickle.load(reader)

X_train, y_train = np.array(data["train_set"]["review"]), np.array(data["train_set"]["polarity"])
X_val, y_val = np.array(data["val_set"]["review"]), np.array(data["val_set"]["polarity"])
X_test, y_test = np.array(data["test_set"]["review"]), np.array(data["test_set"]["polarity"])
class_names = data['class_names']

print("LEN TRAIN: " + str(len(X_train)))
print("LEN VAL: " + str(len(X_val)))
print("LEN TEST: " + str(len(X_test)))
```

LEN TRAIN: 160000  
LEN VAL: 20000  
LEN TEST: 20000

train/validation/test 셋 각각 160000/20000/20000, 8:1:1로 잘 나누어진 것을 확인합니다.

그 다음, 첫번째 모델을 구현해줍니다. Scikit-Learn의 TF-IDF를 이용한 임베딩과 Logistic Regression을 pipeline에 담아서, 우선 모든 parameters를 디폴트값으로 두고 아래와 같은 모델을 만들고 테스트 해보았습니다.

```

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline

tfidf_clf = Pipeline([
    ('tfidf', TfidfVectorizer()),
    ('clf', LogisticRegression(n_jobs=-1, verbose=1)),
])

```

tfidf\_clf.fit(X\_train, y\_train)

Training Accuracy: 0.9415625

Validation Accuracy: 0.92465

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| Negative     | 0.93      | 0.92   | 0.93     | 10204   |
| Positive     | 0.92      | 0.93   | 0.92     | 9796    |
| accuracy     |           |        | 0.92     | 20000   |
| macro avg    | 0.92      | 0.92   | 0.92     | 20000   |
| weighted avg | 0.92      | 0.92   | 0.92     | 20000   |

평가 척도는 precision, recall, f1-score를 사용했습니다. f1-score가 부정적 리뷰에 대해서는 0.93, 긍정적 리뷰에 대해서는 0.92로 괜찮은 성능을 보여주었습니다.

하지만 이에 그치지 않고 더 좋은 성능을 낼 수 있도록 Grid Search method를 적용시켜 튜닝을 시켜주겠습니다. Grid search(격자 탐색)이란 모델 hyperparameters에 넣을 수 있는 값들을 순차적으로 입력한뒤에 가장 높은 성능을 보이는 hyperparameters를 찾는 탐색 방법입니다. 아래는 이 연구에서 튜닝을 할때 사용한 Grid search 모델과 hyperparameters 값들입니다.

```

from sklearn.model_selection import GridSearchCV

param_grid = {
    'tfidf_lowercase': (True, False),
    'tfidf_ngram_range': [(1, 1), (1, 2)],
    'tfidf_max_df': [0.60, 0.65, 0.70, 0.75, 0.85, 1],
    'clf_C': np.logspace(-4, 4, 10),
}

tfidf_clf = Pipeline([
    ('tfidf', TfidfVectorizer()),
    ('clf', LogisticRegression(penalty='l2', n_jobs=-1, verbose=1)),
])

grid_search = GridSearchCV(
    tfidf_clf, param_grid, cv=ps,
    scoring='accuracy', return_train_score=True,
    n_jobs=-1, verbose=1
)

grid_search.fit(X, y)

```

튜닝후에는 다음과 같은 결과를 얻었습니다.

```

print(grid_search.best_params_)
print(grid_search.best_score_)

best_clf = grid_search.best_estimator_

{'clf_C': 1291.5496650148827, 'tfidf_lowercase': True, 'tfidf_max_df': 0.75, 'tfidf_ngram_range': (1, 2)}
0.94385

```

정확도가 약 2 퍼센트 정도 상승한 것을 볼 수 있습니다.

Grid Search를 실행하면서 느낀 사실은 작업 시간이 매우 오래 걸린다는 것입니다. 특히 저희의 경우 선언한 hyperparameters들의 개수가 여러개였기에 상당히 오래 걸렸습니다.

이제 튜닝된 결과를 바탕으로 'C: 1291.5496650148827(=1300)', 'lowercase: True', 'max\_df: 0.75', 'ngram\_range: (1, 2)'를 hyperparameters로 설정하고 향상된 모델을 훈련시켜 보겠습니다.

```

Val Accuracy: 94.36
Val F1-Score: 94.30

      precision    recall  f1-score   support

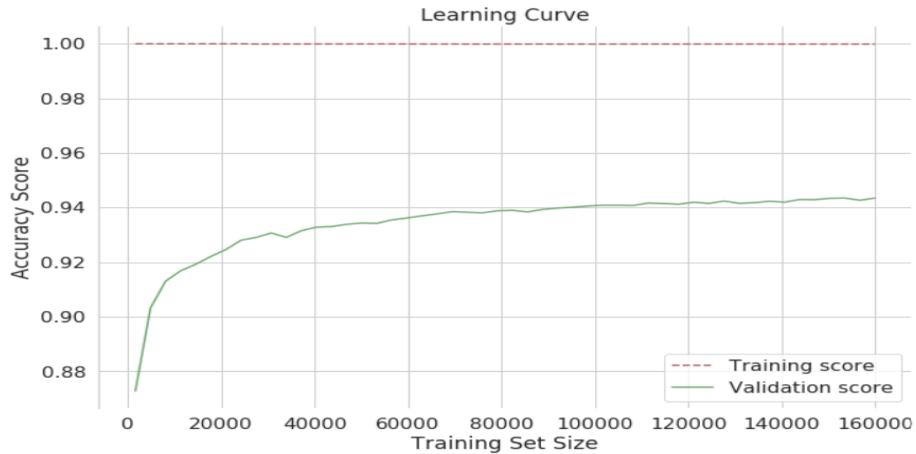
  Negative       0.95      0.93      0.94     10204
  Positive       0.93      0.95      0.94      9796

   accuracy          0.94      0.94      0.94     20000
   macro avg       0.94      0.94      0.94     20000
weighted avg       0.94      0.94      0.94     20000

```

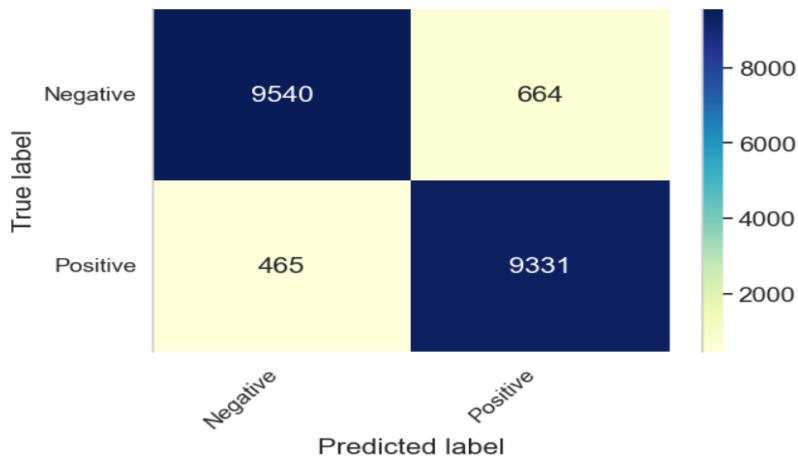
기본 모델과 비교했을때 긍정적 리뷰와 부정적 리뷰 모두 f1-score가 올라간 것이 확인 되었습니다.

그 다음으로는 데이터셋의 크기를 높여가며 크기대비 학습 성능이 얼마나 올라가는지 보여주는 학습 곡선(Learning Curve)을 구현해 보았습니다. 데이터셋의 크기는 전체의 1 퍼센트에 해당하는 1,600개부터 전체 160,000개 까지 50개의 구간으로 나누어 진행을 하였고, 아래 그래프로 결과를 시각화 해보았습니다.

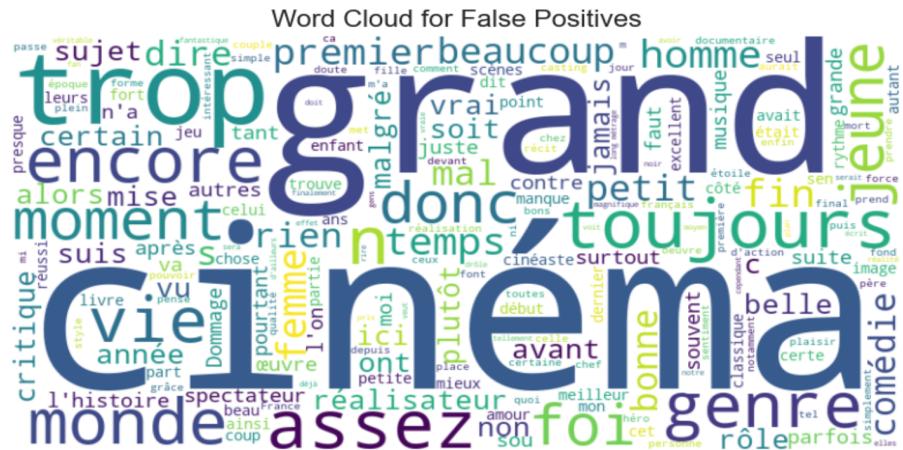


train 데이터셋의 크기가 80000개를 기점으로 성능이 상승하는것이 더뎌지기는 하나, 데이터셋 크기가 커질수록 더 성능이 좋아진다는 사실은 확인할 수 있습니다.

이제 모델이 데이터셋에 대해 만든 prediction들의 총량 중 맞춘 것과 못 맞춘것의 비율, 특히 false negative(모델은 부정적인 리뷰라고 예측했지만 golden label은 긍정인 리뷰)와 false positive(반대로 긍정적이라고 예측했지만 사실은 부정적인 리뷰)가 얼마나 되는지, 혼동행렬(confusion matrix)를 그려 관찰해 보도록 하겠습니다.



관찰결과 false positive predictions이 false negative보다 많은 것으로 관찰되었습니다. 여기서 ”그렇다면 모델이 헷갈려하는 리뷰는 어떤 단어들을 포함하기에 에러가 나는 것일까?”라는 질문을 할 수 있습니다. 이를위해 모델이 예측하는데 성공하지 못한 리뷰들만 뽑아서 단어구름을 만들어 어떤 단어들이 가장 많이 사용되어 혼란을 야기하였는지 살펴보겠습니다.



## Word Cloud for False Negatives



단어 분포만 봤을때, 결과적으로는 모델이 왜 그런 결정을 내렸는지 이해가 가는 부분이었습니다: False positives에 사용된 단어들을 보면, grand(위대한), cinéma(영화(산업)), toujours(항상), moment(순간), premier(첫번째), monde(세상), vie(삶), foi(신념), genre(장르)와 같이 긍정적인 리뷰에서 볼 법한, 긍정적이거나 중립적인 뜻을 가진 단어들이 많았고, 반대로 False negatives에서는 assez(적당히), trop(너무), mal(나쁜), rien(아무것도), l'histoire(이야기)와 같이 일반적으로 부정적 리뷰에서 기대할 만한 단어들이 주를 이루는 것을 볼 수 있었습니다.

Mean Length of Movie Reviews in False Positives: 90.09487951807229

Mean Length of Movie Reviews in False Negatives: 102.11182795698925

나타나는 단어들을 봤을 때, 정확한 예측을 하지 못한 이유가 충분해 보였습니다. 그렇기에 아마도 상당히 길고 복잡한 리뷰 때문에 사용 단어와 문장의 구조가 다양해(예를 들어, 긍정적 리뷰에서 부정적인 단어들을 사용했을 경우/ 그 반대인 경우; 이런 경우는 보통 자세한 코멘트들을 포함하기에 리뷰 길이가 길 것이라고 예상 할 수 있습니다.) 이런 현상이

야기되었다고 생각하고 각각 평균 문장 길이를 살펴 보았으나, 둘 다 전체 평균보다 낮은것이 관찰되었기에 그럴 확률은 높지 않을것 같습니다.

마지막으로, 훈련된 최종 모델을 테스트함으로써 연구를 끝마치도록 하겠습니다.

Test Accuracy: 94.36  
Test F1-Score: 94.16

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| Negative     | 0.95      | 0.94   | 0.95     | 10408   |
| Positive     | 0.94      | 0.95   | 0.94     | 9592    |
| accuracy     |           |        | 0.94     | 20000   |
| macro avg    | 0.94      | 0.94   | 0.94     | 20000   |
| weighted avg | 0.94      | 0.94   | 0.94     | 20000   |

가장 성능이 높게 나온것을 관찰할 수 있습니다.

## 4 결론과 발전 방향

이 연구를 통해 저희는 두 가지를 살펴보았는데, 첫번째는 데이터셋 구축을 통해 데이터를 스크래핑, 전처리 해보고 그 과정에서 나타나는 수치들을 살펴보는 것이었습니다. 스크래핑 부분에 대해서는 본문에서 많이 다루지 않았지만, 추출해와야 했던 데이터 양이 매우 크고 또 그에 필요한 장치들을 코드로 작성하는데 상당한 시간이 걸렸기에 사실상 모델 구현보다 더 복잡했던 작업이었습니다. 하지만 언어 데이터 과학 수업이 지향하는 방향과 일치하는 부분이 많고, 개인적으로도 이런 작업에 사용되는 도구들과 친밀도를 높이고 감각을 키울 수 있었기에 의미가 커다고 생각합니다. 전처리 부분에서도 진행 과정에서 개인적 해석과 결정이 필요한 부분이 꽤 있어서 어떻게하면 더 모델의 정확도를 향상 시키는데에 기여할 수 있는 방향으로 전처리를 할 것인지 생각해 볼 수 있는 기회였다고 생각합니다.

두번째는 모델을 직접 구현해보고 준비한 데이터를 사용해 학습과 시험을 해보는 것 이었습니다. CountVectorizer() 대신 수업 시간에 배운 TF-IDF 개념을 사용해 만들어진 TF-IDFVectorizer()를 사용해 로지스틱 회귀 모델은 만들어 보았는데, 거기에 그치지 않고 Grid Search 기법으로 모델 성능을 업그레이드 했습니다. 놀라웠던 점은, 튜닝을 하지 않은 디폴트 모델조차 매우 좋은 성능을 발휘 했다는 것입니다. 또한, 학습곡선을 통하여 데이터셋 대비 학습속도를 시각화 해보고, 오차분석을 통하여 예측의 분포 확인, 단어 구름을 통하여 잘못된 예측에 대해서 모델이 왜 그런 판단을 했을지 추측해 보았습니다.

이 연구는 확장할 수 있는 방향이 많아보입니다. 완성된 데이터셋에서 다른 통계적 수치(ex. 부정적 리뷰와 긍정적 리뷰 중에서 어떤것이 더 문장의 평균 길이가 긴가? 년도 별로 둘 중 어떤것이 더 많이 작성되었나?)들을 살펴보는 것이 하나가 될 수 있겠고, 데이터셋을 대상으로 여러 다른 모델들을 학습시켜(e.g. BERT, RNN etc.) 학습 속도와 정확도를 비교하는 것도 흥미로울듯 합니다.

## 5 참고문헌

Allociné.fr 사이트 : <https://www.allocine.fr/>