Anime Genre Classification

Final Project
Dohun Kim 2018115809

0. Index

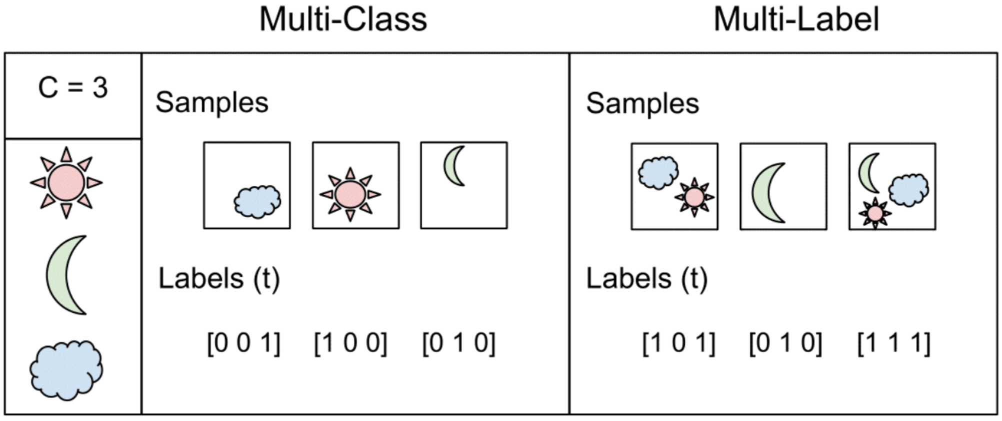
1. Problem Statement
2. Data Generation
3. Data Preprocessing
4. Model Design
5. Conclusion
6. Description of Submission Code

1. Problem Statement

1-1. Predicting Genres of Anime from Synopsis

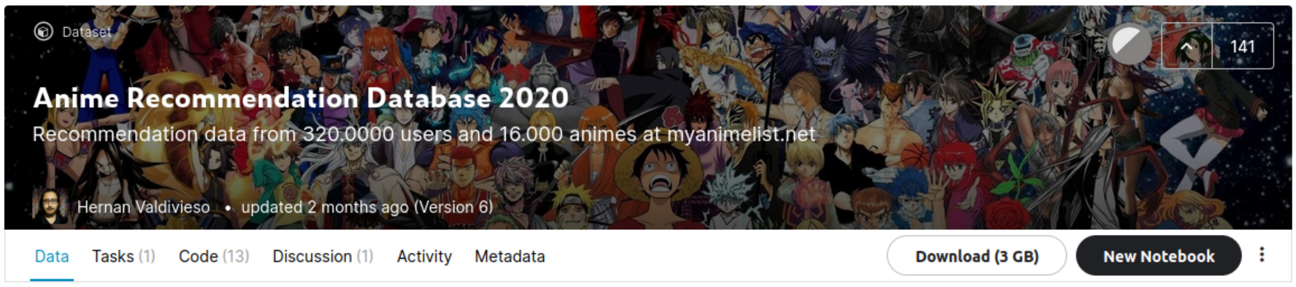
Input: Synopsis	Koyomi Araragi, a third-year high school student, manages to survive a vampire attack with the help of Meme Oshino, a strange man residing in an abandoned building. Though being saved from vampirism and now a human again, several side effects such as superhuman healing abilities and enhanced vision still remain. Regardless, Araragi tries to live the life of a normal student, with the help of his friend and the class president, Tsubasa Hanekawa. When fellow classmate Hitagi Senjougahara falls down the stairs and is caught by Araragi, the boy realizes that the girl is unnaturally weightless. Despite Senjougahara's protests, Araragi insists he help her, deciding to enlist the aid of Oshino, the very man who had once helped him with his own predicament. Through several tales involving demons and gods, Bakemonogatari follows Araragi as he attempts to help those who suffer from supernatural maladies.
Output: Genres	Romance, Supernatural, Mystery, Vampire

1-2. Multi-Label Classification



2. Data Generation

2-1. Original Dataset on Kaggle



Anime Recommendation Database 2020

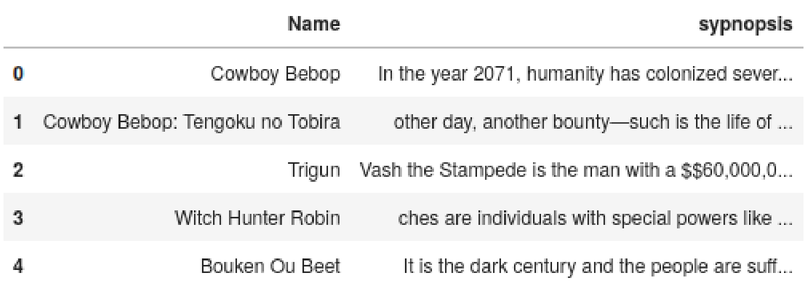
Recommendation data from 320.0000 users and 16.000 animes at myanimelist.net

Hernan Valdivieso • updated 2 months ago (Version 6)

처음에 최종 프로젝트로 선정하였던 데이터셋은 Kaggle의 `Anime Recommendation Database 2020` 입니다. 이 데이터셋은 애니메이션 정보 사이트인 `MyAnimeList.net`으로부터 수집된 애니메이션과 사용자에 대한 정보를 포함합니다. 따라서 데이터셋의 이름과 같이 사용자별 선호도에 따른 애니메이션 추천 모델을 만드는 데에 주로 사용됩니다.

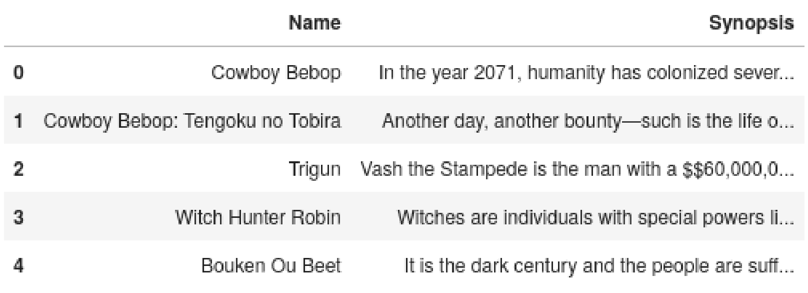
총 3GB 가량의 방대한 용량의 데이터 중 대부분은 유저별 애니메이션 점수 평가에 관한 것이며, 각 애니메이션 자체에 대한 정보는 `anime.csv`, `anime_with_synopsis.csv` 두개로 약 12MB에 그칩니다. 30MB 용량 제한이 주어진 이번 프로젝트에서는 이 두 파일만은 사용하여 애니메이션의 줄거리를 입력받았을 때, 그 애니메이션의 장르를 구분하는 모델을 만들어보고자 하였습니다.

- Original Dataset on Kaggle



	Name	sypnopsis
0	Cowboy Bebop	In the year 2071, humanity has colonized sever...
1	Cowboy Bebop: Tengoku no Tobira	other day, another bounty—such is the life of ...
2	Trigun	Vash the Stampede is the man with a $$60,000,0...
3	Witch Hunter Robin	ches are individuals with special powers like ...
4	Bouken Ou Beet	It is the dark century and the people are suff...

- Correct Data



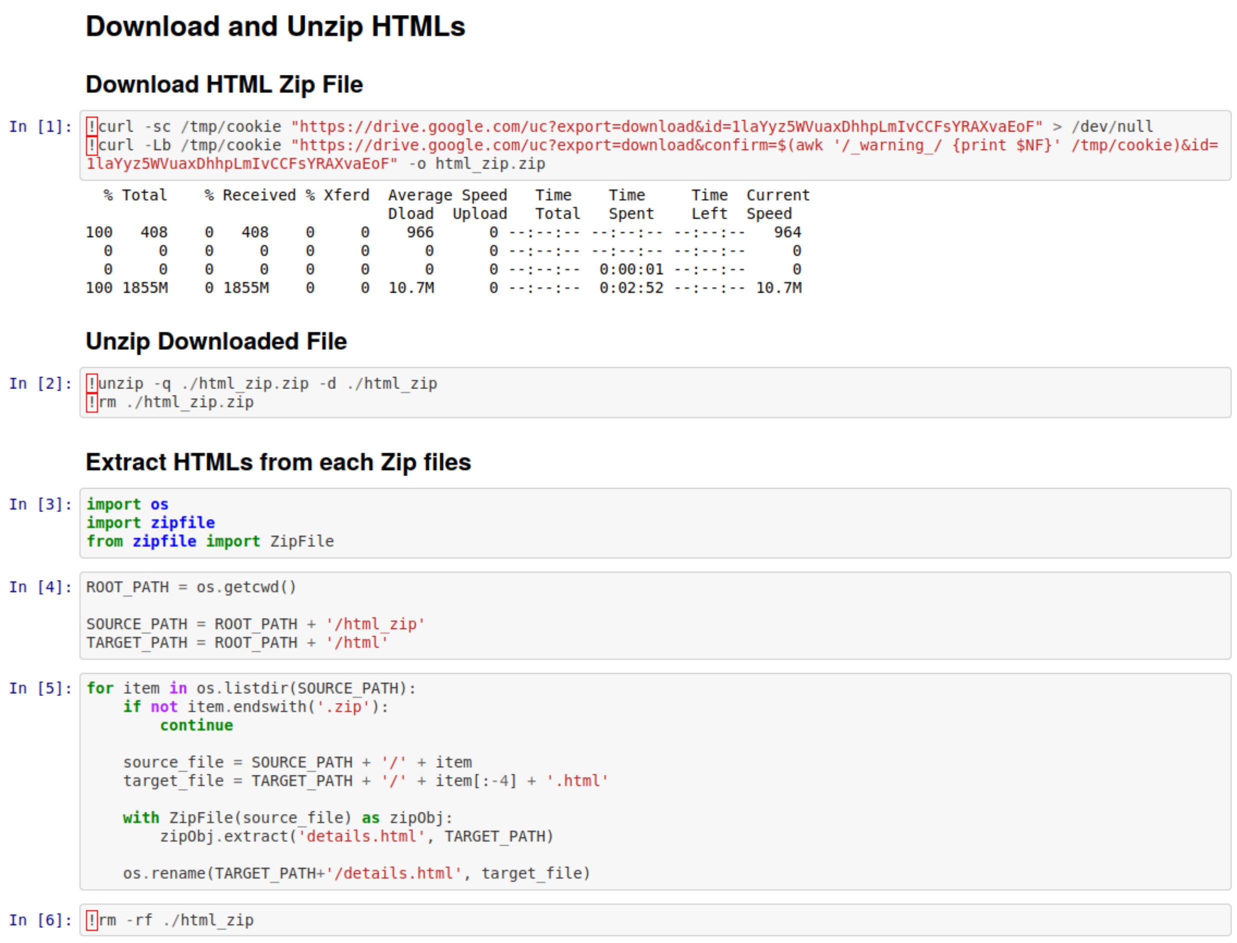
	Name	Synopsis
0	Cowboy Bebop	In the year 2071, humanity has colonized sever...
1	Cowboy Bebop: Tengoku no Tobira	Another day, another bounty—such is the life o...
2	Trigun	Vash the Stampede is the man with a $$60,000,0...
3	Witch Hunter Robin	Witches are individuals with special powers li...
4	Bouken Ou Beet	It is the dark century and the people are suff...

하지만, 코드 작업 진행 중 해당 데이터셋의 줄거리 데이터는 위의 그림과 같이 일부 단어들의 글자가 소실되어있다는 문제점을 발견하였습니다. Kaggle에서 이 데이터셋을 대체할만한 새로운 데이터셋이 마땅하지 않았으며, 이 시점까지 작업이 많이 진행되어있었기 때문에, 자체 데이터셋을 직접 생성해서 사용하기로 결정하였습니다.

2-2. Web Scrapping

`MyAnimeList.net`의 웹페이지 데이터를 수집하기 위해서는, 사이트 보안 차원의 IP 밴 조치를 우회하기 위해 4~7초 정도의 시간 간격이 필요합니다. 총 17000여개의 애니메이션의 상세정보 페이지를 모두 수집하기 위해서는 적어도 24시간이 필요한 작업입니다. 다행히 위 Kaggle 데이터셋의 제작자가 데이터셋을 생성할 때 수집된 모든 원본 html 데이터를 Google Drive 링크를 통해 공유하고 있기 때문에, 이를 활용하여 직접 웹스크래핑을 오랜 시간동안 실행하지 않고 바로 자체 데이터셋 생성을 시작할 수 있었습니다.

get_html.ipynb



Download and Unzip HTMLs

Download HTML Zip File

In [1]:
```
!curl -sc /tmp/cookie "https://drive.google.com/uc?export=download&id=1laYyz5WVuaxDhhpLmIvCCFsYRAXvaEoF" > /dev/null
!curl -Lb /tmp/cookie "https://drive.google.com/uc?export=download&confirm=$(awk '/_warning_/ {print $NF}' /tmp/cookie)&id=
1laYyz5WVuaxDhhpLmIvCCFsYRAXvaEoF" -o html_zip.zip
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   408    0   408    0     0    966      0 --:--:-- --:--:-- --:--:--   964
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
  0     0    0     0    0     0      0      0 --:--:--  0:00:01 --:--:--     0
100 1855M    0 1855M    0     0  10.7M      0 --:--:--  0:02:52 --:--:-- 10.7M
```

Unzip Downloaded File

In [2]:
```
!unzip -q ./html_zip.zip -d ./html_zip
!rm ./html_zip.zip
```

Extract HTMLs from each Zip files

In [3]:
```
import os
import zipfile
from zipfile import ZipFile
```

In [4]:
```
ROOT_PATH = os.getcwd()

SOURCE_PATH = ROOT_PATH + '/html_zip'
TARGET_PATH = ROOT_PATH + '/html'
```

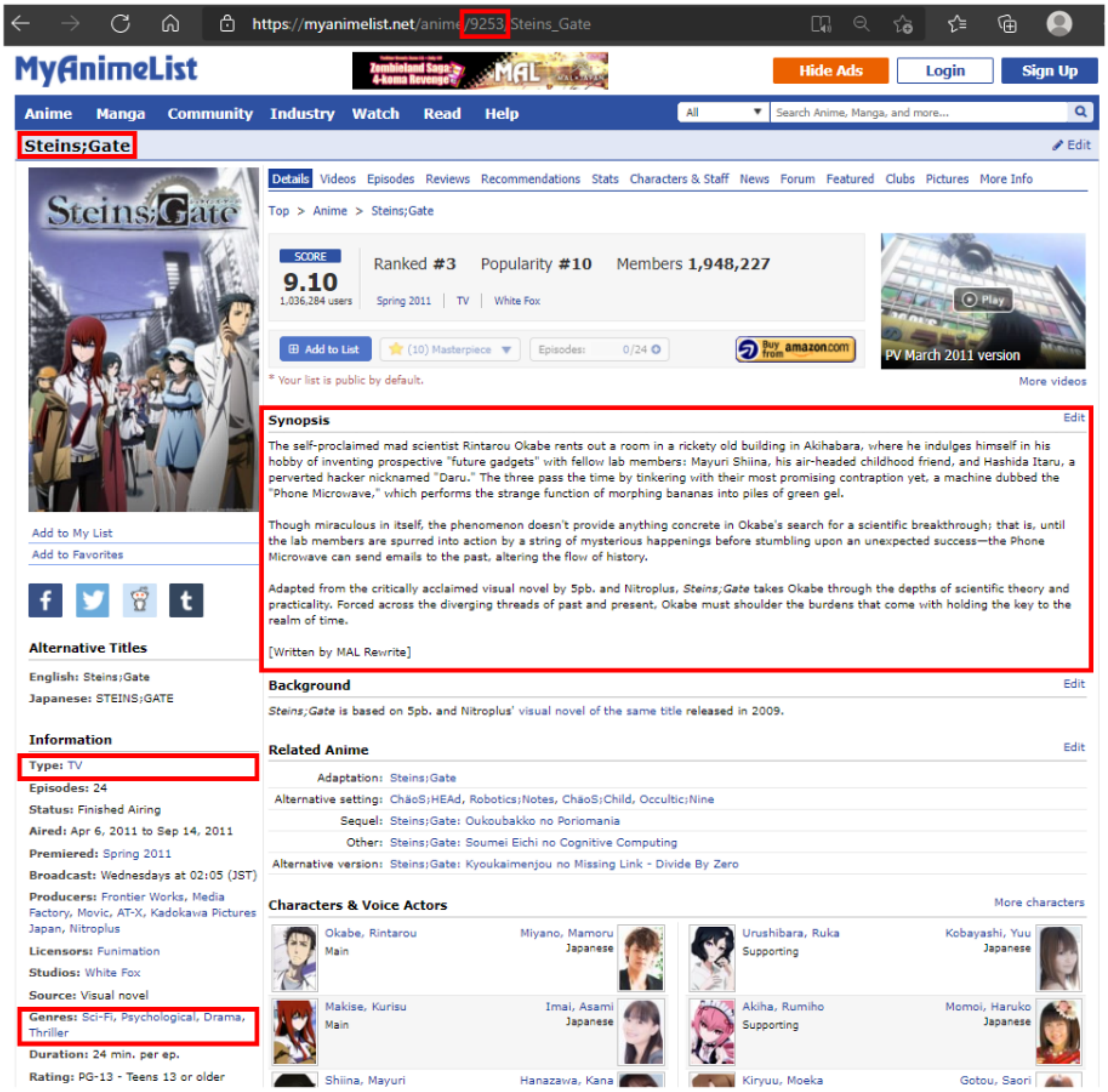
In [5]:
```
for item in os.listdir(SOURCE_PATH):
    if not item.endswith('.zip'):
        continue

    source_file = SOURCE_PATH + '/' + item
    target_file = TARGET_PATH + '/' + item[:-4] + '.html'

    with ZipFile(source_file) as zipObj:
        zipObj.extract('details.html', TARGET_PATH)

    os.rename(TARGET_PATH+'/details.html', target_file)
```

In [6]:
```
!rm -rf ./html_zip
```

Ubuntu 명령어 `curl`을 통해 원본 데이터셋 제작자의 Google Drive 링크로부터 압축 파일을 다운로드하고 압축 해제하여 모든 html 파일들을 저장합니다. 이 노트북을 실행한 후에는 다음과 같이 모든 html 파일들을 포함한 `html` 디렉터리가 생성됩니다.

```
anime-genre-classification/
├─ gen_data/
│  ├─ html/
│  │  ├─ 1.html
│  │  ├─ 5.html
│  │  ├─ ...
│  ├─ generate_data.ipynb
│  ├─ get_html.ipynb
├─ ...
```

2-3. HTML Parsing



Steins;Gate

SCORE 9.10 1,036,284 users

Ranked #3 Popularity #10 Members 1,948,227

Spring 2011 | TV | White Fox

* Your list is public by default.

Synopsis

The self-proclaimed mad scientist Rintarou Okabe rents out a room in a rickety old building in Akihabara, where he indulges himself in his hobby of inventing prospective "future gadgets" with fellow lab members: Mayuri Shiina, his air-headed childhood friend, and Hashida Itaru, a perverted hacker nicknamed "Daru." The three pass the time by tinkering with their most promising contraption yet, a machine dubbed the "Phone Microwave," which performs the strange function of morphing bananas into piles of green gel.

Though miraculous in itself, the phenomenon doesn't provide anything concrete in Okabe's search for a scientific breakthrough; that is, until the lab members are spurred into action by a string of mysterious happenings before stumbling upon an unexpected success—the Phone Microwave can send emails to the past, altering the flow of history.

Adapted from the critically acclaimed visual novel by 5pb. and Nitroplus, *Steins;Gate* takes Okabe through the depths of scientific theory and practicality. Forced across the diverging threads of past and present, Okabe must shoulder the burdens that come with holding the key to the realm of time.

[Written by MAL Rewrite]

Background

Steins;Gate is based on 5pb. and Nitroplus' visual novel of the same title released in 2009.

Related Anime

Adaptation:	Steins;Gate
Alternative setting:	ChäoS;HEAd, Robotics;Notes, ChäoS;Child, Occultic;Nine
Sequel:	Steins;Gate: Oukoubakko no Poriomania
Other:	Steins;Gate: Soumei Eichi no Cognitive Computing
Alternative version:	Steins;Gate: Kyoukaimenjou no Missing Link - Divide By Zero

Characters & Voice Actors

Okabe, Rintarou	Main	Miyano, Mamoru	Japanese
Urushibara, Ruka	Supporting	Kobayashi, Yuu	Japanese
Makise, Kurisu	Main	Imai, Asami	Japanese
Akiha, Rumiho	Supporting	Momoi, Haruko	Japanese
Shiina, Mayuri		Hanazawa, Kana	
Kiryuu, Moeka		Gotou, Saori	

Alternative Titles

English: Steins;Gate
Japanese: STEINS;GATE

Information

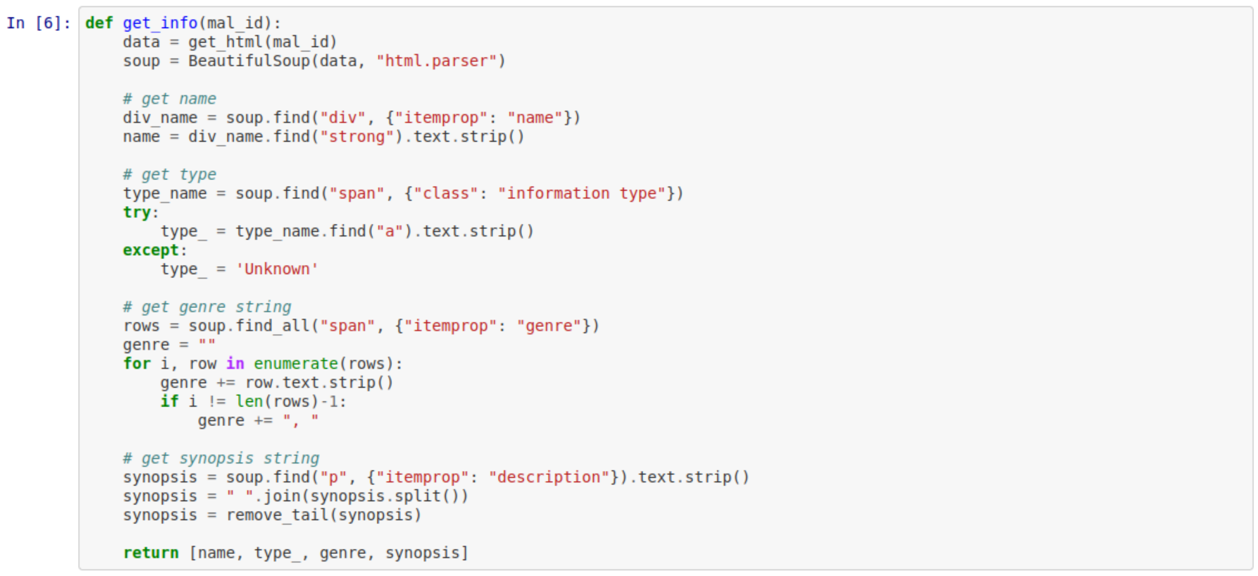
Type: TV
Episodes: 24
Status: Finished Airing
Aired: Apr 6, 2011 to Sep 14, 2011
Premiered: Spring 2011
Broadcast: Wednesdays at 02:05 (JST)
Producers: Frontier Works, Media Factory, Movic, AT-X, Kadokawa Pictures Japan, Nitroplus
Licensors: Funimation
Studios: White Fox
Source: Visual novel
Genres: Sci-Fi, Psychological, Drama, Thriller
Duration: 24 min. per ep.
Rating: PG-13 - Teens 13 or older

generate_data.ipynb

웹 브라우저의 개발자 도구 기능을 통해 html 코드에서 필요한 부분을 찾고, 그 값을 html로부터 `beautifulsoup4` 라이브러리를 통해 추출합니다. 이 노트북을 실행한 후에는 기존의 `html` 디렉터리가 삭제되고 필요한 모든 데이터를 정리한 `anime.csv` 가 생성됩니다.

```
anime-genre-classification/
├─ gen_data/
│  ├─ html/
│  ├─ anime.csv
│  ├─ generate_data.ipynb
│  ├─ get_html.ipynb
├─ ...
```

- generate_data.ipynb (일부분)



```
In [6]: def get_info(mal_id):
            data = get_html(mal_id)
            soup = BeautifulSoup(data, "html.parser")

            # get name
            div_name = soup.find("div", {"itemprop": "name"})
            name = div_name.find("strong").text.strip()

            # get type
            type_name = soup.find("span", {"class": "information type"})
            try:
                type_ = type_name.find("a").text.strip()
            except:
                type_ = 'Unknown'

            # get genre string
            rows = soup.find_all("span", {"itemprop": "genre"})
            genre = ""
            for i, row in enumerate(rows):
                genre += row.text.strip()
                if i != len(rows)-1:
                    genre += ", "

            # get synopsis string
            synopsis = soup.find("p", {"itemprop": "description"}).text.strip()
            synopsis = " ".join(synopsis.split())
            synopsis = remove_tail(synopsis)

            return [name, type_, genre, synopsis]
```

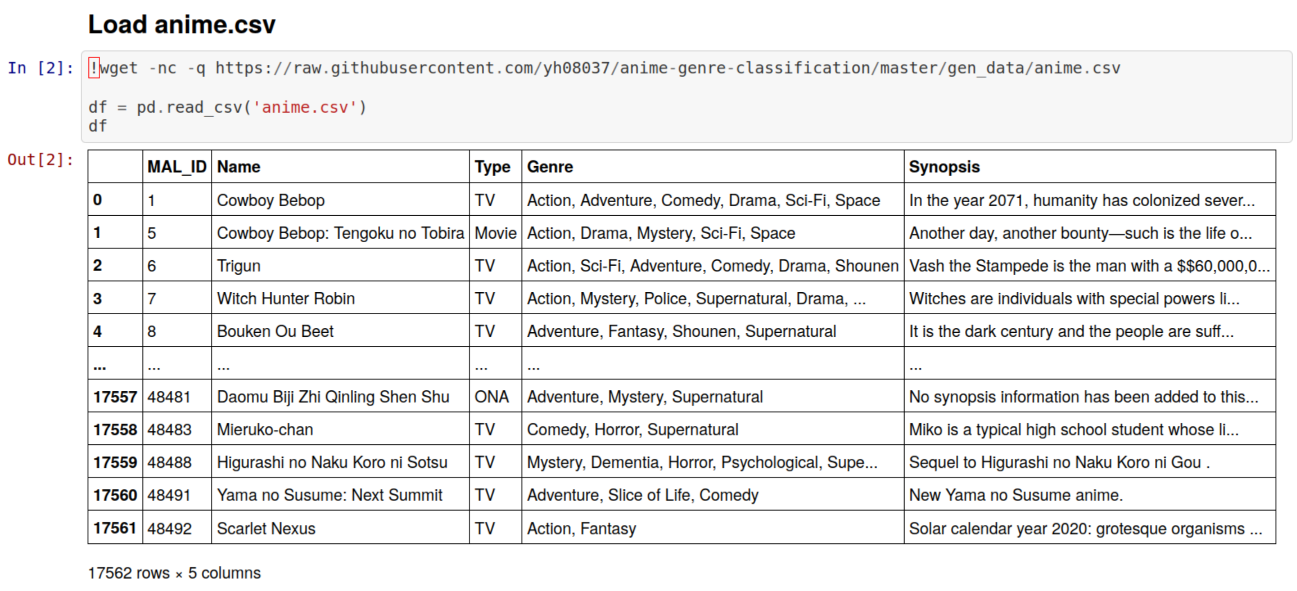
- anime.csv (과제 선정 데이터셋)

MAL_ID	Name	Type	Genre	Synopsis
9253	Steins;Gate	TV	Sci-Fi, Psychological, Drama, Thriller	The self-proclaimed mad scientist Rintarou Okabe rents out a room in a rickety old building in Akihabara, where he indulges himself in his hobby of inventing prospective "future gadgets" with fellow lab members: Mayuri Shiina, his air-headed childhood friend, and Hashida Itaru, a perverted hacker nicknamed "Daru." The three pass the time by tinkering with their most promising contraption yet, a machine dubbed the "Phone Microwave," which performs the strange function of morphing bananas into piles of green gel. Though miraculous in itself, the phenomenon doesn't provide anything concrete in Okabe's search for a scientific breakthrough; that is, until the lab members are spurred into action by a string of mysterious happenings before stumbling upon an unexpected success—the Phone Microwave can send emails to the past, altering the flow of history. Adapted from the critically acclaimed visual novel by 5pb. and Nitroplus, Steins;Gate takes Okabe through the depths of scientific theory and practicality. Forced across the diverging threads of past and present, Okabe must shoulder the burdens that come with holding the key to the realm of time.

3. Data Preprocessing

3-1. Load Data from Github

clean_data.ipynb



Load anime.csv

```
In [2]: !wget -nc -q https://raw.githubusercontent.com/yh08037/anime-genre-classification/master/gen_data/anime.csv

        df = pd.read_csv('anime.csv')
        df
```

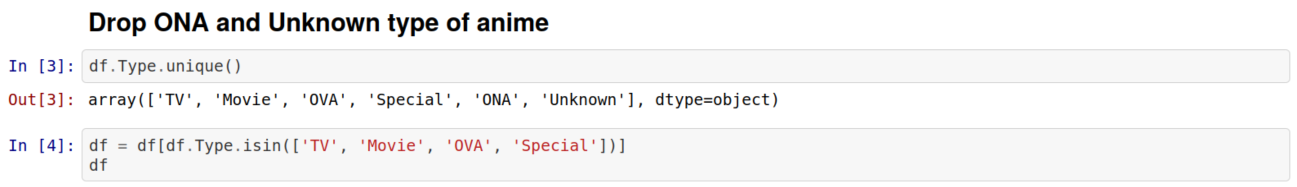
Out[2]:

	MAL_ID	Name	Type	Genre	Synopsis
0	1	Cowboy Bebop	TV	Action, Adventure, Comedy, Drama, Sci-Fi, Space	In the year 2071, humanity has colonized sever...
1	5	Cowboy Bebop: Tengoku no Tobira	Movie	Action, Drama, Mystery, Sci-Fi, Space	Another day, another bounty—such is the life o...
2	6	Trigun	TV	Action, Sci-Fi, Adventure, Comedy, Drama, Shounen	Vash the Stampede is the man with a $$60,000,0...
3	7	Witch Hunter Robin	TV	Action, Mystery, Police, Supernatural, Drama, ...	Witches are individuals with special powers li...
4	8	Bouken Ou Beet	TV	Adventure, Fantasy, Shounen, Supernatural	It is the dark century and the people are suff...
...	...	...	...	...	...
17557	48481	Daomu Biji Zhi Qinling Shen Shu	ONA	Adventure, Mystery, Supernatural	No synopsis information has been added to this...
17558	48483	Mieruko-chan	TV	Comedy, Horror, Supernatural	Miko is a typical high school student whose li...
17559	48488	Higurashi no Naku Koro ni Sotsu	TV	Mystery, Dementia, Horror, Psychological, Supe...	Sequel to Higurashi no Naku Koro ni Gou .
17560	48491	Yama no Susume: Next Summit	TV	Adventure, Slice of Life, Comedy	New Yama no Susume anime.
17561	48492	Scarlet Nexus	TV	Action, Fantasy	Solar calendar year 2020: grotesque organisms ...

17562 rows × 5 columns

앞서 **2. Data Generation** 에서 생성한 `anime.csv` 는 Github를 통해 공개되어 있으며, Ubuntu의 `wget` 명령어를 통해 웹에서 다운로드하여 데이터를 사용합니다. 데이터를 받은 후, 데이터셋에 포함된 애니메이션들 중 Type이 Unknown이거나 ONA인 애니메이션을 제외합니다. 이는 Type이 명시되어있지 않거나 인터넷 애니메이션으로, 이번 문제의 범위에 속하지 않습니다. 그 후 모든 데이터에 대해 결측치, 즉 비어있는 값이 존재하는 행을 제외하였습니다.

clean_data.ipynb



Drop ONA and Unknown type of anime

```
In [3]: df.Type.unique()
Out[3]: array(['TV', 'Movie', 'OVA', 'Special', 'ONA', 'Unknown'], dtype=object)

In [4]: df = df[df.Type.isin(['TV', 'Movie', 'OVA', 'Special'])]
        df
```

- 애니메이션의 구분

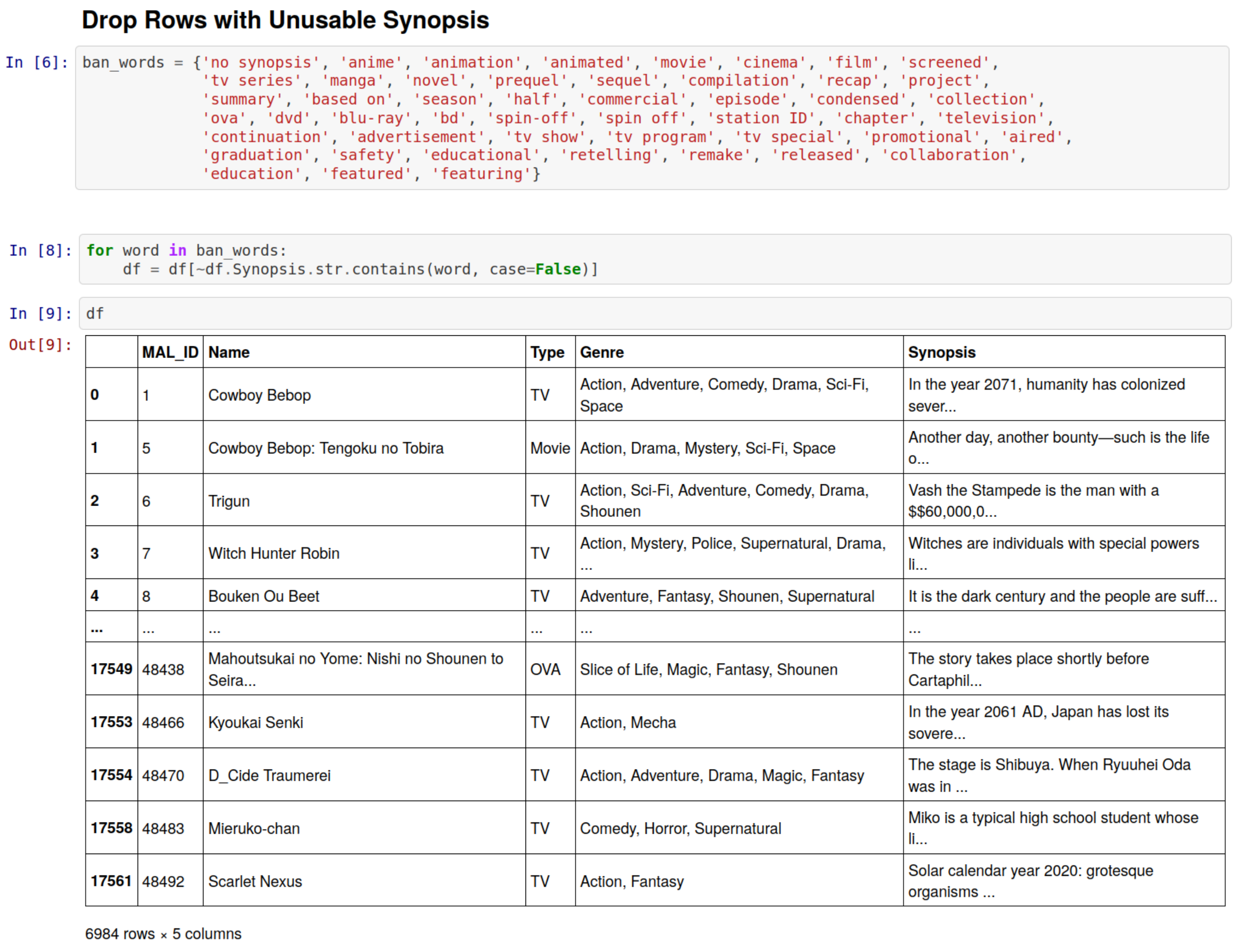
Type	Description
TV	TV 방영 애니메이션
Movie	극장판 애니메이션
Special	행사/특전 제공 애니메이션
OVA	Original Video Animation, DVD/BD를 통해서만 판매되는 애니메이션
ONA	Original Net Animation, 인터넷을 통해서만 공개되는 애니메이션

3-2. Remove Unusable Synopsis

Examples of unusable synopsis	Counts
Contains "no synopsis" (ex: No synopsis has been added for this series yet.)	665
Contains "sequel". (ex: Sequel to the first series.)	116
Contains "season". (ex: Second season of One Room.)	535
Contains "recap". (ex: Recap of the first 12 episodes of Carole & Tuesday.)	270
Contains "summary". (ex: A summary of the full series aired as a TV specials.)	76
Contains "educational" (ex: An educational film teaching about traffic safety within a city.)	229
Contains "collection" (ex: A collection of short stories, all of which are perverted to various degrees.)	98
...	...

MyAnimeList.net 에서 수집한 줄거리 정보 중 위의 경우와 같이 실제 줄거리가 아닌 작품 자체의 정보에 해당하는 경우가 상당수 존재합니다. 예를 들어 "줄거리 정보가 없습니다."라던가 "-의 후속작입니다." 등 작품 외적인 정보를 기술하는 경우 학습 대상에서 제외하였습니다.

clean_data.ipynb



Drop Rows with Unusable Synopsis

```
In [6]: ban_words = {'no synopsis', 'anime', 'animation', 'animated', 'movie', 'cinema', 'film', 'screened',
                     'tv series', 'manga', 'novel', 'prequel', 'sequel', 'compilation', 'recap', 'project',
                     'summary', 'based on', 'season', 'half', 'commercial', 'episode', 'condensed', 'collection',
                     'ova', 'dvd', 'blu-ray', 'bd', 'spin-off', 'spin off', 'station ID', 'chapter', 'television',
                     'continuation', 'advertisement', 'tv show', 'tv program', 'tv special', 'promotional', 'aired',
                     'graduation', 'safety', 'educational', 'retelling', 'remake', 'released', 'collaboration',
                     'education', 'featured', 'featuring'}
```

```
In [8]: for word in ban_words:
            df = df[~df.Synopsis.str.contains(word, case=False)]
```

```
In [9]: df
```

Out[9]:

	MAL_ID	Name	Type	Genre	Synopsis
0	1	Cowboy Bebop	TV	Action, Adventure, Comedy, Drama, Sci-Fi, Space	In the year 2071, humanity has colonized sever...
1	5	Cowboy Bebop: Tengoku no Tobira	Movie	Action, Drama, Mystery, Sci-Fi, Space	Another day, another bounty—such is the life o...
2	6	Trigun	TV	Action, Sci-Fi, Adventure, Comedy, Drama, Shounen	Vash the Stampede is the man with a $$60,000,0...
3	7	Witch Hunter Robin	TV	Action, Mystery, Police, Supernatural, Drama, ...	Witches are individuals with special powers li...
4	8	Bouken Ou Beet	TV	Adventure, Fantasy, Shounen, Supernatural	It is the dark century and the people are suff...
...	...	...	...	...	...
17549	48438	Mahoutsukai no Yome: Nishi no Shounen to Seira...	OVA	Slice of Life, Magic, Fantasy, Shounen	The story takes place shortly before Cartaphil...
17553	48466	Kyoukai Senki	TV	Action, Mecha	In the year 2061 AD, Japan has lost its sovere...
17554	48470	D_Cide Traumerei	TV	Action, Adventure, Drama, Magic, Fantasy	The stage is Shibuya. When Ryuuhei Oda was in ...
17558	48483	Mieruko-chan	TV	Comedy, Horror, Supernatural	Miko is a typical high school student whose li...
17561	48492	Scarlet Nexus	TV	Action, Fantasy	Solar calendar year 2020: grotesque organisms ...

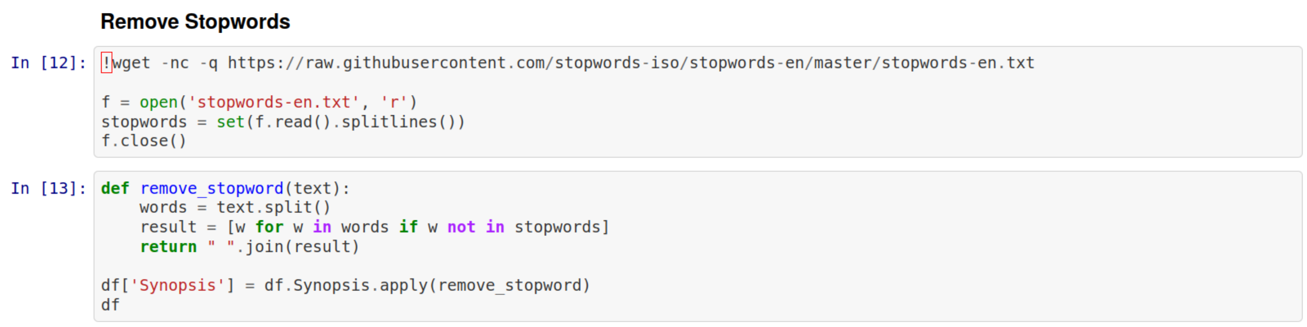
6984 rows × 5 columns

3-3. Text Cleaning

Methods	Example Result
Original Synopsis	In the year 2071, humanity has colonized sever...
To lower case	in the year 2071, humanity has colonized sever...
Remove non-alphabetic characters	in the year humanity has colonized sever...
Remove stopwords	humanity colonized planets moons solar leaving...
Remove non-PTB words	solar leaving surface planet earth solar polic...

줄거리 데이터에 CBOW 등의 Word2Vec 모델을 적용하기 위해서는 먼저 줄거리 텍스트를 정리해 줄 필요가 있습니다. 소문자로 만들기, 특수문자 제거하기, 불용어 제거하기, PTB 데이터셋에 없는 어휘 제거하기를 순서대로 진행하였습니다.

clean_data.ipynb



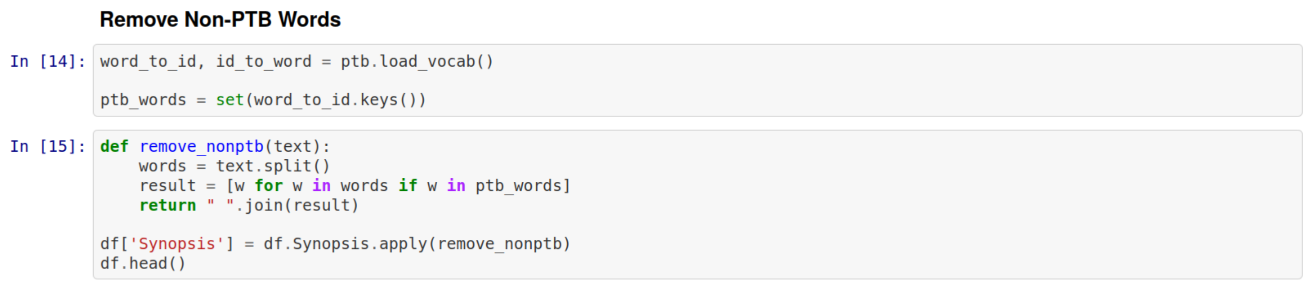
Remove Stopwords

```
In [12]: !wget -nc -q https://raw.githubusercontent.com/stopwords-iso/stopwords-en/master/stopwords-en.txt

         f = open('stopwords-en.txt', 'r')
         stopwords = set(f.read().splitlines())
         f.close()
```

```
In [13]: def remove_stopword(text):
             words = text.split()
             result = [w for w in words if w not in stopwords]
             return " ".join(result)

         df['Synopsis'] = df.Synopsis.apply(remove_stopword)
         df
```

줄거리 텍스트로 부터 불용어를 처리하기 위해서 `stopwords-iso` 에서 제공하는 영어의 불용어 리스트를 공식 Github repo에서 `wget` 명령을 통해 위와 같이 웹에서 다운로드하여 사용하였습니다.

clean_data.ipynb



Remove Non-PTB Words

```
In [14]: word_to_id, id_to_word = ptb.load_vocab()

         ptb_words = set(word_to_id.keys())
```

```
In [15]: def remove_nonptb(text):
             words = text.split()
             result = [w for w in words if w in ptb_words]
             return " ".join(result)

         df['Synopsis'] = df.Synopsis.apply(remove_nonptb)
         df.head()
```

또한 교재의 Github repo에서 제공하는 PTB 데이터셋에 사전 학습된 CBOW 모델 파라미터 pickle 파일 또한 `wget` 명령을 통해 위와 같이 웹에서 다운로드하여 어휘 집합을 사용하였습니다.

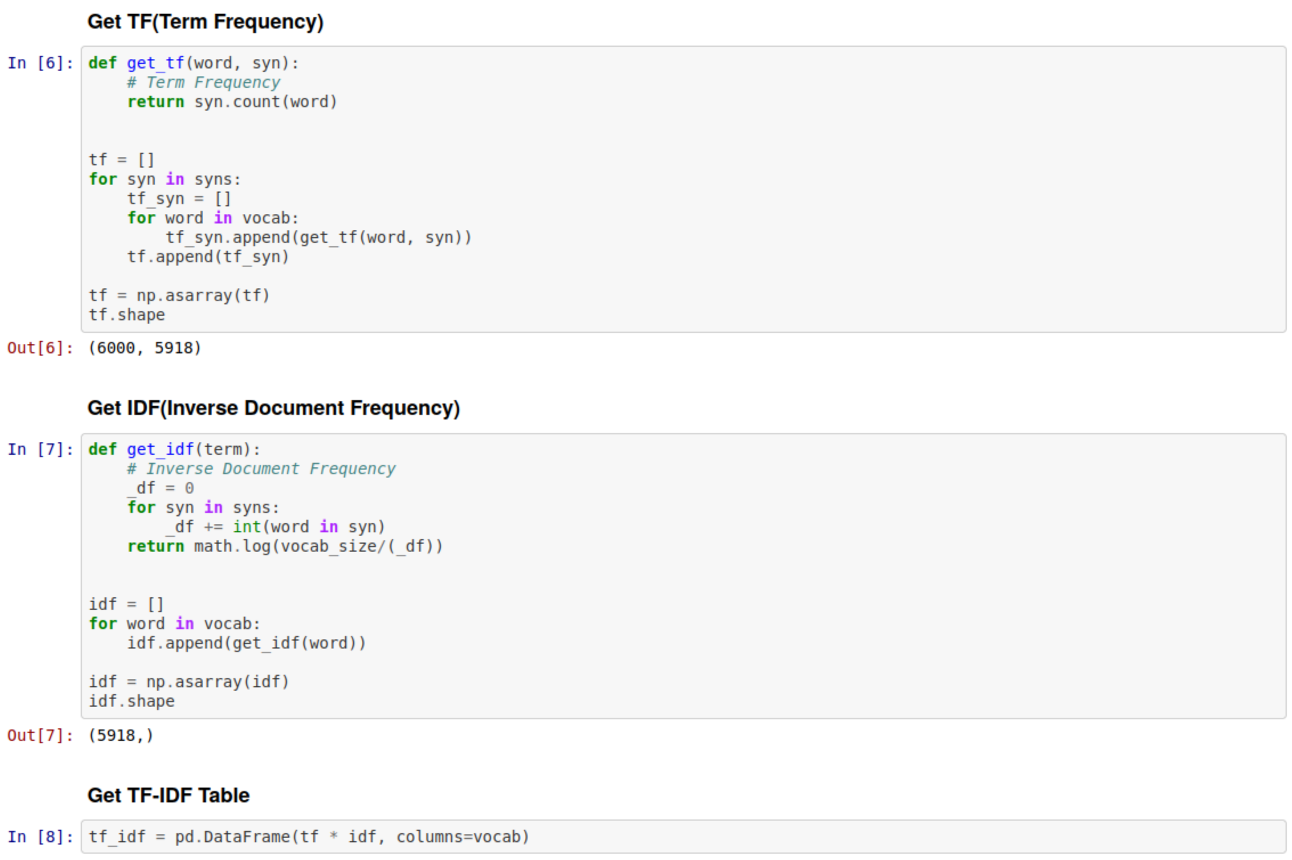
3-4. Synopsis to Vector

강의 내용을 에서 배운 Word2Vec의 CBOW, Skipgram의 경우 한 단어를 고정된 길이의 벡터로 변환하고, RNNLM의 경우 이전 단어로부터 다음 단어를 예측하므로, 전체 문장이 표현하는 장르를 예측하는 모델을 만들기 어려울 것이라고 판단하였습니다. 여기서 Word2Vec에서와 같이 한 줄거리 텍스트 전체를 하나의 고정 길이 벡터에 대응시키는 방법을 떠올리게 되었습니다. 한 문장 내의 각 단어들의 Word2Vec의 평균으로 해당 문장에 대응하는 문장 벡터를 만들어낼 수 있으며, 단순 평균이 아닌 TF-IDF라는 값을 가중치로써 곱하여 평균을 구하면 더 효과적임을 알게되었습니다.

TF-IDF

Term Frequency - Inverse Document Frequency의 약자로, TF와 IDF의 곱으로 구해집니다. TF는 특정 문서 d에서의 특정 단어 t의 등장 횟수이며, IDF는 특정 단어 t가 등장한 문서의 수에 반비례하는 수입니다. TF-IDF는 모든 문서에서 자주 등장하는 단어는 중요도가 낮다고 판단하며, 특정 문서에서만 자주 등장하는 단어는 중요도가 높다고 판단합니다. TF-IDF 값이 낮으면 중요도가 낮은 것이며, TF-IDF 값이 크면 중요도가 큰 것입니다.

synopsis_vector.ipynb



Get TF(Term Frequency)

```
In [6]: def get_tf(word, syn):
            # Term Frequency
            return syn.count(word)


        tf = []
        for syn in syns:
            tf_syn = []
            for word in vocab:
                tf_syn.append(get_tf(word, syn))
            tf.append(tf_syn)

        tf = np.asarray(tf)
        tf.shape

Out[6]: (6000, 5918)
```

Get IDF(Inverse Document Frequency)

```
In [7]: def get_idf(term):
            # Inverse Document Frequency
            _df = 0
            for syn in syns:
                _df += int(word in syn)
            return math.log(vocab_size/(_df))


        idf = []
        for word in vocab:
            idf.append(get_idf(word))

        idf = np.asarray(idf)
        idf.shape

Out[7]: (5918,)
```

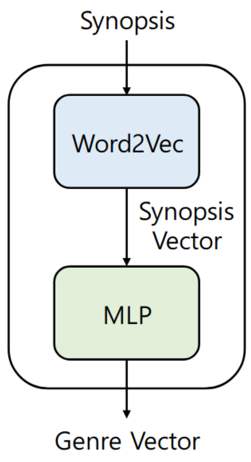
Get TF-IDF Table

```
In [8]: tf_idf = pd.DataFrame(tf * idf, columns=vocab)
```

4. Model Design

4-1. MLP with Transfer Learning

- Word2Vec: Frozen
  - Pre-trained CBOW model with PTB dataset
  - Only for feature extraction
  - Input: Synopsis - plain text data
  - Output: Sentence vector of synopsis
- MLP: Trainable
  - Learnable MLP classifier
  - Input : Sentence vector of a synopsis
  - Output : Probabilities of each genre



애니메이션 장르 분류를 위한 모델을 위의 그림과 같이 설계하였습니다. 먼저 앞서 본 것과 같이 TF-IDF를 곱한 Word2Vec의 평균으로 줄거리에 대한 고정 길이 벡터(줄거리 벡터)를 생성하는 고정된 모델과, 이 줄거리 벡터를 바탕으로 장르를 예측하는 학습 가능한 다층 퍼셉트론 모델의 조합으로, 특징 추출을 위해 전이 학습을 사용하는 것이라고 해석할 수 있습니다.

4-2. Issue 1: Multi-Label Classification

모델을 작성하는 과정에서 첫번째로 만난 난관은 풀려고 하는 문제가 Multi-Class 분류가 아닌 Multi-Label 분류 문제라는 것입니다. Multi-Class 분류 문제는 정답이 단 한개 존재하는, 즉 one-hot 벡터로 표현되는 경우를 말하며, Multi-Label 분류 문제는 정답이 여러개 존재할 수 있는 문제입니다.

for Multi-Class Classification

$$\begin{aligned} \hat{y_i} &= \mathrm{softmax}(x_i) \\ \mathrm{CCELoss} &= -\sum_i y_i \log(\hat{y_i}) \end{aligned}$$

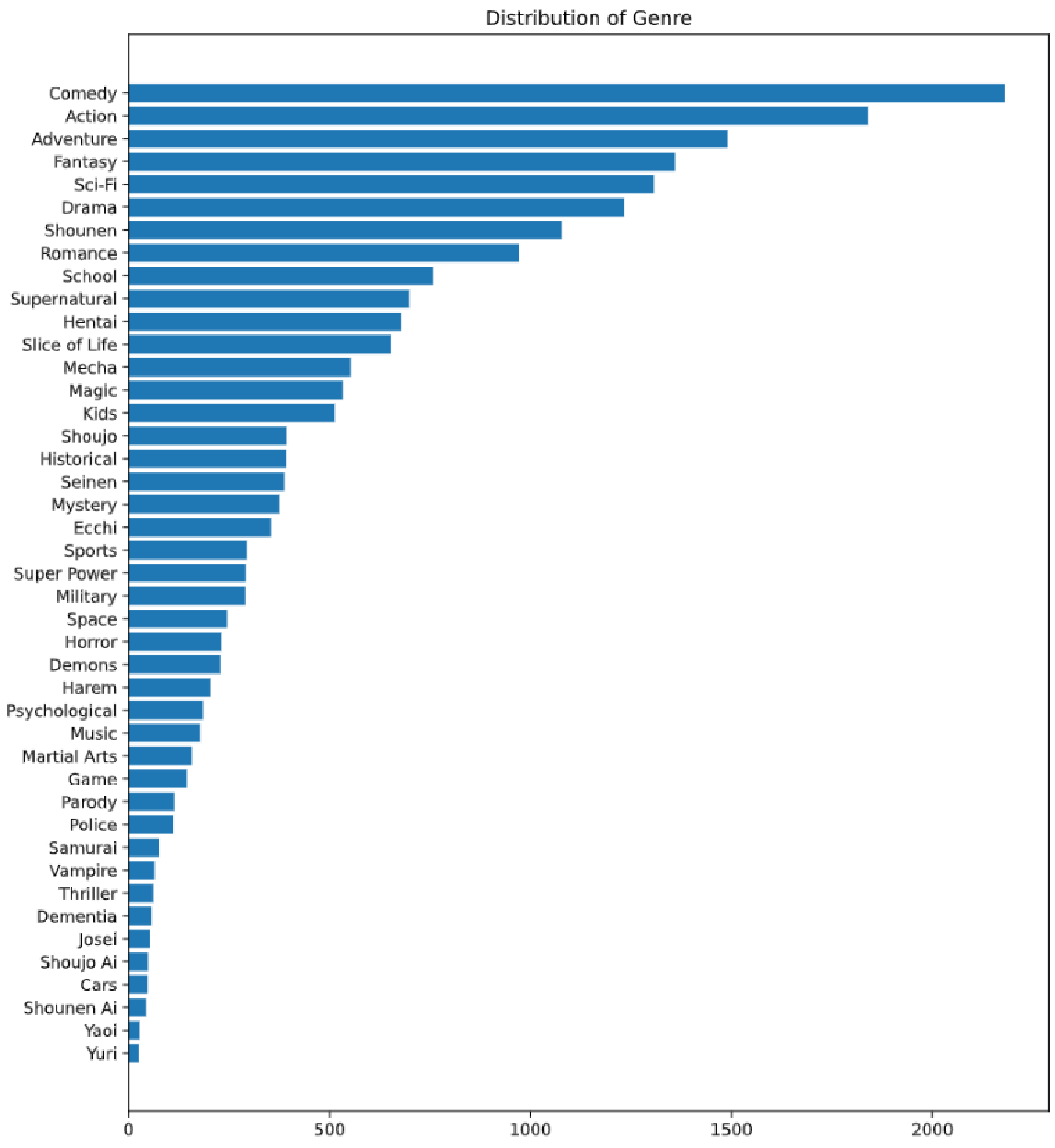
for Multi-Label Classification

$$\begin{aligned} \hat{y_i} &= \mathrm{sigmoid}(x_i) \\ \mathrm{BCELoss}(\hat{y_i}, y_i) &= -\left(y_i \log(\hat{y_i}) + (1-y_i)\log(1-\hat{y_i})\right) \\ \text{Total Loss} &= \sum_i \mathrm{BCELoss}(\hat{y_i}, y_i) \end{aligned}$$

Multi-Label 분류에서는 기존에 Multi-Class 분류에 사용하던 Softmax와 Categorical Cross Entropy Loss의 조합은 사용할 수 없습니다. 그 대신 각각의 label에 대해서 $0/1$ 여부를 예측하는 Binary 분류를 수행하는 것으로 생각할 수 있습니다. 따라서 활성화 함수로 Softmax 대신 Sigmoid를 사용하고, 손실 함수로 Label별 Binary Cross Entropy의 합을 사용할 수 있습니다.

이 경우 모델의 출력은 각 label에 대해 해당 label이 1일 확률이며, 따라서 0.5와 같은 경계값을 기준으로 확률값으로부터 0 또는 1을 결정할 수 있습니다.

4-3 Issue 2: Class Imbalance



두번째로 만난 난관은 매우 큰 클래스의 불균형입니다. 위의 그림은 전체 애니메이션에 대한 장르의 분포입니다. 이러한 불균형이 큰 데이터를 그대로 학습에 사용하였을 때는 데이터의 수가 많은 몇개의 장르가 나머지를 압도하여 제대로 학습이 진행되지 않았습니다.

이 문제에 대한 해결 방안으로 먼저 데이터를 조작하여 모든 장르의 비율을 맞추어 주는 방법이 있습니다. 개수가 많은 장르를 개수가 적은 장르에 맞추어 사용하는 under sampling이나, 개수가 많은 장르에 개수가 적은 장르를 반복하여 맞추는 over sampling 등이 있는데, 데이터의 양이 충분하지 않고 양의 차이가 너무 크게 나서 사용하기 곤란한 방법이었습니다.

다음으로는 손실 함수를 변경하여 해결하는 방법이 있습니다. 앞서 확인했던 Binary Cross Entropy Loss에 label별로 가중치를 다르게 두는 Weighted BCELoss와, 각 label에 focal modulation factor를 곱하는 Focal Loss 등을 사용하면 클래스 불균형 문제를 해결할 수 있는데, 구현의 난이도와 이론적 정밀성을 검토하여 Focal Loss를 사용하기로 정하였습니다.

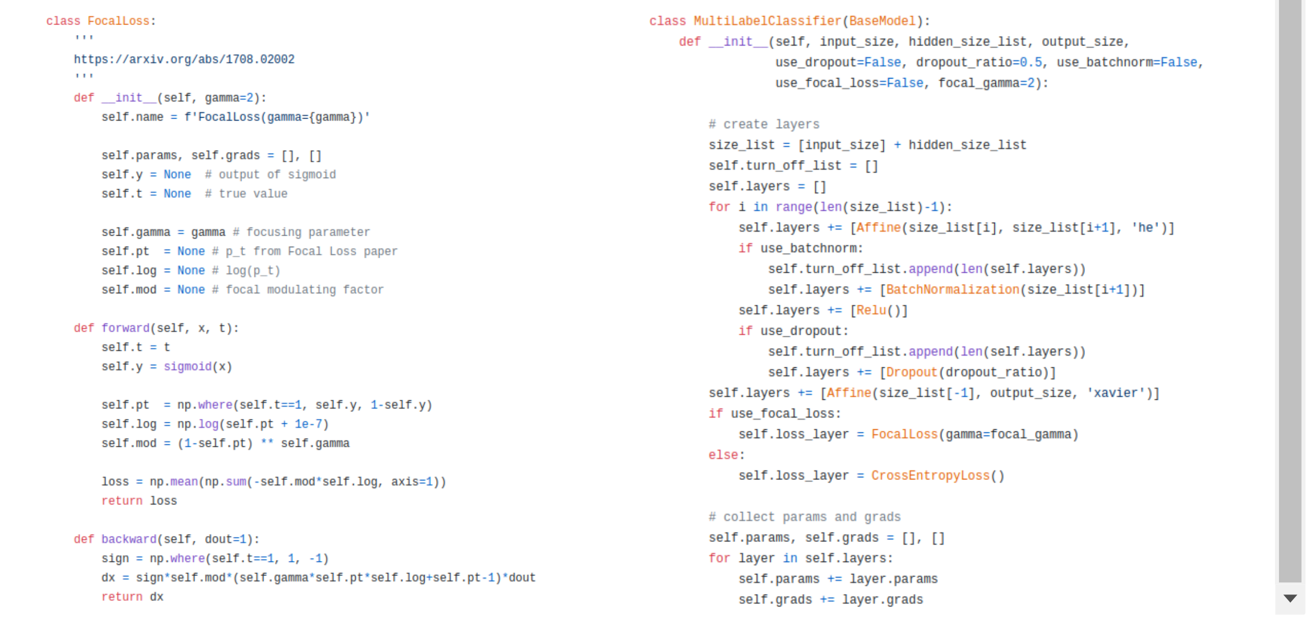
4-4. Focal Loss

Focal Loss에 대한 정확한 정의와 미분식을 확인하기 위해 arXiv에 게재된 논문 ***Focal Loss for Dense Object Detection*** 을 읽고 정리하였습니다.

$$
\begin{aligned}
\mathrm{CE}(p_t) &= -\log(p_t) \\
\mathrm{FL}(p_t) &= -(1-p_t)^\gamma \log(p_t)
\end{aligned}
\qquad\qquad
p_t = \begin{cases} p, & \text{if } y = 1 \\ 1-p, & \text{otherwise} \end{cases}
$$

Focal Loss는 object detection 시나리오에서 훈련 중에 극단적인 클래스 불균형이 존재할 경우를 위해 만들어졌습니다. 기존의 일반적인 방법은 Cross Entropy에 weight를 도입한 Balanced Cross Entropy인데, Focal Loss는 여기서 Cross Entropy에 weight 대신 modulating factor $(1-p_t)^\gamma$를 도입합니다. 이는 훈련 중에 쉬운 label의 가중치를 낮추어, 손실 함수가 어려운 label에 좀 더 집중할 수 있게 해줍니다.

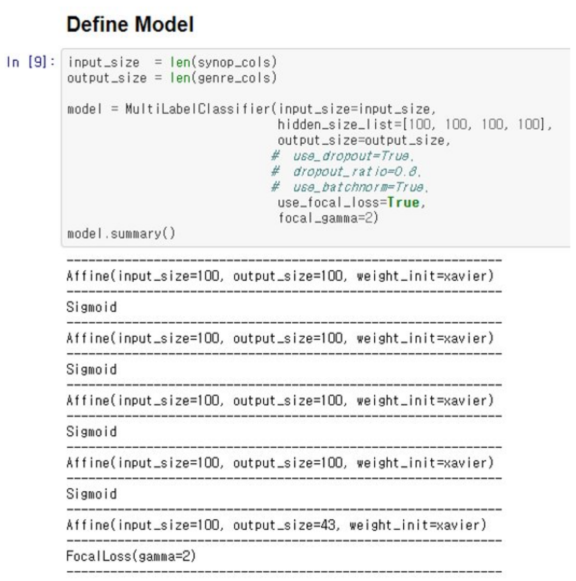
anime_classification.py



```python
class FocalLoss:
    '''
    https://arxiv.org/abs/1708.02002
    '''
    def __init__(self, gamma=2):
        self.name = f'FocalLoss(gamma={gamma})'

        self.params, self.grads = [], []
        self.y = None  # output of sigmoid
        self.t = None  # true value

        self.gamma = gamma # focusing parameter
        self.pt  = None # p_t from Focal Loss paper
        self.log = None # log(p_t)
        self.mod = None # focal modulating factor

    def forward(self, x, t):
        self.t = t
        self.y = sigmoid(x)

        self.pt  = np.where(self.t==1, self.y, 1-self.y)
        self.log = np.log(self.pt + 1e-7)
        self.mod = (1-self.pt) ** self.gamma

        loss = np.mean(np.sum(-self.mod*self.log, axis=1))
        return loss

    def backward(self, dout=1):
        sign = np.where(self.t==1, 1, -1)
        dx = sign*self.mod*(self.gamma*self.pt*self.log+self.pt-1)*dout
        return dx
```

```python
class MultiLabelClassifier(BaseModel):
    def __init__(self, input_size, hidden_size_list, output_size,
                 use_dropout=False, dropout_ratio=0.5, use_batchnorm=False,
                 use_focal_loss=False, focal_gamma=2):

        # create layers
        size_list = [input_size] + hidden_size_list
        self.turn_off_list = []
        self.layers = []
        for i in range(len(size_list)-1):
            self.layers += [Affine(size_list[i], size_list[i+1], 'he')]
            if use_batchnorm:
                self.turn_off_list.append(len(self.layers))
                self.layers += [BatchNormalization(size_list[i+1])]
            self.layers += [Relu()]
            if use_dropout:
                self.turn_off_list.append(len(self.layers))
                self.layers += [Dropout(dropout_ratio)]
        self.layers += [Affine(size_list[-1], output_size, 'xavier')]
        if use_focal_loss:
            self.loss_layer = FocalLoss(gamma=focal_gamma)
        else:
            self.loss_layer = CrossEntropyLoss()

        # collect params and grads
        self.params, self.grads = [], []
        for layer in self.layers:
            self.params += layer.params
            self.grads += layer.grads
```

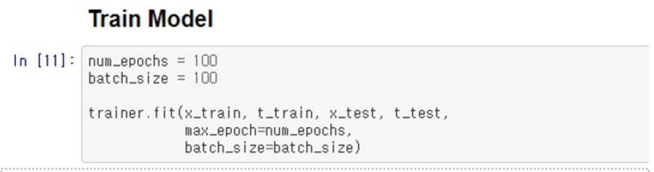
model.ipynb



Define Model

```
In [9]: input_size  = len(synop_cols)
        output_size = len(genre_cols)

        model = MultiLabelClassifier(input_size=input_size,
                                     hidden_size_list=[100, 100, 100, 100],
                                     output_size=output_size,
                                    #  use_dropout=True,
                                    #  dropout_ratio=0.8,
                                    #  use_batchnorm=True,
                                     use_focal_loss=True,
                                     focal_gamma=2)
        model.summary()
```

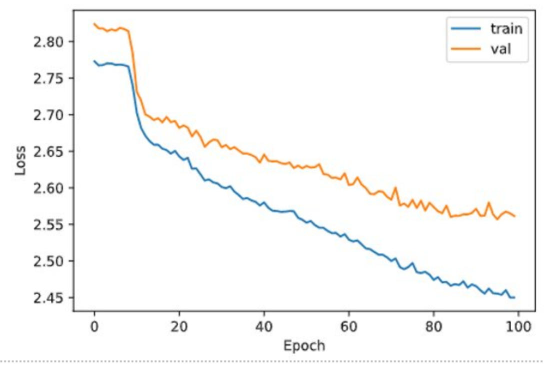
```
------------------------------------------------------------
Affine(input_size=100, output_size=100, weight_init=xavier)
------------------------------------------------------------
Sigmoid
------------------------------------------------------------
Affine(input_size=100, output_size=100, weight_init=xavier)
------------------------------------------------------------
Sigmoid
------------------------------------------------------------
Affine(input_size=100, output_size=100, weight_init=xavier)
------------------------------------------------------------
Sigmoid
------------------------------------------------------------
Affine(input_size=100, output_size=100, weight_init=xavier)
------------------------------------------------------------
Sigmoid
------------------------------------------------------------
Affine(input_size=100, output_size=43, weight_init=xavier)
------------------------------------------------------------
FocalLoss(gamma=2)
------------------------------------------------------------
```



Train Model

```
In [11]: num_epochs = 100
         batch_size = 100

         trainer.fit(x_train, t_train, x_test, t_test,
                     max_epoch=num_epochs,
                     batch_size=batch_size)
```

```
In [12]: trainer.plot()
```



5. Conclusion

5-1. Result

앞장의 그림과 같이 train loss와 validation loss가 함께 줄어들어 그래프 상 학습이 잘 된 것처럼 보이지만, 모델의 실제 prediction의 정확도는 매우 좋지 않습니다. 아래는 test dataset의 줄거리 벡터를 입력했을 때 모델의 출력을 높은 순으로 정렬하고, 해당 label에 대한 참값을 병기한 것입니다.

ex1 $(\gamma = 2)$

```
Action         47.01% 1
Sci-Fi         43.11% 0
Comedy         38.46% 0
Adventure      36.39% 0
Drama          36.06% 0
Shounen        31.63% 0
Mecha          30.58% 0
Hentai         29.33% 0
Fantasy        29.25% 0
Romance        28.87% 0
Military       26.51% 0
Historical     24.69% 0
Seinen         24.63% 0
Supernatural   23.13% 0
Mystery        23.04% 0
School         21.82% 0
Space          20.24% 0
Super Power    19.52% 0
Slice of Life  19.06% 0
Magic          18.11% 0
Ecchi          18.06% 0
Police         17.47% 0
Kids           17.09% 0
Horror         15.39% 0
Sports         14.20% 0
Psychological  14.17% 0
Shoujo         13.04% 0
Martial Arts   13.03% 0
Music          12.96% 1
Harem          12.37% 0
Demons         11.38% 0
Parody          9.41% 0
Game            8.76% 0
Samurai         8.24% 0
Thriller        6.50% 0
Vampire         6.28% 0
Cars            6.04% 0
Yuri            4.75% 1
Dementia        4.64% 0
Josei           4.39% 0
Shoujo Ai       4.05% 0
Shounen Ai      3.12% 0
Yaoi            2.73% 0
```

ex2 $(\gamma = 2)$

```
Sci-Fi         52.58% 0
Adventure      50.94% 0
Comedy         46.04% 0
Fantasy        41.06% 0
Action         40.81% 0
Mecha          34.78% 0
Kids           34.04% 0
Shounen        32.05% 0
Magic          21.68% 0
Drama          20.00% 0
Space          16.07% 0
Slice of Life  13.09% 0
Romance        12.50% 0
Super Power    11.57% 0
Shoujo         11.14% 0
Ecchi          10.51% 0
School         10.22% 0
Supernatural    9.82% 0
Hentai          9.19% 1
Music           8.75% 0
Military        7.98% 0
Seinen          7.65% 0
Historical      6.86% 0
Sports          6.05% 0
Parody          5.08% 0
Mystery         4.63% 0
Demons          4.52% 0
Martial Arts    4.13% 0
Horror          3.86% 1
Harem           3.39% 0
Game            3.24% 0
Police          2.27% 0
Cars            2.13% 0
Psychological   1.90% 0
Vampire         0.87% 0
Dementia        0.66% 0
Samurai         0.48% 0
Josei           0.40% 0
Yaoi            0.35% 0
Yuri            0.20% 0
Thriller        0.19% 0
Shoujo Ai       0.17% 0
Shounen Ai      0.17% 0
```

ex3 $(\gamma = 6)$

```
Action         49.91% 1
Sci-Fi         47.73% 0
Adventure      47.61% 0
Comedy         46.81% 0
Fantasy        46.01% 0
Shounen        45.78% 0
Drama          45.36% 0
Mecha          44.31% 0
Supernatural   44.00% 0
Romance        42.89% 0
Magic          42.43% 0
Historical     41.51% 0
Military       41.12% 0
Mystery        40.99% 0
Super Power    40.87% 0
Hentai         40.55% 0
School         40.03% 0
Seinen         39.61% 0
Ecchi          39.60% 0
Demons         39.43% 0
Slice of Life  39.42% 0
Space          39.30% 0
Horror         39.19% 0
Psychological  38.89% 0
Kids           38.69% 0
Martial Arts   37.53% 0
Shoujo         37.14% 0
Harem          36.37% 0
Music          36.26% 1
Sports         35.76% 0
Police         35.54% 0
Game           35.48% 0
Parody         35.25% 0
Samurai        35.10% 0
Vampire        34.12% 0
Cars           31.88% 0
Josei          31.70% 0
Shounen Ai     30.97% 1
Thriller       30.81% 0
Yuri           29.40% 0
Dementia       29.26% 0
Yaoi           28.73% 0
Shoujo Ai      27.53% 0
```

ex4 $(\gamma = 6)$

```
Sci-Fi         53.27% 0
Adventure      50.32% 0
Mecha          49.13% 0
Action         48.94% 0
Comedy         46.96% 0
Fantasy        46.00% 0
Space          45.88% 0
Kids           45.26% 0
Shounen        44.65% 0
Military       42.33% 0
Drama          41.04% 0
Magic          40.04% 0
Super Power    38.36% 0
Romance        38.23% 0
Seinen         36.25% 0
Hentai         35.91% 0
School         35.81% 0
Ecchi          35.61% 0
Sports         35.23% 1
Shoujo         35.15% 0
Slice of Life  34.54% 0
Historical     33.67% 0
Music          33.40% 0
Supernatural   33.31% 0
Demons         31.92% 0
Martial Arts   31.48% 0
Mystery        31.45% 0
Horror         30.86% 0
Game           30.65% 1
Parody         30.58% 0
Harem          29.13% 0
Cars           28.87% 0
Police         26.89% 0
Psychological  26.08% 0
Josei          21.19% 0
Vampire        21.13% 0
Samurai        20.34% 0
Dementia       20.26% 0
Shoujo Ai      19.70% 0
Shounen Ai     16.82% 0
Yuri           16.41% 0
Yaoi           15.98% 0
Thriller       14.94% 0
```

출력값으로 나온 각 label들의 확률을 살펴보면 대부분 장르의 분포 비율이 그대로 나오며, 50%가 넘는 label이 거의 나오지 않음을 확인할 수 있습니다. Focal Loss의 Focal Parameter(γ)의 매우 값을 크게 하여 억지로 출력값의 크기를 비슷하게 맞추어줄 수 있으나, 실제 정답과는 상관없이 단순히 비율만 맞추어지게 되어 의미있는 결과가 아님을 확인했습니다.

5-2. Review

모델의 예측 정확도를 개선하기 위해 매우 다양한 시도를 해보았지만 끝내 성공하지 못했습니다. 앞서 설명한 방법 이외에 시도해보았던 방법들을 몇가지 나열하면 다음과 같습니다.

1. 장르들 간의 종속성을 고려해 통합하여 장르의 총 가짓수 줄이기
2. 각 장르를 word vector로 표현하여 유사도를 기반으로 학습하기
3. 더 큰 데이터셋으로 사전 학습된 Word2Vec 모델 사용해보기
4. 줄거리 벡터 생성할 때 TF-IDF 가중치 생략하기
5. TF-IDF 중요도가 가장 높은 n개의 word vector만의 평균으로 줄거리 벡터 생성하기
6. 전체 장르 데이터로부터 장르 벡터를 만드는 Autoencoder를 만들어 label로 사용하기
7. Focal Loss 대신 MSE Loss를 통해 회귀문제로 풀어보기
8. 출력 확률이 해당 데이터 분포 비율보다 크면 1로 predict하도록 후처리하기

많은 노력을 기울였음에도 기대했던만큼의 결과물이 완성되지 않아 실망스럽긴 하지만, 자연어 처리, 멀티 레이블 분류 등 기존에 해보지 못했던 분야로 주제를 정했던 만큼 이번 최종 프로젝트를 진행하면서 상당히 많은 새로운 것을 배울 수 있었습니다. 이 주제는 지능시스템설계 과목의 프로젝트로 끝내지 않고 앞으로 개인 프로젝트로 Github와 Kaggle에서 계속 진행하게 될 것 같습니다.

6. Description of Submission Code

압축 파일 내부 구조

```
submission.zip/
├─ anime_classification.py
├─ clean_data.ipynb
├─ model.ipynb
├─ ptb.py
├─ synopsis_vector.ipynb
```

실행 순서

1. `clean_data.ipynb` : 데이터 다운로드 및 전처리
2. `synopsis_vector.ipynb` : 텍스트 데이터를 문장 벡터로 변환
3. `model.ipynb` : 모델 생성 및 학습 실행

부가 설명

- `ptb.py` : 교재 제공 코드, PTB 데이터셋 어휘 집합 불러올 때 사용
- `anime_classification.py` : 함수, 커스텀 레이어, 커스텀 모델 정의 포함