Steam Game Review Aspect Extraction Data Generation

Data generation ini adalah tahap pertama dari aspect extraction. Data generation ini perlu dilakukan karena kebanyakan data di luar sana merupakan data unsupervised. Sehingga di data generation ini akan dicari aspect dari masing-masing review, sehingga nantinya dapat dilakukan training yang bersifat supervised. Setelah ini akan dilakukan dataParser.

In [1]:
```
import numpy as np
import pandas as pd
import spacy
from tqdm import tqdm


import re
import os
```

Read datasets

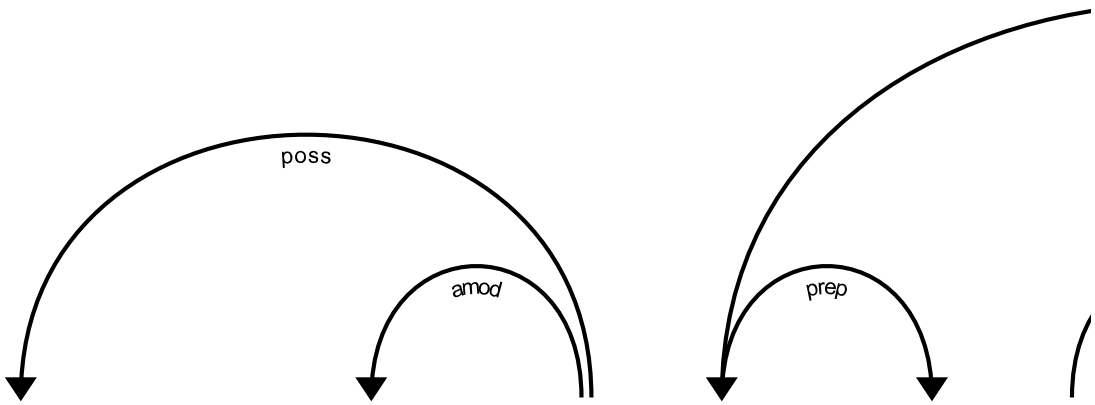
In [2]:
```
toy_rev = pd.read_csv('./data/data.csv')
```

SpaCy Dependency Parser

In [3]:
```
nlp = spacy.load('en_core_web_sm', parse=True, tag=True, entity=True)
```

SpaCy displacy to show dependency parser

In [4]:
```
txt = toy_rev['review'][0]
doc = nlp(txt)
spacy.displacy.render(doc,style='dep',jupyter=True)
```


poss
amod
prep

My	first	game	on
DET	ADJ	NOUN	ADP

Aspect extraction algorithm

Competitors List

In [5]:
```
competitors = ['epic games', 'origin', 'gog', 'humble bundle']
```

In [ ]:
```
aspect_terms = []
comp_terms = []
easpect_terms = []
ecomp_terms = []
enemy = []
for x in tqdm(range(len(toy_rev['review']))):
    amod_pairs = []
    advmod_pairs = []
    compound_pairs = []
    xcomp_pairs = []
    neg_pairs = []
    eamod_pairs = []
    eadvmod_pairs = []
    ecompound_pairs = []
    eneg_pairs = []
    excomp_pairs = []
    enemlist = []
    if len(str(toy_rev['review'][x])) != 0:
        lines = str(toy_rev['review'][x]).replace('*',' ').replace('-',' ').re
place('so ',' ').replace('be ',' ').replace('are ',' ').replace('just ',' ').r
eplace('get ','').replace('were ',' ').replace('When ','').replace('when ','')
.replace('again ',' ').replace('where ','').replace('how ',' ').replace('has '
,' ').replace('Here ',' ').replace('here ',' ').replace('now ',' ').replace('s
ee ',' ').replace('why ',' ').split('.')
        for line in lines:
            enem_list = []
            for eny in competitors:
                enem = re.search(eny,line)
                if enem is not None:
                    enem_list.append(enem.group())
            if len(enem_list)==0:
                doc = nlp(line)
                str1=''
                str2=''
                for token in doc:
                    if token.pos_ is 'NOUN':
                        for j in token.lefts:
                            if j.dep_ == 'compound':
                                compound_pairs.append((j.text+' '+token.text,t
oken.text))
                            if j.dep_ is 'amod' and j.pos_ is 'ADJ': #primary
 condition
                                str1 = j.text+' '+token.text
                                amod_pairs.append(j.text+' '+token.text)
                                for k in j.lefts:
                                    if k.dep_ is 'advmod': #secondary conditio
n to get adjective of adjectives
                                        str2 = k.text+' '+j.text+' '+token.tex
t
                                        amod_pairs.append(k.text+' '+j.text+'
 '+token.text)
                            mtch = re.search(re.escape(str1),re.escape(str
2))
                            if mtch is not None:
                                amod_pairs.remove(str1)
                    if token.pos_ is 'VERB':
```

```
                        for j in token.lefts:
                            if j.dep_ is 'advmod' and j.pos_ is 'ADV':
                                advmod_pairs.append(j.text+' '+token.text)
                            if j.dep_ is 'neg' and j.pos_ is 'ADV':
                                neg_pairs.append(j.text+' '+token.text)
                        for j in token.rights:
                            if j.dep_ is 'advmod'and j.pos_ is 'ADV':
                                advmod_pairs.append(token.text+' '+j.text)
                    if token.pos_ is 'ADJ':
                        for j,h in zip(token.rights,token.lefts):
                            if j.dep_ is 'xcomp' and h.dep_ is not 'neg':
                                for k in j.lefts:
                                    if k.dep_ is 'aux':
                                        xcomp_pairs.append(token.text+' '+k.te
xt+' '+j.text)
                            elif j.dep_ is 'xcomp' and h.dep_ is 'neg':
                                for k in j.lefts:
                                    if k.dep_ is 'aux':
                                        neg_pairs.append(h.text +' '+token.tex
t+' '+k.text+' '+j.text)

            else:
                enemlist.append(enem_list)
                doc = nlp(line)
                str1=''
                str2=''
                for token in doc:
                    if token.pos_ is 'NOUN':
                        for j in token.lefts:
                            if j.dep_ == 'compound':
                                ecompound_pairs.append((j.text+' '+token.text,
token.text))
                            if j.dep_ is 'amod' and j.pos_ is 'ADJ': #primary
 condition
                                str1 = j.text+' '+token.text
                                eamod_pairs.append(j.text+' '+token.text)
                                for k in j.lefts:
                                    if k.dep_ is 'advmod': #secondary conditio
n to get adjective of adjectives
                                        str2 = k.text+' '+j.text+' '+token.tex
t
                                        eamod_pairs.append(k.text+' '+j.text+'
'+token.text)
                                mtch = re.search(re.escape(str1),re.escape(str
2))
                                if mtch is not None:
                                    eamod_pairs.remove(str1)
                    if token.pos_ is 'VERB':
                        for j in token.lefts:
                            if j.dep_ is 'advmod' and j.pos_ is 'ADV':
                                eadvmod_pairs.append(j.text+' '+token.text)
                            if j.dep_ is 'neg' and j.pos_ is 'ADV':
                                eneg_pairs.append(j.text+' '+token.text)
                        for j in token.rights:
                            if j.dep_ is 'advmod'and j.pos_ is 'ADV':
                                eadvmod_pairs.append(token.text+' '+j.text)
                    if token.pos_ is 'ADJ':
                        for j in token.rights:
```

```
                            if j.dep_ is 'xcomp':
                                for k in j.lefts:
                                    if k.dep_ is 'aux':
                                        excomp_pairs.append(token.text+' '+k.t
ext+' '+j.text)
        pairs = list(set(amod_pairs+advmod_pairs+neg_pairs+xcomp_pairs))
        epairs = list(set(eamod_pairs+eadvmod_pairs+eneg_pairs+excomp_pairs))
        for i in range(len(pairs)):
            if len(compound_pairs)!=0:
                for comp in compound_pairs:
                    mtch = re.search(re.escape(comp[1]),re.escape(pairs[i]))
                    if mtch is not None:
                        pairs[i] = pairs[i].replace(mtch.group(),comp[0])
        for i in range(len(epairs)):
            if len(ecompound_pairs)!=0:
                for comp in ecompound_pairs:
                    mtch = re.search(re.escape(comp[1]),re.escape(epairs[i]))
                    if mtch is not None:
                        epairs[i] = epairs[i].replace(mtch.group(),comp[0])
    aspect_pairs = []
    for i in range(len(pairs)):
        words = pairs[i].split()
        aspect_pairs.append(words[-1])

    aspect_terms.append(aspect_pairs)
    comp_terms.append(compound_pairs)
    easpect_terms.append(epairs)
    ecomp_terms.append(ecompound_pairs)
    enemy.append(enemlist)
toy_rev['compound_nouns'] = comp_terms
toy_rev['aspect_keywords'] = aspect_terms
toy_rev['competition'] = enemy
toy_rev['competition_comp_nouns'] = ecomp_terms
toy_rev['competition_aspects'] = easpect_terms
toy_rev.head()
```

```
  0%|                                                                        | 2
8/79437 [00:14<8:30:34,  2.59it/s]
```

In [ ]:
```
toy_rev.to_csv('./data/data-aspect.csv')
```

Data Parser for Aspect Extraction

Pada tahapan kedua ini akan dilakukan data parsing. Data parsing yang dimaksud adalah melakukan pembagian data dari berbentuk review string biasa menjadi kolom-kolom word_before, word_now, word_after, dan pos_tag nya serta class nya. Parsing ini harus dilakukan sebelum melakukan training atau testing. Di dalam data parsing ini digunakan program posTagger milik anggota kelompok kami yakni Nixon Andhika.

In [1]:
```
import numpy as np
import pandas as pd
import nltk
from posTagger import posTagger
nltk.download('punkt')
nltk.download('maxent_treebank_pos_tagger')

import os
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\ferdy\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package maxent_treebank_pos_tagger to
[nltk_data]     C:\Users\ferdy\AppData\Roaming\nltk_data...
[nltk_data]   Package maxent_treebank_pos_tagger is already up-to-
[nltk_data]       date!
```

Read data

In [2]:
```
data = pd.read_csv('./data/data-aspect.csv')
```

Create Pos Tagger Class

In [3]:
```
pos_tagger = posTagger()
```

Parse Data

In [4]:
```
def split_string_to_list(string_list):
    splitted_strings = []
    for string_comp in string_list:
        splitted_strings.append(string_comp.split())
    return splitted_strings
```

In [ ]:
```
aspect_data = pd.DataFrame()
arr_word_before = []
arr_word_now = []
arr_word_after = []
arr_pos_tag = []
arr_class = []
string_list = split_string_to_list(data['review'])
pos_tagged_data = pos_tagger.pos_tag(string_list)

for i in range(500):
    aspect = data['aspect_keywords'][i]

    for j in range(len(pos_tagged_data[i])):
        if (j == 0):
            arr_word_before.append('[START]')
        else:
            arr_word_before.append(pos_tagged_data[i][j-1][0])

        arr_word_now.append(pos_tagged_data[i][j][0])

        arr_pos_tag.append(pos_tagged_data[i][j][1])

        if (j == (len(pos_tagged_data[i]) - 1)):
            arr_word_after.append('[END]')
        else:
            arr_word_after.append(pos_tagged_data[i][j+1][0])

        if (pos_tagged_data[i][j][0] in aspect):
            arr_class.append('true')
        else:
            arr_class.append('false')

aspect_data['word_before'] = arr_word_before
aspect_data['word_now'] = arr_word_now
aspect_data['word_after'] = arr_word_after
aspect_data['pos_tag'] = arr_pos_tag
aspect_data['class'] = arr_class
```

In [ ]:
```
aspect_data.head()
```

In [ ]:
```
aspect_data.to_csv('./data/steam-aspect.csv', index=False)
```

Steam Game Review Aspect Extraction

Deskripsi Task: Pada task ini akan dilakukan aspect extraction.

1. Aspect extraction pada awalnya akan dilakukan dengan cara melakukan generate data yang bersifat supervised, karena pada awalnya data belum supervised.
2. Dari data yang masih berupa review saja akan dicari aspectnya apa saja dengan program jupyter notebook dataGeneration.ipynb.
3. Setelah itu akan dilakukan parsing data sehingga menjadi bentuk word_before, word_now, word_after, dan pos_tag beserta class nya melalui program dataParser.ipynb.
4. Setelah itu data telah siap dilakukan training.
5. Training dilakukan dengan melakukan konkatenasi word_before, word_now, word_after dan pos_tag jika dibutuhkan. Lalu akan dicari tfidf nya.
6. Setelah itu akan dilakukan pemisahan data untuk data training dan testing.
7. Setelah itu akan dilakukan training dengan menggunakan model machine learning yang sudah ada di sklearn. Di tugas ini saya menggunakan LogisticRegression dan SVM.
8. Setelah dilakukan training akan dilakukan testing, dan hasil score akan muncul.

In [1]:
```
import numpy as np
import pandas as pd
import pickle

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC

import os
```

Read Data

In [2]:
```
data = pd.read_csv('./data/steam-aspect.csv')
data.head()
```

Out[2]:

	word_before	word_now	word_after	pos_tag	class
0	[START]	My	first	PRP$	False
1	My	first	game	JJ	False
2	first	game	on	NN	True
3	game	on	A3	IN	False
4	on	A3	brought	NNP	False

Combine columns

In [3]:
```
aspect_data = pd.DataFrame()
aspect_data_no_pos_tag = pd.DataFrame()
arr_words = []
arr_words_no_pos_tag = []

for i in range(len(data['word_now'])):
    word = ""
    if (data['word_before'][i] != '[START]'):
        word += str(data['word_before'][i])

    word += " " + str(data['word_now'][i])

    if (data['word_after'][i] != '[END]'):
        word += " " + str(data['word_after'][i])

    arr_words_no_pos_tag.append(word)

    word_pos_tag = word + " " + str(data['pos_tag'][i])

    arr_words.append(word_pos_tag)

aspect_data['review'] = arr_words
aspect_data['class'] = data['class'].copy()
aspect_data_no_pos_tag['review'] = arr_words_no_pos_tag
aspect_data_no_pos_tag['class'] = data['class'].copy()
aspect_data_no_pos_tag.head()
```

Out[3]:

	review	class
0	My first	False
1	My first game	False
2	first game on	True
3	game on A3	False
4	on A3 brought	False

Train Test Split Data

In [4]:
```
X_train, X_test, y_train, y_test = train_test_split(aspect_data['review'], asp
ect_data['class'], test_size=0.33)
X_train_no_pos_tag, X_test_no_pos_tag, y_train_no_pos_tag, y_test_no_pos_tag =
train_test_split(aspect_data_no_pos_tag['review'], aspect_data_no_pos_tag['cla
ss'], test_size=0.33)
```

Feature Extraction

1. With Pos Tag

In [5]:
```
tfidf = TfidfVectorizer(binary=True, use_idf = True, max_features=256)
tfidf = tfidf.fit(X_train)

X_train_tfidf = pd.DataFrame(tfidf.transform(X_train).toarray(), columns=[tfid
f.get_feature_names()])
X_test_tfidf = pd.DataFrame(tfidf.transform(X_test).toarray(), columns=[tfidf.
get_feature_names()])

X_train_tfidf
```

Out[5]:

	10	able	about	actually	after	again	ai	all	almost	alpha	...	why	will	with
0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
1	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
2	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
3	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
4	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
5	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
6	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
7	0.0	0.0	0.650489	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
8	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
9	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
10	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
11	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
12	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
13	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
14	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
15	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
16	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.706833	0.0	...	0.0	0.0	0.0
17	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
18	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
19	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
20	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
21	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
22	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
23	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
24	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
25	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
26	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
27	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
28	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
29	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
48053	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
48054	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
48055	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
48056	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0

	10	able	about	actually	after	again	ai	all	almost	alpha	...	why	will	with
48057	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
48058	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
48059	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
48060	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
48061	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
48062	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
48063	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
48064	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
48065	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
48066	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
48067	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
48068	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
48069	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
48070	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
48071	0.0	0.0	0.000000	0.755466	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
48072	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
48073	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
48074	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
48075	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
48076	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
48077	0.0	0.0	0.801329	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
48078	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
48079	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
48080	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0
48081	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	1.000000	0.0	...	0.0	0.0	0.0
48082	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0

48083 rows × 256 columns

2. No Pos Tag

In [6]:
```
tfidf_no_pos_tag = TfidfVectorizer(binary=True, use_idf = True, max_features=2
56)
tfidf_no_pos_tag = tfidf_no_pos_tag.fit(X_train_no_pos_tag)

X_train_tfidf_no_pos_tag = pd.DataFrame(tfidf_no_pos_tag.transform(X_train_no_
pos_tag).toarray(), columns=[tfidf_no_pos_tag.get_feature_names()])
X_test_tfidf_no_pos_tag = pd.DataFrame(tfidf_no_pos_tag.transform(X_test_no_po
s_tag).toarray(), columns=[tfidf_no_pos_tag.get_feature_names()])

X_train_tfidf_no_pos_tag
```

Out[6]:

	10	20	able	about	actually	add	after	again	ai	all	...	with	wor
0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
1	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.706919	0.000000	...	0.0	0.0(
2	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
3	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
4	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
5	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
6	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
7	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
8	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
9	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
10	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
11	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
12	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
13	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
14	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
15	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
16	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
17	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
18	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
19	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
20	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
21	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
22	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
23	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
24	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
25	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.565119	...	0.0	0.0(
26	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
27	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
28	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
29	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
...	...	...	...	...	...	...	...	...	...	...	...	...	
48053	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
48054	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
48055	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
48056	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(

	10	20	able	about	actually	add	after	again	ai	all	...	with	wor
48057	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
48058	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
48059	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
48060	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
48061	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
48062	0.709944	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
48063	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
48064	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
48065	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
48066	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
48067	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
48068	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
48069	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
48070	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
48071	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
48072	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
48073	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
48074	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
48075	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.842182	...	0.0	0.0(
48076	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
48077	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
48078	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
48079	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
48080	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.6
48081	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(
48082	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.0	0.0(

48083 rows × 256 columns

Classification

1.a. Logistic Regression With Pos Tag

In [7]:
```
lg = LogisticRegression(C=1000, solver='liblinear')
```

In [8]:
```
lg.fit(X_train_tfidf, y_train)
```

Out[8]:
```
LogisticRegression(C=1000, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='liblinear', tol=0.0001, verbose
=0,
                   warm_start=False)
```

In [9]:
```
lg.score(X_test_tfidf, y_test)
```

Out[9]:
```
0.7896892416821483
```

1.b. Logistic Regression Without Pos Tag

In [10]:
```
lg_no_pos_tag = LogisticRegression(C=1000, solver='liblinear')
```

In [11]:
```
lg_no_pos_tag.fit(X_train_tfidf_no_pos_tag, y_train_no_pos_tag)
```

Out[11]:
```
LogisticRegression(C=1000, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='liblinear', tol=0.0001, verbose
=0,
                   warm_start=False)
```

In [12]:
```
lg_no_pos_tag.score(X_test_tfidf_no_pos_tag, y_test_no_pos_tag)
```

Out[12]:
```
0.7720401959128526
```

2.a. SVM With Pos Tag

In [13]:
```
svc = SVC(C=1, kernel='linear')
```

In [14]:
```
svc.fit(X_train_tfidf, y_train)
```

Out[14]:
```
SVC(C=1, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [15]:
```
svc.score(X_test_tfidf, y_test)
```

Out[15]:
```
0.777402465799696
```

2.a. SVM Without Pos Tag

In [16]:
```
svc_no_pos_tag = SVC(C=1, kernel='linear')
```

In [17]:
```
svc_no_pos_tag.fit(X_train_tfidf_no_pos_tag, y_train_no_pos_tag)
```

Out[17]:
```
SVC(C=1, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [18]:
```
svc_no_pos_tag.score(X_test_tfidf_no_pos_tag, y_test_no_pos_tag)
```

Out[18]:
```
0.7719135281202499
```

Save Model

In [19]:
```
pickle.dump(lg, open("./model/aspect_lg.pkl", "wb"))
```

In [20]:
```
pickle.dump(lg_no_pos_tag, open("./model/aspect_lg_no_pos_tag.pkl", "wb"))
```

In [21]:
```
pickle.dump(svc, open("./model/aspect_svc.pkl", "wb"))
```

In [22]:
```
pickle.dump(svc_no_pos_tag, open("./model/aspect_svc_no_pos_tag.pkl", "wb"))
```