movie_analysis

March 8, 2024

#Classifcation Problem: English difficulty of movies based on subtitles (Apiary Project)

0.1 Introduction

Hello and welcome to my take on this project launched by the Apiary Team at Practicum by Yandex! This project is part of my learning path as a Data Scientist at Practicum USA Bootcamp.

English is one of the most spoken languages worldwide, that's not a secret to anybody. As present as it is on the internet, learning it can still be a challenging task for natives of different languages. A fun way to improve listening skills and familiarize with a language could be watching movies and TV series with subtitles, but how much English is too much English?

Being a linguist enthusiast myself and an intermediate Chinese speaker, I know that choosing the right show to watch in a second language can be challenging! It would be great to know how do vocabulary and grammar used in a show compare to my current skills, otherwise things cat get frustrating when each new sentence needs to be paused to look up the meaning of a sentence.

In this project we seek to develop a classification model to decide on movie difficulty based on subtitles. It works by checking English words in a subtitle file and trying to predict how would a professional linguist classify it based on its difficulty. The levels range A2 to B2, based on the English CEFR (Common European Framework of Reference for Languages).

Enjoy!

0.2 Importing modules

0.3 Reading target data

First we will check what data is available from the linguists, which is a table about the movies and respective CEFR levels!

	Movie	Level	Subtitles	Kinopoisk
0	10 Cloverfield Lane	B1	Yes	NaN
1	10 things I hate about you	B1	Yes	No subs
2	A knight's tale	B2	Yes	Everything
3	A star is born	B2	Yes	Nope
4	Aladdin	A2/A2+	Yes	Everything

```
88
```

Data from 88 movies is available. This data will be split between training and testing to be used as learning material for our prediction model!

0.4 Reading wordlists

A good starting point to classify movie difficulties based on their subtitles is to analyze vocabulary. For that, we will use Oxford and Cambridge wordlists previously scraped from the internet to create a word reference table!

```
Creating A1 df
Creating A2 df
Creating B1 df
Creating B2 df
Creating B2 df
Creating C1 df
```

```
Empty DataFrame
Columns: [word, pos, level]
Index: []
```

```
            word pos level
798   infinitive        A1
1152      expert        A2
```

```
0
```

```
array(['prep', 'n', 'adv', 'conj', 'number', 'pron', 'v', 'det', 'adj',
       'exclam', 'article'], dtype='<U7')
```

```
          word   pos level
2542  academic  NOUN    B2
1733  academic   ADJ    B1
2545   account  VERB    B2
1736   account  NOUN    B1
3257      acid  NOUN    B2
…          …     …     …
5242      well  NOUN    C1
1726     while  CONJ    A2
2522     while  NOUN    B1
2523     whole  NOUN    B1
1727     whole   ADJ    A2

[698 rows x 3 columns]
```

```
5263
```

```
     word        pos level
0       a        DET    A1
1   about  SCONJ ADV    A1
2   above  SCONJ ADV    A1
3  across  SCONJ ADV    A1
4  action       NOUN    A1
```

```
  Base Word Guideword Level Part of Speech          Topic  Details
0    cattle       NaN    B1            NaN        animals  Details
1   clothes       NaN    A1            NaN        clothes  Details
2    albeit       NaN    C2            NaN            NaN  Details
3  although       BUT    B1            NaN  communication  Details
4  although   DESPITE    B1            NaN  communication  Details

          word        pos level
0        donor       noun    C2
1    classical  adjective    C2
2  broadminded  adjective    C2
3    crackdown       noun    C2
4        crack       noun    C2

1928

          word        pos level
0        donor       noun    C2
1    classical  adjective    C2
2  broadminded  adjective    C2
3    crackdown       noun    C2
4        crack       noun    C2

1854

          word        pos level
0        donor       noun    C2
1    classical  adjective    C2
2  broadminded  adjective    C2
3    crackdown       noun    C2
4        crack       noun    C2

<ipython-input-18-4cb24d2574a6>:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  word_df['pos'] = word_df['pos'].map(pos_dict)

          word   pos level
0        donor  NOUN    C2
1    classical   ADJ    C2
2  broadminded   ADJ    C2
3    crackdown  NOUN    C2
4        crack  NOUN    C2

 1854
```

```
             word        pos level_x level_y
0           donor       NOUN      C2      C1
1       classical        ADJ      C2      A2
2     broadminded        ADJ      C2     NaN
3       crackdown       NOUN      C2     NaN
4           crack       NOUN      C2     NaN
…             …       …      …       …
6015  worthwhile         ADJ     NaN      C1
6016      worthy         ADJ     NaN      C1
6017        yell        VERB     NaN      C1
6018       yield   NOUN VERB     NaN      C1
6019   youngster        NOUN     NaN      C1

[6020 rows x 4 columns]
```

```
6020
```

```
       word        pos level
1818  about  SCONJ ADV    A1
732   about        ADV    B2
141   above        ADV    C2
1819  above  SCONJ ADV    A1
4932  abuse  NOUN VERB    C1
…     …         …    …
3172  while       CONJ    A2
3798  while       NOUN    B1
1093  while        NaN    B2
3173  whole        ADJ    A2
3799  whole       NOUN    B1

[1237 rows x 3 columns]
```

```
Creating lookup table to define word levels in movies…

The following words are repeated for same POS in the wordlists: ['bank', 'any',
'about', 'above', 'across', 'answer', 'around', 'back', 'back', 'behind',
'below', 'black', 'blue', 'blue', 'break', 'brown', 'brown', 'call', 'capital',
'change', 'clean', 'clean', 'cold', 'complete', 'cost', 'cost', 'dance',
'design', 'design', 'dress', 'dress', 'drink', 'east', 'email', 'email', 'end',
'exercise', 'have', 'have', 'ice', 'need', 'after', 'all', 'alone', 'along',
'anywhere', 'arrangement', 'assistant', 'attack', 'attention', 'average',
'average', 'before', 'best', 'blank', 'bottom', 'brush', 'camp', 'camp', 'care',
'care', 'cause', 'cause', 'chat', 'circle', 'circle', 'control', 'control',
'copy', 'cross', 'cross', 'cycle', 'cycle', 'download', 'dream', 'expert',
'light', 'rest', 'ring', 'rock', 'access', 'access', 'aim', 'arrest', 'arrest',
'balance', 'ban', 'base', 'bend', 'bite', 'bite', 'block', 'bomb', 'brand',
'calm', 'campaign', 'charge', 'charge', 'cheat', 'chemical', 'claim', 'claim',
'click', 'click', 'commercial', 'contact', 'contact', 'contrast', 'contrast',
```

```
'damage', 'direct', 'dislike', 'doubt', 'doubt', 'escape', 'escape', 'exchange',
'exchange', 'export', 'extra', 'lie', 'race', 'used', 'advance', 'aid',
'appeal', 'appeal', 'approach', 'attempt', 'attempt', 'bet', 'bet', 'beyond',
'blame', 'blame', 'broadcast', 'capture', 'capture', 'cast', 'cast',
'characteristic', 'characteristic', 'chief', 'classic', 'collapse', 'collapse',
'comfort', 'command', 'concern', 'concern', 'conduct', 'conflict', 'contest',
'contract', 'contract', 'core', 'crash', 'cure', 'cure', 'curve', 'debate',
'decline', 'decrease', 'defeat', 'defeat', 'delay', 'delay', 'delight',
'demand', 'desire', 'desire', 'display', 'display', 'dozen', 'draft', 'draft',
'encounter', 'estimate', 'evil', 'excuse', 'excuse', 'executive', 'tear',
'besides', 'bid', 'boost', 'chase', 'cheer', 'cheer', 'comic', 'concrete',
'crack', 'crack', 'cruise', 'cruise', 'dive', 'dive', 'divorce', 'divorce',
'equivalent', 'exhibit', 'abuse', 'advocate', 'alert', 'alert', 'alike',
'alike', 'amateur', 'assault', 'attribute', 'blend', 'breed', 'civilian',
'compromise', 'compromise', 'consent', 'consent', 'dispute', 'distress', 'ease',
'echo', 'explosive']
```

By the end of this section, we have python dictionaries with words and their respective parts of speech. This is important because different parts of speech may define different levels for the same word! An additional dictionary has been created for cases where the part of speech is not correctly recognized.

0.5 Reading subtitle files

Now subtitle files will be read and organized into tables by lines. This data will be used for later analysis for each movie.

```
                              name  year  \
0                        mamma_mia  2008
1                         die_hard  1988
2                   the_blind_side  2009
3         the_theory_of_everything  2014
4   the_secret_life_of_walter_mitty  2013
..                             …   …
81                    pleasantville  1998
82                the_invisible_man  2020
83               back_to_the_future  1985
84                     notting_hill  1999
85                   a_star_is_born  2018

                                 filename
0                      Mamma_Mia(2008).srt
1                       Die_hard(1988).srt
2                 The_blind_side(2009).srt
3       The_theory_of_everything(2014).srt
4   The_secret_life_of_Walter_Mitty(2013).srt
..                                       …
81                  Pleasantville(1998).srt
82              The_invisible_man(2020).srt
```

```
83              Back_to_the_future(1985).srt
84                    Notting_Hill(1999).srt
85                  A_star_is_born(2018).srt

[86 rows x 3 columns]
```

```
Empty DataFrame
Columns: [name, year, filename]
Index: []
```

```
 array(['mamma_mia', 'die_hard', 'the_blind_side',
        'the_theory_of_everything', 'the_secret_life_of_walter_mitty',
        'the_man_called_flintstone', 'the_hangover', 'up', 'titanic',
        'ready_or_not', 'dredd', 'the_fault_in_our_stars',
        'it_s_a_wonderful_life', 'groundhog_day', 'good_will_hunting',
        'venom', 'the_terminator', 'the_terminal', 'pulp_fiction',
        'finding_nemo', 'lion', 'babe', 'catch_me_if_you_can', 'toy_story',
        'soul', 'pirates_of_the_caribbean', 'the_kings_speech',
        'the_break-up', 'aladdin', 'home_alone', 'shrek', 'knives_out',
        'braveheart', 'inside_out', 'an_american_tail', 'meet_the_parents',
        'the_lion_king', 'dune', 'moulin_rouge', 'we_are_the_millers',
        'the_usual_suspects', 'the_jungle_book', 'before_sunset',
        'love_actually', 'the_lord_of_the_rings', 'the_graduate', 'logan',
        'matilda', 'the_holiday', 'mary_poppins_returns',
        'eurovision_song_contest_', 'mrs_doubtfire', 'house_of_gucci',
        'forrest_gump', 'before_sunrise', 'bridget_jones_diary',
        'the_greatest_showman', 'the_social_network', 'her', 'twilight',
        'cast_away', 'liar_liar', 'fight_club',
        'harry_potter_and_the_philosophers_stone',
        'my_big_fat_greek_wedding', 'deadpool', 'powder',
        'all_dogs_go_to_heaven', 'clueless', 'beauty_and_the_beast',
        'warm_bodies', 'a_knights_tale', 'kubo_and_the_two_strings',
        'the_shawshank_redemption', '10_cloverfield_lane',
        'the_cabin_in_the_woods', 'before_i_go_to_sleep', 'batman_begins',
        'hook', 'sleepless_in_seattle', '10_things_i_hate_about_you',
        'pleasantville', 'the_invisible_man', 'back_to_the_future',
        'notting_hill', 'a_star_is_born'], dtype=object)
```

```
CPU times: user 11 s, sys: 390 ms, total: 11.4 s
Wall time: 19.2 s
```

```
Word table for movie "The Blind Side"
```

	lemma	pos	line	start	end
0	crowd	NOUN	1	29	29
1	cheer	VERB	1	29	29
2	in	ADP	1	29	29
3	distance	NOUN	1	29	29
4	lelgh	PROPN	2	32	32

Tables have been created, one for each movie. Each of these tables contain all words, their respective parts of speech, what line they are in and what time of the movie they appear.

0.6 Merging ref tables

After reading subtitle files available, we will join information of tables to make sure data is available for each movie analyzed. During this section, we will also standardize levels for classification.

```
          word   pos level
0        donor  NOUN    C2
1    classical   ADJ    C2
2  broadminded   ADJ    C2
3    crackdown  NOUN    C2
4        crack  NOUN    C2
```

```
                         Movie   Level
0         10 Cloverfield Lane      B1
1  10 things I hate about you      B1
2             A knight's tale      B2
3              A star is born      B2
4                     Aladdin  A2/A2+
```

```
array(['B1', 'B2', 'A2/A2+', 'B1,B2', 'A2/A2+,B1'], dtype=object)
```

```
array(['B1', 'B2', 'A2'], dtype=object)
```

```
                         Movie Level
0         10_cloverfield_lane    B1
1  10_things_i_hate_about_you    B1
2              a_knights_tale    B2
3              a_star_is_born    B2
4                     aladdin    A2
```

```
                                              movie level  \
6                                 an__american_tail    A2
7                                  harry_potter_(1)    B1
8                              its_a_wonderful_life    A2
9                               lie_to_me_(series)    B2
10                        the_man_called_flinstone    A2
11                     the_walking_dead__(series)     A2
12                                 were_the_millers    B1
29  eurovision_song_contest_the_story_of_fire_saga    A2
```

```
                      name year filename
6                  Unknown  NaN      NaN
7                  Unknown  NaN      NaN
8                  Unknown  NaN      NaN
9                  Unknown  NaN      NaN
```

```
10                 Unknown  NaN      NaN
11                 Unknown  NaN      NaN
12                 Unknown  NaN      NaN
29  eurovision_song_contest  NaN      NaN

   movie level                                   name  year  \
88   NaN   NaN               the_man_called_flintstone  1966
89   NaN   NaN                   it_s_a_wonderful_life  1946
90   NaN   NaN                        an_american_tail  1986
91   NaN   NaN                      we_are_the_millers  2013
92   NaN   NaN                eurovision_song_contest_  2020
93   NaN   NaN  harry_potter_and_the_philosophers_stone  2001

                                            filename
88               The_man_called_Flintstone(1966).srt
89                   It_s_a_wonderful_life(1946).srt
90                        An_American_tail(1986).srt
91                      We_are_the_Millers(2013).srt
92                Eurovision_song_contest_(2020).srt
93  Harry_Potter_and_the_philosophers_stone(2001).srt

                      movie level     name year filename
39         lie_to_me_(series)    B2  Unknown  NaN      NaN
40  the_walking_dead__(series)    A2  Unknown  NaN      NaN

Empty DataFrame
Columns: [movie, level, name, year, filename]
Index: []

                       movie level                     name  year  \
0         10_cloverfield_lane    B1        10_cloverfield_lane  2016
1  10_things_i_hate_about_you    B1  10_things_i_hate_about_you  1999
2              a_knights_tale    B2             a_knights_tale  2001
3              a_star_is_born    B2             a_star_is_born  2018
4                     aladdin    A2                    aladdin  1992

                            filename
0         10_Cloverfield_lane(2016).srt
1  10_things_I_hate_about_you(1999).srt
2              A_knights_tale(2001).srt
3              A_star_is_born(2018).srt
4                     Aladdin(1992).srt
```

We know that 86 movies have their respective subtitles, the table above shows the first five entries, and connects movies with their subtitle files.

0.7 Analyzing movies

Movies will now be analyzed and useful features will be drawn to feed into the classifcation model. Words in each movie will be classified according to their difficulty levels, and we will count these words, as well as other useful information, such as how many words per minute appear for each movie. Words not contained in our wordlist reference will be marked as 'Unk' (unknown)!

```
All movies have their respective subtitles!
Reindexing…
```

```
<ipython-input-73-d7511fb8b9c5>:25: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  movie_df['level'] = movie_df.apply(find_word_level, axis=1).values
```

```
                                           movie level  year  \
name
10_cloverfield_lane              10_cloverfield_lane    B1  2016
10_things_i_hate_about_you  10_things_i_hate_about_you    B1  1999
a_knights_tale                        a_knights_tale    B2  2001
a_star_is_born                        a_star_is_born    B2  2018
aladdin                                      aladdin    A2  1992

                                                  filename A1_count  \
name
10_cloverfield_lane              10_Cloverfield_lane(2016).srt     2366
10_things_i_hate_about_you  10_things_I_hate_about_you(1999).srt     3726
a_knights_tale                        A_knights_tale(2001).srt     2831
a_star_is_born                        A_star_is_born(2018).srt     6248
aladdin                                      Aladdin(1992).srt     3322

                           A2_count B1_count B2_count C1_count C2_count  …  \
name                                                                      …
10_cloverfield_lane             336      183      160       72       65  …
10_things_i_hate_about_you      482      236      198       95       59  …
a_knights_tale                  457      227      241      110       67  …
a_star_is_born                  835      234      265       78      102  …
aladdin                         559      294      256      104       55  …

                              C2_pct    Unk_pct 1_verb_lines 2_verb_lines  \
name
10_cloverfield_lane         1.173921  42.532057          485          207
10_things_i_hate_about_you  0.679332  44.778353          478          295
a_knights_tale              0.895124   47.45491          521          222
a_star_is_born              0.722175  45.043897          913          539
aladdin                     0.618534  48.380567          628          295

                           3_verb_lines 4+_verb_lines    duration         wpm  \
name
```

10_cloverfield_lane	55	14	96.433333	57.417905
10_things_i_hate_about_you	127	29	95.366667	91.069556
a_knights_tale	66	12	131.866667	56.761881
a_star_is_born	134	30	135.2	104.467456
aladdin	67	25	87.55	101.56482

	sconj_count	word_count
name		
10_cloverfield_lane	654	5537
10_things_i_hate_about_you	1362	8685
a_knights_tale	798	7485
a_star_is_born	1914	14124
aladdin	834	8892

[5 rows x 26 columns]

	lemma	pos	line	start	end	level
0	phone	NOUN	1	38	38	A1
1	line	NOUN	1	38	38	A1
2	dialing	NOUN	1	38	38	Unk
3	then	ADV	1	38	38	A1
4	ring	VERB	1	38	38	A2

	word	pos	level
0	donor	NOUN	C2
1	classical	ADJ	C2
2	broadminded	ADJ	C2
3	crackdown	NOUN	C2
4	crack	NOUN	C2

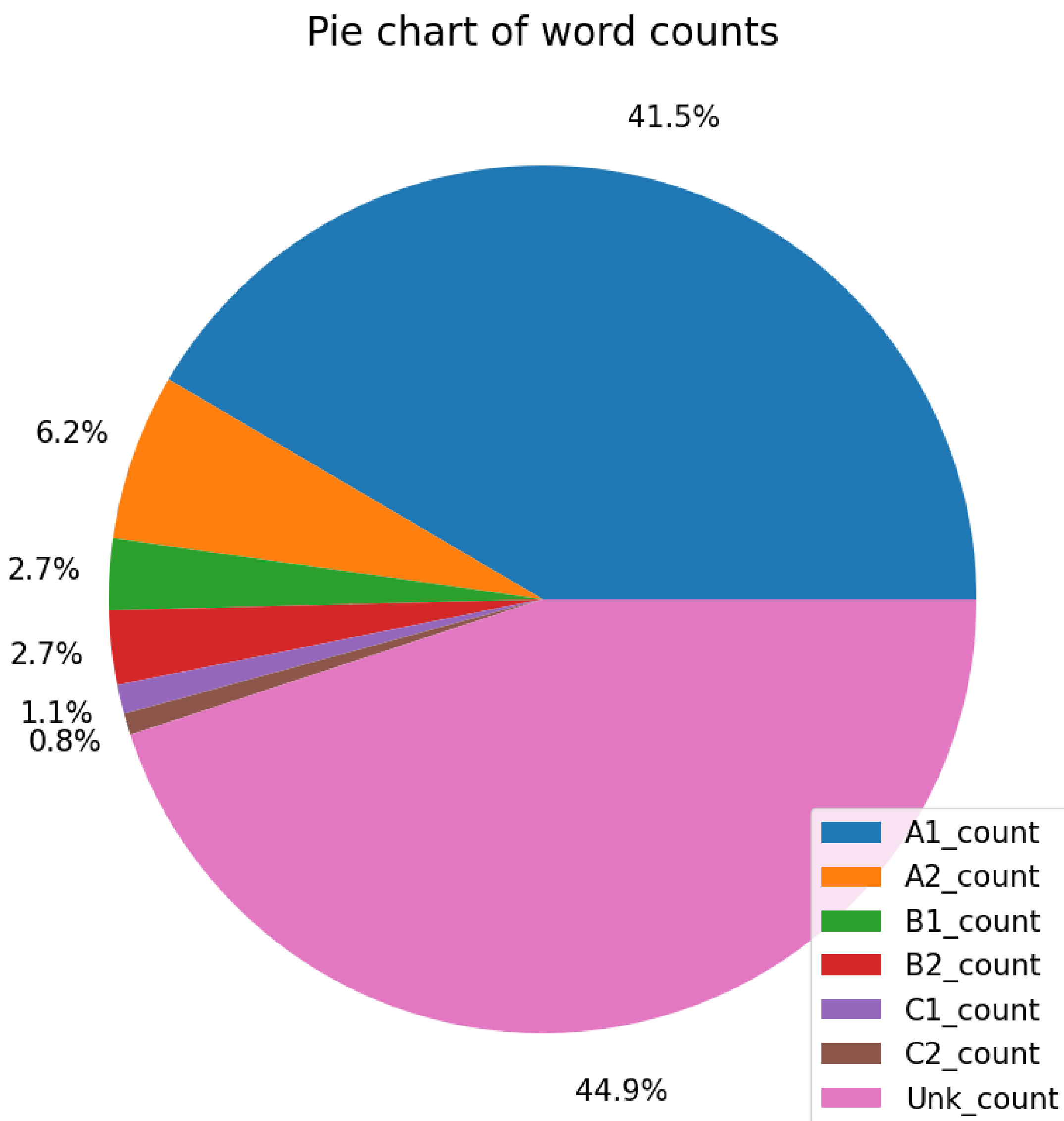
0.8 EDA

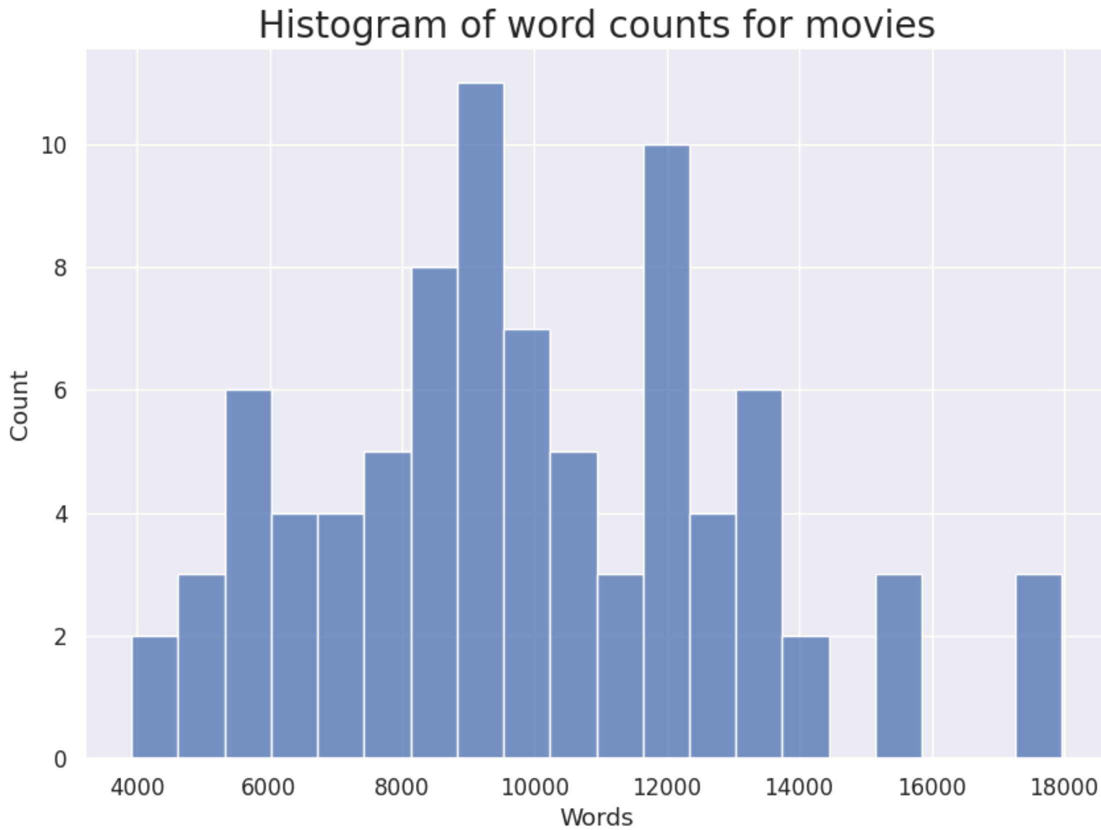
	word	pos	level
0	donor	NOUN	C2
1	classical	ADJ	C2
2	broadminded	ADJ	C2
3	crackdown	NOUN	C2
4	crack	NOUN	C2
…	…	…	…
6015	worthwhile	ADJ	C1
6016	worthy	ADJ	C1
6017	yell	VERB	C1
6018	yield	NOUN VERB	C1
6019	youngster	NOUN	C1

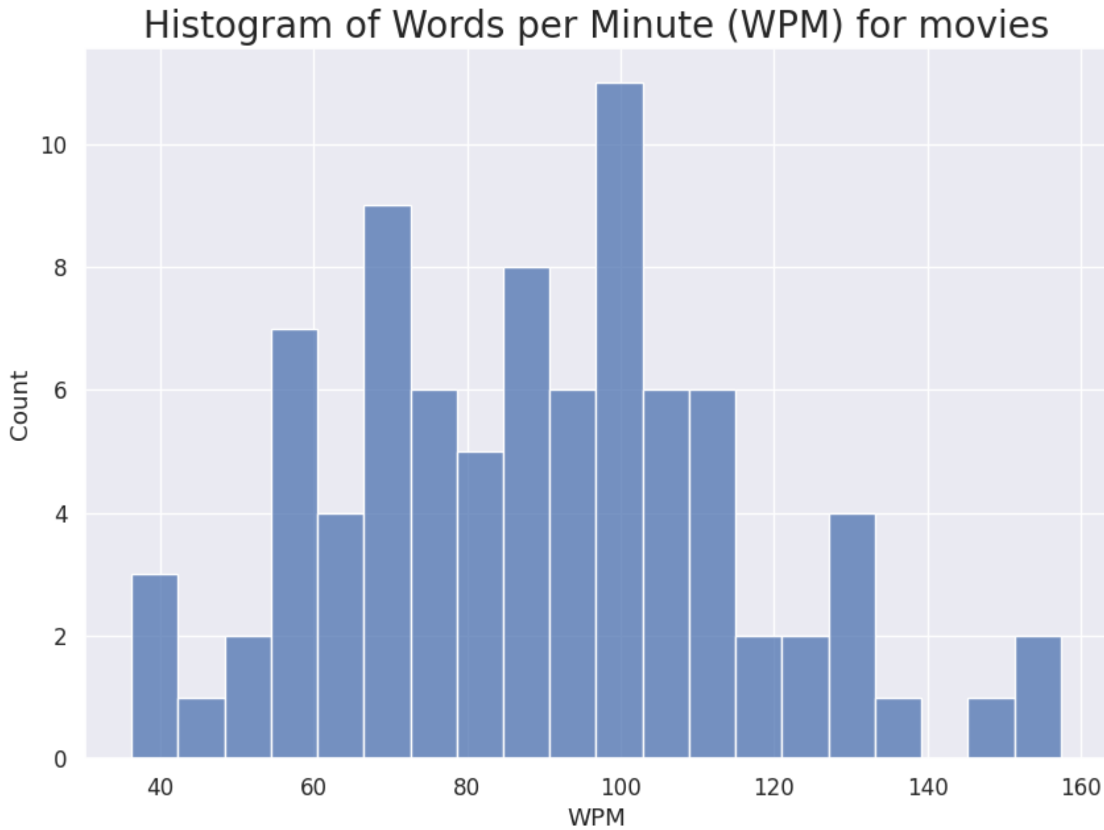
[5945 rows x 3 columns]

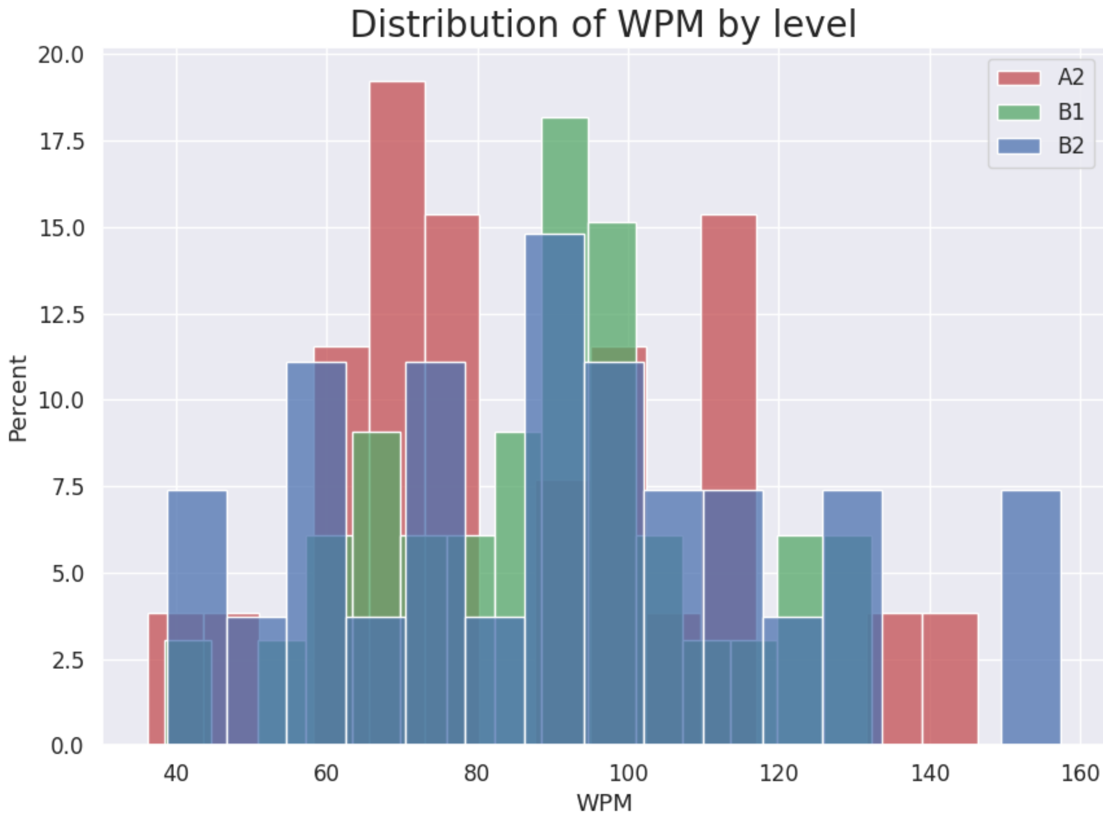
```
                                              movie level  year  \
name
10_cloverfield_lane                10_cloverfield_lane    B1  2016
10_things_i_hate_about_you  10_things_i_hate_about_you    B1  1999
a_knights_tale                          a_knights_tale    B2  2001
a_star_is_born                          a_star_is_born    B2  2018
aladdin                                        aladdin    A2  1992
…                                                …   …    …
twilight                                      twilight    A2  2008
up                                                  up    A2  2009
venom                                            venom    B2  2018
warm_bodies                                warm_bodies    B1  2013
we_are_the_millers                    were_the_millers    B1  2013

                                                     filename A1_count  \
name
10_cloverfield_lane                10_Cloverfield_lane(2016).srt     2366
10_things_i_hate_about_you  10_things_I_hate_about_you(1999).srt     3726
a_knights_tale                          A_knights_tale(2001).srt     2831
a_star_is_born                          A_star_is_born(2018).srt     6248
aladdin                                        Aladdin(1992).srt     3322
…                                                          …       …
twilight                                      Twilight(2008).srt     4127
up                                                  Up(2009).srt     2440
venom                                            Venom(2018).srt     3574
warm_bodies                                Warm_bodies(2013).srt     2159
we_are_the_millers                  We_are_the_Millers(2013).srt     6676

                           A2_count B1_count B2_count C1_count C2_count  …  \
name                                                                     …
10_cloverfield_lane             336      183      160       72       65  …
10_things_i_hate_about_you      482      236      198       95       59  …
a_knights_tale                  457      227      241      110       67  …
a_star_is_born                  835      234      265       78      102  …
aladdin                         559      294      256      104       55  …
…                                 …        …        …        …      …  …
twilight                        592      222      250       61       77  …
up                              362      160      179       42       39  …
venom                           543      219      260      102       68  …
warm_bodies                     307      112      117       51       44  …
we_are_the_millers             1017      364      409      167       90  …

                              C2_pct    Unk_pct 1_verb_lines 2_verb_lines  \
name
10_cloverfield_lane         1.173921  42.532057          485          207
10_things_i_hate_about_you  0.679332  44.778353          478          295
a_knights_tale              0.895124   47.45491          521          222
```

```
a_star_is_born               0.722175  45.043897         913          539
aladdin                      0.618534  48.380567         628          295
…                               …          …         …          …
twilight                      0.80806  44.075979         794          362
up                           0.643777   46.81413         372          224
venom                        0.793558  44.380908         561          285
warm_bodies                  0.902009  42.804428         380          199
we_are_the_millers           0.574896  44.279783         715          514

                           3_verb_lines 4+_verb_lines    duration         wpm  \
name
10_cloverfield_lane                  55            14   96.433333   57.417905
10_things_i_hate_about_you          127            29   95.366667   91.069556
a_knights_tale                       66            12  131.866667   56.761881
a_star_is_born                      134            30       135.2  104.467456
aladdin                              67            25       87.55   101.56482
…                                  …             …           …          …
twilight                             71             4       121.1   78.687036
up                                   59            13   94.966667   63.790804
venom                                99            20   94.716667   90.469822
warm_bodies                          41             4   90.833333   53.702752
we_are_the_millers                  210            70      118.35  132.277144

                           sconj_count word_count
name
10_cloverfield_lane                654       5537
10_things_i_hate_about_you        1362       8685
a_knights_tale                     798       7485
a_star_is_born                    1914      14124
aladdin                            834       8892
…                                 …         …
twilight                          1284       9529
up                                 618       6058
venom                             1152       8569
warm_bodies                        570       4878
we_are_the_millers                1578      15655

[86 rows x 26 columns]
```

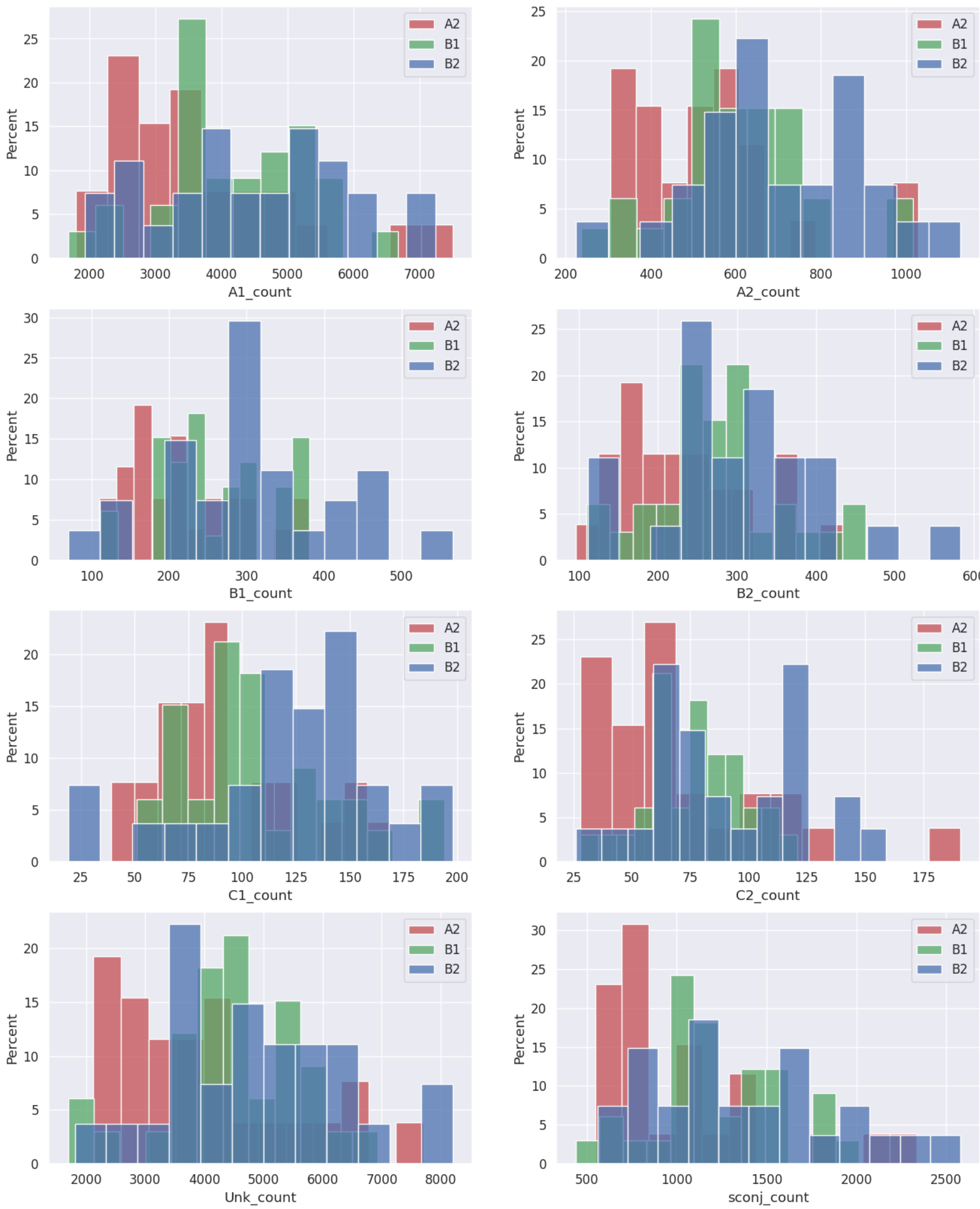

Pie chart of word counts
41.5%
6.2%
2.7%
2.7%
1.1%
0.8%
44.9%
A1_count
A2_count
B1_count
B2_count
C1_count
C2_count
Unk_count

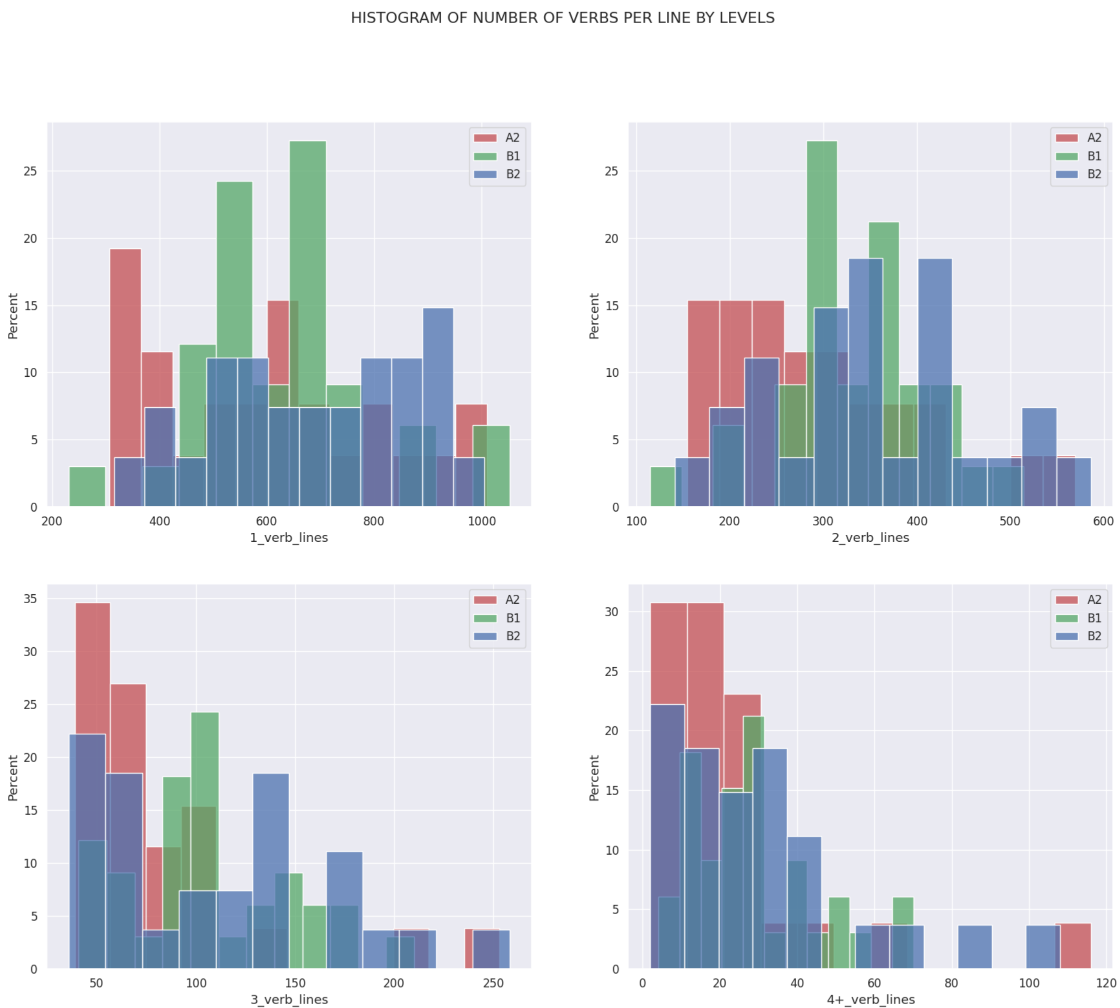

Histogram of word counts for movies
Count
0
2
4
6
8
10
4000
6000
8000
10000
12000
14000
16000
18000
Words


Histogram of Words per Minute (WPM) for movies
Count
WPM
0
2
4
6
8
10
40
60
80
100
120
140
160


Distribution of WPM by level
A2
B1
B2
Percent
WPM
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
40
60
80
100
120
140
160

DISTRIBUTION OF WORD COUNTS BY LEVELS





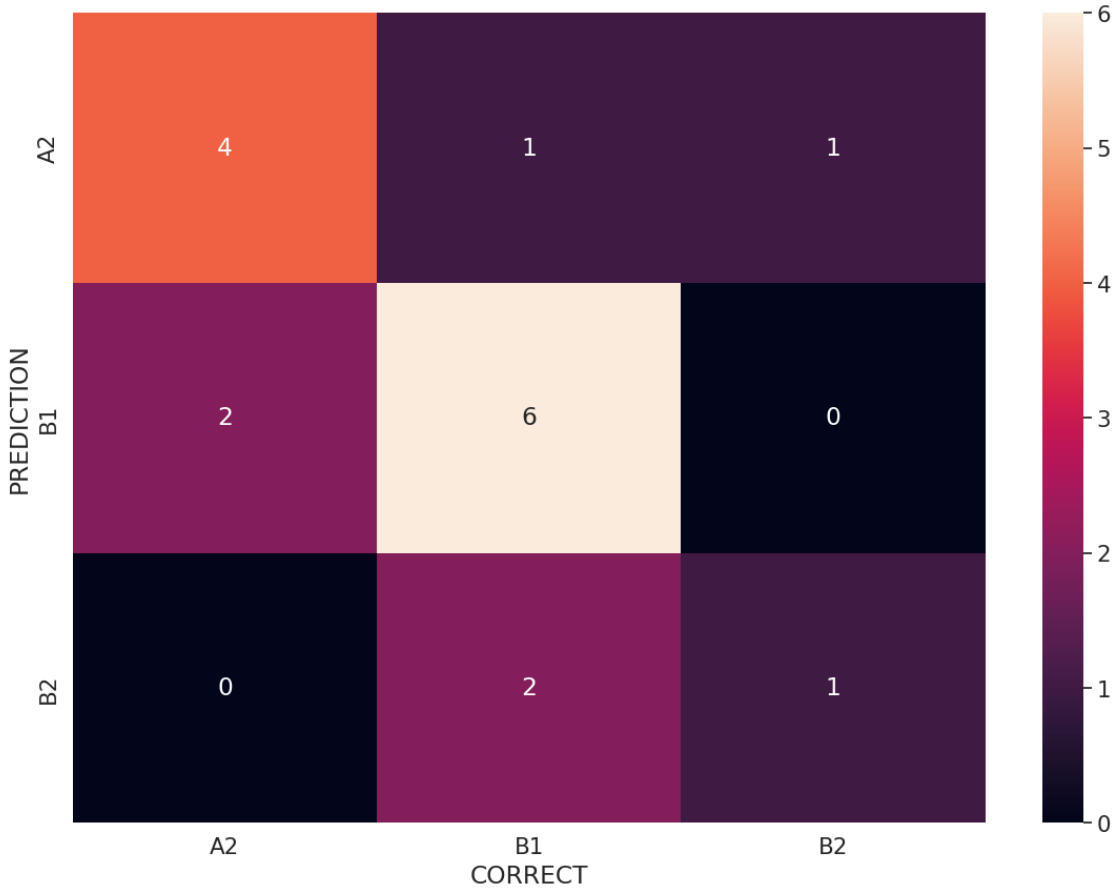
0.9 Model training

	A1_pct	A2_pct	B1_pct	B2_pct	C1_pct	\
name						
twilight	43.309896	6.212614	2.32973	2.62357	0.640151	
mary_poppins_returns	42.203804	7.259676	2.723303	3.42633	0.932435	
mrs_doubtfire	42.855951	6.31781	2.503756	2.562177	1.076615	
the_usual_suspects	42.5023	6.828171	2.851886	2.524788	1.144843	
mamma_mia	45.450131	6.584442	1.92289	2.563854	0.670098	
…	…	…	…	…	…	
the_lion_king	38.570517	5.909381	2.373963	2.169751	0.98277	
pleasantville	42.699782	5.289411	2.244622	2.514808	1.028785	
liar_liar	40.344284	6.594739	2.897321	3.091284	1.127409	
up	40.277319	5.975569	2.641136	2.954771	0.693298	
knives_out	41.63696	6.402396	3.415086	2.673606	1.283331	

```
                       C2_pct    Unk_pct 1_verb_lines 2_verb_lines  \
name
twilight              0.80806  44.075979          794          362
mary_poppins_returns 0.710427  42.744024         1052          473
mrs_doubtfire        0.742781  43.940911          541          356
the_usual_suspects   0.817745  43.330267          506          345
mamma_mia            0.563271  42.245314          676          370
…                         …         …          …           …
the_lion_king        0.727505  49.266114          587          259
pleasantville        0.613114  45.609477          654          307
liar_liar            0.909201  45.035762          499          284
up                   0.643777   46.81413          372          224
knives_out           1.133609  43.455012          710          437

                     3_verb_lines 4+_verb_lines sconj_count         wpm level
name
twilight                       71             4        1284   78.687036    A2
mary_poppins_returns           91             4        2016  107.988812    B1
mrs_doubtfire                 158            38        1236  100.211876    B1
the_usual_suspects            128            41        1158   98.851465    B2
mamma_mia                     105            32        1620   99.793248    B1
…                            …             …          …          …   …
the_lion_king                  39            10         786   89.816584    A2
pleasantville                  90            17        1152    82.73105    B1
liar_liar                      84            23        1020  100.393509    B1
up                             59            13         618   63.790804    A2
knives_out                    166            62        1962  114.763398    B2

[86 rows x 14 columns]
```

```
Splitting dataframes in proportion: 3, 1
Train dataframe size: 51
Test dataframe size: 17
```

```
0.6470588235294118
```



PREDICTION
A2
B1
B2
4
1
1
2
6
0
0
2
1
A2
B1
B2
CORRECT
0
1
2
3
4
5
6