Assignment 5: Topic Models

net id: sa857

Due: Tuesday, November 5.

This assignment has three problems. The first is about Bayesian inference. The second two are about topic models. You will first work with abstracts of scientific articles. These abstracts are obtained from arXiv.org, an open access repository for e-prints of articles in scientific fields maintained by Cornell University. You will then work with a collection of movie plots.

For your convenience, we have separated the problems into three notebooks: assn5_problem1.ipynb, assn5_problem2.ipynb, and assn5_problem3.ipynb. Submit your solutions in these three notebooks, printing out each as a separate pdf.

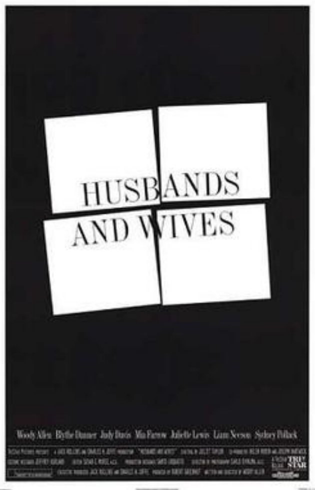
We provide significant "starter code" as discussed in lecture. We then ask you build topic models using the Python library gensim, and do some analysis over the topics obtained.

We ask that you please at least start the assignment right away. If you have any difficulties running gensim we would like to know!

Problem 3: Topic Models on Movie Plots

In this problem we will continue working with topic models, but this time with a new dataset. Instead of abstracts of scientific articles, we will create topic models over movie plot descriptions. This is a dataset containing descriptions of movies from Wikipedia. The dataset was obtained (https://www.kaggle.com/jrobischon/wikipedia-movie-plots) from Kaggle, an online community of data scientists. We again provide extensive starter code to process the data.

Spoiler alert! We will use the movie "Husbands and Wives (https://en.wikipedia.org/wiki/Husbands_and_Wives)" as a running example...



(https://en.wikipedia.org/wiki/Husbands_and_Wives)

In [1]:
```
import numpy as np
import re
import gensim
import pandas as pd
from collections import Counter

import logging
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logg
ing.ERROR)
logging.root.level = logging.CRITICAL

import warnings
warnings.filterwarnings("ignore",category=DeprecationWarning)

# direct plots to appear within the cell, and set their style
%matplotlib inline
import matplotlib.pyplot as plots
plots.style.use('fivethirtyeight')
```

This time around, the movie plot descriptions are in a CSV format in `movie_plots.csv` . The file is hosted on the Amazon Web Service s3. We'll use the `datascience` package to read this CSV file.

In [2]:
```
filename = "https://s3.amazonaws.com/sds171/labs/lab07/movie_plots.csv"
data = pd.read_csv(filename)
data.head(5)
```

Out[2]:

	Release Year	Title	Origin/Ethnicity	Director	Cast	Genre	Wiki Page	Plot
0	1901	Kansas Saloon Smashers	American	Unknown	NaN	unknown	https://en.wikipedia.org /wiki/Kansas_Saloon_Sm...	A bartender is working at a saloon, serving dr...
1	1901	Love by the Light of the Moon	American	Unknown	NaN	unknown	https://en.wikipedia.org /wiki/Love_by_the_Ligh...	The moon, painted with a smiling face hangs ov...
2	1901	The Martyred Presidents	American	Unknown	NaN	unknown	https://en.wikipedia.org /wiki/The_Martyred_Pre...	The film, just over a minute long, is composed...
3	1901	Terrible Teddy, the Grizzly King	American	Unknown	NaN	unknown	https://en.wikipedia.org /wiki/Terrible_Teddy,_...	Lasting just 61 seconds and consisting of two ...
4	1902	Jack and the Beanstalk	American	George S. Fleming, Edwin S. Porter	NaN	unknown	https://en.wikipedia.org /wiki/Jack_and_the_Bea...	The earliest known adaptation of the classic f...

To make the data a little more manageable, we restrict to movies that were released after 1980. We then pull out the titles and plots as lists, for convenience.

In [3]:
```
movies = data[data['Release Year'] > 1980]
titles = list(movies['Title'])
plots = list(movies['Plot'])
```

In [4]:
```
movies.head()
```

Out[4]:

	Release Year	Title	Origin/Ethnicity	Director	Cast	Genre	Wiki Page	Plot
9796	1981	Absence of Malice	American	Sydney Pollack	Paul Newman, Sally Field, Bob Balaban, Melinda...	drama	https://en.wikipedia.org /wiki/Absence_of_Malice	Miami liquor wholesaler Michael Gallagher (Pau...
9797	1981	All Night Long	American	Jean-Claude Tramont	Gene Hackman, Barbra Streisand, Diane Ladd, De...	comedy	https://en.wikipedia.org /wiki/All_Night_Long_(...	George Dupler (Gene Hackman), a married man ne...
9798	1981	...All the Marbles	American	Robert Aldrich	Peter Falk	comedy, drama	https://en.wikipedia.org /wiki/...All_the_Marbles	Harry is the manager of a tag team of gorgeous...
9799	1981	The Amateur	American	Charles Jarrott	John Savage, Christopher Plummer	spy	https://en.wikipedia.org /wiki/The_Amateur_(198...	When his fiancée is murdered by terrorists, a ...
9800	1981	American Pop	American	Ralph Bakshi	Lisa Jane Persky, Ron Thompson	drama, animated	https://en.wikipedia.org /wiki/American_Pop	In Imperial Russia during the late 1890s, a ra...

In [5]:
```
sample = 2015
print("Number of movies: %d\n" % movies.shape[0])
print("Plot of \"%s\":\n" % titles[2015])
print(plots[2015])
```

```
Number of movies: 19994

Plot of "Husbands and Wives":

The film is about two couples: Jack (Pollack) and Sally (Davis), and Gabe (Alle
n) and Judy (Farrow). The film starts when Jack and Sally arrive at Gabe and Jud
y's apartment and announce their separation. Gabe is shocked, but Judy takes the
news personally and is very hurt. Still confused, they go out for dinner at a Ch
inese restaurant.
A few weeks later Sally goes to the apartment of a colleague. They plan to go ou
t together to the opera and then to dinner. Sally asks if she can use his phone,
and calls Jack. Learning from him that he has met someone, she accuses him of ha
ving had an affair during their marriage.
Judy and Gabe are introduced to Jack's new girlfriend, Sam, an aerobics trainer.
While Judy and Sam shop, Gabe calls Jack's new girlfriend a "cocktail waitress"
and tells him that he is crazy for leaving Sally for her. About a week later, Ju
dy introduces Sally to Michael (Neeson), Judy's magazine colleague who she clear
ly is interested in herself. Michael asks Sally out, and they begin dating; Mich
ael is smitten, but Sally is dissatisfied with the relationship.
Meanwhile, Gabe has developed a friendship with a young student of his, Rain, an
d has her read the manuscript of his novel. She comments on its brilliance, but
has several criticisms, to which Gabe reacts defensively.
At a party, Jack learns from a friend that Sally is seeing someone, and flies in
to a jealous rage. He and Sam break up after an intense argument, and Jack drive
s back to his house to find Sally in bed with Michael. He asks Sally to give the
ir marriage another chance, but she tells him to leave.
Less than two weeks later, however, Jack and Sally are back together and the cou
ple meet Judy and Gabe for dinner like old times. After dinner, Judy and Gabe ge
t into an argument about her not sharing her poetry. After Gabe makes a failed p
ass at her, Judy tells him she thinks the relationship is over; a week later Gab
e moves out. Judy begins seeing Michael.
Gabe goes to Rain's 21st birthday party, and gives her a music box as a present.
She asks him to kiss her, and though the two share a romantic moment, Gabe tells
her they should not pursue it any further. As he walks home in the rain, he real
izes that he has ruined his relationship with Judy.
Michael tells Judy he needs time alone, then says he can't help still having fee
lings for Sally. Angry and hurt, Judy walks out into the rain. Highlighting her
"passive aggressiveness," Michael follows and begs her to stay with him. A year
and a half later they marry.
At the end, the audience sees a pensive Jack and Sally back together. Jack and S
ally admit their marital problems still exist (her frigidity is not solved), but
they find they accept their problems as simply the price they have to pay to rem
ain together.
Gabe is living alone because he says he is not dating for the time being, as he
does not want to hurt anyone. The film ends with an immediate cut to black after
Gabe asks the unseen documentary crew, "Can I go? Is this over?"
```

This plot description is from the movie "[Husbands and Wives (https://en.wikipedia.org/wiki/Husbands_and_Wives)](https://en.wikipedia.org/wiki/Husbands_and_Wives)"

We don't have LaTeX markup in these documents, but we'll still use some regular expressions to do some simpe pre-processing of punctuation. There are lots of names in the plot descriptions, so we'll remove all the words that have a capitalized first letter. This will remove lots of non-name words as well, but this'll be sufficient for our goal of building a basic topic model.

In [6]:
```
# replace '-' with ' ', then remove punctuation
plots = [re.sub('-', ' ', plot) for plot in plots]
plots = [re.sub('[^\w\s]', '', plot) for plot in plots]

# remove tokens with a capitalized first letter
# (broad stroke to remove names)
plots = [re.sub('[A-Z]\w*', '', plot) for plot in plots]
# replace multiple spaces by a single space
plots = [re.sub('[ ]+', ' ', plot) for plot in plots]

print(plots[sample])
```

```
 film is about two couples and and and film starts when and arrive at and apartm
ent and announce their separation is shocked but takes the news personally and i
s very hurt confused they go out for dinner at a restaurant
 few weeks later goes to the apartment of a colleague plan to go out together to
the opera and then to dinner asks if she can use his phone and calls from him th
at he has met someone she accuses him of having had an affair during their marri
age
 and are introduced to new girlfriend an aerobics trainer and shop calls new gir
lfriend a cocktail waitress and tells him that he is crazy for leaving for her a
week later introduces to magazine colleague who she clearly is interested in her
self asks out and they begin dating is smitten but is dissatisfied with the rela
tionship
 has developed a friendship with a young student of his and has her read the man
uscript of his novel comments on its brilliance but has several criticisms to wh
ich reacts defensively
 a party learns from a friend that is seeing someone and flies into a jealous ra
ge and break up after an intense argument and drives back to his house to find i
n bed with asks to give their marriage another chance but she tells him to leave
 than two weeks later however and are back together and the couple meet and for
dinner like old times dinner and get into an argument about her not sharing her
poetry makes a failed pass at her tells him she thinks the relationship is over
a week later moves out begins seeing
 goes to 21st birthday party and gives her a music box as a present asks him to
kiss her and though the two share a romantic moment tells her they should not pu
rsue it any further he walks home in the rain he realizes that he has ruined his
relationship with
 tells he needs time alone then says he cant help still having feelings for and
hurt walks out into the rain her passive aggressiveness follows and begs her to
stay with him year and a half later they marry
 the end the audience sees a pensive and back together and admit their marital p
roblems still exist her frigidity is not solved but they find they accept their
problems as simply the price they have to pay to remain together
 is living alone because he says he is not dating for the time being as he does
not want to hurt anyone film ends with an immediate cut to black after asks the
unseen documentary crew go this over
```

Now, we further process each plot description by converting it to lower case, stripping leading and trailing white space, and then tokenizing by splitting on spaces.

In [7]:
```
plots_tok = []
for plot in plots:
    processed = plot.lower().strip().split(' ')
    plots_tok.append(processed)
```

3.1 Further cleaning

As in problem 2, we will remove tokens that have digits, possessives or contractions, or are empty strings.

- `is_numeric(string)` checks if `string` has any numbers
- `has_poss_contr(string)` checks if `string` has possessives or contractions
- `empty_string(string)` checks if `string` is an empty string
- `remove_string(string)` checcks if `string` should be removed

In [8]:
```python
def is_numeric(string):
    return bool(re.search('\d', string))

def has_poss_contr(string):
    blacklist_in = ["'s", "'m", "'re", "'ve", "'d", "'ll", "'t"]
    return any(i in string for i in blacklist_in)

def empty_string(string):
    return not bool(string)

def remove_string(string):
    return is_numeric(string) | has_poss_contr(string) | empty_string(string)
```

In [9]:
```python
temp = []
for plot in plots_tok:
    filtered = []
    for token in plot:
        if not remove_string(token):
            filtered.append(token)
    temp.append(filtered)
plots_tok = temp
```

Recall that to build topic models, we require the following components:

- A vocabulary of tokens that appear across all documents.
- A mapping of those tokens to a unique integer identifier, because topic model algorithms treat words by these identifiers, and not the strings themselves. For example, we represent `'epidemic'` as `word2id['epidemic'] = 50`
- The corpus, where each document in the corpus is a collection of tokens, where each token is represented by the identifier and the number of times it appears in the document. For example, in the first document above the token `'epidemic'`, which appears twice, is represented as `(50, 2)`

Now we will build a vocabulary representing the tokens that have appeared across all the plot descriptions we have.

Recall that we can use the `Counter` class to build the vocabulary. The `Counter` is an extension of the Python dictionary, and also has key-value pairs. For the `Counter`, keys are the objects to be counted, while values are their counts.

```
In [10]: vocab = Counter()
         for plot in plots_tok:
             vocab.update(plot)

         print("Number of unique tokens: %d" % len(vocab))
```

```
Number of unique tokens: 56508
```

Recall that removing rare words helps prevent our vocabulary from being too large. Many tokens appear only a few times across all the plot descriptions. Keeping them in the vocabulary increases subsequent computation time. Furthermore, their presence tends not to carry much significance for a document, since they can be considered as anomalies.

We remove rare words by only keeping tokens that appear more than 25 times across all plot descriptions.

```
In [11]: tokens = []
         for token in vocab.elements():
             if vocab[token] >= 50:
                 tokens.append(token)
         vocab = Counter(tokens)

         print("Number of unique tokens: %d" % len(vocab))
```

```
Number of unique tokens: 7793
```

Recall that stop words are defined as very common words such as `'the'` and `'a'`. Removing stop words is important because their presence also does not carry much significance, since they appear in all kinds of texts.

We will remove stop words by removing the 200 most common tokens across all the plot descriptions.

```
In [12]: stop_words = []
         for item in vocab.most_common(200):
             stop_word = item[0]
             stop_words.append(stop_word)
         tokens = []
         for token in vocab.elements():
             if token not in stop_words:
                 tokens.append(token)
         vocab = Counter(tokens)

         print("Number of unique tokens: %d" % len(vocab))
```

```
Number of unique tokens: 7593
```

Now we create a mapping for tokens to unique identifiers.

In [13]:
```
items = vocab.items()
id2word = {}
word2id = {}
idx = 0
for word, count in vocab.items():
    id2word[idx] = word
    word2id[word] = idx
    idx += 1

print("Number of tokens mapped: %d" % len(id2word))
print("Identifier for 'photograph': %d" % word2id['photograph'])
print("Word for identifier %d: %s" % (word2id['photograph'], id2word[word2id['phot
ograph']]))
```

```
Number of tokens mapped: 7593
Identifier for 'photograph': 1781
Word for identifier 1781: photograph
```

Now, we will remove, for each plot description, the tokens that are not found in our vocabulary.

In [14]:
```
temp = []
for plot in plots_tok:
    filtered = []
    for token in plot:
        if token in vocab:
            filtered.append(token)
    temp.append(filtered)
plots_tok = temp
```

Let's create the corpus. Recall that the corpus should have the format

```
[(1841, 2), (2095, 2), (2096, 1), (2097, 1), (2098, 2), (105, 2), (2099, 1), (2100,
1), (270, 2), (1763, 1), (1870, 1), (2101, 1), (2017, 4), (633, 1), (1270, 1), (1093,
1), (2102, 1), (1197, 1), (113, 1), (1583, 1), (2103, 1), (2104, 2), (2105, 1), (873,
1), (1950, 1), (107, 1), (2106, 1), (2107, 1), (116, 1), (1436, 1), (62, 1), (2108,
1), (213, 1), (2109, 1), (1205, 1), (2110, 1), (1042, 1), (1275, 1), (1259, 1), (1342,
1), (2111, 1), (440, 1), (1662, 1), (374, 1), (663, 1)]
```

where each element is a pair containing the identifier for the token and the count of that token in just that plot description.

In [15]:
```
corpus = []
for plot in plots_tok:
    plot_count = Counter(plot)
    corpus_doc = []
    for item in plot_count.items():
        pair = (word2id[item[0]], item[1])
        corpus_doc.append(pair)
    corpus.append(corpus_doc)

print("Plot, tokenized:\n", plots_tok[sample], "\n")
print("Plot, in corpus format:\n", corpus[sample])
```

```
Plot, tokenized:
 ['couples', 'arrive', 'apartment', 'announce', 'separation', 'shocked', 'news',
'personally', 'very', 'hurt', 'confused', 'dinner', 'few', 'weeks', 'apartment',
'colleague', 'plan', 'opera', 'dinner', 'use', 'phone', 'calls', 'met', 'someone
', 'accuses', 'affair', 'marriage\r\n', 'introduced', 'girlfriend', 'trainer', '
shop', 'calls', 'girlfriend', 'cocktail', 'waitress', 'crazy', 'leaving', 'week
', 'introduces', 'magazine', 'colleague', 'clearly', 'interested', 'herself', 'b
egin', 'dating', 'smitten', 'dissatisfied', 'relationship\r\n', 'developed', 'fr
iendship', 'student', 'read', 'manuscript', 'novel', 'comments', 'several', 'rea
cts', 'learns', 'seeing', 'someone', 'flies', 'jealous', 'rage', 'break', 'inten
se', 'argument', 'drives', 'bed', 'give', 'chance', 'leave\r\n', 'than', 'weeks
', 'however', 'couple', 'dinner', 'times', 'dinner', 'argument', 'sharing', 'poe
try', 'failed', 'pass', 'thinks', 'week', 'moves', 'seeing', 'birthday', 'music
', 'box', 'present', 'kiss', 'though', 'share', 'romantic', 'moment', 'should',
'pursue', 'any', 'further', 'walks', 'rain', 'realizes', 'ruined', 'needs', 'alo
ne', 'cant', 'feelings', 'hurt', 'walks', 'rain', 'follows', 'begs', 'stay', 'ye
ar', 'half', 'audience', 'admit', 'marital', 'problems', 'exist', 'solved', 'acc
ept', 'problems', 'simply', 'price', 'pay', 'remain', 'together\r\n', 'living',
'alone', 'dating', 'want', 'hurt', 'anyone', 'immediate', 'cut', 'black', 'unsee
n', 'documentary', 'crew']

Plot, in corpus format:
 [(2759, 1), (1274, 1), (512, 2), (2701, 1), (4987, 1), (2088, 1), (734, 1), (50
08, 1), (669, 1), (3567, 3), (3177, 1), (673, 4), (3519, 1), (346, 2), (3132,
2), (71, 1), (3832, 1), (77, 1), (96, 1), (116, 2), (929, 1), (1783, 2), (2534,
1), (93, 1), (4373, 1), (2137, 1), (1303, 2), (2291, 1), (4051, 1), (6758, 1),
(350, 1), (1233, 1), (1280, 1), (2655, 2), (2774, 1), (3817, 1), (793, 1), (171
4, 1), (1608, 1), (884, 1), (2987, 2), (2515, 1), (3599, 1), (102, 1), (1799,
1), (1429, 1), (4070, 1), (58, 1), (3896, 1), (5566, 1), (5045, 1), (646, 1), (3
597, 1), (718, 1), (1749, 2), (2124, 1), (1560, 1), (1037, 1), (2310, 1), (844,
1), (1182, 2), (342, 1), (1013, 1), (80, 1), (1123, 1), (5869, 1), (627, 1), (25
2, 1), (681, 1), (840, 1), (3607, 1), (5688, 1), (1029, 1), (2306, 1), (206, 1),
(220, 1), (3796, 1), (409, 1), (3803, 1), (3012, 1), (2636, 1), (120, 1), (2823,
1), (2107, 1), (571, 1), (434, 1), (194, 1), (166, 1), (131, 1), (913, 2), (115
3, 2), (202, 1), (4783, 1), (700, 1), (639, 2), (1456, 1), (624, 1), (802, 1),
(2214, 1), (436, 1), (935, 1), (280, 1), (2627, 1), (3018, 1), (3490, 1), (2929,
2), (6241, 1), (7357, 1), (677, 1), (3017, 1), (5104, 1), (299, 1), (2775, 1),
(4654, 1), (664, 1), (50, 1), (482, 1), (3220, 1), (503, 1), (1299, 1), (1832,
1), (6176, 1), (1269, 1)]
```

Now, we are ready to create our topic model!

We again use gensim, a Python library to create topic models. Also, we again use the algorithm called latent dirichlet allocation implemented in the gensim library.

This step takes about 2 minutes

In [16]:
```
#%%time
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=10,
                                            random_state=100,
                                            update_every=1,
                                            chunksize=100,
                                            passes=10,
                                            alpha='auto',
                                            per_word_topics=True)
```

In [17]:
```
num_topics = 10
num_words = 15
top_words = pd.DataFrame({'word rank': np.arange(1,num_words+1)})
for k in np.arange(num_topics):
    topic = lda_model.get_topic_terms(k, num_words)
    words = [id2word[topic[i][0]] for i in np.arange(num_words)]
    probs = [topic[i][1] for i in np.arange(num_words)]
    top_words['topic %d' % k] = words

top_words
```

Out[17]:

	word rank	topic 0	topic 1	topic 2	topic 3	topic 4	topic 5	topic 6	topic 7	topic 8	topic
0	1	suicide	attack	power	gang	even	movie	train	town	student	so
1	2	due	soldiers	world	murder	know	company	call	boy	girls	he
2	3	these	war	battle	case	marry	show	phone	found	college	apartmer
3	4	law	mission	save	officer	husband	part	runs	appears	high	clu
4	5	many	ship	human	revenge	very	dream	calls	blood	students	jo
5	6	any	military	use	plan	live	play	station	search	woo	[illegible]
6	7	incident	orders	uses	killing	never	dreams	scene	mysterious	teacher	tap
7	8	under	army	form	ho	without	world	inside	herself	class	phot
8	9	order	crew	destroy	arrested	much	around	outside	ago	game	alier
9	10	those	forces	city	prison	too	match	morning	revealed	year	com
10	11	several	government	defeat	killer	child	different	head	around	won	baseba
11	12	state	battle	villagers	boss	doesnt	music	video	face	win	journalis
12	13	further	leader	using	shot	since	called	stop	suddenly	boys	nar
13	14	court	bomb	called	crime	days	big	door	across	four	woc
14	15	including	camp	control	henchmen	good	director	leaving	dog	kids	stadiur

Topics for Movies

Your task is now to carry out the same steps as for problem 2 (arXiv abstracts), but now for this dataset of movie plots

3.2 Label the Topics

Label all the 10 topics with your interpretation of what the topics are.

In [18]:
```
mapTopicsToLabels = {
    0 : "murder mystery",
    1 : "military war",
    2 : "alien invasion",
    3 : "gang war",
    4 : "love story",
    5 : "hollywood ambition",
    6 : "travel",
    7 : "thriller",
    8 : "school life",
    9 : "k drama",
}
```

In [19]:
```
new_names = ["word rank"] + [v for v in mapTopicsToLabels.values()]
top_words.columns=new_names
top_words
```

Out[19]:

	word rank	murder mystery	military war	alien invasion	gang war	love story	hollywood ambition	travel	thriller	school life	k dra
0	1	suicide	attack	power	gang	even	movie	train	town	student	
1	2	due	soldiers	world	murder	know	company	call	boy	girls	
2	3	these	war	battle	case	marry	show	phone	found	college	apartn
3	4	law	mission	save	officer	husband	part	runs	appears	high	(
4	5	many	ship	human	revenge	very	dream	calls	blood	students	
5	6	any	military	use	plan	live	play	station	search	woo	
6	7	incident	orders	uses	killing	never	dreams	scene	mysterious	teacher	t
7	8	under	army	form	ho	without	world	inside	herself	class	ph
8	9	order	crew	destroy	arrested	much	around	outside	ago	game	ali
9	10	those	forces	city	prison	too	match	morning	revealed	year	cc
10	11	several	government	defeat	killer	child	different	head	around	won	base
11	12	state	battle	villagers	boss	doesnt	music	video	face	win	journa
12	13	further	leader	using	shot	since	called	stop	suddenly	boys	r
13	14	court	bomb	called	crime	days	big	door	across	four	w
14	15	including	camp	control	henchmen	good	director	leaving	dog	kids	stad

In [20]:
```
labels = pd.DataFrame(
    zip(mapTopicsToLabels.keys(), mapTopicsToLabels.values()),
    columns=["topic_num", "topic_label"])
labels
```

Out[20]:

	topic_num	topic_label
0	0	murder mystery
1	1	military war
2	2	alien invasion
3	3	gang war
4	4	love story
5	5	hollywood ambition
6	6	travel
7	7	thriller
8	8	school life
9	9	k drama

3.3 Table of Topics for Movies

Create a function `create_movie_table(data, abstracts, corpus, lda_model)` which does the following:

- Goes through every movie plot and finds the most likely topic for that plot.
- Creates a table `movie_table` that has the following columns
  - `title` : the title of the movie
  - `topic` : the topic number of the most likely topic for each abstract
  - `label` : the topic label of that topic number, which you assigned in part 1
  - `prob` : the probability of that topic number
  - `plot` : a string containing the first 200 characters of the plot
- Show the first 10 rows of the table, then return the table

In [31]:
```
def create_topic_table(data, plots, corpus, lda_model):
    # initialize some arrays
    title = []
    topic = []
    label = []
    probs = []

    for sample in np.arange(len(corpus)):
        # topic, label and probs
        topic_dist = lda_model.get_document_topics(corpus[sample])
        _topics = [pair[0] for pair in topic_dist]
        _probabilities = [pair[1] for pair in topic_dist]
        _t = np.argmax(_probabilities)
        probs.append(_probabilities[_t])
        topic.append(_topics[_t])
        label.append(mapTopicsToLabels[_topics[_t]])

    table = pd.DataFrame()
    table['title'] = titles
    table['topic'] = topic # attach the entire column
    table['label'] = label
    table['prob'] = probs
    table['plot'] = plots
    # You'll need to add the topic, label, and probability for each abstract
    return table
```

In [32]:
```
topic_table = create_topic_table(data, plots, corpus, lda_model)
topic_table.head(20)
```

Out[32]:

	title	topic	label	prob	plot
0	Absence of Malice	3	gang war	0.333318	liquor wholesaler who is the son of a decease...
1	All Night Long	4	love story	0.363166	a married man nearing middle age is demoted a...
2	...All the Marbles	7	thriller	0.268781	is the manager of a tag team of gorgeous lady...
3	The Amateur	4	love story	0.360593	his fiancée is murdered by terrorists a crypt...
4	American Pop	5	hollywood ambition	0.352425	during the late 1890s a rabbis wife and her y...
5	An American Werewolf in London	7	thriller	0.327727	backpackers and are trekking across the moors...
6	Amy	4	love story	0.307463	is a dutiful housewife of the early 20th cent...
7	Arthur	4	love story	0.382241	is a spoiled alcoholic from who likes to be d...
8	Back Roads	2	alien invasion	0.199472	is a 20 a trick hooker in night she entertain...
9	Blow Out	6	travel	0.231888	in post production on a low budget slasher fi...
10	Body Heat	3	gang war	0.248278	a particularly intense heatwave inept lawyer ...
11	Buddy Buddy	0	murder mystery	0.225376	has been hired to eliminate before he testifi...
12	Burned at the Stake	4	love story	0.274218	the of 1692 a group of witches are burned at ...
13	The Burning	6	travel	0.388388	night at several campers pull a prank on the ...
14	Bustin' Loose	6	travel	0.195808	is a convict who violates his parole after a ...
15	The Cannonball Run	6	travel	0.298051	teams have gathered in to start a cross count...
16	Carbon Copy	4	love story	0.273836	a black man is the long lost son of a white b...
17	Cattle Annie and Little Britches	4	love story	0.204206	outlaws the girls find are the demoralized re...
18	Caveman	7	thriller	0.221793	is a bullied and scrawny caveman living in 9t...
19	Charlie Chan and the Curse of the Dragon Queen	4	love story	0.282651	detective is asked for his help by the police...

3.4 Analysis for selected movies

Choose at least five movies, including 'Husbands and Wives' and discuss how the assignment of topics either does or does not make sense, according to your own understanding of the movies. Note that Wikipedia pages are given for most of the movies in the original data. For example, https://en.wikipedia.org/wiki/Absence_of_Malice (https://en.wikipedia.org/wiki/Absence_of_Malice) is the page for "Absence of Malice"

```
In [33]: topic_table.iloc[30:35]
```

Out[33]:

	title	topic	label	prob	plot
30	Death Hunt	7	thriller	0.267537	the in 1931 a solitary trapper comes across a...
31	The Devil and Max Devlin	4	love story	0.292125	is a shady landlord of a rundown tenement in ...
32	Dragonslayer	2	alien invasion	0.397013	sixth century post kingdom called is being te...
33	Endless Love	4	love story	0.358182	suburban teenagers and fall in love after the...
34	Enter the Ninja	5	hollywood ambition	0.280889	a veteran of the completes his ninjutsu train...

- 30: The movie is actually an action film.
- 31: It is actually fantasy-comedy. In a sense, it is close, if we were to consider 'love story' as romcom
- 32: It is actually fantasy-action. In a sense, it is close, if we were to connsider 'alien invasion' more generally as a fantasy/sci-fi topic.
- 33: It is the correct topic.
- 34: This is action/martial arts so it is not close at all. Although it might be similar because of concepts like training and ambition to become a good ninja.

In summary, my own conceptions of the topics did not capture an appropriate level of generality. It's possible the topic model is looking at more general similarities e.g. fantasy, action or ambition.

3.5 Extra credit: Improve the model

For extra credit, improve the topic model by improving the processing of the data and the vocabulary, and selecting a more appropriate number of topics. Describe how your new model gives an improvement over the "quick and dirty" topic model built above.