STEVENS INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY®

Classification of Movie Reviews with Latent Semantic Analysis

Using SVD

Taekkeun Nam, Houming Kuang, Kristin Kim

Motivation

A. **Something interesting!**
B. **Taking advantage of what we learned in the course!**

1. **Shared interests?** Movies
2. **Model?** Movies & Knowledge from the course
3. **Why LSA?** Simple yet powerful with SVD & TF-IDF
4. **On the side?** CounterVectorizer & SVM
5. **Object?** To classify movie reviews into Romance/Comedy vs. Action/Horror

Introduction

[Work Flow]

Data Importing -> Data Cleaning ->

- **LSA part**

1. **CountVectorizer(BOW) -> SVD -> Fitting LSA**
2. **TF-IDF -> SVD -> Fitting LSA**

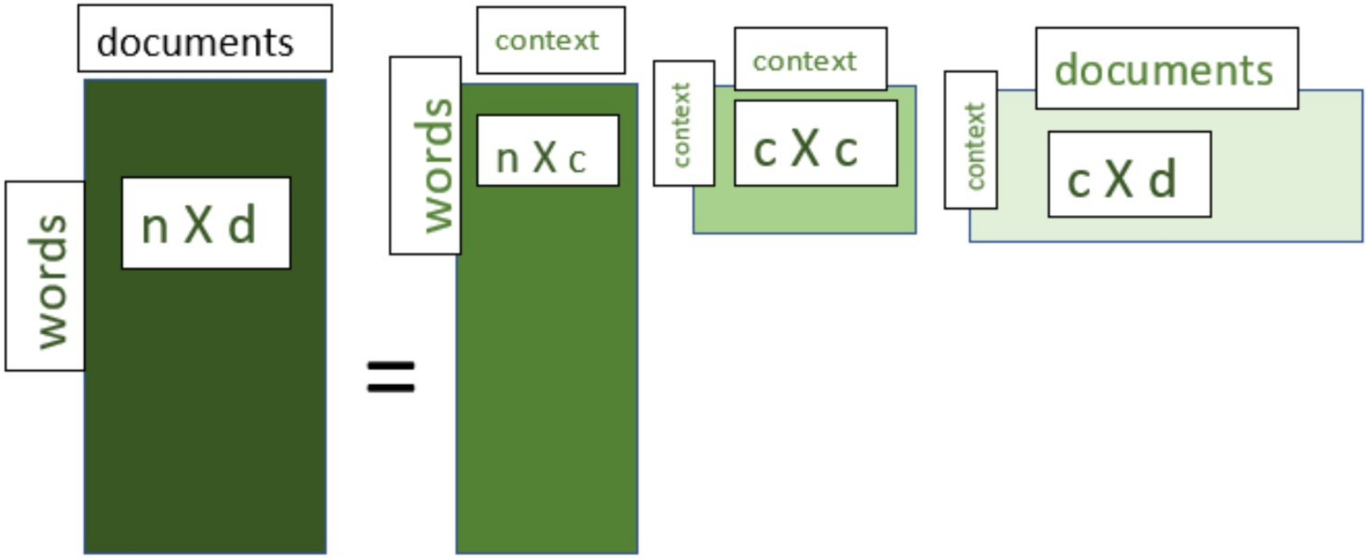
- **SVM part**

1. **Fitting SVM(without parameters)**
2. **GridsearchCV -> Fitting SVM(with parameters)**

- Dataset : “Rotten Tomatoes Movies and critic Reviews” from <Kaggle>

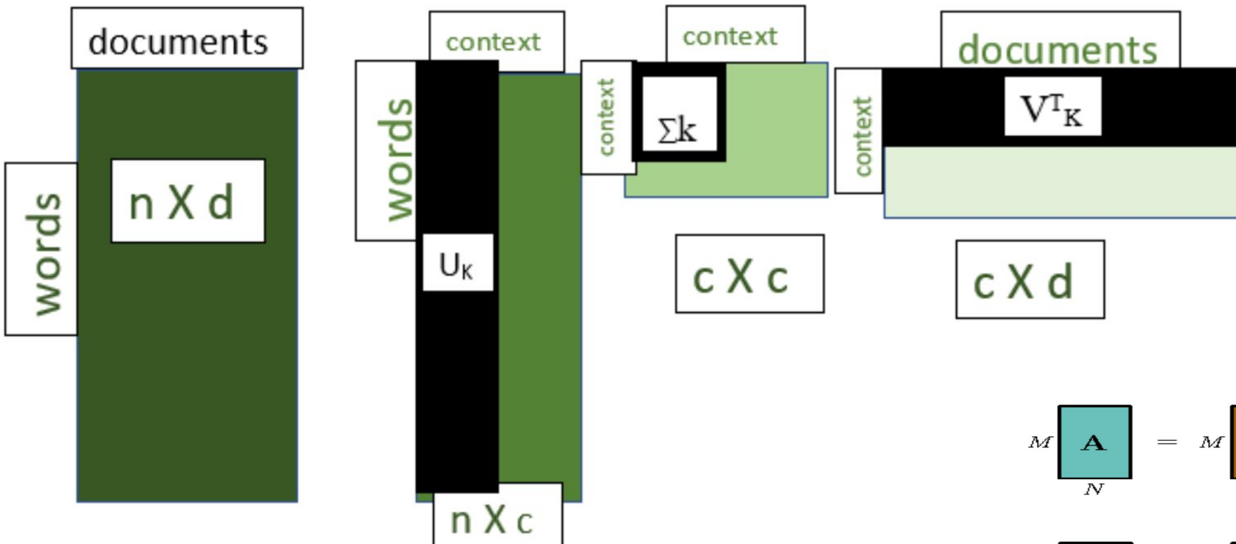
Latent Semantic Analysis (LSA)

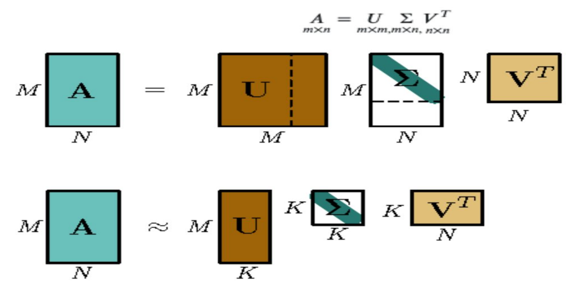
- unsupervised learning
- natural language processing for topic modeling
- discovers relationships between documents and the words that they contain
- uses the statistical approach to identify the association
- uses Single Value Decomposition(SVD) to extract the distinct features and finds hidden patterns and insights



Mathematics of SVD ($U\Sigma V^T$)

- In LSA, SVD allows to truncate few contexts that are not necessarily required by us





(c) truncated SVD

∑ matrix

- provides the diagonal values which represent the significance of the context from highest to the lowest.
- allows to reduce the dimensions
- K : the number of topic

TF-IDF
(term frequency-inverse document frequency)

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF
Term x within document y

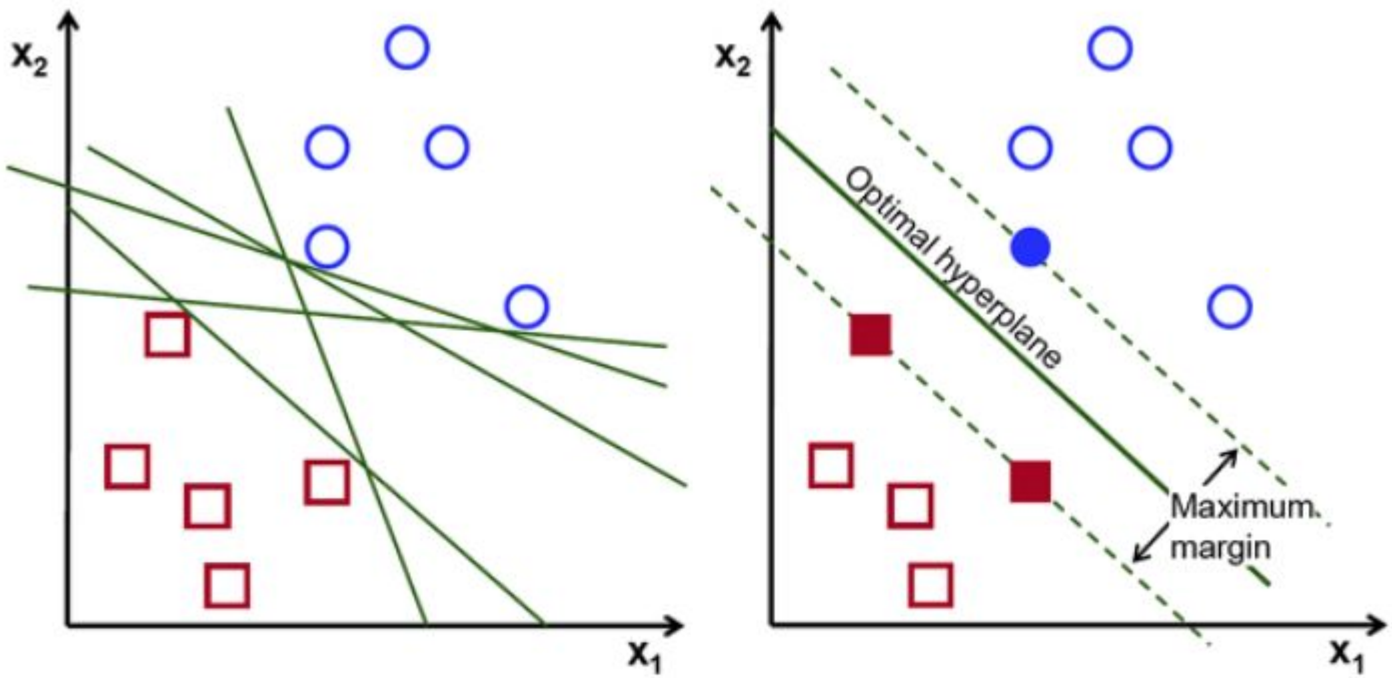
$tf_{x,y}$ = frequency of x in y
df_x = number of documents containing x
N = total number of documents

Word	TF		IDF	TF*IDF	
	A	B		A	B
The	1/7	1/7	log(2/2) = 0	0	0
Car	1/7	0	log(2/1) = 0.3	0.043	0
Truck	0	1/7	log(2/1) = 0.3	0	0.043
Is	1/7	1/7	log(2/2) = 0	0	0
Driven	1/7	1/7	log(2/2) = 0	0	0
On	1/7	1/7	log(2/2) = 0	0	0
The	1/7	1/7	log(2/2) = 0	0	0
Road	1/7	0	log(2/1) = 0.3	0.043	0
Highway	0	1/7	log(2/1) = 0.3	0	0.043

A = "The car is driven on the road"; B = "The truck is driven on the highway" Image from freeCodeCamp - How to

SVM
(Support Vector Machine)

- this algorithm can be used for both regression and classification tasks
- more widely used in classification objectives
- objective : to find an optimal hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points



Possible hyperplanes

Implementation

Now, Time to check out our code!

https://colab.research.google.com/drive/1vIebBf_mgPH9yjfRi2Oweip_9yTsAqjf?authuser=2#scrollTo=oYsCHxxIlz82

Dataset - Initial Dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17712 entries, 0 to 17711
Data columns (total 22 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   rotten_tomatoes_link             17712 non-null  object
 1   movie_title                      17712 non-null  object
 2   movie_info                       17391 non-null  object
 3   critics_consensus                9134 non-null   object
 4   content_rating                   17712 non-null  object
 5   genres                           17693 non-null  object
 6   directors                        17518 non-null  object
 7   authors                          16170 non-null  object
 8   actors                           17360 non-null  object
 9   original_release_date            16546 non-null  object
 10  streaming_release_date           17328 non-null  object
 11  runtime                          17398 non-null  float64
 12  production_company               17213 non-null  object
 13  tomatometer_status               17668 non-null  object
 14  tomatometer_rating               17668 non-null  float64
 15  tomatometer_count                17668 non-null  float64
 16  audience_status                  17264 non-null  object
 17  audience_rating                  17416 non-null  float64
 18  audience_count                   17415 non-null  float64
 19  tomatometer_top_critics_count    17712 non-null  int64
 20  tomatometer_fresh_critics_count  17712 non-null  int64
 21  tomatometer_rotten_critics_count 17712 non-null  int64
dtypes: float64(5), int64(3), object(14)
memory usage: 3.0+ MB
```

Dataset - Initial Dataset

```
array(['Action & Adventure, Comedy, Drama, Science Fiction & Fantasy',
       'Comedy', 'Comedy, Romance', ...,
       'Animation, Art House & International, Drama, Science Fiction & Fantasy, Romance',
       'Art House & International, Romance',
       'Action & Adventure, Drama, Horror, Kids & Family, Mystery & Suspense'],
      dtype=object)
```

Dataset - Final Dataset after cleaning

- 53 rows of Action / 53 rows of Romantic Comedy
- 2 columns of Reviews / Genres

	critics_consensus	genres
2307	Cobbling together an unfinished satire on the ...	Comedy, Romance
12048	Primeval is a low-quality horror film, which d...	Action & Adventure, Drama, Horror, Mystery & S...
3012	Provides lots of laughs with Myers at the heal...	Comedy, Romance
5456	Though it is ultimately somewhat undone by its...	Action & Adventure, Drama, Horror, Mystery & S...
3598	Insubstantial yet charming, Billy's Hollywood ...	Comedy, Romance

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 106 entries, 2 to 17645
Data columns (total 2 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   critics_consensus  106 non-null    object
 1   genres             106 non-null    object
dtypes: object(2)
memory usage: 2.5+ KB
```

(CountVectorizer) LSA

1. **CountVectorizer()** to vectorize the words of reviews

```
count_vec = CountVectorizer(min_df=1, stop_words='english')
bag_of_words_count = count_vec.fit_transform(action_romcom.critics_consensus)
bag_of_words_count.todense()
```

2. TruncatedSVD() to fit Bag of Words

```
svd = TruncatedSVD(n_components=2,n_iter=100)
lsa = svd.fit_transform(bag_of_words_count)
```

(CountVectorizer) LSA

Red) result of LSA (finding hidden patterns & separate into two categories)

Blue) Convert categorical values to binary values

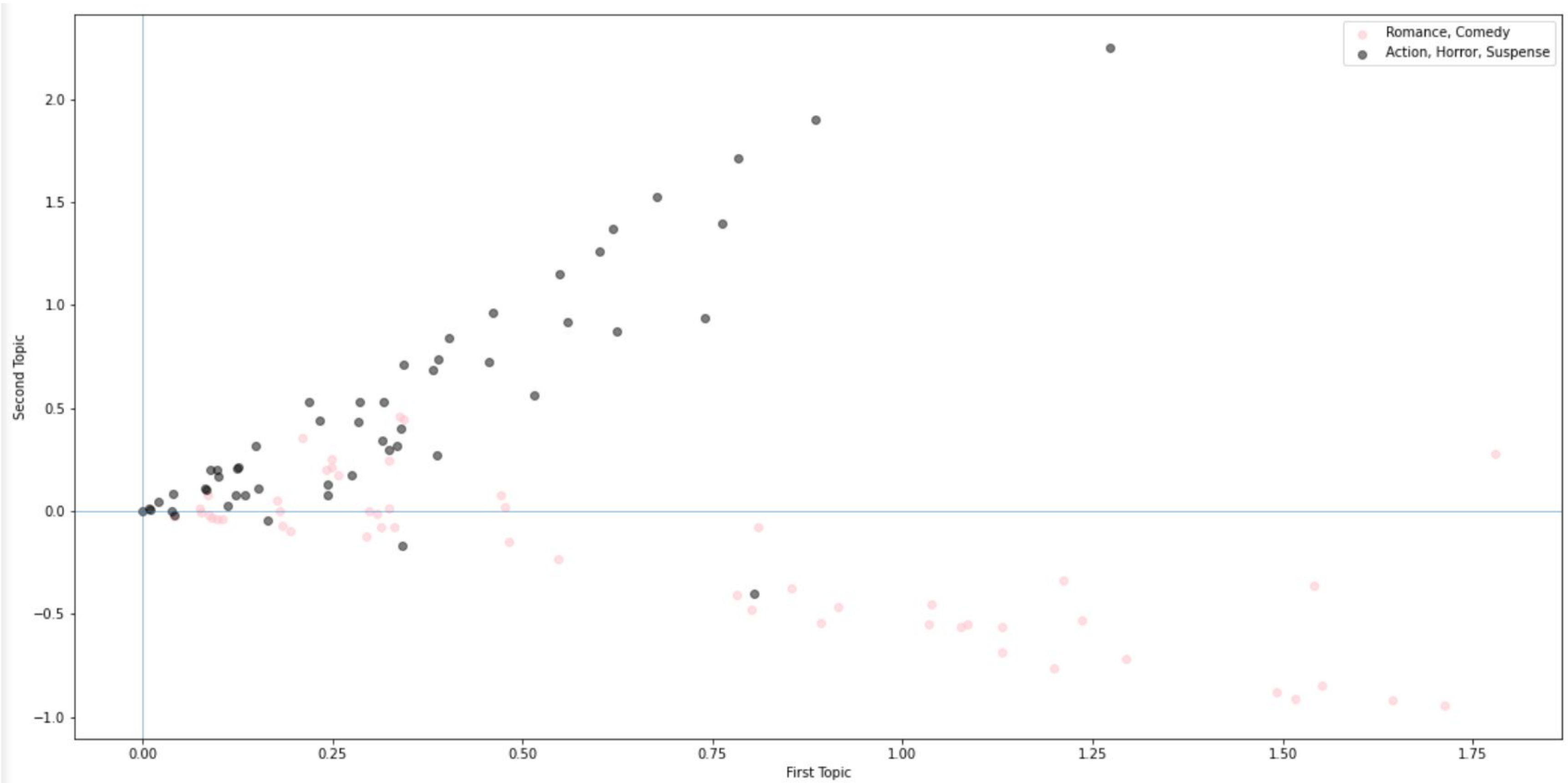
	topic_1	topic_2	review	genre	Is_Romance
0	1.03447	-0.55144	Blake Edwards' bawdy comedy may not score a pe...	Comedy, Romanc	1
1	1.03861	-0.45070	Matched by Garson Kanin's witty, sophisticated...	Classics, Comedy, Romanc	1
2	0.80111	-0.47994	he Baxter is good-natured, but there are simp...	Comedy, Romanc	1
3	1.64536	-0.92025	What Happens in Vegas has a few laughs, but mo...	Comedy, Romanc	1
4	0.17707	0.05142	Sandra Bullock and Ryan Reynolds exhibit plent...	Comedy, Romanc	1
...	...	...	...	.	...
101	0.78395	1.71309	Train to Busan delivers a thrillingly unique -...	Action & Adventure, Art House International,.	0
102	0.32414	0.30112	A visual and aural assault on the senses, this...	Action & Adventure, Horror, Myster & Suspense.	0
103	0.28342	0.43371	oefully deficient in thrills or common sense,...	Action & Adventure, Horror, Myster & Suspense.	0
104	0.00000	0.00000	Though Wolf Creek is effectively horrific, it ...	Action & Adventure, Horror, Myster & Suspens	0
105	0.16527	-0.04145	Young Sherlock Holmes is a charming, if unnece...	Action & Adventure, Drama, Horro Kids & Fami.	0

(CountVectorizer) LSA

	topic_1	topic_2
count	979.00000	979.00000
mean	0.01623	0.00685
std	0.02755	0.03123
min	0.00000	-0.32067
25%	0.00409	-0.00312
50%	0.00812	0.00205
75%	0.02260	0.01170
max	0.56403	0.37479

(CountVectorizer) LSA

Romance/Comedy vs. **Action/Horror**



(TF-IDF) LSA

1. **TfidfVectorizer()** to vectorize the words of reviews

```
tfidf_vec = TfidfVectorizer(min_df=1, stop_words='english')
bag_of_words = tfidf_vec.fit_transform(action_romcom.critics_consensus)
bag_of_words.todense()
```

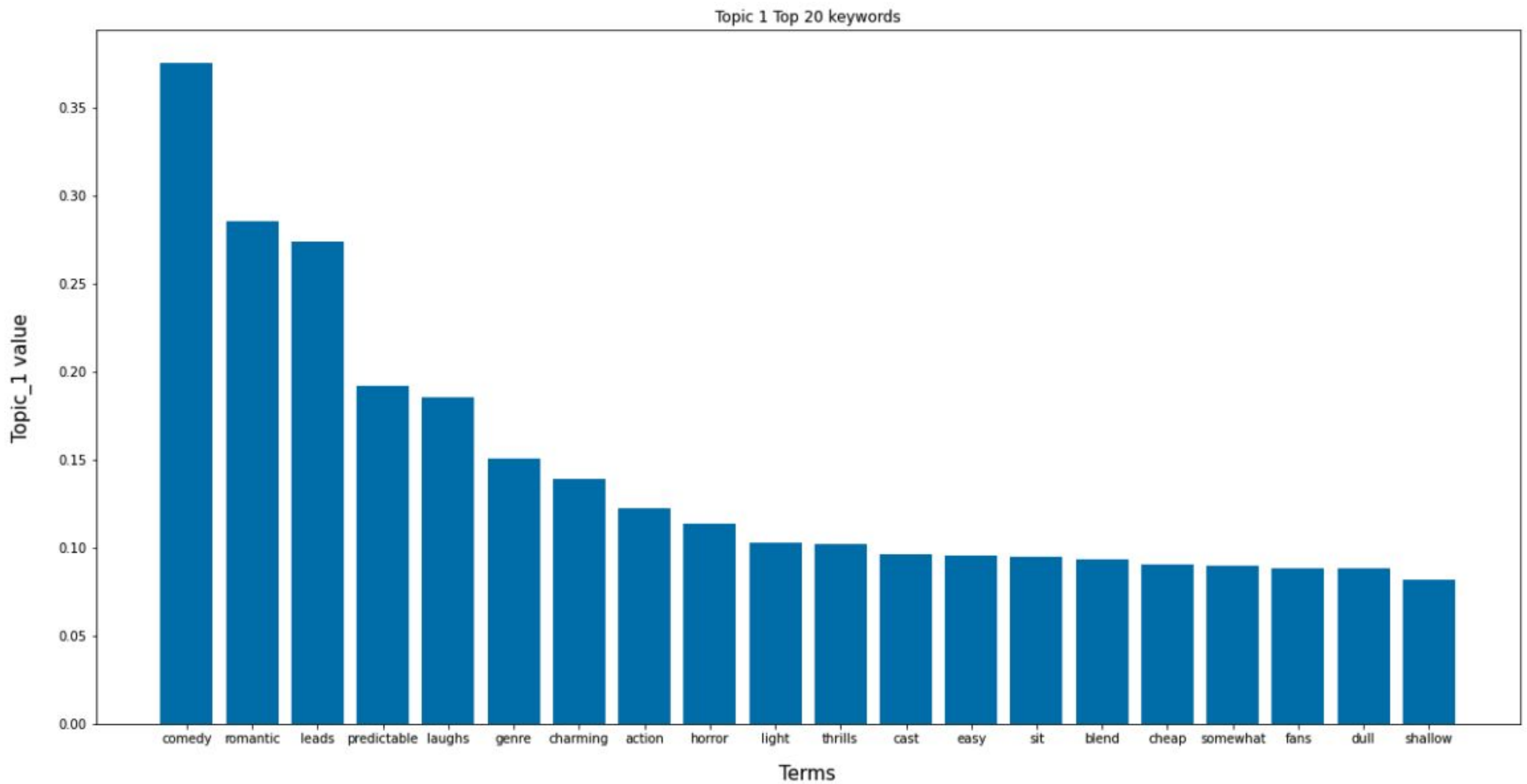
2. TruncatedSVD() to fit Bag of Words

```
svd = TruncatedSVD(n_components=2,n_iter=100)
lsa = svd.fit_transform(bag_of_words_count)
```

(TF-IDF) LSA - Distinct Terms

'**Topic 1 (Romantic Comedy)** Top 20 keywords'

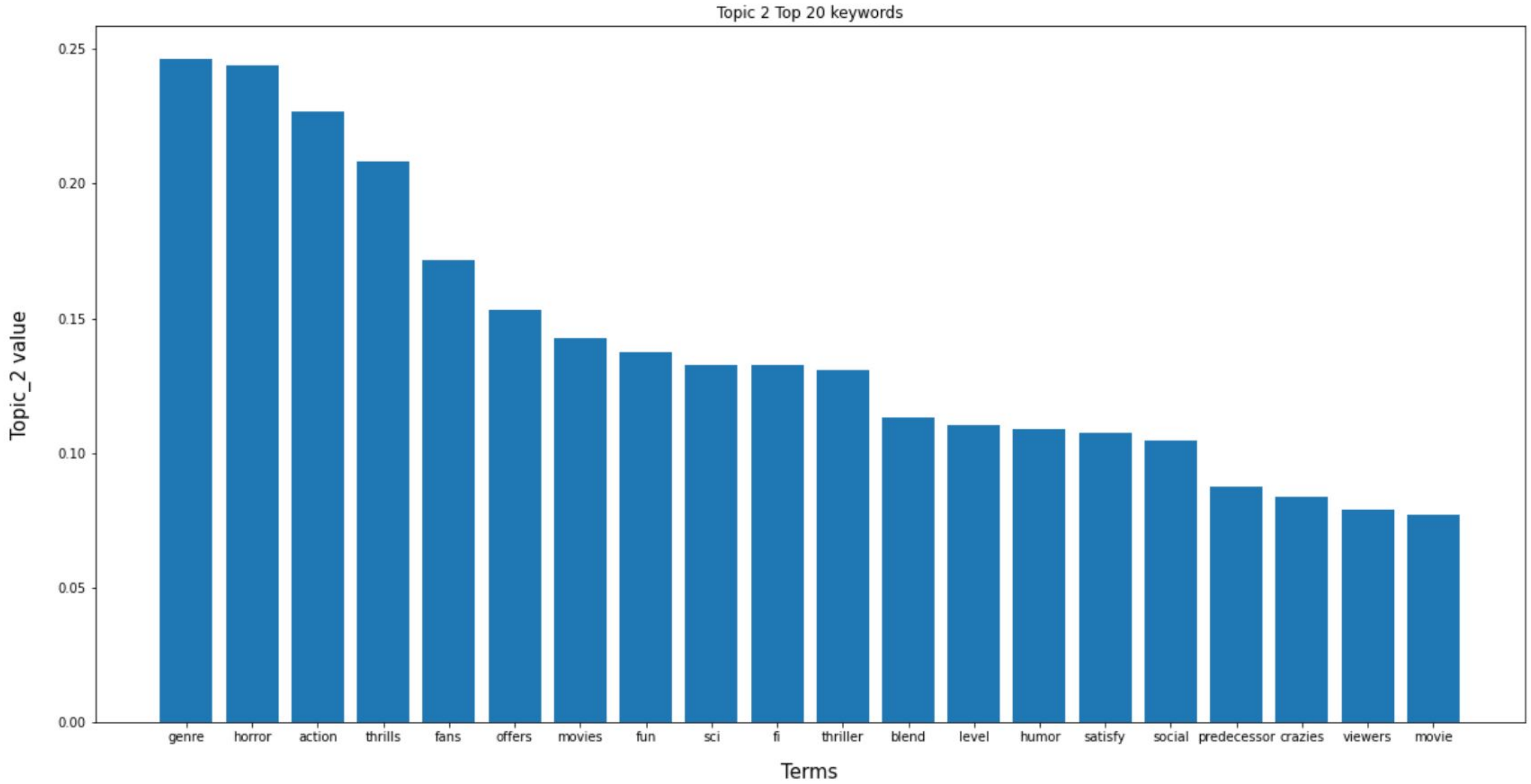
“comedy”, “romantic”, “predictable”, “laughs”, “charming”, “easy”, “shallow”



(TF-IDF) LSA - Distinct Terms

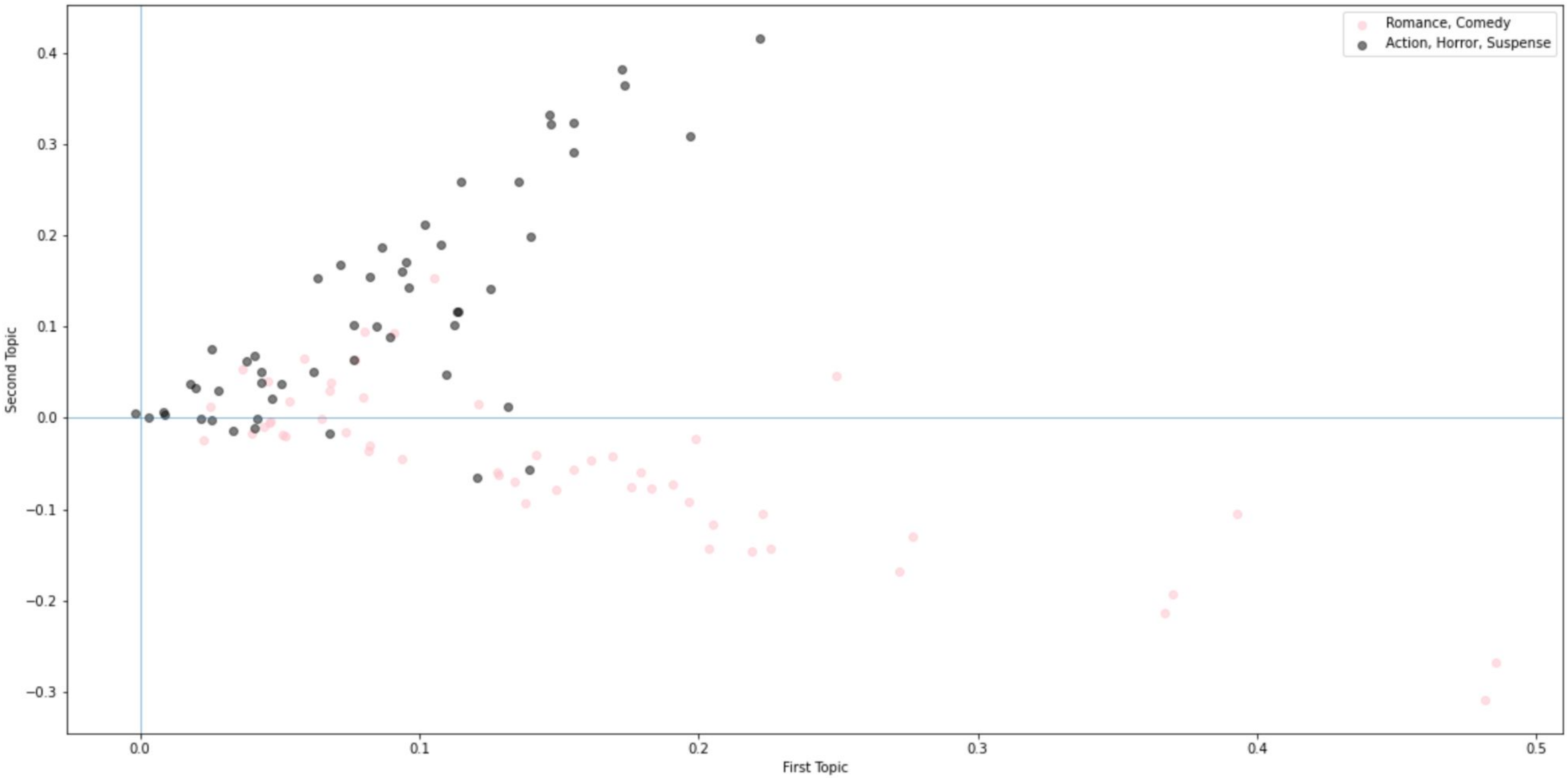
'**Topic 2 (Action/Horror)** Top 20 keywords'

"genre", "horror", "action", "thrills", "fans", "crazies"



(TF-IDF) LSA

Romance/Comedy vs. Action/Horror



(CountVectorizer) vs (TF-IDF) LSA

Classification Report

(CountVectorizer)

	precision	recall	f1-score	support
0	0.90	0.70	0.79	53
1	0.75	0.92	0.83	53
accuracy			0.81	106
macro avg	0.83	0.81	0.81	106
weighted avg	0.83	0.81	0.81	106

(TF-IDF)

	precision	recall	f1-score	support
False	0.91	0.57	0.70	53
True	0.68	0.94	0.79	53
accuracy			0.75	106
macro avg	0.80	0.75	0.75	106
weighted avg	0.80	0.75	0.75	106

SVM - 1 (without parameters)

```
#train data
clf = LinearSVC()
clf.fit(X_train, y_train)
```

```
LinearSVC()
```

```
#evaluation
y_pred = clf.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.76      0.81      0.79        16
           1       0.80      0.75      0.77        16

    accuracy                           0.78        32
   macro avg       0.78      0.78      0.78        32
weighted avg       0.78      0.78      0.78        32
```

GridSearchCV
(for parameter tuning)

```
def search_params(train_review, train_in_romance):
    #pipeline
    text_clf = Pipeline([("tfidf", TfidfVectorizer()),
                        ("clf", svm.LinearSVC())])
    #ranges of parameters for GridSearchCV
    parameters = {"tfidf__use_idf":[True, False],
                 "tfidf__ngram_range":[(1,1),(1,2),(1,3)],   #n_grams,
                 "tfidf__stop_words":[None, "english"],   #exclude unnecessary words
                  "tfidf__min_df":[1, 3, 5],  #minimum frequency in documents
                  "clf__C":[0.1, 0.5, 1, 2]}   #lower C param : softer margin, higher C param : harder margin
    metric = "f1_macro"
    gs_clf = GridSearchCV(text_clf, param_grid=parameters, refit=True,
                          scoring=metric, cv=10)
    gs_clf.fit(train_review, train_in_romance)
    best_param = gs_clf.best_params_
    best_score = gs_clf.best_score_
    return [best_param, best_score]
```

```
clf__C: 2
tfidf__min_df:  1
tfidf__ngram_range:     (1, 1)
tfidf__stop_words:      english
tfidf__use_idf: True
best f1-score: 86.381%
```

SVM - 2
(with parameters based on GridSearchcv)

```
def svm_params(dtm_options, svm_options):
  tfidf_vect = TfidfVectorizer(stop_words=dtm_options["stop_words"],
                               min_df=dtm_options["min_df"],
                               use_idf=dtm_options["use_idf"],
                               ngram_range=dtm_options["ngram_range"])
  dtm = tfidf_vect.fit_transform(svm_train["review"])

  # data split
  X_train = dtm
  y_train = svm_train["Is_Romance"]
  X_test = tfidf_vect.transform(svm_test["review"])
  y_test = svm_test["Is_Romance"]

  # train data
  clf = svm.LinearSVC(C=svm_options["C"])
  clf.fit(X_train, y_train)

  # prediction result
  y_pred = clf.predict(X_test)
  report = classification_report(y_test, y_pred)
  return report
```

	precision	recall	f1-score	support
0	0.93	0.81	0.87	16
1	0.75	0.90	0.82	10
accuracy			0.85	26
macro avg	0.84	0.86	0.84	26
weighted avg	0.86	0.85	0.85	26

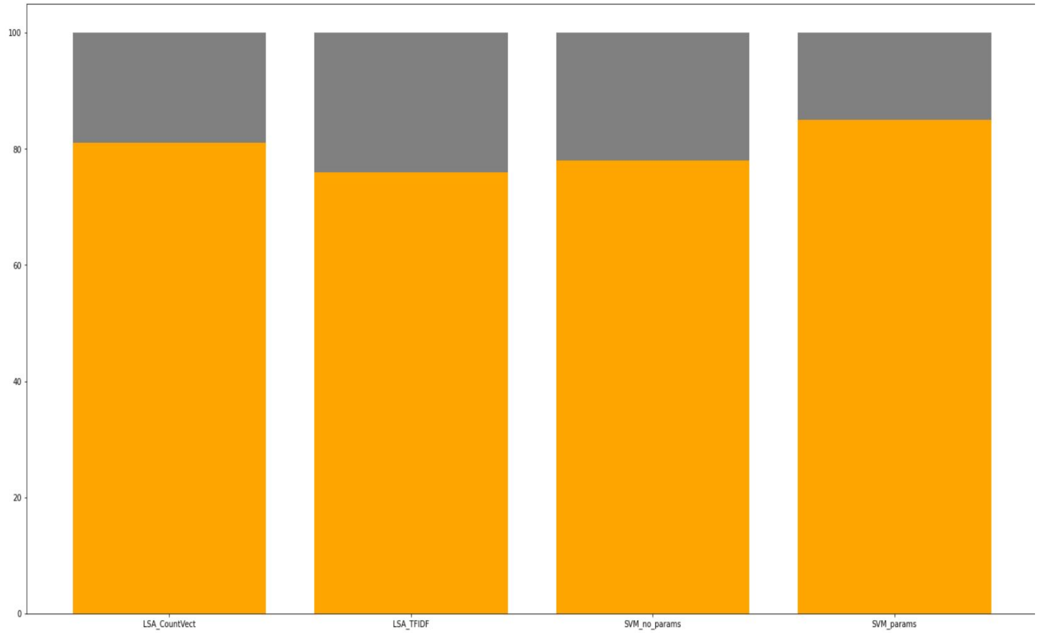
Conclusion

F-1 score of CountVectorizer LSA: 81%

F-1 score of TFIDF LSA: 75%

F-1 score of SVM w/o Parameter (F1 score): 78%

F-1 score of SVM w/ parameters (F1 score): 85%



Limitations and Future Work

Limitations

Lack of Valid measurements in Evaluations of LSA

- Inconsistent classification since they weigh more in the difference of two values than each value itself

TF-IDF producing a lower F1-score than CountVectorizer

- requires more research to reveal a hidden logic

Future Work

- Analyzing more variations of genres, not confined to 2 distinct genres
- Apply these methods for another types of data set
  - restaurant reviews
  - book reviews
- Improve the accuracy of current work by trying various classifications method

	LSA		SVM	
	CounterVectorizer	TF-IDF	Without Parameters	With Parameters
f1-score	0.83	0.79	0.77	0.82
accuracy	0.81	0.75	0.78	0.85

Implementation

Check out our code!

https://colab.research.google.com/drive/1vIebBf_mgPH9yjfRi2Oweip_9yTsAqjf?authuser=2#scrollTo=oYsCHxxIlz82

Thank you