In [99]:
```python
# POEtryimport pandas as pd
import numpy as np
import os
import pandas as pd
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer

import re

import string
from nltk.corpus import stopwords
```

In [103]:
```python
print(stopwords)
```

```
<WordListCorpusReader in '.../corpora/stopwords' (not loaded yet)>
```

In [37]:
```python
DATA = pd.read_csv("all.csv")
tDATA = DATA["type"]
```

DATA

	author	content	poem name	age	type
0	WILLIAM SHAKESPEARE	Let the bird of loudest lay\r\nOn the sole Ara...	The Phoenix and the Turtle	Renaissance	Mythology & Folklore
1	DUCHESS OF NEWCASTLE MARGARET CAVENDISH	Sir Charles into my chamber coming in,\r\nWhen...	An Epilogue to the Above	Renaissance	Mythology & Folklore
2	THOMAS BASTARD	Our vice runs beyond all that old men saw,\r\n...	Book 7, Epigram 42	Renaissance	Mythology & Folklore
3	EDMUND SPENSER	Lo I the man, whose Muse whilome did maske,\r\...	from The Faerie Queene: Book I, Canto I	Renaissance	Mythology & Folklore
4	RICHARD BARNFIELD	Long have I longd to see my love againe,\r\nSt...	Sonnet 16	Renaissance	Mythology & Folklore
5	RICHARD BARNFIELD	Cherry-lipt Adonis in his snowie shape,\r\n ...	Sonnet 17	Renaissance	Mythology & Folklore
6	SIR WALTER RALEGH	Praisd be Dianas fair and harmless light;\r\nP...	Praisd be Dianas Fair and Harmless Light	Renaissance	Mythology & Folklore
7	QUEEN ELIZABETH I	When I was fair and young, then favor graced m...	When I Was Fair and Young	Renaissance	Mythology & Folklore
8	JOHN DONNE	When by thy scorn, O murd'ress, I am dead\r\n ...	The Apparition	Renaissance	Mythology & Folklore
9	JOHN SKELTON	Pla ce bo,\r\nWho is there, who?\r\nDi le xi,\...	The Book of Phillip Sparrow	Renaissance	Mythology & Folklore
10	EDMUND SPENSER	Ye learned sisters which have oftentimes\r\nBe...	Epithalamion	Renaissance	Mythology & Folklore
11	CHRISTOPHER MARLOWE	On Hellespont, guilty of true love's blood,\r\...	Hero and Leander	Renaissance	Mythology & Folklore
12	EDMUND SPENSER	By that he ended had his ghostly sermon,\r\nTh...	Prosopopoia: or Mother Hubbard's Tale	Renaissance	Mythology & Folklore
13	EDMUND SPENSER	CALM was the day, and through the trembling ai...	Prothalamion	Renaissance	Mythology & Folklore
14	EDMUND SPENSER	THENOT & HOBBINOLL\r\nTell me good Hob...	from The Shepheardes Calender: April	Renaissance	Mythology & Folklore
15	EDMUND SPENSER	PIERCE & CUDDIE\r\nCuddie, for s...	from The Shepheardes Calender: October	Renaissance	Mythology & Folklore

	author	content	poem name	age	type
16	JOHN DONNE	Go and catch a falling star,\r\n Get with c...	Song: Go and catch a falling star	Renaissance	Mythology & Folklore
17	WILLIAM SHAKESPEARE	Orpheus with his lute made trees, \r\nAnd the ...	Song: Orpheus with his lute made trees	Renaissance	Mythology & Folklore
18	WILLIAM SHAKESPEARE	What is your substance, whereof are you made,\...	Sonnet 53: What is your substance, whereof are...	Renaissance	Mythology & Folklore
19	WILLIAM SHAKESPEARE	Why didst thou promise such a beauteous day,\r...	Sonnet 34: Why didst thou promise such a beaut...	Renaissance	Nature
20	THOMAS BASTARD	The welcome Sun from sea Freake is returned,\r...	Book 1, Epigram 34: Ad. Thomam Freake armig. d...	Renaissance	Nature
21	THOMAS BASTARD	I met a courtier riding on the plain,\r\nWell-...	Book 2, Epigram 22	Renaissance	Nature
22	THOMAS BASTARD	Walking the fields a wantcatcher I spied,\r\nT...	Book 2, Epigram 8	Renaissance	Nature
23	THOMAS BASTARD	Fishing, if I a fisher may protest,\r\nOf plea...	Book 6, Epigram 14: De Piscatione.	Renaissance	Nature
24	LADY MARY WROTH	Come darkest night, becoming sorrow best;\r\n ...	from Pamphilia to Amphilanthus: 19	Renaissance	Nature
25	EDMUND SPENSER	Januarie. gloga prima. ARGVMENT.\r\n\r\n IN...	The Shepheardes Calender: January	Renaissance	Nature
26	WILLIAM SHAKESPEARE	Where the bee sucks, there suck I:\r\nIn a cow...	Song: Where the bee sucks, there suck I	Renaissance	Nature
27	JOHN DONNE	Tis true, tis day, what though it be?\r\nO wil...	Break of Day	Renaissance	Nature
28	ROBERT SOUTHWELL, SJ	As I in hoary winters night stood shivering in...	The Burning Babe	Renaissance	Nature
29	WILLIAM BYRD	Care for thy soul as thing of greatest price,\...	Care for Thy Soul as Thing of Greatest Price	Renaissance	Nature
...	...	...	...	...	...
543	MARJORIE PICKTHALL	When the first dark had fallen around them\r\n...	Adam and Eve	Modern	Love
544	D. H. LAWRENCE	My love looks like a girl to-night,\r\n B...	The Bride	Modern	Love

	author	content	poem name	age	type
545	WILLIAM BUTLER YEATS	The jester walked in the garden:\r\nThe garden...	The Cap and Bells	Modern	Love
546	D. H. LAWRENCE	What large, dark hands are those at the window...	Cruelty and Love	Modern	Love
547	SARA TEASDALE	Supper comes at five o'clock,\r\nAt six, the e...	Eight O'Clock	Modern	Love
548	EZRA POUND	Go, dumb-born book,\r\nTell her that sang me o...	Envoi	Modern	Love
549	SARA TEASDALE	They came to tell your faults to me,\r\nThey n...	Faults	Modern	Love
550	LOUIS UNTERMEYER	I never knew the earth had so much gold\r\nThe...	Feuerzauber	Modern	Love
551	D. H. LAWRENCE	When she rises in the morning\r\nI linger to w...	Gloire de Dijon	Modern	Love
552	LOUIS UNTERMEYER	Louis Untermeyer, Infidelity from The New Poet...	Infidelity	Modern	Love
553	D. H. LAWRENCE	Version 1 (1921)\r\nYours is the shame and sor...	Last Words to Miriam	Modern	Love
554	SARA TEASDALE	Strephon kissed me in the spring,\r\nRobin in ...	The Look	Modern	Love
555	T. S. ELIOT	Let us go then, you and I,\r\nWhen the evening...	The Love Song of J. Alfred Prufrock	Modern	Love
556	EDGAR LEE MASTERS	I went to the dances at Chandlerville,\r\nAnd ...	Lucinda Matlock	Modern	Love
557	PAUL LAURENCE DUNBAR	Seen my lady home las' night,\r\nJump back, ho...	A Negro Love Song	Modern	Love
558	PAUL LAURENCE DUNBAR	W'en daih's chillun in de house,\r\nDey keep o...	The Old Front Gate	Modern	Love
559	SARA TEASDALE	I saw her in a Broadway car,\r\nThe woman I mi...	The Old Maid	Modern	Love
560	WALLACE STEVENS	I\r\...	Peter Quince at the Clavier	Modern	Love
561	T. S. ELIOT	I\r\nAmong the smoke and fog of a December aft...	Portrait of a Lady	Modern	Love

	author	content	poem name	age	type
562	EDGAR LEE MASTERS	Maurice, weep not, I am not here under this pi...	Sarah Brown	Modern	Love
563	MARJORIE PICKTHALL	I shall not go with pain\r\nWhether you hold m...	Song	Modern	Love
564	PAUL LAURENCE DUNBAR	Wintah, summah, snow er shine,\r\nHit's all de...	Song (Wintah, summah, snow er shine)	Modern	Love
565	LOUISE BOGAN	This youth too long has heard the break\r\nOf ...	A Tale	Modern	Love
566	D. H. LAWRENCE	Making his advances\r\nHe does not look at her...	Tortoise Gallantry	Modern	Love
567	D. H. LAWRENCE	I thought he was dumb,\r\nI said he was dumb,\...	Tortoise Shout	Modern	Love
568	SARA TEASDALE	With the man I love who loves me not,\r\nI wal...	Union Square	Modern	Love
569	HART CRANE	Hart Crane, "Voyages I, II, III, IV, V, VI" fr...	Voyages	Modern	Love
570	WILLIAM BUTLER YEATS	When you are old and grey and full of sleep,\r...	When You Are Old	Modern	Love
571	CARL SANDBURG	Give me hunger,\r\nO you gods that sit and giv...	At a Window	Modern	Love
572	RICHARD ALDINGTON	Potuia, potuia\r\nWhite grave goddess,\r\nPity...	To a Greek Marble	Modern	Love

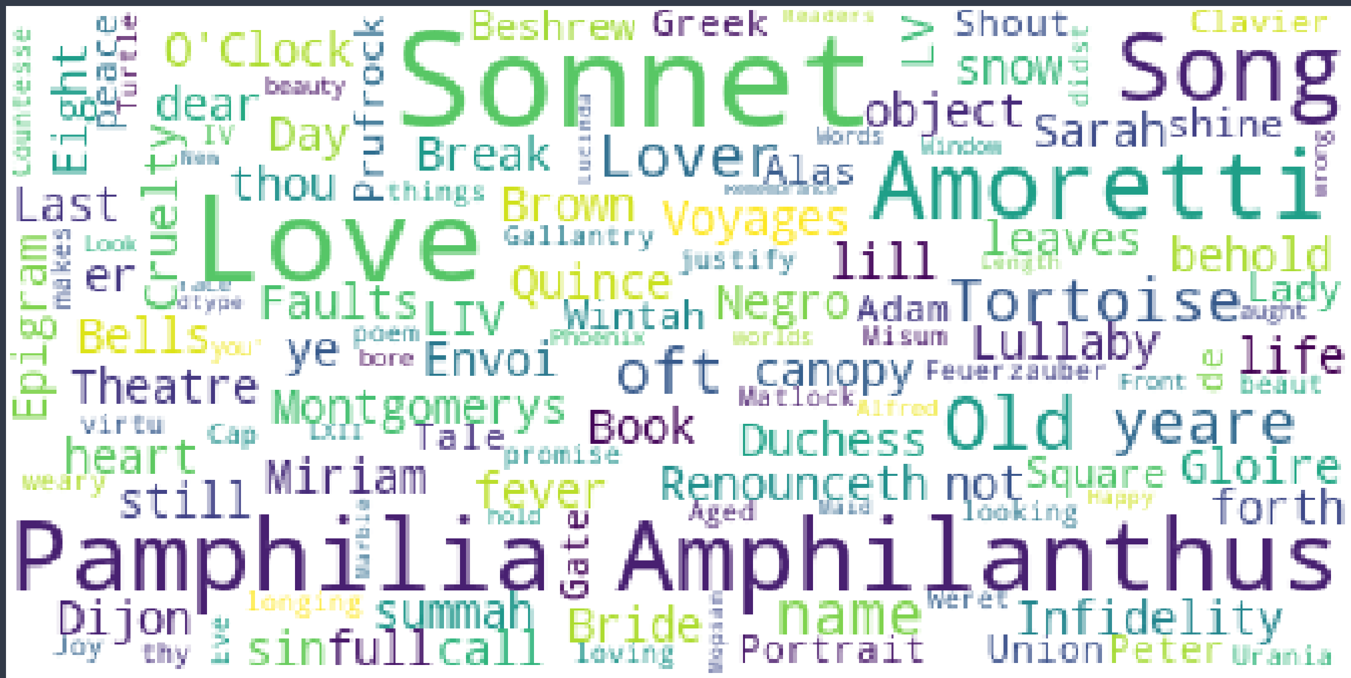
573 rows × 5 columns

Exploratory data analysis and visualization via Dimenionality Reduction

In [109]:

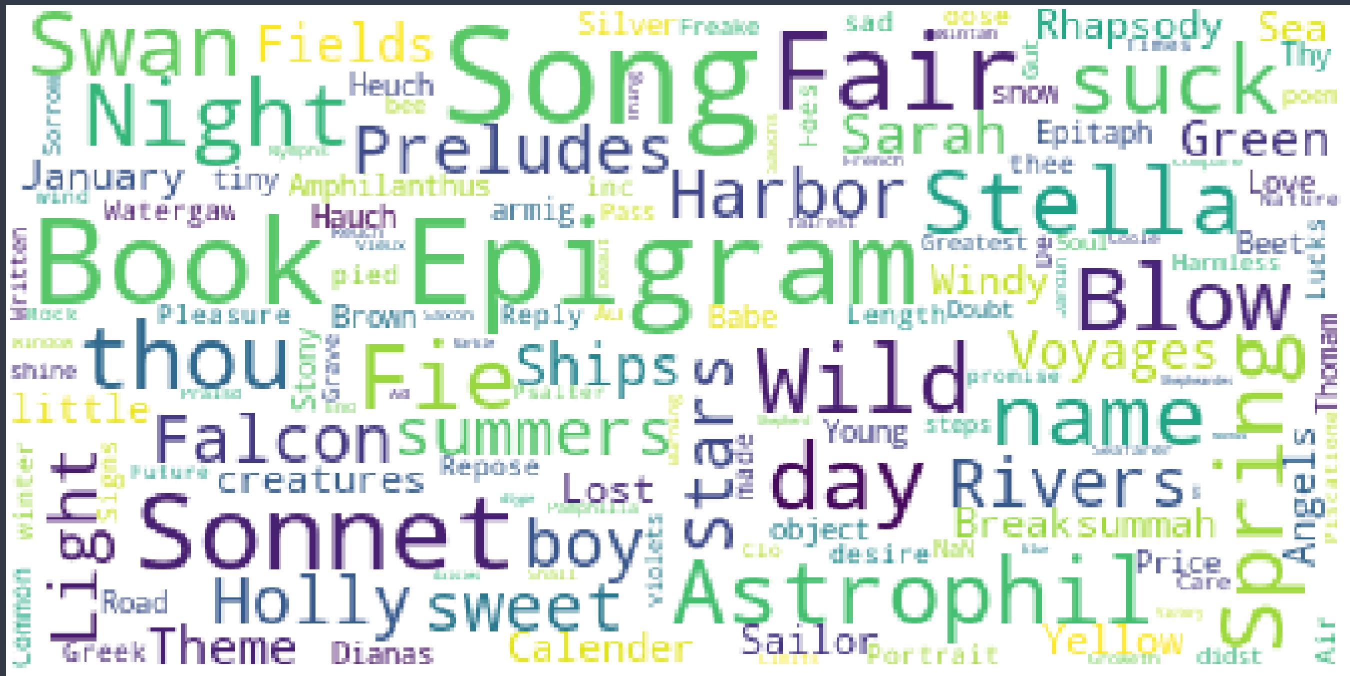
```
from wordcloud import WordCloud
wordcloud = WordCloud(
                      background_color='white',
```

```
                          stopwords=stopwords,
                          max_words=200,
                          max_font_size=40,
                          random_state=42
                         ).generate(str(DATA[DATA['type']=='Love']['poem name']))

fig = plt.figure(1,figsize=(12,18))
plt.imshow(wordcloud)
plt.axis('off')
plt.show()
```

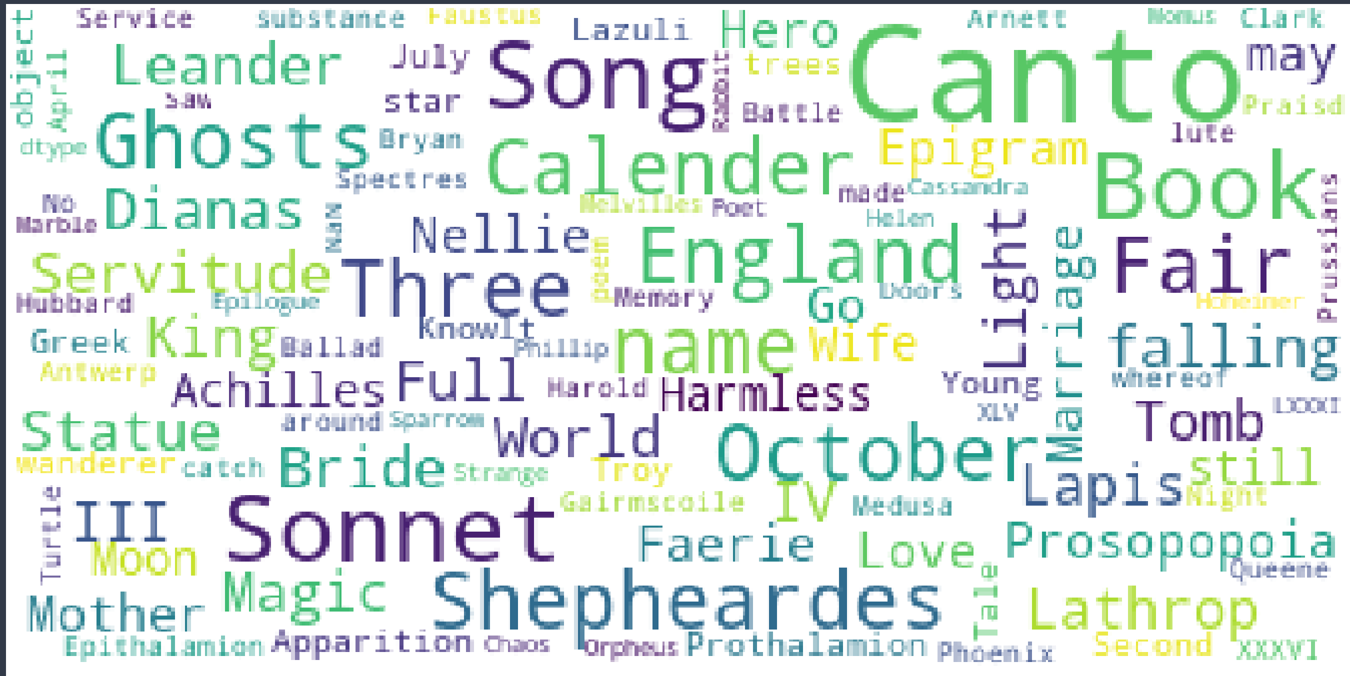



```
In [110]: wordcloud = WordCloud(
```

```
                          background_color='white',
                          stopwords=stopwords,
                          max_words=200,
                          max_font_size=40,
                          random_state=42
                         ).generate(str(DATA[DATA['type']=='Nature']['poem name']))

fig = plt.figure(1,figsize=(12,18))
plt.imshow(wordcloud)
plt.axis('off')
plt.show()
```



```python
In [111]: wordcloud = WordCloud(
                              background_color='white',
                              stopwords=stopwords,
                              max_words=200,
                              max_font_size=40,
                              random_state=42
                             ).generate(str(DATA[DATA['type']=='Mythology & Folklore']['poem name']))

          fig = plt.figure(1,figsize=(12,18))
          plt.imshow(wordcloud)
          plt.axis('off')
          plt.show()
```



In [113]:

```python
print(DATA['type'].value_counts())
print(DATA['author'].value_counts())
```

```
Love                      326
Nature                    188
Mythology & Folklore       59
Name: type, dtype: int64
WILLIAM SHAKESPEARE                                    71
SIR PHILIP SIDNEY                                      42
JOHN DONNE                                             41
EDMUND SPENSER                                         34
WILLIAM BUTLER YEATS                                   26
SIR THOMAS WYATT                                       22
CARL SANDBURG                                          16
EZRA POUND                                             16
```

```
THOMAS CAMPION                                        15
HART CRANE                                            14
D. H. LAWRENCE                                        14
SARA TEASDALE                                         14
WALLACE STEVENS                                       14
EN JONSON                                             13
PAUL LAURENCE DUNBAR                                  12
IVOR GURNEY                                           11
LOUISE BOGAN                                          11
MICHAEL ANANIA                                        10
EDGAR LEE MASTERS                                     10
SIR WALTER RALEGH                                      9
LADY MARY WROTH                                        8
HUGH MACDIARMID                                        7
MARJORIE PICKTHALL                                     7
SAMUEL DANIEL                                          7
ARCHIBALD MACLEISH                                     7
ELINOR WYLIE                                           7
LOUIS UNTERMEYER                                       6
CHRISTOPHER MARLOWE                                    6
QUEEN ELIZABETH I                                      6
RICHARD ALDINGTON                                      6
                                                      ..
JOHN SKELTON                                           4
GEORGE GASCOIGNE                                       4
CONRAD AIKEN                                           3
GUILLAUME APOLLINAIRE                                  3
HENRY HOWARD, EARL OF SURREY                           3
SAMUEL GREENBERG                                       3
ASIL BUNTING                                           3
MARIANNE MOORE                                         2
STEPHEN SPENDER                                        2
MINA LOY                                               2
THOMAS NASHE                                           2
GERTRUDE STEIN                                         2
GEORGE CHAPMAN                                         2
KENNETH FEARING                                        2
DUCHESS OF NEWCASTLE MARGARET CAVENDISH                2
JAMES JOYCE                                            2
```

```
EDITH SITWELL                                      2
SECOND BARON VAUX OF HARROWDEN THOMAS, LORD VAUX   1
WILLIAM BYRD                                       1
FORD MADOX FORD                                    1
ISABELLA WHITNEY                                   1
MALCOLM COWLEY                                     1
GIOVANNI BATTISTA GUARINI                          1
THOMAS HEYWOOD                                     1
SIR EDWARD DYER                                    1
ROBERT SOUTHWELL, SJ                               1
KATHERINE MANSFIELD                                1
THOMAS LODGE                                       1
JOHN FLETCHER                                      1
ORLANDO GIBBONS                                    1
Name: author, Length: 67, dtype: int64
```

Observation

Here we can see that our data is imbalanced with more poems in love and very less in Mythological and folklore , here we can also see most of the poems are of William Shakespear

TSNE plots

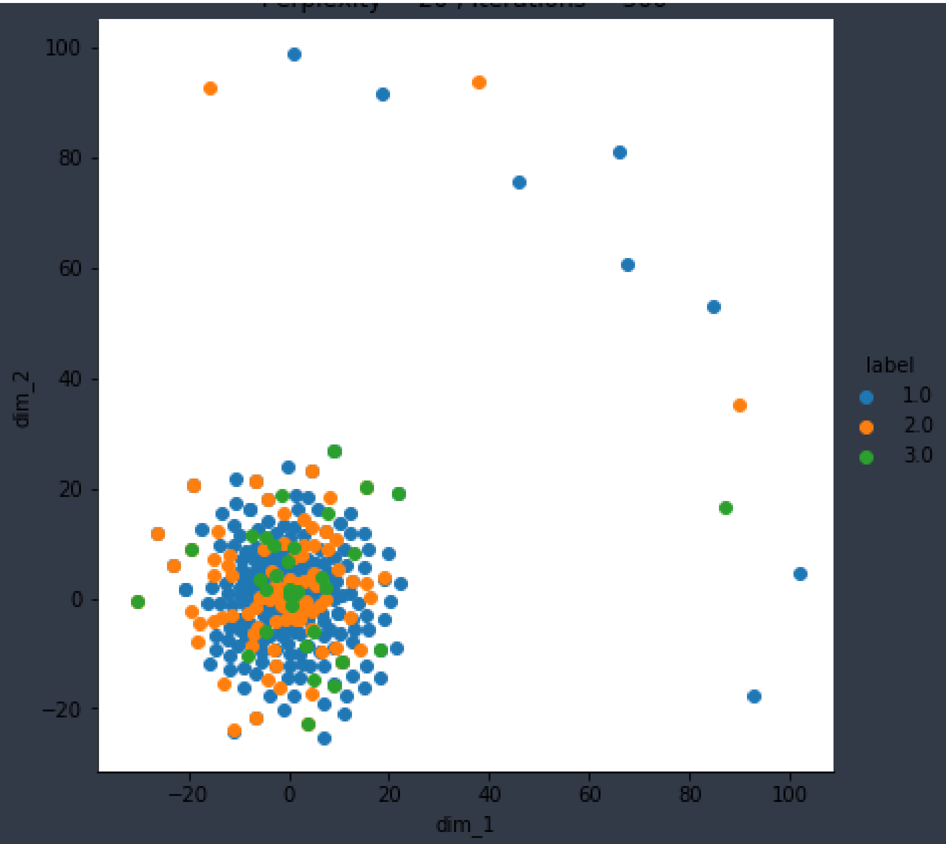
For BOW

In [155]:
```
from sklearn.manifold import TSNE
from sklearn.preprocessing import StandardScaler

model = TSNE(n_components=2,perplexity=20,n_iter=500,random_state=0)
# converting sparse to a dense array
final_count= BOW_Train.toarray()
# standardizing our data making mean=0 and std_dev=1
```

```
standardize = StandardScaler().fit_transform(final_count)
BOW_data = model.fit_transform(standardize)
#creating a new data which help us in plotting the result data
BOW_data = np.vstack((BOW_data.T,y_tr)).T
BOW_df = pd.DataFrame(data = BOW_data,columns=("dim_1","dim_2","label"))
# Plotting the result of tsne
sns.FacetGrid(BOW_df,hue="label",size=6).map(plt.scatter,"dim_1","dim_2").add_legend()
plt.title('Perplexity = 20 , iterations = 500')
plt.show()
```

```
/usr/local/lib/python3.5/dist-packages/sklearn/utils/validation.py:475: DataConversionWarning: Data with input dtype int64 w
as converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
/usr/local/lib/python3.5/dist-packages/sklearn/utils/validation.py:475: DataConversionWarning: Data with input dtype int64 w
as converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
/usr/local/lib/python3.5/dist-packages/seaborn/axisgrid.py:230: UserWarning: The `size` paramter has been renamed to `height
`; please update your code.
  warnings.warn(msg, UserWarning)
```

Perplexity = 20 , iterations = 500

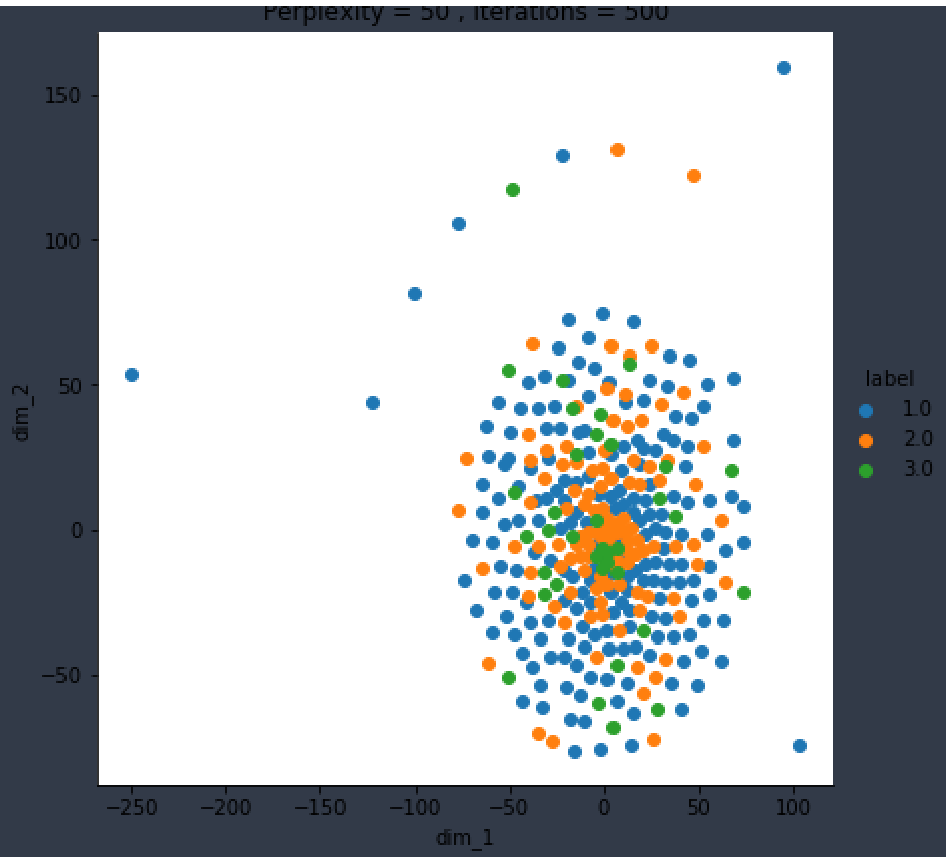


In [160]:

```
from sklearn.preprocessing import StandardScaler

model = TSNE(n_components=2,perplexity=50,n_iter=500,random_state=0)
# converting sparse to a dense array
final_count= BOW_Train.toarray()
# standardizing our data making mean=0 and std_dev=1
standardize = StandardScaler().fit_transform(final_count)
BOW_data = model.fit_transform(standardize)
#creating a new data which help us in plotting the result data
BOW_data = np.vstack((BOW_data.T,y_tr)).T
```

```python
BOW_df = pd.DataFrame(data = BOW_data,columns=("dim_1","dim_2","label"))
# Plotting the result of tsne
sns.FacetGrid(BOW_df,hue="label",size=6).map(plt.scatter,"dim_1","dim_2").add_legend()
plt.title('Perplexity = 50 , iterations = 500')
plt.show()
```

```
/usr/local/lib/python3.5/dist-packages/sklearn/utils/validation.py:475: DataConversionWarning: Data with input dtype int64 w
as converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
/usr/local/lib/python3.5/dist-packages/sklearn/utils/validation.py:475: DataConversionWarning: Data with input dtype int64 w
as converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
/usr/local/lib/python3.5/dist-packages/seaborn/axisgrid.py:230: UserWarning: The `size` paramter has been renamed to `height
`; please update your code.
  warnings.warn(msg, UserWarning)
```

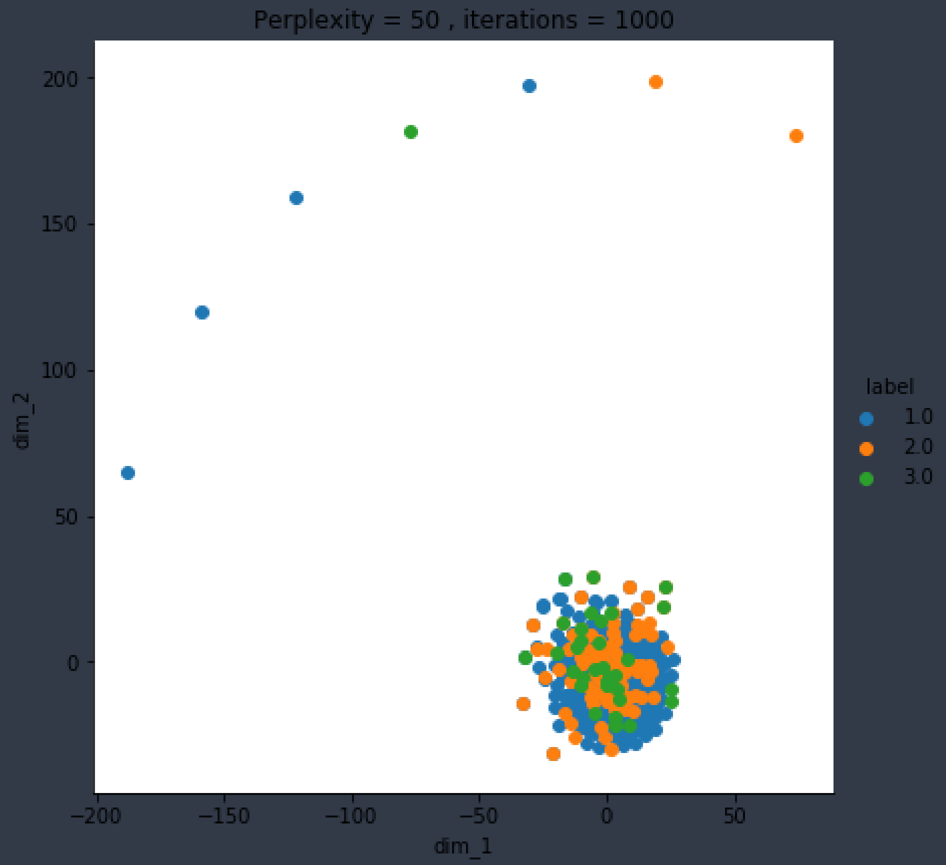


In [161]:

```
from sklearn.preprocessing import StandardScaler

model = TSNE(n_components=2,perplexity=50,n_iter=1000,random_state=0)
# converting sparse to a dense array
final_count= BOW_Train.toarray()
# standardizing our data making mean=0 and std_dev=1
standardize = StandardScaler().fit_transform(final_count)
BOW_data = model.fit_transform(standardize)
#creating a new data which help us in plotting the result data
```

```python
BOW_data = np.vstack((BOW_data.T,y_tr)).T
BOW_df = pd.DataFrame(data = BOW_data,columns=("dim_1","dim_2","label"))
# Plotting the result of tsne
sns.FacetGrid(BOW_df,hue="label",size=6).map(plt.scatter,"dim_1","dim_2").add_legend()
plt.title('Perplexity = 50 , iterations = 1000')
plt.show()
```

```
/usr/local/lib/python3.5/dist-packages/sklearn/utils/validation.py:475: DataConversionWarning: Data with input dtype int64 w
as converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
/usr/local/lib/python3.5/dist-packages/sklearn/utils/validation.py:475: DataConversionWarning: Data with input dtype int64 w
as converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
/usr/local/lib/python3.5/dist-packages/seaborn/axisgrid.py:230: UserWarning: The `size` paramter has been renamed to `height
`; please update your code.
  warnings.warn(msg, UserWarning)
```



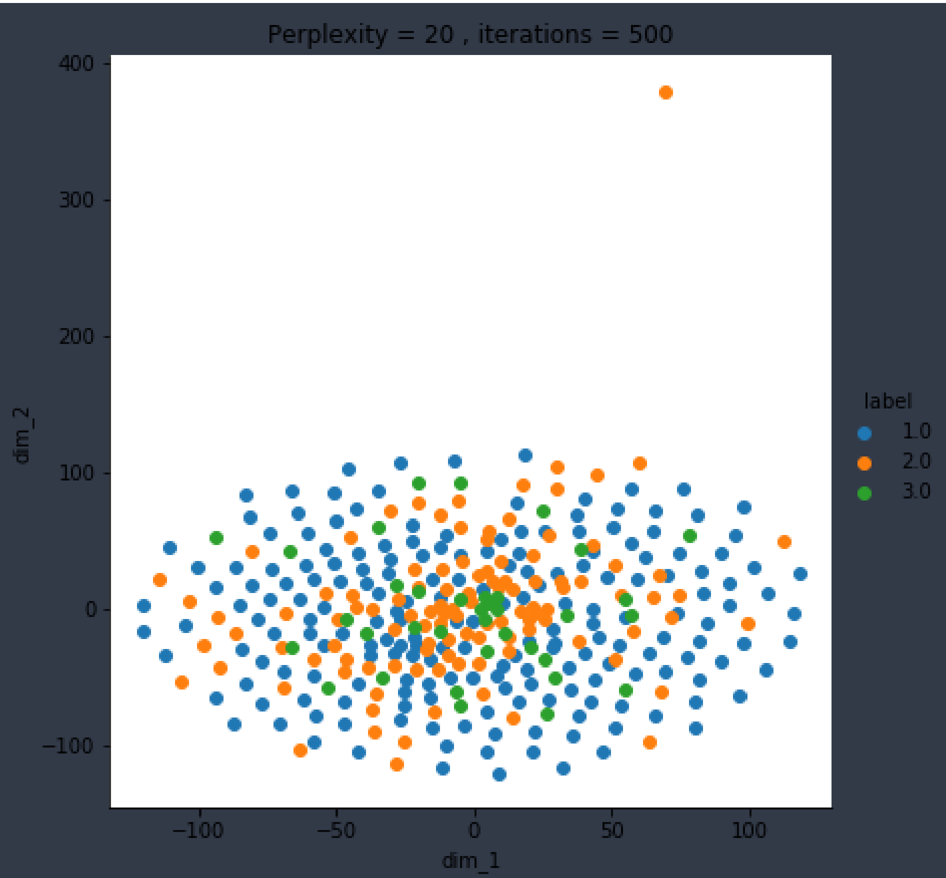
For TFIDF

In [164]:

```
from sklearn.manifold import TSNE
from sklearn.preprocessing import StandardScaler

model = TSNE(n_components=2,perplexity=20,n_iter=500,random_state=0)
# converting sparse to a dense array
final_count= TFIDF_Train.toarray()
```

```python
# standardizing our data making mean=0 and std_dev=1
standardize = StandardScaler().fit_transform(final_count)
tfidf_data = model.fit_transform(standardize)
#creating a new data which help us in plotting the result data
tfidf_data = np.vstack((tfidf_data.T,y_tr)).T
tfidf_df = pd.DataFrame(data = tfidf_data,columns=("dim_1","dim_2","label"))
# Plotting the result of tsne
sns.FacetGrid(tfidf_df,hue="label",size=6).map(plt.scatter,"dim_1","dim_2").add_legend()
plt.title('Perplexity = 20 , iterations = 500')
plt.show()
```

```
/usr/local/lib/python3.5/dist-packages/seaborn/axisgrid.py:230: UserWarning: The `size` paramter has been renamed to `height
`; please update your code.
  warnings.warn(msg, UserWarning)
```

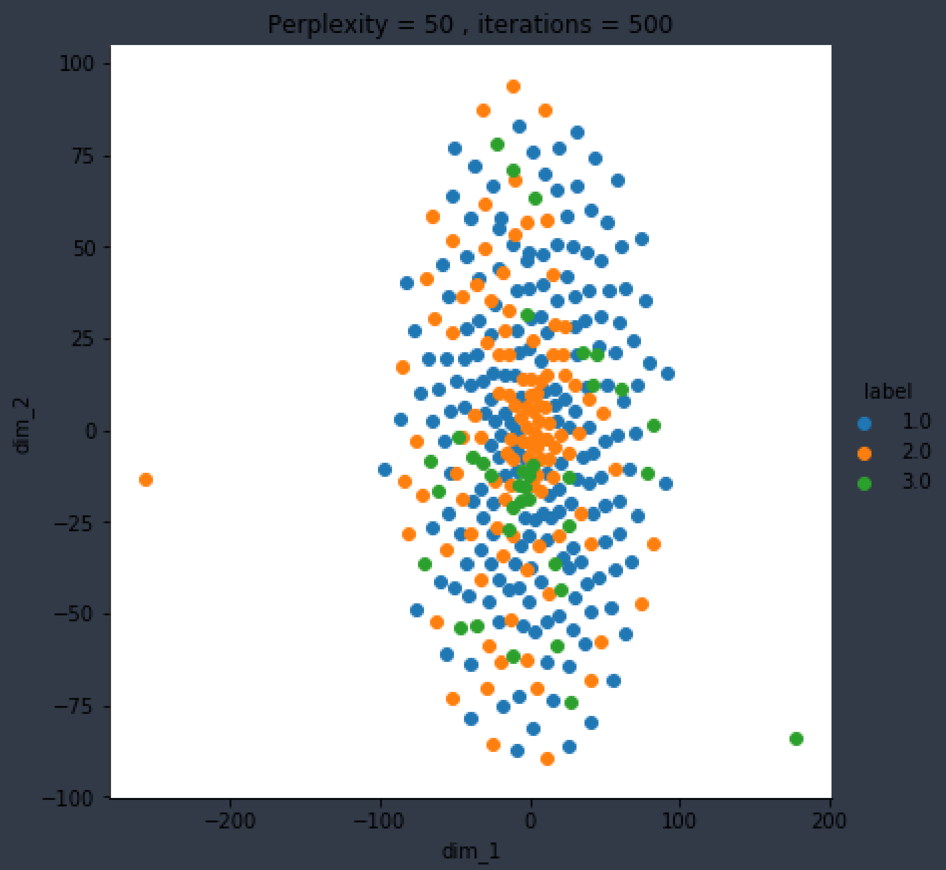


In [165]:

```
from sklearn.preprocessing import StandardScaler

model = TSNE(n_components=2,perplexity=50,n_iter=500,random_state=0)
# converting sparse to a dense array
final_count= TFIDF_Train.toarray()
# standardizing our data making mean=0 and std_dev=1
standardize = StandardScaler().fit_transform(final_count)
tfidf_data = model.fit_transform(standardize)
#creating a new data which help us in plotting the result data
```

```
tfidf_data = np.vstack((tfidf_data.T,y_tr)).T
tfidf_df = pd.DataFrame(data = tfidf_data,columns=("dim_1","dim_2","label"))
# Plotting the result of tsne
sns.FacetGrid(tfidf_df,hue="label",size=6).map(plt.scatter,"dim_1","dim_2").add_legend()
plt.title('Perplexity = 50 , iterations = 500')
plt.show()
```

```
/usr/local/lib/python3.5/dist-packages/seaborn/axisgrid.py:230: UserWarning: The `size` paramter has been renamed to `height
`; please update your code.
  warnings.warn(msg, UserWarning)
```



DATA Preprocessing

In [107]:
```
poems = DATA["content"].values
print(poems[0])
```

```
Let the bird of loudest lay
On the sole Arabian tree
Herald sad and trumpet be,
To whose sound chaste wings obey.

But thou shrieking harbinger,
Foul precurrer of the fiend,
Augur of the fever's end,
To this troop come thou not near.

From this session interdict
Every fowl of tyrant wing,
Save the eagle, feather'd king;
Keep the obsequy so strict.

Let the priest in surplice white,
That defunctive music can,
Be the death-divining swan,
Lest the requiem lack his right.

And thou treble-dated crow,
That thy sable gender mak'st
With the breath thou giv'st and tak'st,
'Mongst our mourners shalt thou go.

Here the anthem doth commence:
Love and constancy is dead;
Phoenix and the Turtle fled
In a mutual flame from hence.

So they lov'd, as love in twain
```

```
Had the essence but in one;
Two distincts, division none:
Number there in love was slain.

Hearts remote, yet not asunder;
Distance and no space was seen
'Twixt this Turtle and his queen:
But in them it were a wonder.

So between them love did shine
That the Turtle saw his right
Flaming in the Phoenix' sight:
Either was the other's mine.

Property was thus appalled
That the self was not the same;
Single nature's double name
Neither two nor one was called.

Reason, in itself confounded,
Saw division grow together,
To themselves yet either neither,
Simple were so well compounded;

That it cried, "How true a twain
Seemeth this concordant one!
Love has reason, reason none,
If what parts can so remain."

Whereupon it made this threne
To the Phoenix and the Dove,
Co-supremes and stars of love,
As chorus to their tragic scene:

                 threnos

Beauty, truth, and rarity,
Grace in all simplicity,
Here enclos'd, in cinders lie.
```

```
Death is now the Phoenix' nest,
And the Turtle's loyal breast
To eternity doth rest,

Leaving no posterity:
'Twas not their infirmity,
It was married chastity.

Truth may seem but cannot be;
Beauty brag but 'tis not she;
Truth and beauty buried be.

To this urn let those repair
That are either true or fair;
For these dead birds sigh a prayer.
```

In [53]:

```
Category = DATA['type']
CAT = []
for i in Category:
    if i=='Love':
        CAT.append(1)
    if i=='Nature':
        CAT.append(2)
    if i=='Mythology & Folklore':
        CAT.append(3)
print(CAT[57])
```

```
2
```

In [104]:

```
stopwords= set(['br', 'the', 'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves'
, 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him',
```

```
'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'th
ey', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "tha
t'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'ha
d', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as',
 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through',
 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'ove
r', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any'
, 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too',
 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'no
w', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't",
'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'might
n', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wa
sn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"])
```

In [40]:

```
# https://stackoverflow.com/a/47091490/4084039
import re
```

```python
def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

In [44]:

```python
from tqdm import tqdm
from bs4 import BeautifulSoup
preprocessed_reviews = []
# tqdm is for printing the status bar
for sentance in tqdm(poems):
    sentance = re.sub(r"http\S+", "", sentance)
    sentance = BeautifulSoup(sentance, 'lxml').get_text()
    sentance = decontracted(sentance)
    sentance = re.sub("\S*\d\S*", "", sentance).strip()
    sentance = re.sub('[^A-Za-z]+', ' ', sentance)
    # https://gist.github.com/sebleier/554280
    sentance = ' '.join(e.lower() for e in sentance.split() if e.lower() not in stopword
s)
    preprocessed_reviews.append(sentance.strip())
```

```
100%|██████████| 573/573 [00:00<00:00, 2435.07it/s]
```

```
In [47]: from sklearn.cross_validation import train_test_split
from sklearn.metrics import accuracy_score,roc_auc_score,roc_curve,confusion_matrix,auc
from sklearn import cross_validation
from scipy.sparse import csr_matrix,hstack
```

```
In [54]: X_1, X_test, y_1, y_test = cross_validation.train_test_split(preprocessed_reviews,CAT, t
est_size=0.2, random_state=0)
X_tr, X_cv, y_tr, y_cv = cross_validation.train_test_split(X_1, y_1, test_size=0.25)
print(np.asarray(X_1).shape,np.asarray(X_test).shape,np.asarray(X_tr).shape,np.asarray(X
_test).shape,np.asarray(X_cv).shape)
```

```
(458,) (115,) (343,) (115,) (115,)
```

Bag of Words

```
In [56]: count_vect = CountVectorizer() #in scikit-learn
BOW_Train = count_vect.fit_transform(X_tr)
BOW_test = count_vect.transform(X_test)
BOW_CV = count_vect.transform(X_cv)
print("some feature names ", count_vect.get_feature_names()[:10])
print('='*50)
print("the type of count vectorizer ",type(BOW_Train))
print("the shape of out text BOW vectorizer ",BOW_Train.get_shape())
print("the number of unique words ", BOW_Train.get_shape()[1])
```

```
some feature names  ['abandon', 'abandonment', 'abasht', 'abate', 'abed', 'abhor', 'abhorring', 'abject', 'abjure', 'abler']
==================================================
the type of count vectorizer  <class 'scipy.sparse.csr.csr_matrix'>
```

```
the shape of out text BOW vectorizer  (343, 7988)
the number of unique words  7988
```

TFIDF

In [57]:

```
tf_idf_vect = TfidfVectorizer(ngram_range=(1,2), min_df=10)
TFIDF_Train = tf_idf_vect.fit_transform(X_tr)
TFIDF_Test = tf_idf_vect.transform(X_test)
TFIDF_Validation = tf_idf_vect.transform(X_cv)
print("the type of count vectorizer ",type(TFIDF_Train))
print("the shape of out text TFIDF vectorizer ",TFIDF_Train.get_shape())
print("the number of unique words including both unigrams and bigrams ", TFIDF_Train.get
_shape()[1])
```

```
the type of count vectorizer  <class 'scipy.sparse.csr.csr_matrix'>
the shape of out text TFIDF vectorizer  (343, 508)
the number of unique words including both unigrams and bigrams  508
```

Word2vec

In [119]:

```
i=0
list_of_sentance=[]
list_of_sentance_cv=[]
list_of_sentance_test=[]
for sentance in X_tr:
    list_of_sentance.append(sentance.split())
for sentance in X_cv:
    list_of_sentance_cv.append(sentance.split())
```

```
for sentance in X_test:
    list_of_sentance_test.append(sentance.split())
```

In [120]:

```
from gensim.models import Word2Vec
is_your_ram_gt_16g=False
want_to_use_google_w2v = False
want_to_train_w2v = True

if want_to_train_w2v:
    # min_count = 5 considers only words that occured atleast 5 times
    w2v_model=Word2Vec(list_of_sentance,min_count=5,size=100, workers=4)
    print(w2v_model.wv.most_similar('great'))
    print('='*50)
    print(w2v_model.wv.most_similar('worst'))

elif want_to_use_google_w2v and is_your_ram_gt_16g:
    if os.path.isfile('GoogleNews-vectors-negative300.bin'):
        w2v_model=KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin'
, binary=True)
        print(w2v_model.wv.most_similar('great'))
        print(w2v_model.wv.most_similar('worst'))
    else:
        print("you don't have gogole's word2vec file, keep want_to_train_w2v = True, to
 train your own w2v ")
```

```
[('not', 0.9999395608901978), ('eyes', 0.9999392628669739), ('yet', 0.9999386072158813), ('love', 0.999935507774353), ('lik
e', 0.9999346733093262), ('still', 0.9999300241470337), ('would', 0.9999276399612427), ('must', 0.9999268054962158), ('thy',
0.9999232888221741), ('might', 0.9999213218688965)]
==================================================
[('rise', 0.9924914240837097), ('slow', 0.9924129843711853), ('name', 0.9922633171081543), ('fresh', 0.9922454357147217),
('fields', 0.9922425150871277), ('stars', 0.9921298623085022), ('fly', 0.9921197295188904), ('place', 0.9920812249183655),
('right', 0.99204421043396), ('even', 0.9920337200164795)]
```

```
/usr/local/lib/python3.5/dist-packages/gensim/matutils.py:737: FutureWarning: Conversion of the second argument of issubdtyp
e from `int` to `np.signedinteger` is deprecated. In future, it will be treated as `np.int64 == np.dtype(int).type`.
  if np.issubdtype(vec.dtype, np.int):
```

In [121]:

```
w2v_words = list(w2v_model.wv.vocab)
print("number of words that occured minimum 5 times ",len(w2v_words))
print("sample words ", w2v_words[0:50])
```

```
number of words that occured minimum 5 times  1354
sample words  ['mayst', 'takes', 'lack', 'land', 'brought', 'nights', 'gan', 'faces', 'whistles', 'withered', 'reason', 'sno
w', 'faithful', 'watched', 'happy', 'painted', 'joy', 'note', 'iron', 'gaine', 'unhappy', 'er', 'broken', 'amorous', 'man',
'await', 'fell', 'silence', 'turned', 'lay', 'graves', 'even', 'hung', 'sorrow', 'fu', 'desires', 'shape', 'village', 'ban
k', 'praise', 'nor', 'melts', 'leaves', 'estate', 'mellow', 'bene', 'sin', 'idle', 'dear', 'bronze']
```

Avg W2v

In [122]:

```
sent_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sent in tqdm(list_of_sentance): # for each review/sentence
    sent_vec = np.zeros(100) # as word vectors are of zero length 50, you might need to
 change this to 300 if you use google's w2v
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sent: # for each word in a review/sentence
        if word in w2v_words:
            vec = w2v_model.wv[word]
            sent_vec += vec
            cnt_words += 1
    if cnt_words != 0:
        sent_vec /= cnt_words
    sent_vectors.append(sent_vec)
sent_vectors_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sent in tqdm(list_of_sentance_cv): # for each review/sentence
```

```
    sent_vec = np.zeros(100) # as word vectors are of zero length 50, you might need to
 change this to 300 if you use google's w2v
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sent: # for each word in a review/sentence
        if word in w2v_words:
            vec = w2v_model.wv[word]
            sent_vec += vec
            cnt_words += 1
    if cnt_words != 0:
        sent_vec /= cnt_words
    sent_vectors_cv.append(sent_vec)
sent_vectors_test = []; # the avg-w2v for each sentence/review is stored in this list
for sent in tqdm(list_of_sentance_test): # for each review/sentence
    sent_vec = np.zeros(100) # as word vectors are of zero length 50, you might need to
 change this to 300 if you use google's w2v
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sent: # for each word in a review/sentence
        if word in w2v_words:
            vec = w2v_model.wv[word]
            sent_vec += vec
            cnt_words += 1
    if cnt_words != 0:
        sent_vec /= cnt_words
    sent_vectors_test.append(sent_vec)
print(len(sent_vectors))
print(len(sent_vectors[0]))
```

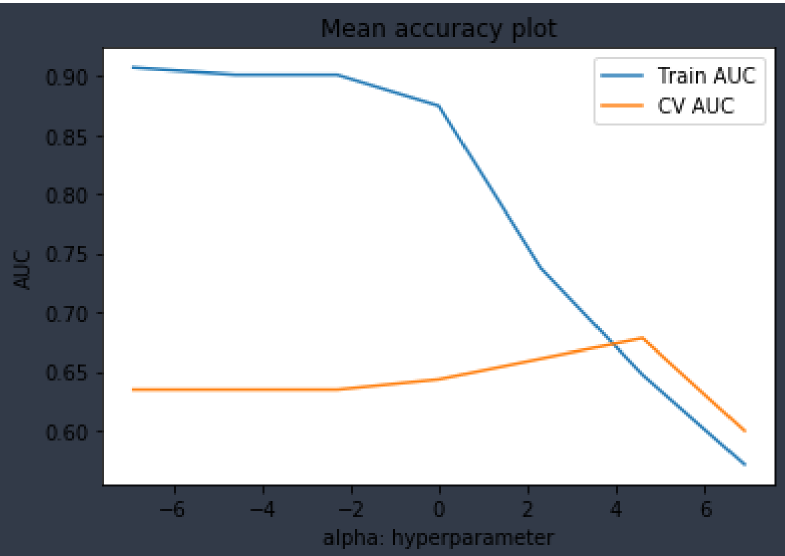
```
100%|██████████| 343/343 [00:00<00:00, 353.39it/s]
100%|██████████| 115/115 [00:00<00:00, 320.57it/s]
100%|██████████| 115/115 [00:00<00:00, 254.52it/s]
```

```
343
100
```

Multinomial Naive Bayes

For BOW

In [59]:
```
from sklearn.naive_bayes import MultinomialNB
alph = [10**(-3),10**(-2),10**(-1),1,10,100,1000]
BOW_Train_Accuracy = []
BOW_CV_Accuracy = []
for i in alph:
    model = MultinomialNB(alpha=i)
    model.fit(BOW_Train,y_tr)
    BOW_Train_Accuracy.append(model.score(BOW_Train,y_tr))
    BOW_CV_Accuracy.append(model.score(BOW_CV,y_cv))

plt.plot(np.log(np.asarray(alph)), BOW_Train_Accuracy, label='Train AUC')
plt.plot(np.log(np.asarray(alph)), BOW_CV_Accuracy, label='CV AUC')
plt.legend()
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("Mean accuracy plot")
plt.show()
```



In [75]:
```
best_alpha = 1
```

Testing on test data

In [87]:
```
model = MultinomialNB(alpha=best_alpha)
model.fit(BOW_Train,y_tr)
Final_accuracy = model.score(BOW_test,y_test)
print("The accuracy of our NAIVE BAYES model is %f "%(Final_accuracy))
```

```
The accuracy of our NAIVE BAYES model is 0.634783
```

Important features

In [92]:
```
a = model.feature_log_prob_
b =  a[0].argsort()[0:-10][::-1]
print(" So the top 10 features of positive class are--")
```

```
for i in range(10):
    print("feature name : %s , value : %f"%(count_vect.get_feature_names()[b[i]],float(a
[1,b[i]])))
print("*"*50)
a = model.feature_log_prob_
b =  a[1].argsort()[0:10][::-1]
print(" So the top 10 features of positive class are--")
for i in range(10):
    print("feature name : %s , value : %f"%(count_vect.get_feature_names()[b[i]],float(a
[1,b[i]])))
print('*'*50)
a = model.feature_log_prob_
b =  a[2].argsort()[0:10][::-1]
print(" So the top 10 features of positive class are--")
for i in range(10):
    print("feature name : %s , value : %f"%(count_vect.get_feature_names()[b[i]],float(a
[1,b[i]])))
```

```
 So the top 10 features of positive class are--
feature name : doth , value : -5.888079
feature name : shall , value : -6.078122
feature name : yet , value : -6.201736
feature name : eyes , value : -6.312962
feature name : heart , value : -6.703828
feature name : still , value : -6.312962
feature name : hath , value : -6.342815
feature name : time , value : -6.794800
feature name : see , value : -6.342815
feature name : us , value : -6.472026
**************************************************
 So the top 10 features of positive class are--
feature name : gnaw , value : -9.839322
feature name : gnat , value : -9.839322

feature name : scope , value : -9.839322
```

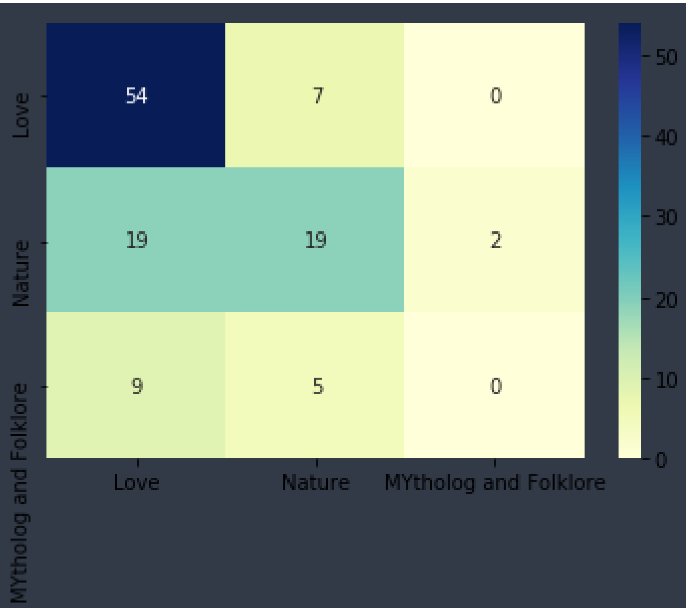
```
feature name : glowing , value : -9.839322
feature name : scorned , value : -9.839322
feature name : gloves , value : -9.839322
feature name : scornful , value : -9.839322
feature name : gloriously , value : -9.839322
feature name : scorning , value : -9.839322
feature name : zone , value : -9.839322
**************************************************
 So the top 10 features of positive class are--
feature name : passenger , value : -9.839322
feature name : passers , value : -9.839322
feature name : passing , value : -7.759881
feature name : passionate , value : -9.839322
feature name : passives , value : -9.839322
feature name : pastime , value : -9.839322
feature name : pastimes , value : -9.839322
feature name : pastry , value : -9.839322
feature name : pastures , value : -9.146175
feature name : abandon , value : -9.146175
```

Confusion Matrix

In [78]:

```
ytest = model.predict(BOW_test)
ctest = confusion_matrix(y_test,ytest)
class_label=["Love","Nature","MYtholog and Folklore"]
df = pd.DataFrame(ctest, index=class_label, columns=class_label)
sns.heatmap(df, annot= True, fmt="d", cmap="YlGnBu")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb8aee9af60>
```

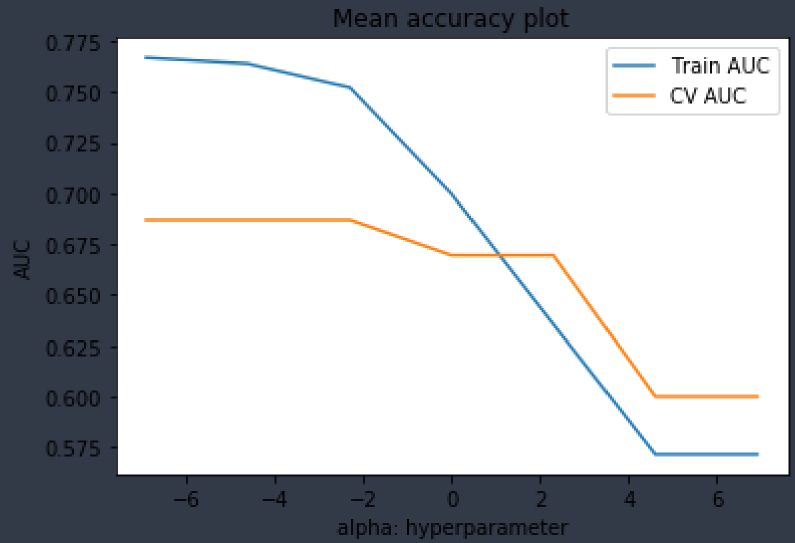


For TFIDF

In [81]:

```
alph = [10**(-3),10**(-2),10**(-1),1,10,100,1000]
TFIDF_Train_Accuracy = []
TFIDF_CV_Accuracy = []
for i in alph:
    model = MultinomialNB(alpha=i)
    model.fit(TFIDF_Train,y_tr)
    TFIDF_Train_Accuracy.append(model.score(TFIDF_Train,y_tr))
    TFIDF_CV_Accuracy.append(model.score(TFIDF_Validation,y_cv))

plt.plot(np.log(np.asarray(alph)), TFIDF_Train_Accuracy, label='Train AUC')
plt.plot(np.log(np.asarray(alph)), TFIDF_CV_Accuracy, label='CV AUC')
plt.legend()
plt.xlabel("alpha: hyperparameter")
```

```
plt.ylabel("AUC")
plt.title("Mean accuracy plot")
plt.show()
```



In [84]:

```
best_alpha = 1
```

In [93]:

```
model = MultinomialNB(alpha=best_alpha)
model.fit(TFIDF_Train,y_tr)
Final_accuracy = model.score(TFIDF_Test,y_test)
print("The accuracy of our NAIVE BAYES model is %f "%(Final_accuracy))
```

```
The accuracy of our NAIVE BAYES model is 0.617391
```

Important Features

In [97]:

```
a = model.feature_log_prob_
```

```
b =  a[0].argsort()[0:10][::-1]
print(" So the top 10 features of positive class are--")
for i in range(10):
    print("feature name : %s , value : %f"%(count_vect.get_feature_names()[b[i]],float(a
[1,b[i]])))
print('*'*50)
a = model.feature_log_prob_
b =  a[1].argsort()[0:10][::-1]
print(" So the top 10 features of positive class are--")
for i in range(10):
    print("feature name : %s , value : %f"%(count_vect.get_feature_names()[b[i]],float(a
[1,b[i]])))
print("*"*50)
a = model.feature_log_prob_
b =  a[2].argsort()[0:10][::-1]
print(" So the top 10 features of positive class are--")
for i in range(10):
    print("feature name : %s , value : %f"%(count_vect.get_feature_names()[b[i]],float(a
[1,b[i]])))
```

```
 So the top 10 features of positive class are--
feature name : aspiring , value : -6.247188
feature name : autumn , value : -5.977974
feature name : act , value : -6.547932
feature name : acherontes , value : -6.441643
feature name : assayde , value : -6.192822
feature name : accept , value : -6.199810
feature name : aiken , value : -6.529133
feature name : asked , value : -5.465876
feature name : bagpype , value : -6.003149
feature name : apply , value : -5.098805
**************************************************
 So the top 10 features of positive class are--
feature name : affairs , value : -6.825026
```

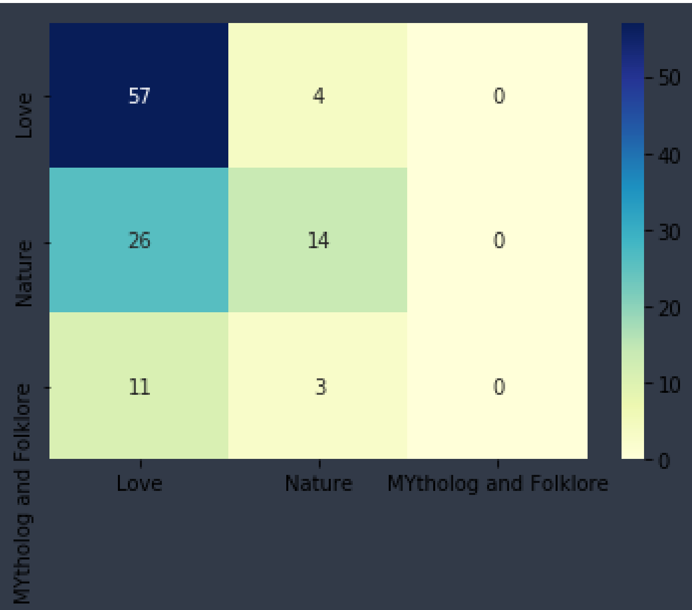
```
feature name : absurdly , value : -6.827404
feature name : allah , value : -6.832129
feature name : adam , value : -6.835932
feature name : april , value : -6.840817
feature name : abused , value : -6.841681
feature name : became , value : -6.862383
feature name : arriv , value : -6.891910
feature name : bacon , value : -6.891910
feature name : ban , value : -6.891910
**************************************************
 So the top 10 features of positive class are--
feature name : although , value : -6.455448
feature name : amorous , value : -6.483691
feature name : anachorit , value : -6.254788
feature name : anotamise , value : -6.769768
feature name : apollo , value : -6.539308
feature name : approve , value : -6.122475
feature name : approved , value : -6.135747
feature name : around , value : -6.564033
feature name : arts , value : -6.676836
feature name : abandon , value : -6.537122
```

Confusion Matrix

In [98]:

```
ytest = model.predict(TFIDF_Test)
ctest = confusion_matrix(y_test,ytest)
class_label=["Love","Nature","MYtholog and Folklore"]
df = pd.DataFrame(ctest, index=class_label, columns=class_label)
sns.heatmap(df, annot= True, fmt="d", cmap="YlGnBu")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb8ad2bb5f8>
```

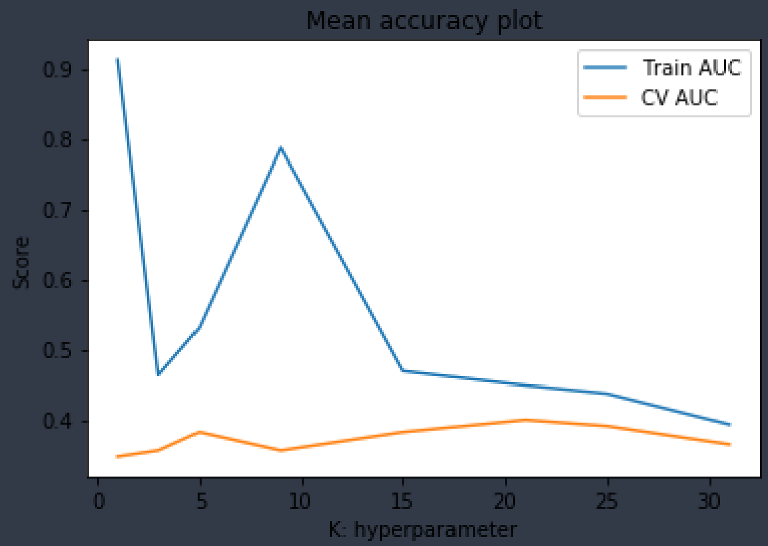


K Nearest Neighbour

In [127]:

```
from sklearn.neighbors import KNeighborsClassifier
k = [1,3,5,9,15,21,25,31]
BOW_Train_Accuracy = []
BOW_CV_Accuracy = []
for i in k:
    model = KNeighborsClassifier(n_neighbors=i)
    model.fit(BOW_Train,y_tr)
    BOW_Train_Accuracy.append(model.score(BOW_Train,y_tr))
    BOW_CV_Accuracy.append(model.score(BOW_CV,y_cv))

plt.plot(np.asarray(k), BOW_Train_Accuracy, label='Train AUC')
plt.plot(np.asarray(k), BOW_CV_Accuracy, label='CV AUC')
```

```
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("Score")
plt.title("Mean accuracy plot")
plt.show()
```



In [129]:
```
best_k = 21
```

Testing on test data

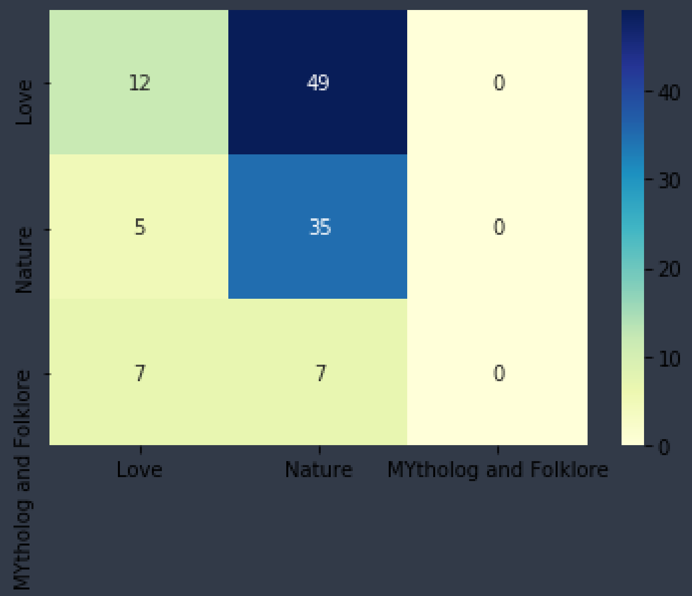
In [132]:
```
model = KNeighborsClassifier(n_neighbors=best_k)
model.fit(BOW_Train,y_tr)
Final_accuracy = model.score(BOW_test,y_test)
print("The final accuracy obtained is %f%%"%(Final_accuracy*100))
```

```
The final accuracy obtained is 40.869565%
```

Confusion Matrix

In [134]:
```
ytest = model.predict(BOW_test)
ctest = confusion_matrix(y_test,ytest)
class_label=["Love","Nature","MYtholog and Folklore"]
df = pd.DataFrame(ctest, index=class_label, columns=class_label)
sns.heatmap(df, annot= True, fmt="d", cmap="YlGnBu")
```

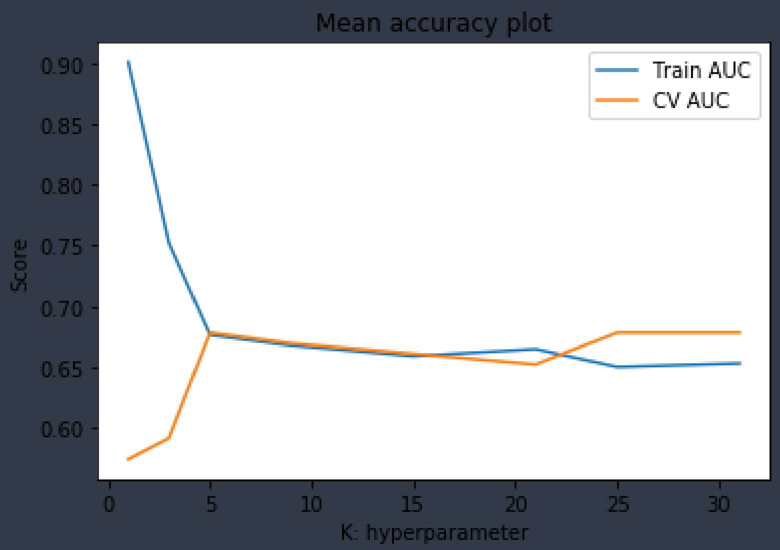
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb893ec1fd0>
```



For W2V

In [135]:
```
k = [1,3,5,9,15,21,25,31]
```

```python
w2v_Train_Accuracy = []
w2v_CV_Accuracy = []
for i in k:
    model = KNeighborsClassifier(n_neighbors=i)
    model.fit(sent_vectors,y_tr)
    w2v_Train_Accuracy.append(model.score(sent_vectors,y_tr))
    w2v_CV_Accuracy.append(model.score(sent_vectors_cv,y_cv))

plt.plot(np.asarray(k), w2v_Train_Accuracy, label='Train AUC')
plt.plot(np.asarray(k), w2v_CV_Accuracy, label='CV AUC')
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("Score")
plt.title("Mean accuracy plot")
plt.show()
```



In [136]:
```python
best_k = 25
```

TEsting on test data

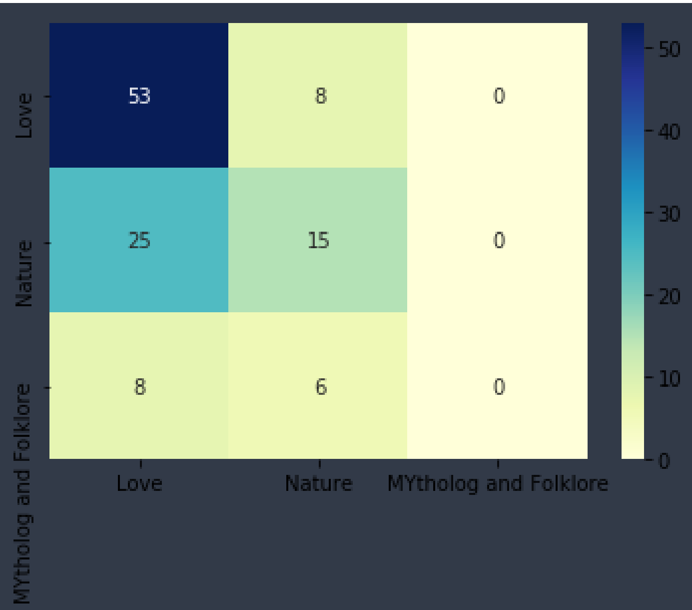
In [137]:
```python
model = KNeighborsClassifier(n_neighbors=best_k)
model.fit(sent_vectors,y_tr)
Final_accuracy = model.score(sent_vectors_test,y_test)
print("The final accuracy obtained is %f%%"%(Final_accuracy*100))
```

```
The final accuracy obtained is 59.130435%
```

Confusion Matrix

In [138]:
```python
ytest = model.predict(sent_vectors_test)
ctest = confusion_matrix(y_test,ytest)
class_label=["Love","Nature","MYtholog and Folklore"]
df = pd.DataFrame(ctest, index=class_label, columns=class_label)
sns.heatmap(df, annot= True, fmt="d", cmap="YlGnBu")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb8a20bb160>
```

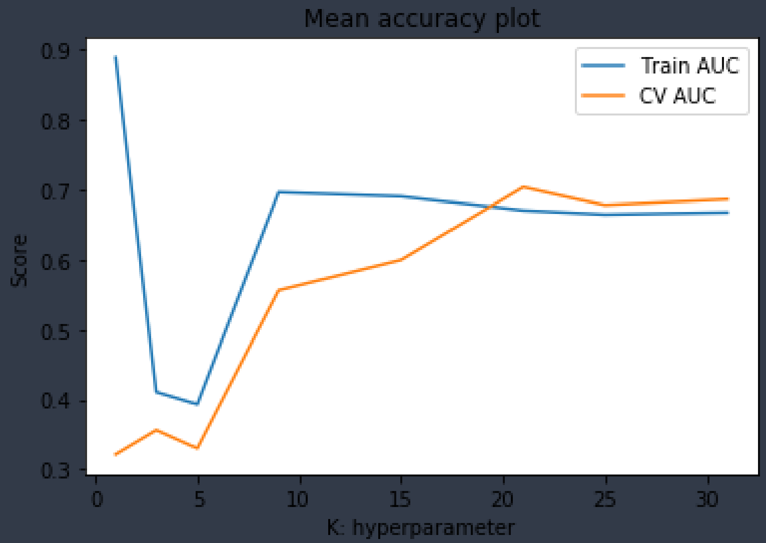


FOR TFIDF

In [141]:

```
k = [1,3,5,9,15,21,25,31]
TFIDF_Train_Accuracy = []
TFIDF_CV_Accuracy = []
for i in k:
    model = KNeighborsClassifier(n_neighbors=i)
    model.fit(TFIDF_Train,y_tr)
    TFIDF_Train_Accuracy.append(model.score(TFIDF_Train,y_tr))
    TFIDF_CV_Accuracy.append(model.score(TFIDF_Validation,y_cv))

plt.plot(np.asarray(k), TFIDF_Train_Accuracy, label='Train AUC')
plt.plot(np.asarray(k), TFIDF_CV_Accuracy, label='CV AUC')
plt.legend()
plt.xlabel("K: hyperparameter")
```

```
plt.ylabel("Score")
plt.title("Mean accuracy plot")
plt.show()
```



In [148]:

```
best_k =19
```

Testing on test data

In [149]:

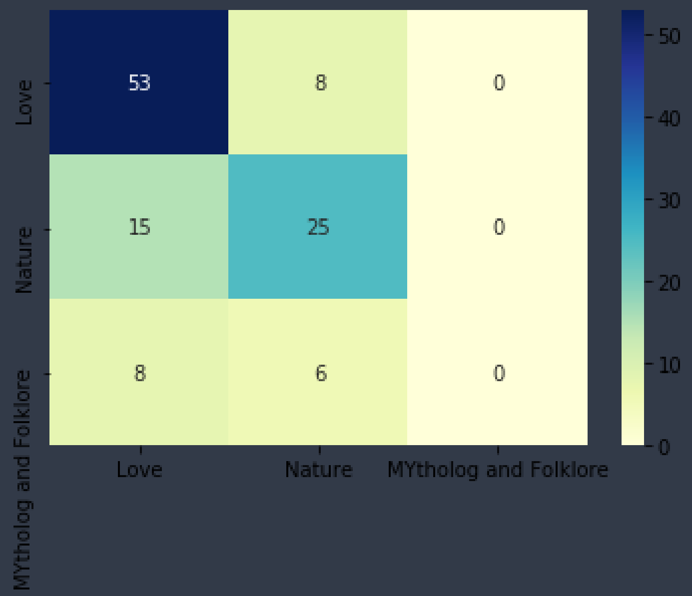
```
model = KNeighborsClassifier(n_neighbors=best_k)
model.fit(TFIDF_Train,y_tr)
Final_accuracy = model.score(TFIDF_Test,y_test)
print("The final accuracy obtained is %f%%"%(Final_accuracy*100))
```

```
The final accuracy obtained is 67.826087%
```

Confusion Matrix

In [150]:
```
ytest = model.predict(TFIDF_Test)
ctest = confusion_matrix(y_test,ytest)
class_label=["Love","Nature","MYtholog and Folklore"]
df = pd.DataFrame(ctest, index=class_label, columns=class_label)
sns.heatmap(df, annot= True, fmt="d", cmap="YlGnBu")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb893de1240>
```



XG BOOST

FOR BOW

In [167]:

```
scipy.sparse.csr.csr_matrix
```

Deep learning

In [173]:

```python
from keras.models import Sequential
from keras.layers import Dense, Activation
from keras.utils import np_utils
```

```
Using TensorFlow backend.
```

In [186]:

```python
Y_TRAIN = []
Y_TEST = []
for i in y_tr:
    if i==1:
        Y_TRAIN.append(0)
    if i==2:
        Y_TRAIN.append(1)
    if i==3:
        Y_TRAIN.append(2)
for i in y_test:
    if i==1:
        Y_TEST.append(0)
    if i==2:
        Y_TEST.append(1)
    if i==3:
        Y_TEST.append(2)
```

In [187]:

```python
Y_train = np_utils.to_categorical(Y_TRAIN, 3)
```

```python
Y_test = np_utils.to_categorical(Y_TEST, 3)

print("After converting the output into a vector : ",Y_train[0])
```

```
After converting the output into a vector :  [0. 1. 0.]
```

In [191]:

```python
ou_dim=3
input_dim = 100
batch_size = 40
epoch = 12
```

In [190]:

```python
model = Sequential()
model.add(Dense(60,activation='sigmoid',input_shape=(input_dim,)))
model.add(Dense(30,activation='sigmoid'))
model.add(Dense(ou_dim,activation = 'softmax'))
model.summary()
```

```
WARNING:tensorflow:From /usr/local/lib/python3.5/dist-packages/tensorflow/python/framework/op_def_library.py:263: colocate_w
ith (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_2 (Dense)              (None, 60)                6060
_________________________________________________________________
dense_3 (Dense)              (None, 30)                1830
_________________________________________________________________
dense_4 (Dense)              (None, 3)                 93
=================================================================
Total params: 7,983
Trainable params: 7,983
Non-trainable params: 0
_________________________________________________________________
```

In [196]:
```
model.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])
hist = model.fit(np.asarray(sent_vectors), Y_train, batch_size=batch_size, epochs=epoch,
 verbose=1, validation_data=(np.asarray(sent_vectors_test), Y_test))
```

```
WARNING:tensorflow:From /usr/local/lib/python3.5/dist-packages/tensorflow/python/ops/math_ops.py:3066: to_int32 (from tensor
flow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 343 samples, validate on 115 samples
Epoch 1/12
343/343 [==============================] - 2s 7ms/step - loss: 1.4944 - acc: 0.1050 - val_loss: 1.3119 - val_acc: 0.1217
Epoch 2/12
343/343 [==============================] - 0s 48us/step - loss: 1.2255 - acc: 0.1050 - val_loss: 1.1512 - val_acc: 0.1217
Epoch 3/12
343/343 [==============================] - 0s 62us/step - loss: 1.0957 - acc: 0.5277 - val_loss: 1.0715 - val_acc: 0.5304
Epoch 4/12
343/343 [==============================] - 0s 62us/step - loss: 1.0284 - acc: 0.5714 - val_loss: 1.0290 - val_acc: 0.5304
Epoch 5/12
343/343 [==============================] - 0s 77us/step - loss: 0.9904 - acc: 0.5714 - val_loss: 1.0042 - val_acc: 0.5304
Epoch 6/12
343/343 [==============================] - 0s 60us/step - loss: 0.9675 - acc: 0.5714 - val_loss: 0.9894 - val_acc: 0.5304
Epoch 7/12
343/343 [==============================] - 0s 98us/step - loss: 0.9536 - acc: 0.5714 - val_loss: 0.9796 - val_acc: 0.5304
Epoch 8/12
343/343 [==============================] - 0s 110us/step - loss: 0.9434 - acc: 0.5714 - val_loss: 0.9732 - val_acc: 0.5304
Epoch 9/12
343/343 [==============================] - 0s 64us/step - loss: 0.9369 - acc: 0.5714 - val_loss: 0.9688 - val_acc: 0.5304
Epoch 10/12
343/343 [==============================] - 0s 66us/step - loss: 0.9327 - acc: 0.5714 - val_loss: 0.9663 - val_acc: 0.5304
Epoch 11/12
343/343 [==============================] - 0s 80us/step - loss: 0.9294 - acc: 0.5714 - val_loss: 0.9649 - val_acc: 0.5304
Epoch 12/12
343/343 [==============================] - 0s 95us/step - loss: 0.9277 - acc: 0.5714 - val_loss: 0.9636 - val_acc: 0.5304
```

In [200]:
```
model = Sequential()
model.add(Dense(50,activation='relu',input_shape=(input_dim,)))
model.add(Dense(25,activation='relu'))
```

```
model.add(Dense(ou_dim,activation = 'softmax'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_8 (Dense)              (None, 50)                5050      
_________________________________________________________________
dense_9 (Dense)              (None, 25)                1275      
_________________________________________________________________
dense_10 (Dense)             (None, 3)                 78        
=================================================================
Total params: 6,403
Trainable params: 6,403
Non-trainable params: 0
_________________________________________________________________
```

In [201]:

```
model.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])
hist = model.fit(np.asarray(sent_vectors), Y_train, batch_size=batch_size, epochs=epoch,
 verbose=1, validation_data=(np.asarray(sent_vectors_test), Y_test))
```

```
Train on 343 samples, validate on 115 samples
Epoch 1/12
343/343 [==============================] - 0s 909us/step - loss: 1.0941 - acc: 0.3265 - val_loss: 1.0723 - val_acc: 0.3739
Epoch 2/12
343/343 [==============================] - 0s 43us/step - loss: 1.0597 - acc: 0.2653 - val_loss: 1.0479 - val_acc: 0.2522
Epoch 3/12
343/343 [==============================] - 0s 45us/step - loss: 1.0352 - acc: 0.5102 - val_loss: 1.0310 - val_acc: 0.5304
Epoch 4/12
343/343 [==============================] - 0s 54us/step - loss: 1.0176 - acc: 0.5714 - val_loss: 1.0181 - val_acc: 0.5304
Epoch 5/12
343/343 [==============================] - 0s 69us/step - loss: 1.0028 - acc: 0.5714 - val_loss: 1.0069 - val_acc: 0.5304
Epoch 6/12
343/343 [==============================] - 0s 56us/step - loss: 0.9897 - acc: 0.5714 - val_loss: 0.9971 - val_acc: 0.5304
Epoch 7/12
343/343 [==============================] - 0s 74us/step - loss: 0.9778 - acc: 0.5714 - val_loss: 0.9891 - val_acc: 0.5304
Epoch 8/12
```

```
343/343 [==============================] - 0s 54us/step - loss: 0.9680 - acc: 0.5714 - val_loss: 0.9824 - val_acc: 0.5304
Epoch 9/12
343/343 [==============================] - 0s 47us/step - loss: 0.9596 - acc: 0.5714 - val_loss: 0.9764 - val_acc: 0.5304
Epoch 10/12
343/343 [==============================] - 0s 49us/step - loss: 0.9517 - acc: 0.5714 - val_loss: 0.9714 - val_acc: 0.5304
Epoch 11/12
343/343 [==============================] - 0s 45us/step - loss: 0.9450 - acc: 0.5714 - val_loss: 0.9671 - val_acc: 0.5304
Epoch 12/12
343/343 [==============================] - 0s 43us/step - loss: 0.9389 - acc: 0.5714 - val_loss: 0.9637 - val_acc: 0.5304
```

In [202]:

```
ou_dim=3
input_dim = 508
batch_size = 25
epoch = 25
```

In [210]:

```
from keras.layers.normalization import BatchNormalization
model = Sequential()
model.add(Dense(200,activation='relu',input_shape=(input_dim,)))
model.add(BatchNormalization())
model.add(Dense(200,activation='relu'))
model.add(BatchNormalization())
model.add(Dense(25,activation='sigmoid'))
model.add(BatchNormalization())
model.add(Dense(ou_dim,activation = 'softmax'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_22 (Dense)             (None, 200)               101800
_________________________________________________________________
batch_normalization_2 (Batch (None, 200)               800
_________________________________________________________________
dense_23 (Dense)             (None, 200)               40200
```

```
_________________________________________________________________
batch_normalization_3 (Batch (None, 200)               800
_________________________________________________________________
dense_24 (Dense)             (None, 25)                5025
_________________________________________________________________
batch_normalization_4 (Batch (None, 25)                100
_________________________________________________________________
dense_25 (Dense)             (None, 3)                 78
=================================================================
Total params: 148,803
Trainable params: 147,953
Non-trainable params: 850
_________________________________________________________________
```

In [211]:

```
model.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])
hist = model.fit(TFIDF_Train, Y_train, batch_size=batch_size, epochs=epoch, verbose=1, v
alidation_data=(TFIDF_Test, Y_test))
```

```
Train on 343 samples, validate on 115 samples
Epoch 1/25
343/343 [==============================] - 1s 3ms/step - loss: 1.3717 - acc: 0.3615 - val_loss: 1.1973 - val_acc: 0.4870
Epoch 2/25
343/343 [==============================] - 0s 317us/step - loss: 0.9280 - acc: 0.5598 - val_loss: 1.1003 - val_acc: 0.5565
Epoch 3/25
343/343 [==============================] - 0s 293us/step - loss: 0.7714 - acc: 0.6764 - val_loss: 1.0659 - val_acc: 0.5826
Epoch 4/25
343/343 [==============================] - 0s 303us/step - loss: 0.6441 - acc: 0.7493 - val_loss: 1.0462 - val_acc: 0.6000
Epoch 5/25
343/343 [==============================] - 0s 335us/step - loss: 0.5560 - acc: 0.8251 - val_loss: 1.0487 - val_acc: 0.5826
Epoch 6/25
343/343 [==============================] - 0s 356us/step - loss: 0.4986 - acc: 0.8571 - val_loss: 1.0333 - val_acc: 0.6174
Epoch 7/25
343/343 [==============================] - 0s 295us/step - loss: 0.4599 - acc: 0.8513 - val_loss: 1.0746 - val_acc: 0.6000
Epoch 8/25
343/343 [==============================] - 0s 287us/step - loss: 0.4045 - acc: 0.8484 - val_loss: 1.0968 - val_acc: 0.5913
Epoch 9/25
343/343 [==============================] - 0s 280us/step - loss: 0.3922 - acc: 0.8688 - val_loss: 1.0883 - val_acc: 0.5913
Epoch 10/25
```

```
343/343 [==============================] - 0s 351us/step - loss: 0.3834 - acc: 0.8688 - val_loss: 1.1137 - val_acc: 0.5565
Epoch 11/25
343/343 [==============================] - 0s 302us/step - loss: 0.3460 - acc: 0.8688 - val_loss: 1.0776 - val_acc: 0.6174
Epoch 12/25
343/343 [==============================] - 0s 343us/step - loss: 0.3499 - acc: 0.8542 - val_loss: 1.0954 - val_acc: 0.6087
Epoch 13/25
343/343 [==============================] - 0s 337us/step - loss: 0.3350 - acc: 0.8601 - val_loss: 1.1186 - val_acc: 0.6174
Epoch 14/25
343/343 [==============================] - 0s 238us/step - loss: 0.3401 - acc: 0.8630 - val_loss: 1.1205 - val_acc: 0.6087
Epoch 15/25
343/343 [==============================] - 0s 274us/step - loss: 0.2959 - acc: 0.8717 - val_loss: 1.1513 - val_acc: 0.5826
Epoch 16/25
343/343 [==============================] - 0s 238us/step - loss: 0.2785 - acc: 0.8805 - val_loss: 1.1357 - val_acc: 0.6261
Epoch 17/25
343/343 [==============================] - 0s 296us/step - loss: 0.2850 - acc: 0.8863 - val_loss: 1.1445 - val_acc: 0.6261
Epoch 18/25
343/343 [==============================] - 0s 254us/step - loss: 0.2834 - acc: 0.8863 - val_loss: 1.1600 - val_acc: 0.6000
Epoch 19/25
343/343 [==============================] - 0s 343us/step - loss: 0.2851 - acc: 0.8688 - val_loss: 1.1491 - val_acc: 0.6348
Epoch 20/25
343/343 [==============================] - 0s 332us/step - loss: 0.2731 - acc: 0.8688 - val_loss: 1.1564 - val_acc: 0.6261
Epoch 21/25
343/343 [==============================] - 0s 343us/step - loss: 0.2583 - acc: 0.8863 - val_loss: 1.1679 - val_acc: 0.6261
Epoch 22/25
343/343 [==============================] - 0s 339us/step - loss: 0.2423 - acc: 0.8834 - val_loss: 1.1796 - val_acc: 0.6174
Epoch 23/25
343/343 [==============================] - 0s 341us/step - loss: 0.2554 - acc: 0.8630 - val_loss: 1.1778 - val_acc: 0.6522
Epoch 24/25
343/343 [==============================] - 0s 322us/step - loss: 0.2817 - acc: 0.8659 - val_loss: 1.1963 - val_acc: 0.6435
Epoch 25/25
343/343 [==============================] - 0s 346us/step - loss: 0.2455 - acc: 0.8921 - val_loss: 1.2063 - val_acc: 0.6435
```

In [212]:

```
from keras.layers.normalization import BatchNormalization
from keras.layers import Dropout
model = Sequential()
model.add(Dense(200,activation='relu',input_shape=(input_dim,)))
model.add(Dropout(0.5))
```

```
model.add(BatchNormalization())
model.add(Dense(200,activation='relu'))
model.add(BatchNormalization())
model.add(Dense(25,activation='relu'))
model.add(Dropout(0.5))
model.add(BatchNormalization())
model.add(Dense(ou_dim,activation = 'softmax'))
model.summary()
```

```
WARNING:tensorflow:From /usr/local/lib/python3.5/dist-packages/keras/backend/tensorflow_backend.py:3445: calling dropout (fr
om tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_26 (Dense)             (None, 200)               101800
_________________________________________________________________
dropout_1 (Dropout)          (None, 200)               0
_________________________________________________________________
batch_normalization_5 (Batch (None, 200)               800
_________________________________________________________________
dense_27 (Dense)             (None, 200)               40200
_________________________________________________________________
batch_normalization_6 (Batch (None, 200)               800
_________________________________________________________________
dense_28 (Dense)             (None, 25)                5025
_________________________________________________________________
dropout_2 (Dropout)          (None, 25)                0
_________________________________________________________________
batch_normalization_7 (Batch (None, 25)                100
_________________________________________________________________
dense_29 (Dense)             (None, 3)                 78
=================================================================
Total params: 148,803
Trainable params: 147,953
```

```
Non-trainable params: 850
_________________________________________________________________
```

In [213]:
```
model.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])
hist = model.fit(TFIDF_Train, Y_train, batch_size=batch_size, epochs=epoch, verbose=1, v
alidation_data=(TFIDF_Test, Y_test))
```

```
Train on 343 samples, validate on 115 samples
Epoch 1/25
343/343 [==============================] - 1s 3ms/step - loss: 1.4229 - acc: 0.3469 - val_loss: 1.1277 - val_acc: 0.4348
Epoch 2/25
343/343 [==============================] - 0s 228us/step - loss: 1.2137 - acc: 0.4402 - val_loss: 1.1087 - val_acc: 0.4522
Epoch 3/25
343/343 [==============================] - 0s 274us/step - loss: 1.2682 - acc: 0.4111 - val_loss: 1.1077 - val_acc: 0.4435
Epoch 4/25
343/343 [==============================] - 0s 293us/step - loss: 1.1732 - acc: 0.4315 - val_loss: 1.0580 - val_acc: 0.5130
Epoch 5/25
343/343 [==============================] - 0s 275us/step - loss: 1.1106 - acc: 0.4898 - val_loss: 1.0294 - val_acc: 0.5478
Epoch 6/25
343/343 [==============================] - 0s 270us/step - loss: 1.0622 - acc: 0.5394 - val_loss: 0.9999 - val_acc: 0.5652
Epoch 7/25
343/343 [==============================] - 0s 282us/step - loss: 1.0779 - acc: 0.5423 - val_loss: 0.9890 - val_acc: 0.6087
Epoch 8/25
343/343 [==============================] - 0s 285us/step - loss: 1.0251 - acc: 0.5306 - val_loss: 0.9668 - val_acc: 0.6522
Epoch 9/25
343/343 [==============================] - 0s 304us/step - loss: 1.0128 - acc: 0.5423 - val_loss: 0.9553 - val_acc: 0.6435
Epoch 10/25
343/343 [==============================] - 0s 309us/step - loss: 0.9458 - acc: 0.5510 - val_loss: 0.9599 - val_acc: 0.6609
Epoch 11/25
343/343 [==============================] - 0s 246us/step - loss: 0.9832 - acc: 0.5627 - val_loss: 0.9507 - val_acc: 0.6609
Epoch 12/25
343/343 [==============================] - 0s 275us/step - loss: 0.9374 - acc: 0.5831 - val_loss: 0.9556 - val_acc: 0.6435
Epoch 13/25
343/343 [==============================] - 0s 306us/step - loss: 0.8619 - acc: 0.6385 - val_loss: 0.9414 - val_acc: 0.6348
Epoch 14/25
343/343 [==============================] - 0s 279us/step - loss: 0.9335 - acc: 0.5977 - val_loss: 0.9457 - val_acc: 0.6000
Epoch 15/25
343/343 [==============================] - 0s 252us/step - loss: 0.8477 - acc: 0.6327 - val_loss: 0.9322 - val_acc: 0.6000
```

```
Epoch 16/25
343/343 [==============================] - 0s 353us/step - loss: 0.8690 - acc: 0.6297 - val_loss: 0.9183 - val_acc: 0.6609
Epoch 17/25
343/343 [==============================] - 0s 291us/step - loss: 0.8751 - acc: 0.6152 - val_loss: 0.9078 - val_acc: 0.6783
Epoch 18/25
343/343 [==============================] - 0s 241us/step - loss: 0.8657 - acc: 0.6443 - val_loss: 0.9136 - val_acc: 0.6522
Epoch 19/25
343/343 [==============================] - 0s 260us/step - loss: 0.8085 - acc: 0.6531 - val_loss: 0.9186 - val_acc: 0.6087
Epoch 20/25
343/343 [==============================] - 0s 264us/step - loss: 0.8210 - acc: 0.6385 - val_loss: 0.9169 - val_acc: 0.6087
Epoch 21/25
343/343 [==============================] - 0s 398us/step - loss: 0.8125 - acc: 0.6647 - val_loss: 0.9231 - val_acc: 0.6087
Epoch 22/25
343/343 [==============================] - 0s 317us/step - loss: 0.8187 - acc: 0.6501 - val_loss: 0.9113 - val_acc: 0.6087
Epoch 23/25
343/343 [==============================] - 0s 349us/step - loss: 0.7885 - acc: 0.6793 - val_loss: 0.9126 - val_acc: 0.6000
Epoch 24/25
343/343 [==============================] - 0s 322us/step - loss: 0.7889 - acc: 0.6647 - val_loss: 0.9039 - val_acc: 0.6348
Epoch 25/25
343/343 [==============================] - 0s 322us/step - loss: 0.8021 - acc: 0.6793 - val_loss: 0.9120 - val_acc: 0.6522
```

In [214]:

```
ou_dim=3
input_dim = 7988
batch_size = 25
epoch = 25
```

In [219]:

```
from keras.initializers import RandomNormal
model = Sequential()
model.add(Dense(3000,activation='relu',input_shape=(input_dim,),kernel_initializer=RandomNormal(mean=0.0, stddev=0.039, seed=None)))
model.add(Dropout(0.5))
model.add(BatchNormalization())
model.add(Dense(1000,activation='relu',kernel_initializer=RandomNormal(mean=0.0, stddev=0.039, seed=None)))
model.add(BatchNormalization())
```

```
model.add(Dense(500,activation='relu',kernel_initializer=RandomNormal(mean=0.0, stddev=
0.039, seed=None)))
model.add(Dropout(0.3))
model.add(BatchNormalization())
model.add(Dense(50,activation='relu',kernel_initializer=RandomNormal(mean=0.0, stddev=0.
039, seed=None)))
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(Dense(ou_dim,activation = 'softmax'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_32 (Dense)             (None, 3000)              23967000
_________________________________________________________________
dropout_4 (Dropout)          (None, 3000)              0
_________________________________________________________________
batch_normalization_9 (Batch (None, 3000)              12000
_________________________________________________________________
dense_33 (Dense)             (None, 1000)              3001000
_________________________________________________________________
batch_normalization_10 (Batc (None, 1000)              4000
_________________________________________________________________
dense_34 (Dense)             (None, 500)               500500
_________________________________________________________________
dropout_5 (Dropout)          (None, 500)               0
_________________________________________________________________
batch_normalization_11 (Batc (None, 500)               2000
_________________________________________________________________
dense_35 (Dense)             (None, 50)                25050
_________________________________________________________________
dropout_6 (Dropout)          (None, 50)                0
_________________________________________________________________
batch_normalization_12 (Batc (None, 50)                200
_________________________________________________________________
```

```
dense_36 (Dense)             (None, 3)                 153
=================================================================
Total params: 27,511,903
Trainable params: 27,502,803
Non-trainable params: 9,100
_________________________________________________________________
```

In [220]:

```python
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
hist = model.fit(BOW_Train, Y_train, batch_size=batch_size, epochs=epoch, verbose=1, validation_data=(BOW_test, Y_test))
```

```
Train on 343 samples, validate on 115 samples
Epoch 1/25
343/343 [==============================] - 9s 27ms/step - loss: 1.4879 - acc: 0.3586 - val_loss: 1.3122 - val_acc: 0.5043
Epoch 2/25
343/343 [==============================] - 6s 17ms/step - loss: 1.0318 - acc: 0.5569 - val_loss: 1.1592 - val_acc: 0.5565
Epoch 3/25
343/343 [==============================] - 6s 17ms/step - loss: 0.7085 - acc: 0.7434 - val_loss: 1.1077 - val_acc: 0.6609
Epoch 4/25
343/343 [==============================] - 6s 16ms/step - loss: 0.5108 - acc: 0.8076 - val_loss: 1.1407 - val_acc: 0.7043
Epoch 5/25
343/343 [==============================] - 6s 17ms/step - loss: 0.4254 - acc: 0.8484 - val_loss: 1.2689 - val_acc: 0.6609
Epoch 6/25
343/343 [==============================] - 6s 16ms/step - loss: 0.3591 - acc: 0.8630 - val_loss: 1.1807 - val_acc: 0.6435
Epoch 7/25
343/343 [==============================] - 6s 17ms/step - loss: 0.4262 - acc: 0.8513 - val_loss: 1.1220 - val_acc: 0.6783
Epoch 8/25
343/343 [==============================] - 6s 17ms/step - loss: 0.3041 - acc: 0.8717 - val_loss: 1.1187 - val_acc: 0.6522
Epoch 9/25
343/343 [==============================] - 6s 17ms/step - loss: 0.2754 - acc: 0.8921 - val_loss: 1.1279 - val_acc: 0.6696
Epoch 10/25
343/343 [==============================] - 6s 17ms/step - loss: 0.2647 - acc: 0.8921 - val_loss: 1.2035 - val_acc: 0.6696
Epoch 11/25
343/343 [==============================] - 6s 16ms/step - loss: 0.3160 - acc: 0.8805 - val_loss: 1.1814 - val_acc: 0.6783
Epoch 12/25
343/343 [==============================] - 6s 17ms/step - loss: 0.3436 - acc: 0.8659 - val_loss: 1.1882 - val_acc: 0.6522
Epoch 13/25
343/343 [==============================] - 6s 17ms/step - loss: 0.2685 - acc: 0.8863 - val_loss: 1.1811 - val_acc: 0.6522
```

```
Epoch 14/25
343/343 [==============================] - 6s 17ms/step - loss: 0.2318 - acc: 0.8980 - val_loss: 1.1902 - val_acc: 0.6522
Epoch 15/25
343/343 [==============================] - 6s 17ms/step - loss: 0.2598 - acc: 0.8805 - val_loss: 1.2309 - val_acc: 0.6609
Epoch 16/25
343/343 [==============================] - 6s 17ms/step - loss: 0.2815 - acc: 0.8746 - val_loss: 1.4337 - val_acc: 0.6174
Epoch 17/25
343/343 [==============================] - 6s 17ms/step - loss: 0.2260 - acc: 0.8950 - val_loss: 1.3036 - val_acc: 0.6522
Epoch 18/25
343/343 [==============================] - 6s 16ms/step - loss: 0.2341 - acc: 0.9067 - val_loss: 1.3804 - val_acc: 0.6435
Epoch 19/25
343/343 [==============================] - 6s 17ms/step - loss: 0.2452 - acc: 0.8921 - val_loss: 1.4044 - val_acc: 0.6348
Epoch 20/25
343/343 [==============================] - 6s 18ms/step - loss: 0.2564 - acc: 0.8921 - val_loss: 1.3958 - val_acc: 0.6609
Epoch 21/25
343/343 [==============================] - 6s 17ms/step - loss: 0.2274 - acc: 0.8805 - val_loss: 1.3537 - val_acc: 0.6609
Epoch 22/25
343/343 [==============================] - 6s 17ms/step - loss: 0.2533 - acc: 0.8746 - val_loss: 1.4111 - val_acc: 0.6522
Epoch 23/25
343/343 [==============================] - 6s 17ms/step - loss: 0.2105 - acc: 0.8950 - val_loss: 1.4879 - val_acc: 0.6348
Epoch 24/25
343/343 [==============================] - 6s 17ms/step - loss: 0.2575 - acc: 0.8892 - val_loss: 1.5337 - val_acc: 0.6087
Epoch 25/25
343/343 [==============================] - 6s 16ms/step - loss: 0.2031 - acc: 0.8950 - val_loss: 1.5965 - val_acc: 0.5826
```