

2019-1002 IST 736 Text Mining

Homework Assignment 7 (week 7)

Ryan Timbrook

NetID: RTIMBROO

Course: IST 736 Text Mining

Term: Fall, 2019

Topic: Build MNB and SVM models using sklearn packages on Kaggle Sentiment training data.

Homework Assignment 7 (week 7)

Table of Contents

1	Introduction	3
1.1	Purpose	3
1.2	Scope	3
2	Analysis and Models	5
2.1	About the Data	5
2.1.1	Dataset Info	5
2.1.2	Data Exploration & Cleaning	6
3	Models	9
3.1.1	Classification Models MNB and SVM Comparision	9
4	Conclusions	17
5	Appendix: References	18

Homework Assignment 7 (week 7)

1 Introduction

Every business needs to keep a pulse on how they are perceived in the public eye. The movie-making industry is no exception. And maybe they should be looking very closely at how movie review critics are rating their movies on sites like Rotten Tomatoes.

A headline from Wired reads:

"Rotten Tomatoes and the Unbearable Heaviness of Data

More and more movies are tanking -- and sure, they might be bad, but something else may be at play: metadata dependency." - <https://www.wired.com/story/is-rotten-tomatoes-ruining-movies/>

Rotten tomato has a simple approach to how movie reviews can be compiled into binary pass/fail assessments -- inspired by the thumbs up or down of the critics Gene Siskel and Roger Ebert -- into a quantified selfie that captures a movie's overall quality.

The site maintains fairly straightforward rules about which reviewers and outlets it'll draw from -- about 2,000 critics overall contribute. Some critics have adapted to the binary distinction, sending along with word as to how Rotten Tomatoes should code their possibly more subtle review. "Some days a 2.5 out of 5 out of a particular critic might be fresh, and with a different movie might be a rotten." And Rotten Tomatoes is OK with that.

The challenge and concern are that Rotten Tomatoes scores now show up not only on the site, but also in reviews and articles about the movies they purport to assess, and next to ticket purchase options on Fandango.

Bad reviews are bad news for brands, large or small. The one thing companies can do to listen to their customers. Since companies typically have large amounts of customers, any of whom could be prone to posting comments on the internet, and they need to use sophisticated technology to narrow the playing field and find potential negative reviews or false statements being written about them online before the damage is too great to fix.

1.1 Purpose

Build Multinomial Naive Bayes and Support Vector Machine classification models for comparison on Kaggle Sentiment classification dataset of Rotten Tomatoes Movie Reviews.

["Sentiment Analysis on Movie Reviews"](#) -- Classify the sentiment of sentences from the Rotten Tomatoes dataset

1.2 Scope

Given a labeled data set of movie review's broken into their constituent phrases, perform classification modeling tasks using both Multinomial Naive Bayes and Support Vector Machines to evaluate which modeling technique and hyperparameter configuration perform the best on unseen-held out data. Label phrases on a scale of five values: negative, somewhat negative, neutral, somewhat positive, positive.

- Perform three classification tasks
 - Task 1:
 - Build a unigram MNB model and a unigram SVMs model.

Homework Assignment 7 (week 7)

- Print the top 10 indicative words for the most positive category and the most negative category from the MNB and SVMs models respectively.
- Report the confusion matrix, precisions, and recalls. Explain whether your models performed equally well on all categories, or some categories turn out to be easier or more difficult for MNB or SVMs.
- Task 2:
 - Revise the script to build a MNB model and a SVMs model based on both unigram and bigram. For fair comparison, keep the same 60% for training and the rest 40% for testing. Also, keep vectorization parameters the same as in Task 1.
 - Compare the confusion matrix and other evaluation measures (accuracy, precision, recall). Discuss whether adding bi-grams was helpful for sentiment classification, based on MNB and SVMs respectively.
- Task 3:
 - revise the script to build best SVMs model by tuning parameters and using the entire training data set (changing from 60% to 100%). Report what parameters were used to train the model and its cross validation accuracy.
 - Use this model to predict the Kaggle sentiment test data. Submit the prediction result to Kaggle. <https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/submit>

2 Analysis and Models

2.1 About the Data

[Kaggle Sentiment Analysis on Movie Reviews](#)

The Rotten Tomatoes movie review dataset is a corpus of movie reviews used for sentiment analysis, originally collected by Pang and Lee [1]. In their work on sentiment treebanks, Socher et al. [2] used Amazon's Mechanical Turk to create fine-grained labels for all parsed phrases in the corpus.

The dataset is comprised of tab-separated files with phrases from the Rotten Tomatoes dataset. The train/test split has been preserved for the purposes of benchmarking, but the sentences have been shuffled from their original order. Each Sentence has been parsed into many phrases by the Stanford parser. Each phrase has a Phraseld. Each sentence has a Sentenceld. Phrases that are repeated (such as short/common words) are only included once in the data.

- train.tsv: contains the phrases and their associated sentiment labels.
- test.tsv: contains just phrases. You must assign a sentiment label to each phrase.
 - Label phrases on a scale of five values: negative, somewhat negative, neutral, somewhat positive, positive.

The sentiment labels are:

- 0 - negative
- 1 - somewhat negative
- 2 - neutral
- 3 - somewhat positive
- 4 - positive

2.1.1 Dataset Info

The initial shape of the train data set is (156060, 4). It has 156060 rows and 4 columns. Its initial size is 624240. The Kaggle test submission data set is (66292, 3). It has 66292 rows and 3 columns. Note that the test set does not have sentiment labels. It's to be used for final un-seen test prediction and submission to Kaggle for accuracy scoring.

Train Data Set Object Info	Test Data Set Object Info
<ul style="list-style-type: none"> • RangeIndex: 156060 entries, 0 to 156059 • Data columns (total 4 columns): • Phraseld 156060 non-null int64 • Sentenceld 156060 non-null int64 • Phrase 156060 non-null object • Sentiment 156060 non-null int64 • dtypes: int64(3), object(1) • memory usage: 4.8+ MB 	<ul style="list-style-type: none"> • RangeIndex: 66292 entries, 0 to 66291 • Data columns (total 3 columns): • Phraseld 66292 non-null int64 • Sentenceld 66292 non-null int64 • Phrase 66292 non-null object • dtypes: int64(2), object(1) • memory usage: 1.5+ MB

Figure 2.1 below is a small representation of how the training data looks when first loaded.

Figure 2.1: Train Data Set sample:

Homework Assignment 7 (week 7)

Phraseld	Sentenceld	Phrase	Sentiment
1	1	A series of escapades demonstrating the adage that what is good for the goose is also good for the gander , some of which occasionally amuses but none of which amounts to much of a story .	1
2	1	A series of escapades demonstrating the adage that what is good for the goose	2
3	1	A series	2
4	1	A	2
5	1	series	2
64	2	This quiet , introspective and entertaining independent is worth seeking .	4
65	2	This quiet , introspective and entertaining independent	3
66	2	This	2
67	2	quiet , introspective and entertaining independent	4
68	2	quiet , introspective and entertaining	3

2.1.2 Data Exploration & Cleaning

All cleaning and transformation techniques were performed programmatically in python using a jupyter notebook for code execution and visualization. The python version used was Anaconda 3.6.

For each Task described above, a CountVectorizer is trained on the train data set content and performs cleaning and transformation actions according to it's input configuration parameters. Details for each of the tasks will be covered in section 2.1.2.2 below.

2.1.2.1 Data Exploration

Figure 2.2: Training Data Set WordCloud



Homework Assignment 7 (week 7)

2.1.2.2 Vectorization Steps:

To train the vectorizers on the kaggle training data set, the data was segmented into X (Phrases) and y (Sentiment) numpy arrays. For each modeling task, the data was split using sklearn's train_test_split module to the specifications of the input parameter "test_size". This is used for validation of the model's prediction accuracy before being used on the final kaggle test data set submission.

2.1.2.2.1 CountVectorizer

Utilizing the python package **sklearn.feature_extraction.text CountVectorizer** class, this model converts a collection of customer review text documents to a matrix of token counts. This implementation of CountVectorizer produces a sparse representation of the counts using `scipy.sparse.coo_matrix`.

In-text mining, it is important to create the document-term matrix (DTM) of the corpus we are interested in. A DTM is basically a matrix, with documents designated by rows and words by columns, that the elements are the counts or the weights (usually by tf-idf). The subsequent analysis is usually based creatively on DTM.

CountVectorizer supports counts of N-grams of words or consecutive characters. Once fitted, the vectorizer has built a dictionary of feature indices. The index value of a word in the vocabulary is linked to its frequency in the whole training corpus. You can think of an N-gram as the sequence of N words, by that notion, a 2-gram (or bigram) is a two-word sequence of words like "please turn", "turn your", or "your homework", and a 3-gram (or trigram) is a three-word sequence of words like "please turn your", or "turn your homework". N-grams are used in building predictive language models based on models learned word sequencing.

The data for these vectorization steps is the kaggle train data set. Each of the vectorization steps below were for Multinomial Naive Baise and Support Vector Machine Classification modeling comparison.

Task 1

Initialize CountVectorizer unigram vector objects for MNB and SVM modeling

Vectorizer Name:

t1_mnb_count_vec_unigram

Parameters:

- input='content'
- ngram_range=(1,1)
- max_features=None
- max_df=1.0
- min_df=2
- analyzer=word
- stop_words='english'

Vectorizer Name:

t1_svm_count_vec_unigram

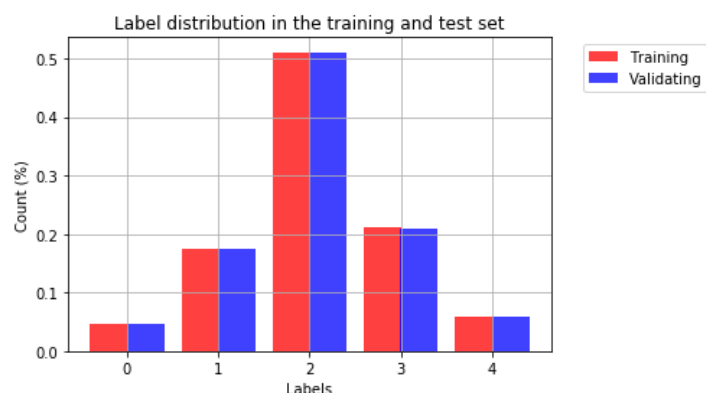
Parameters:

- input='content'
- ngram_range=(1,1)
- max_features=None
- max_df=1.0
- min_df=2
- analyzer=word
- stop_words='english'

Train Data Set Split Configurations:

- train_test_split: test_size=0.4

Figure 2.3: Task 1 Label Distributions



- Transformed Shape: (93636, 12018)
- Vocabulary Size: 12018

Homework Assignment 7 (week 7)

Task 2

Initialize CountVectorizer bigram vector objects for MNB and SVM modeling

Vectorizer Name: t2_mnb_count_vec_bigram

Parameters:

- input='content'
- ngram_range=(1,2)
- max_features=None
- max_df=1.0
- min_df=2
- analyzer=word
- stop_words='english'

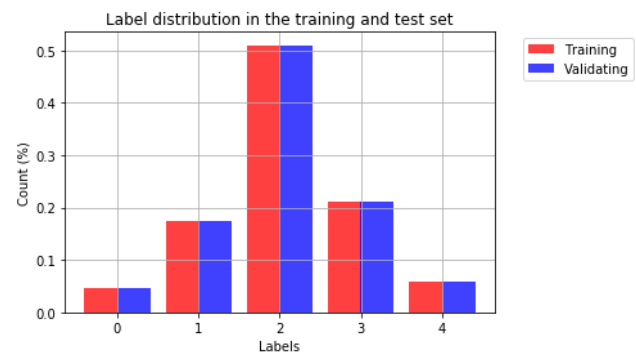
Vectorizer Name: t2_svm_count_vec_bigram

Parameters:

- input='content'
- ngram_range=(1,2)
- max_features=None
- max_df=1.0
- min_df=2
- analyzer=word
- stop_words='english'

Train Data Set Split Configurations:

- train_test_split: test_size=0.4

Figure 2.4: Task 2 Label Distribution

- Transformed Shape: (93636, 34437)
- Vocabulary Size: 500431

Task 3

Initialize CountVectorizer bigram vector objects for MNB and SVM modeling

Vectorizer Name: t3_svm_count_vec_unigram

Parameters:

- input='content'
- ngram_range=(1,1)
- max_features=None
- max_df=1.0
- min_df=2
- analyzer=word
- stop_words='english'

Transformed Shape: (x, x)

Vocabulary Size: x

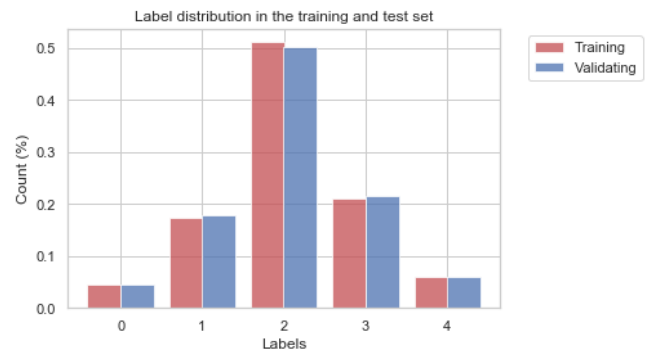
Vectorizer Name: t3_svm_count_vec_bigram

Parameters:

- input='content'
- ngram_range=(1,2)
- max_features=None
- max_df=1.0
- min_df=2
- analyzer=word
- stop_words='english'

Train Data Set Split Configurations:

- train_test_split: test_size=0.1

Figure 2.5: Task 3 Label Distribution

3 Models

Train Test Split Process

Labels for the datasets were encoded using the sklearn.preprocessing LabelEncoder class.

For prediction evaluation and accuracy measurement, 40% of each dataset was held out as unseen data. The method used was sklearn.model_selection train_test_split class.

Build-Test-Validate-Predict MNB and SVM Models

Model training and validation was performed by using sklearn.model_selection cross_validate. A 10 fold cross validation measure was used in training and validating each of vector models training dataset. 40% of each was held out for final, unseen, prediction accuracy evaluation for Task 1 and Task 2, Task 3 was trained at a 90/10 split inorder to maximize training data observations while still being able to report on accuracy scoring.

****Note:** A complete listing of all model result details can be found in the .output/summary_report_final.xlsx

3.1.1 Classification Models MNB and SVM Comparision

3.1.1.1 Multinomial Naive Bayes (MNB) Models

For our text classification task we are using the Naive Bayes classifier for multinomial models. The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work. - scikit-learn MultinomialNB

For this implementation we are using scikit-learn v0.21.3 sklearn.naive_bayes MultinomialNB class. The below steps were taking for each of the CountVectorizer vector models in the given task.

3.1.1.2 Support Vector Machine (SVM) Models

Linear Support Vector Classification (LinearSVC)

Python package scikit-learn v0.21.3 [sklearn.svm.LinearSVC](#)

Similar to SVC with parameter kernel='linear', but implemented in terms of liblinear rather than libsvm, so it has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples.

This class supports both dense and sparse input and the multiclass support is handled according to a one-vs-the-rest scheme.

Read more in the [User Guide](#).

Interpreting LinearSVC models:

- LinearSVC uses a one-vs-all strategy to extend the binary SVM classifier to multi-class problems
- For the Kaggle sentiment classification problem, there are five categories 0,1,2,3,4 with 0 as very negative and 4 very positive
- LinearSVC builds five binary classifier, "very negative vs. others", "negative vs. others", "neutral vs. others", "positive vs. others", "very positive vs. others", and then pick the most confident prediction as the final prediction.
- Linear SVC also ranks all features based on their contribution to distinguish the two concepts in each binary classifier

Homework Assignment 7 (week 7)

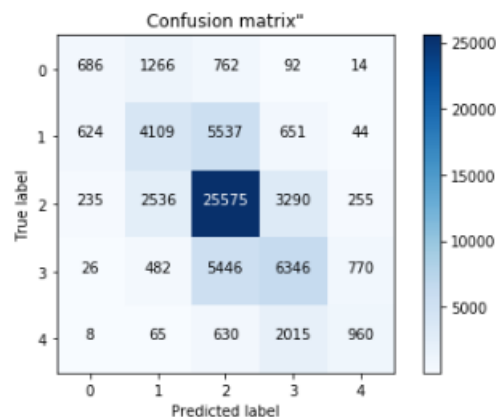
3.1.1.3 Task 1

- Build unigram models
- Print top 10 indicative words for most positive category and the most negative category
- Report confusion matrix, precision, and recall scores.

The SVM model performs marginally better than the MNB model by 2%. It appears that both models had similar challenges with each of the classification categories. The f1-score's have an interesting pattern which aligns to the training data label distribution inequities shown in figure 2.3 above.

3.1.1.3.1 MNB Unigram Results

Most POSITIVE WORDS			Most NEGATIVE WORDS		
Top and Bottom 10 of Most POSITIVE Learned Words			Top and Bottom 10 of Most NEGATIVE Learned Words		
-10.5615	102	-5.7994 great	-10.4308	10th	-6.0120 story
-10.5615	103	-5.7740 performance	-10.4308	112	-5.8876 minutes
-10.5615	104	-5.6712 comedy	-10.4308	12	-5.8665 characters
-10.5615	105	-5.6637 story	-10.4308	127	-5.8561 worst
-10.5615	10th	-5.6057 performances	-10.4308	129	-5.7865 comedy
-10.5615	110	-5.4617 good	-10.4308	12th	-5.6602 just
-10.5615	112	-5.3520 funny	-10.4308	13th	-5.2105 like
-10.5615	120	-5.1816 best	-10.4308	14	-4.8511 film
-10.5615	13th	-4.8282 movie	-10.4308	16	-4.8068 bad
-10.5615	140	-4.2849 film	-10.4308	163	-4.3741 movie

MNB Model Prediction Accuracy Results: 60.35%

Confusion Matrix:

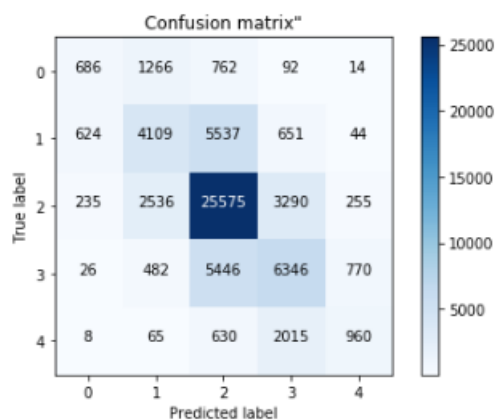
```
[[ 686 1266  762   92   14]
 [ 624 4109 5537  651   44]
 [ 235 2536 25575 3290  255]
 [   26  482  5446 6346  770]
 [    8   65   630 2015  960]]
```

Classification Report:				
	precision	recall	f1-score	support
very_negative	0.43	0.24	0.31	2820
negative	0.49	0.37	0.42	10965
neutral	0.67	0.80	0.73	31891
positive	0.51	0.49	0.50	13070
very_positive	0.47	0.26	0.34	3678
micro avg	0.60	0.60	0.60	62424
macro avg	0.52	0.43	0.46	62424
weighted avg	0.58	0.60	0.59	62424

Homework Assignment 7 (week 7)

3.1.1.3.2 SVM Unigram Results

Most POSITIVE WORDS				Most NEGATIVE WORDS			
Top and Bottom 10 of Most POSITIVE Learned Words				Top and Bottom 10 of Most NEGATIVE Learned Words			
-2.3033	swim	1.6590	defining	-1.6409	won	1.7126	dud
-2.2927	empire	1.6803	uplifter	-1.6203	giving	1.7668	repugnant
-2.0725	guarantee	1.6926	dreamy	-1.5882	month	1.7721	disgusting
-1.9047	losers	1.7024	scientists	-1.5300	sopranos	1.7907	zzzzzzzzzz
-1.8120	stops	1.7614	masterfully	-1.4831	enters	1.7955	flopped
-1.8073	million	1.8522	adorns	-1.3472	passionate	1.7998	loser
-1.7778	costumed	1.9796	flawless	-1.3472	truthful	1.8665	loathsome
-1.7733	maintained	1.9996	miraculous	-1.3002	collar	1.8899	encountering
-1.7083	nonchallenging	2.0021	masterpeice	-1.2982	rare	2.0561	repulsive
-1.6863	placed	2.0272	perfection	-1.2632	extension	2.0998	disappointment

SVM Model Prediction Accuracy Results: 62.34%

Classification Report:				
	precision	recall	f1-score	support
very_negative	0.45	0.33	0.38	2852
negative	0.51	0.39	0.44	10887
neutral	0.69	0.83	0.75	31809
positive	0.55	0.44	0.49	13185
very_positive	0.50	0.37	0.43	3691
micro avg	0.62	0.62	0.62	62424
macro avg	0.54	0.47	0.50	62424
weighted avg	0.60	0.62	0.61	62424

Confusion Matrix:					
[945	1210	598	86	13]
[841	4267	5187	527	65]
[252	2420	26538	2422	177]
[35	432	5828	5798	1092]
[5	44	506	1768	1368]]

3.1.1.4 Task 2

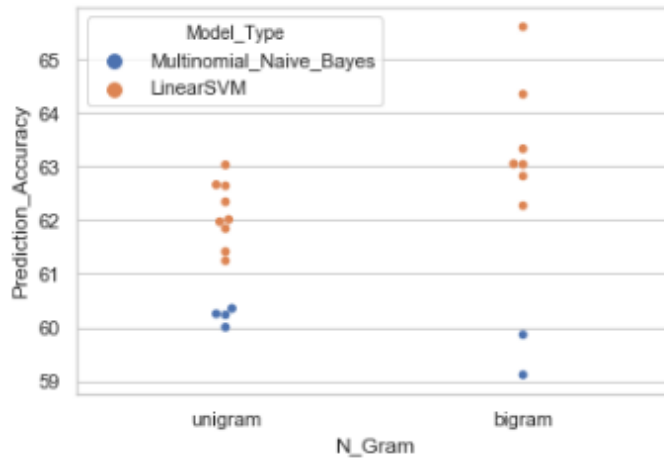
- Build bigram models
- Print top 10 indicative words for most positive category and the most negative category
- Report confusion matrix, precision, and recall scores.
- Compare evaluation measures, was adding bigrams helpful?

The SVM bigram model performs marginally better than the MNB bigram model by 3.48%. It appears that both models had similar challenges with each of the classification categories. The f1-score's have an interesting

Homework Assignment 7 (week 7)

pattern which aligns to the training data label distribution inequities shown in figure 2.3 above.

3.1.1.4.1 Model Comparison Prediction Accuracies

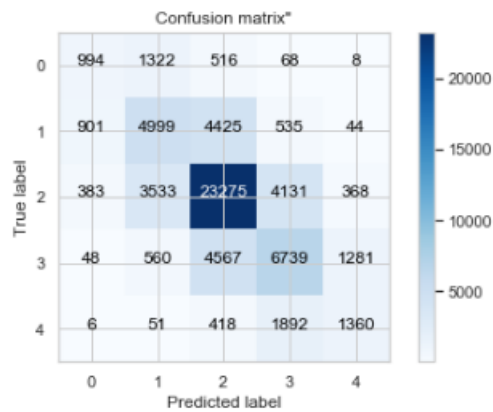


- Overall, LinearSVMs performed the best, with Bigram's being the best predictors

3.1.1.4.2 MNB Bigram Results

Most POSITIVE WORDS				Most NEGATIVE WORDS			
Top and Bottome 10 of Most POSITIVE Learned Words				Top and Bottome 10 of Most NEGATIVE Learned Words			
-11.6239	000	-6.9054	performance	-11.5449	000 leagues	-7.0676	long
-11.6239	000 leagues	-6.8790	great	-11.5449	10 course	-7.0563	characters
-11.6239	000 times	-6.7956	performances	-11.5449	10 year	-7.0563	comedy
-11.6239	10 000	-6.7797	comedy	-11.5449	100 minutes	-7.0123	time
-11.6239	10 15	-6.7797	story	-11.5449	100 year	-6.9910	minutes
-11.6239	10 course	-6.4035	good	-11.5449	101 minutes	-6.8721	just
-11.6239	10 seconds	-6.3560	funny	-11.5449	101 premise	-6.4329	like
-11.6239	10 set	-6.2533	best	-11.5449	101 study	-6.0769	bad
-11.6239	10 years	-5.8370	movie	-11.5449	10th	-6.0115	film
-11.6239	100 minute	-5.4113	film	-11.5449	10th film	-5.5263	movie

Homework Assignment 7 (week 7)

MNB Bigram Model Prediction Accuracy Results: 59.85%

Classification Report:				
	precision	recall	f1-score	support
very_negative	0.43	0.34	0.38	2908
negative	0.48	0.46	0.47	10904
neutral	0.70	0.73	0.72	31690
positive	0.50	0.51	0.51	13195
very_positive	0.44	0.36	0.40	3727
micro avg	0.60	0.60	0.60	62424
macro avg	0.51	0.48	0.49	62424
weighted avg	0.59	0.60	0.59	62424

Confusion Matrix:

```

[[ 994 1322  516    68    8]
 [ 901 4999 4425  535   44]
 [ 383 3533 23275 4131  368]
 [  48  560 4567 6739 1281]
 [   6   51  418 1892 1360]]

```

3.1.1.4.3 SVM Bigram Results**Most POSITIVE WORDS**

Top and Bottom 10 of Most POSITIVE Learned Words

-1.8985 american add	1.5752 ingenious
-1.6432 experience informative	1.5754 hardly ask
-1.5741 locales exceptional	1.5778 masterfully
-1.5552 genuine dramatic	1.5907 riveted
-1.5527 vivid possible	1.5987 breathtaking
-1.4987 laughter surprised	1.6540 excellent
-1.4894 directed highly	1.6570 flawless
-1.4176 devastating experience	1.7120 magnificent
-1.4013 affable	1.8270 masterpeice
-1.3938 dragon worthy	1.8988 perfection

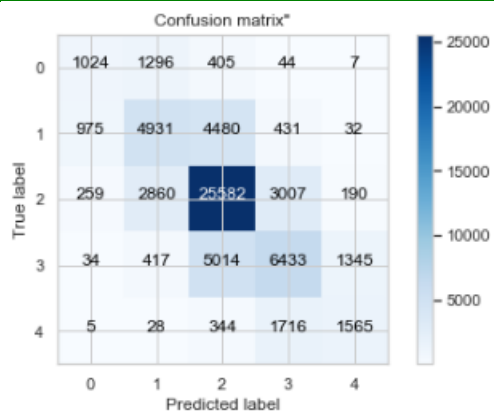
Most NEGATIVE WORDS

Top and Bottom 10 of Most NEGATIVE Learned Words

-1.7975 imitation really	1.6704 baaaaaaaad
-1.7339 indulgent worst	1.7175 disappointingly
-1.6449 humorless chore	1.7703 snoozer
-1.5710 makes passes	1.7889 terrible
-1.4739 fact film	1.8026 turd
-1.4361 acting ensemble	1.8060 repugnant
-1.3901 drama screams	1.8097 snooze
-1.3822 sequels fails	1.8497 unappealing
-1.3816 plotting insultingly	1.8525 dud
-1.3622 poo poo	1.8879 disappointment

SVM Bigram Model Prediction Accuracy Results: 63.33%

Homework Assignment 7 (week 7)



Classification Report:				
	precision	recall	f1-score	support
very_negative	0.45	0.37	0.40	2776
negative	0.52	0.45	0.48	10849
neutral	0.71	0.80	0.76	31898
positive	0.55	0.49	0.52	13243
very_positive	0.50	0.43	0.46	3658
micro avg	0.63	0.63	0.63	62424
macro avg	0.55	0.51	0.52	62424
weighted avg	0.62	0.63	0.62	62424

Confusion Matrix:

```
[[ 1024  1296   405    44     7]
 [  975  4931  4480   431    32]
 [  259  2860 25582  3007   190]
 [   34   417  5014  6433  1345]
 [    5    28   344  1716  1565]]
```

3.1.1.5 Task 3

- Build best SVMs model by tuning parameters using entire training data set
- Report parameters used to train the model and it's cross validation accuracy

Best SVM vectorizer is the Bigram.

To determine the best LinearSVM hyperparameters to configure the model with GridSearchCV was used.

GridSearchCV is a parameter estimation module using grid search with cross-validation.

Hyper-parameters are parameters that are not directly learnt within estimators. In scikit-learn they are passed as arguments to the constructor of the estimator classes.

GridSearchCV was ran with the following parameter ranges:

```
{
'C': [0.1, 1, 10, 100, 1000],
'loss': ['hinge','squared_hinge'],
}
```

- Fitting 3 folds for each of 10 candidates, totalling 30 fits
- grid.best_params_: {'C': 1, 'loss': 'hinge'}

```
GridSearchCV(cv='warn', error_score='raise-deprecating', estimator=LinearSVC(C=1.0,
class_weight=None, dual=True, fit_intercept=True, intercept_scaling=1, loss='squared_hinge',
max_iter=1000, multi_class='ovr', penalty='l2', random_state=None,
tol=0.0001,verbose=0),fit_params=None, iid='warn', n_jobs=None, param_grid={'C': [0.1, 1, 10, 100, 1000],
```

Homework Assignment 7 (week 7)

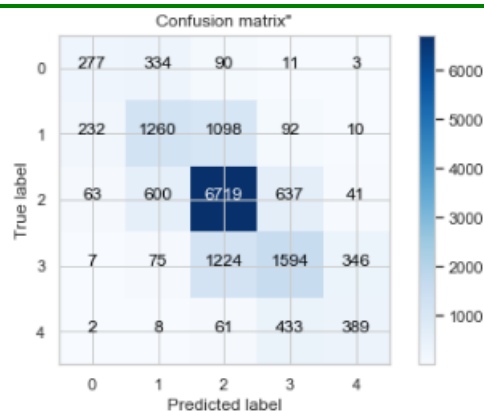
```
'loss': ['hinge', 'squared_hinge']], pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
scoring=None, verbose=3)
```

Most POSITIVE WORDS

```
Top and Bottom 10 of Most POSITIVE Learned Words
-2.0431 catching griffiths      1.9816 ll love
-2.0000 story sweet             1.9872 marvelous
-2.0000 movie highest           2.0000 enriched
-1.7886 wrong film             2.0969 magnificent
-1.7781 turns gripping         2.1072 masterfully
-1.6744 works superior         2.1368 captivating
-1.6485 talent outstanding     2.2931 masterful
-1.5967 directed highly        2.5583 masterpiece
-1.5557 grant lost             2.6241 perfection
-1.4911 captures luminous      2.6241 masterpeice
```

Most NEGATIVE WORDS

```
Top and Bottom 10 of Most NEGATIVE Learned Words
-2.0382 ca actor               1.8557 disappointment
-1.6275 bad painfully          1.8688 pileup cliches
-1.4928 quickly enters         1.9056 ludicrous film
-1.4164 acting ensemble        1.9147 hardly watchable
-1.3333 rejected astronomically 1.9536 thumbs
-1.3305 responsible did        1.9605 like trapped
-1.3200 great terrible         2.0000 execrable
-1.2601 makes passes           2.0000 snoozer
-1.2526 film thought           2.0997 disappointingly
-1.2493 sit crap               2.1561 unwatchable
```

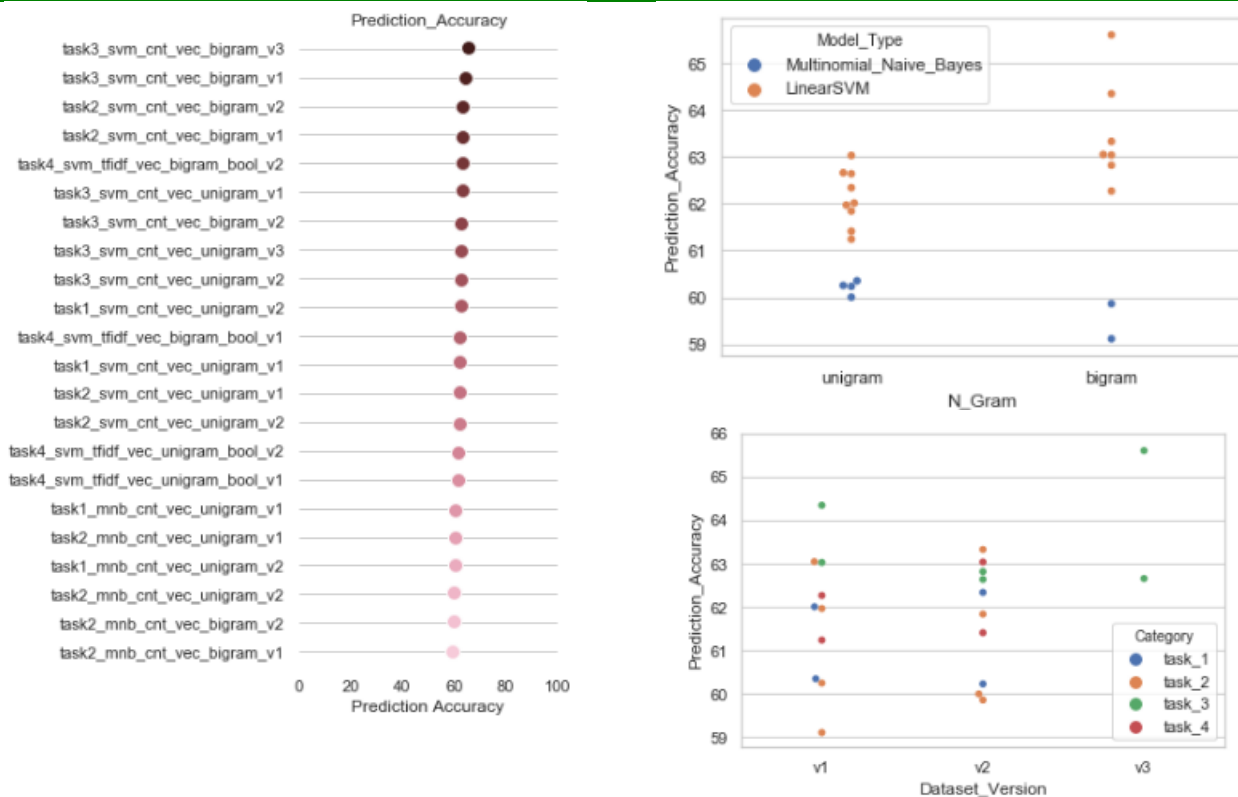
SVM Bigram Model Prediction Accuracy Results: **65.60%**

Classification Report:				
	precision	recall	f1-score	support
very_negative	0.48	0.39	0.43	715
negative	0.55	0.47	0.51	2692
neutral	0.73	0.83	0.78	8060
positive	0.58	0.49	0.53	3246
very_positive	0.49	0.44	0.46	893
micro avg	0.66	0.66	0.66	15606
macro avg	0.57	0.52	0.54	15606
weighted avg	0.64	0.66	0.65	15606

Confusion Matrix:

```
[[ 277  334   90   11    3]
 [ 232 1260 1098   92   10]
 [  63  600 6719  637   41]
 [   7   75 1224 1594  346]
 [   2    8   61  433  389]]
```

Homework Assignment 7 (week 7)

3.1.1.5.1 Model Accuracy Comparison Summary**3.1.1.6 Task 3 - Kaggle Submission Results**

Sentiment Analysis on Movie Reviews

"everything you 'd expect
-- but nothing more"

★★★★★

Classify the sentiment of sentences from the Rotten Tomatoes dataset
861 teams · 5 years ago

Overview Data Notebooks Discussion Leaderboard Rules Team My Submissions **Late Submission**

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
outputkaggle_submission_linearSVC_t...	just now	4 seconds	3 seconds	0.61367

Complete

4 Conclusions

Of the 22 variations of the two Classification models built for comparison, 20 of them scored above at or above 60% predicted accuracy rating on unseen movie reviews with a high of 65.61% by a LinearSVM bigram frequency count model. Comparing Multinomial Naive Bayes with Linear Support Vector Machine models, LinearSVM outperformed MNB in each of the trial tasks for both unigram and bigram vectorizers.

Both modeling classifiers struggled with labeling categories outside of neutral. The labeled dataset was overwhelmingly in favor of neutral sentiment classes. The task of labeling phrases on a scale of five values: negative, somewhat negative, neutral, somewhat positive, positive has many obstacles like sentence negation, sarcasm, terseness, language ambiguity, and many others making this task very challenging.

Additionally, there are labeled datasets that are used as gold standards for building and training models for baseline evaluation of accuracy. Starting with those datasets, then adding context specific domain knowledge into this dataset would greatly enhance the accuracy potential of these models.

Deep Learning for Sentiment Analysis is being expanded on. Below is an excerpt from NLP at Stanford where they describe new Recursive Neural Network techniques that are showing great progress in this very challenging space. - <https://nlp.stanford.edu/sentiment/>

" Most sentiment prediction systems work just by looking at words in isolation, giving positive points for positive words and negative points for negative words and then summing up these points. That way, the order of words is ignored and important information is lost. In contrast, our new deep learning model actually builds up a representation of whole sentences based on the sentence structure. It computes the sentiment based on how words compose the meaning of longer phrases. This way, the model is not as easily fooled as previous models. The underlying technology of this demo is based on a new type of Recursive Neural Network that builds on top of grammatical structures. "

5 Appendix: References

[1] Pang and L. Lee. 2005. *Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales*. In ACL, pages 115–124.

[2] *Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank*, Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Chris Manning, Andrew Ng, and Chris Potts. Conference on Empirical Methods in Natural Language Processing (EMNLP 2013).