TEXT MINING

Assignment # 5

RENU SINGH, AJINKYA RANE, JITENDRA PATEL

IDS 572: DATA MINING

Text mining is a process of deriving high quality information from the text. The focus of our project is to build a model that can classify movie reviews as positive or negative. We are using movie lens data set of 1000 positive and 1000 negative reviews.

QUESTION 1

Question 1. Read a few of the movie-reviews. If you were to classify manually, are there certain words/terms indicative of 'positive' and 'negative' that you would use? Provide lists ('dictionaries') of 'positive' and 'negative' terms. Now consider a manual classification process – you may count the number of 'positive' and 'negative' terms (matching your dictionaries) and classify based on majority count. How effective is this (accuracy?)?

From the collection of reviews, we have taken randomly 7 reviews (mix of positive and negative reviews) and after reading these reviews based on our judgment we have created a dictionary of positive and negative words. This dictionary will help us determine what document is what.

Dictionary:

Negative Words

DULL	EMBARRASSING
STINKS	HELL
BAD	SHODDY
DISASTER	DISASTROUS
AWFULLY	SUCKS
MONOTONOUS	DOWNSHIFTS
HORRIBLE	PAIN
PROSAIC	UNSUCCESSFULLY
YAWNING	MISADVENTURES
ANTI-CLIMATIC	DISTRACTING
MESS	CRAPINESS
DISGUSTED	PRETENDS
SAPPY	PATHETIC
LABORIOUS	PLOTLESSNESS
BLAND	

Positive Words

SUCCESS	WELL
LIKED	INTERESTING
JOY	BEST
AMAZING	UNIQUE
EXCITING	ENJOY
SMILING	APPEALING
GENUINELY	FUNNY

Then a random sample of another 20 document is taken (equal split of negative and positive review). Based on created dictionary of positive and negative words we figured that how many documents we could accurately predict.

<u>Following is the confusion matrix: -</u>

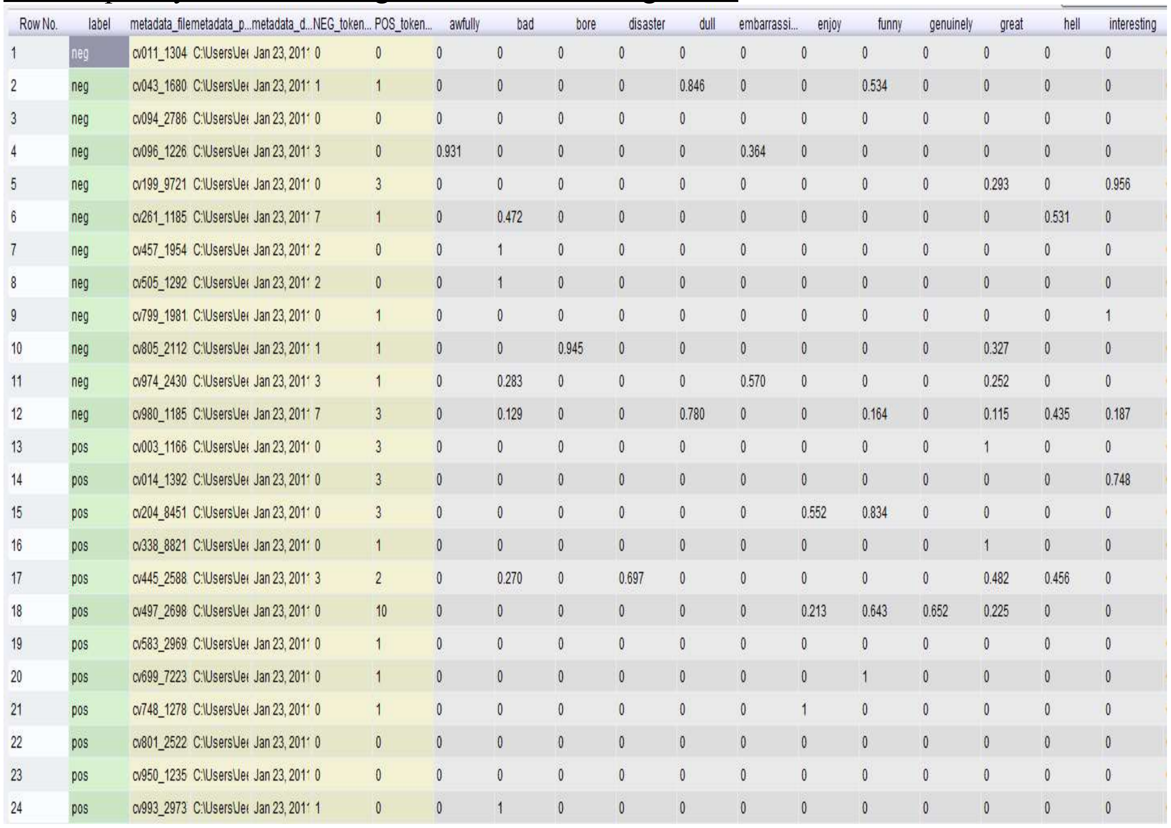
	PREDICTED POS	PREDICTED NEG	
Actual POS	8	2	80%
ACTUAL NEG	2	8	80%
	80%	80%	

<u>**Note:-**</u>

While doing manual mining process we have observed that, there can be cases where the Number of Positive & Negative tokens are equal. In that case we can adjust <u>**those by either predicting them as Positive or Negative based on the better accuracy**</u> we get with any particular choice.

<u>In our case there were 3 cases with equal tokens of opposite sentiments we have predicted them as **Positive** in order to get the better accuracy. Accuracy was way less, if these cases were predicted as negative.</u>

<u>The frequency of the term is given in the following table:</u>



Row No.	label	metadata_file	metadata_p...	metadata_d...	NEG_token...	POS_token...	awfully	bad	bore	disaster	dull	embarrassi...	enjoy	funny	genuinely	great	hell	interesting
1	neg	cv011_1304	C:\Users\Ue	Jan 23, 201	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	neg	cv043_1680	C:\Users\Ue	Jan 23, 201	1	1	0	0	0	0	0.846	0	0	0.534	0	0	0	0
3	neg	cv094_2786	C:\Users\Ue	Jan 23, 201	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	neg	cv096_1226	C:\Users\Ue	Jan 23, 201	3	0	0.931	0	0	0	0	0.364	0	0	0	0	0	0
5	neg	cv199_9721	C:\Users\Ue	Jan 23, 201	0	3	0	0	0	0	0	0	0	0	0	0.293	0	0.956
6	neg	cv261_1185	C:\Users\Ue	Jan 23, 201	7	1	0	0.472	0	0	0	0	0	0	0	0	0.531	0
7	neg	cv457_1954	C:\Users\Ue	Jan 23, 201	2	0	0	1	0	0	0	0	0	0	0	0	0	0
8	neg	cv505_1292	C:\Users\Ue	Jan 23, 201	2	0	0	1	0	0	0	0	0	0	0	0	0	0
9	neg	cv799_1981	C:\Users\Ue	Jan 23, 201	0	1	0	0	0	0	0	0	0	0	0	0	0	1
10	neg	cv805_2112	C:\Users\Ue	Jan 23, 201	1	1	0	0	0.945	0	0	0	0	0	0	0.327	0	0
11	neg	cv974_2430	C:\Users\Ue	Jan 23, 201	3	1	0	0.283	0	0	0	0.570	0	0	0	0.252	0	0
12	neg	cv980_1185	C:\Users\Ue	Jan 23, 201	7	3	0	0.129	0	0	0.780	0	0	0.164	0	0.115	0.435	0.187
13	pos	cv003_1166	C:\Users\Ue	Jan 23, 201	0	3	0	0	0	0	0	0	0	0	0	1	0	0
14	pos	cv014_1392	C:\Users\Ue	Jan 23, 201	0	3	0	0	0	0	0	0	0	0	0	0	0	0.748
15	pos	cv204_8451	C:\Users\Ue	Jan 23, 201	0	3	0	0	0	0	0	0	0.552	0.834	0	0	0	0
16	pos	cv338_8821	C:\Users\Ue	Jan 23, 201	0	1	0	0	0	0	0	0	0	0	0	1	0	0
17	pos	cv445_2588	C:\Users\Ue	Jan 23, 201	3	2	0	0.270	0	0.697	0	0	0	0	0	0.482	0.456	0
18	pos	cv497_2698	C:\Users\Ue	Jan 23, 201	0	10	0	0	0	0	0	0	0.213	0.643	0.652	0.225	0	0
19	pos	cv583_2969	C:\Users\Ue	Jan 23, 201	0	1	0	0	0	0	0	0	0	0	0	0	0	0
20	pos	cv699_7223	C:\Users\Ue	Jan 23, 201	0	1	0	0	0	0	0	0	0	1	0	0	0	0
21	pos	cv748_1278	C:\Users\Ue	Jan 23, 201	0	1	0	0	0	0	0	0	1	0	0	0	0	0
22	pos	cv801_2522	C:\Users\Ue	Jan 23, 201	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	pos	cv950_1235	C:\Users\Ue	Jan 23, 201	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	pos	cv993_2973	C:\Users\Ue	Jan 23, 201	1	0	0	1	0	0	0	0	0	0	0	0	0	0

QUESTION 2

The automated process includes following operators:

1. Tokenize: Tokenization creates a "bag of words" that are contained in a document. This allows us to do further filtering on a document.
2. Filter Stop-words (English)" After we have created bag of words we are ready to filter the documents. We want to filter certain words out of a document that don't really have any meaning to the document itself (such as articles a, an, the, etc.).
3. Filter by length of word: Also, we want to filter out any remaining words that are less than four characters.
4. Transform: After we filtered the bag of words by stop-words and length, we want to transform all of our words to lowercase since the same word would be counted differently if it was in uppercase vs. lowercase.
5. Stemming: the next preprocessing step is stemming. It is a method for collapsing distinct word forms. The main purpose of stemming is to reduce different grammatical forms / word forms of a word like its noun, adjective, verb, adverb etc. to its root form.

After applying the above operators, we found that total number of attributes are:

Without Stemming – **37285**
With Stemming – **23791**

Since the number of attributes are high we can further reduce the number by applying pruning parameter in the process document operator. Pruning will help in excluding too frequent and too infrequent words from the word list. After setting parameter that prune if word is in less than 1 percent of documents and if word is in more than 95% or more of documents.

Following results are observed after applying pruning:

Without stemming the total number of attributes after – **4218**
With Stemming total number of attributes reduce to **3562.**

QUESTION 3

(a) Develop and evaluate knn and naïve-Bayes models. What values of k work best? Evaluation is based on the confusion matrix – overall accuracy, accuracies on positive and negative, precision, specificity, sensitivity, etc.

For the document processing we have used the following four document matrix measures:

Term Frequency: The frequency of the term t in the document d is defined as the number of times that t occurs in d. It places the exact number of occurrences of a term in the intersection cell between the document and the word.

Binary term Occurrence: It places 0 in the intersection cell between the terms and documents if the word does not occur even once else it places 1. The number of occurrences in the documents is ignored.

Term Frequency: It places the relative frequency of the term in the document in the intersection cell of terms and documents. This matrix is calculated by dividing the number of occurrences of the term into the number of total words in that particular document.

TF-IDF: It stands for Term Frequency- Inverse Document Frequency. It is the mostly used weighted frequency measure in Text Mining. It calculates the numerical value that emphasizes both the frequency of the term in a document and the rareness of the same term in the collection of all the documents.

- We have split the training and testing data by 50:50. To train the model we have tired different parameters to observe the high prediction accuracy when tested on the testing data.

Naïve Bayes: -The basic idea is to estimate the probabilities of categories given a test document. The naive part of such a model is the assumption of word independence. So when the model was applied the first thing we observed was that if we use don't use pruning and include all terms, a noise is introduced, which reduces the accuracy rate. The accuracy of testing data without pruning is 67% and accuracy with pruning is 69%. Moreover, use of pruning also reduces the problem of over fit the accuracy with pruning on training data was 99.65% but with pruning it was reduced to 92.25%.

Accuracies across different measures defining document term matrix:

WITHOUT PRUNING	Training Performance	Validation Performance	Recall Neg	Recall Pos	Precision Negative	Precision Positive
TF-IDF	100	64.90	59.53	70.47	67.63	62.68
Term Frequency	100	66.20	64.83	67.62	67.48	64.97
Binary Term Occurrence	100	65.70	52.06	79.84	72.80	61.64
Term Occurrence	100	63.60	54.22	73.32	67.81	60.71

Pruning words based on min and max of the number of documents they appear in: we use pruning to eliminate too frequent and too infrequent terms from the data to remove noise. The parameters we have tried that fit best with this data.

WITH PRUNING: Parameters used were exclude terms that appear in less than 1% of documents and in more than 95% of document

WITH PRUNING	Training Performance	Validation Performance	Recall Neg	Recall Pos	Precision Negative	Precision Positive
TF-IDF	98.10	68.20	64.64	71.89	70.45	66.23
Term Frequency	97.40	67.70	63.85	71.69	70.04	65.67
Binary Term Occurrence	91.40	72.40	71.32	73.52	73.63	71.20
Term Occurrence	93.20	69.30	68.76	69.86	70.28	68.33

With results stated above we can say that pruning helps in getting a more stable model by reducing the difference between the training and testing performance.

Parameters used were exclude terms that appear in less than **5%** of documents and in more than **90%** of documents. The number of terms is now reduced to 909.

WITH PRUNING	Training Performance	Validation Performance	Recall neg	Recall pos	Precision Neg	Precision pos
Binary Term Occurrence	87.50	77.50	80.75	74.13	76.39	79.79
Term Frequency	86.90	74.80	77.41	72.10	74.20	75.48

When the pruning is less restricted we observed that it increases the accuracy of the model on the validation data making the model more stable. So we can conclude that not including too frequent and too infrequent term is essential in making a good model.

Stemming and without stemming: Now we will determine that like pruning does stemming also helps in building a good model. Following is the performance we get when we add stemming operator:

STEMMING	Training Performance	Validation Performance	Recall pos	Recall neg	Precision pos	Precision neg
Term Frequency	86.60	73.30	70.26	76.23	74.03	72.66
Binary Term Occurrence	83.80	75.10	71.60	78.60	76.99	73.46

We observed that stemming does not help much in improving the performance of the model but helps in reducing the problem of over fit. The gap between the training and validation performance is reduced when stemming is used.

POS Tags: - Part of speech tags are used to tag the tokens grammatically. Whether the selected token is adverb, noun, verb, etc. This tagging helps identifying the sentiment of the word according to placement and usage of the particular word. We will introduce this operator right before stemming operator in order to help reduce the process run time.

Part-of-speech tagging also known as grammatical tagging or word-category disambiguation is the process of assigning a word in the text corresponding to a particular part of speech like noun, verb, pronoun, preposition, adverb, adjective or other lexical class marker to each word in a sentence. The output is a single best POS tag for each word. We have taken the expression JJ.*|N.* would keep all adjectives and nouns. Since movie reviews express the opinions of the reviewer it might be important to keep adjective and noun.

Performance measure using POS tag operator: -

POS Tags	Training Performance	Validation Performance	Recall	Precision
Term Frequency	71.35	64.28	68.60 & 72.30	70.45 & 72.32
Binary Term Occurrence	72.20	66.50	69.18 & 75.10	72.75 & 71.72

While we use POS tags and measure the performance, it doesn't seem that the use of this operator doesn't help with improvement of the results. Thus we will avoid using POS tags for the process.

- **K-nearest neighbor: -** With the knowledge of how k-nn operator works? It doesn't seem to give the effective results. In k-nn method a particular word which is near to another related word it will be eliminated with the stemming operator itself. And other nearest word can happen to give the different meaning. We will try different parameters and check for the accuracies & recalls.

Accuracies across different measures defining document term matrix:

Using the different k values & shifting to different parameters. The best performance we got was at k=20. Though the performance does not differ much at different values of k that we tried are k=1, 3, 5, 10, 20, 30, 50. Following are the observations using k=20.

WITHOUT PRUNING	Training Performance	Validation Performance	Recall (pos. & neg.)	Precision (pos. & neg.)
TF-IDF	72.30	66.30	62.55 & 70.20	68.60 & 64.30
Term Frequency	74.70	66.90	57.84 & 76.33	71.78 & 63.50
Binary Term Occurrence	51.50	49.10	0.20 & 100	100 & 49.05
Term Occurrence	52.30	49.60	1.37 & 99.80	87.5 & 49.29

Pruning words based on min and max of the number of documents they appear in: we use pruning to eliminate too frequent and too infrequent terms from the data to remove noise. The parameters we have tried that fit best with this data.

WITH PRUNING: Parameters used were exclude terms that appear in less than 1% of documents and in more 95% of document

WITH PRUNING	Training Performance	Validation Performance	Recall (pos. & neg.)	Precision (pos. & neg.)
TF-IDF	72.30	66.10	60.98 & 71.43	68.96 & 63.75
Term Frequency	71.50	65.00	62.94 & 67.14	66.60 & 63.51

With results stated above we can say that pruning helps in getting a more stable model by reducing the difference between the training and testing performance. Also we can see that for K-nn the best performance we get is from Tf-idf and term frequency matrix, it is different for what we got with Naïve Bayes. We have got best performance on Naïve Bayes from binary term occurrence.

Parameters used were exclude terms that appear in less than **5%** of documents and in more than **90%** of documents. The number of terms is now reduced to 909.

When the pruning is less restricted we observed that it increases the accuracy of the model on the validation data making the model more stable. So we can conclude that not including too frequent and too infrequent term is essential in making a good model.

Stemming and without stemming: Now we will determine that like pruning, stemming also helps in building a good model. Following is the performance we get when we add stemming operator:

STEMMING	Training Performance	Validation Performance	Recall (pos. & neg.)	Precision (pos. & neg.)
TF-IDF	76.50	67.80	66.47 & 69.18	69.18 & 66.47
Term Occurrence	71.80	68.20	59.41 & 77.35	73.19 & 64.68

We observed that stemming does not help much in improving the performance of the model but helps in reducing the problem of over fit. The gap between the training and validation performance is reduced when stemming is used.

POS Tags: - Part of speech tags are used to tag the tokens grammatically. Whether the selected token is adverb, noun, verb, etc. This tagging helps identifying the sentiment of the word according to placement and usage of the particular word. We will introduce this operator right before stemming operator in order to help reduce the process run time.

Performance measure using POS tag operator: -

POS Tagging	Training Performance	Validation Performance	Recall (pos. & neg.)	Precision (pos. & neg.)
TF-IDF	69.60	64.10	60.78 & 67.55	66.10 & 62.34
Term Frequency	72.8	66.30	94.90 & 14.90	51.72&75.25

POS tagging in this case also doesn’t seem to work efficiently as compared to other parameter changes. So we will not consider POS tagging to get the results.

(b) For any one setting from above (for example, using tf-idf, stemmed terms, and using some reasonable pruning), build a support vector machine model. Try SVM with dot-kernel. How does the performance of the SVM model compare with k-nn and naïve-Bayes models? Does using a radial basis function kernel (nonlinear model) perform better? Also try a random forest model and compare performance. What do you conclude overall?

From the best combinations that we tried in question 3(a). We will use those parameters for the further performance evaluations using SVM & Random forest.

The parameters which worked best are:-
- With Term frequency
- With pruning of 10%-90%
- Using steming

SVM:-
By performing various parameter changes in the SVM operator. We found that the value of C is the most important parameter which changes the performance of the process. C value indicates whether the boundaries are smooth or strict. In our case it is observed that the stricter the boundaries better is the performance. We can observe it in the following table. In case of Radial kernel, the results are quite monotonous & not that effective and useful In radial kernel we tried using different values of kernel gamma at various levels and results in table below shows the monotonous performance of the operator.

Results of SVM using dot kernel & radial kernel:-

DOT kernel	Training Performance	Validation Performance	Recall (pos. & neg.)	Precision (pos. & neg.)
C=20	82.40	70.30	81.76 & 58.37	67.15 & 75.46
C=10	82.40	70.30	81.76 & 58.37	67.15 & 75.46
C=0.2	86.50	69.80	86.86 & 52.04	65.34 & 79.19
C=0.1	81.80	69.20	84.31 & 53.47	65.35 & 76.61
Radial kernel	Training Performance	Validation Performance	Recall (pos. & neg.)	Precision (pos. & neg.)
Kernel gamma=0.2	51.00	49.00	0 & 100	0 & 49
Kernel gamma=10	51.00	49.00	0 & 100	0 & 49

Kernel gamma=20	51.00	49.00	0 & 100	0 & 49

From the above tables we can say that computing the dot products using kernel functions seems to perform better to radial kernel function. It seems that simple linear kernel is a better choice and we can say that linear models perform better in this case. Radial kernel function will be suitable for text mining where linear separation is not possible.It is quite clear from the table above with SVM operator & dot kernel performs the best at the value C=0.1. We will select the SVM with dot kernel at C=0.1 as the best model.

<u>Random forest: -</u>

While using the random forest operator, we got the best performance at,
No. of trees =50
Tree depth=20
Without pruning

Results of Random forest: -

Random Forest	Training Performance	Validation Performance	Recall (pos. & neg.)	Precision (pos. & neg.)
Number of trees=100	66.70	56.40	17.06 & 97.35	87.00 & 53.00
Number of trees=50	64.50	58.0	21.37 & 96.12	85.16 & 54.50
Number of tree=200	72.20	60.40	24.71 & 97.55	91.30 55.45

From the table above results we can see that the random forest operator doesn't seem to give the best recall & precision. The problem with the random forest can be, creating a group of words having the similar meaning & similar sentiments. This affects the recalls of the process.

While comparing these two operators with naïve bayes & knn, we can see that Naïve bayes proves to be the best model as it is giving the higher accuracy values & the difference between accuracies over training & validation sets is also less. Recalls and precisions of the naïve bayes are giving the highest values as well.

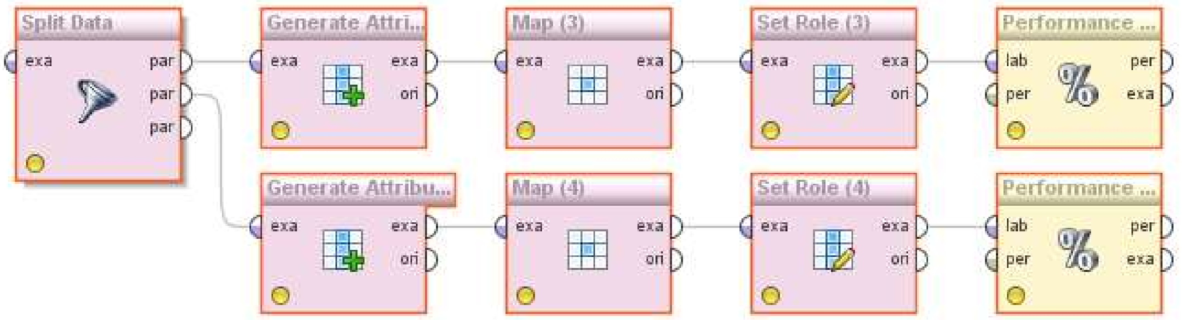
(c) Use the Harvard-IV dictionary of positive and negative terms - Filter Tokens by Content to keep only these terms. How many terms do you get? (Note: for this, the document terms should not be stemmed, since they would otherwise not match the unstemmed dictionary terms. Does it make sense to use pruning of terms or POS tags here?). Use just the count of positive and negative terms to classify sentiment of reviews. How do you do this, and what performance do youfind?
Develop and evaluate a naïve-Bayes, a SVM (either with dot kernel or other kernel based on what you found to perform better in part (b) above), and a random forest model, using only the positive and negative words.

In order to use the Harvard Dictionary of positive & negative words we will organize the process in following manner: -

- Tokenize
- Filter Stop words
- Filter tokens by length
- Transform cases
- Filter POS & NEG words
- Extract tokens number

We will eliminate the use of "Stemming" operator while using Harvard dictionary operator because stemming process might change the terms from its original form. Harvard dictionary contains about 3000 words and we want that as many words should match from the documents to the words from dictionary.

To create a confusion matrix based on positive & negative tokens we will use the following operators: -



Split data: - This operator will help us splitting the data in the segments we require.

Generate attribute: - With help of this operator we will create an attribute "PREDICTION" & set a logic "if (NEG_token_number>POS_token_number, "0", "1"). This logic will help us for predictions based on number of tokens.

MAP: - This operator will help us converting all (0→neg) & (1→pos) in PREDICTION attribute. To match the format of previous label for prediction.

Set Role: - This operator will help us setting the attribute "PREDICTION" which we generated earlier as prediction. This will help us create confusion matrix.

Performance (Classification): - This operator will help us evaluate performance of training & Validation data sets.

As it is mentioned earlier that we set prediction parameters as higher the NEG tokens the sentiment of document is neg & vice versa. The question arises is what about the documents with equal negative & positive tokens??

We will use two techniques and compared those two with each other & see which gives better performance: -

1) We will predict all documents with equal positive & neg tokens → POS
2) We will predict all documents with equal positive & neg tokens → NEG

We will test these data sets without using the pruning parameter as it will improve the number of terms to match with HARVARD dictionary & ultimately improve the reliability of process. We have not included POS tags because it is highly unlikely that the single word will be present in the dictionary if different senses. So to save the time in the process we have excluded POS tags form the process when Harvard dictionary is used.

Accuracy across the training & validation data sets: -

With number of attributes =**2425**

	Training data	Validation data	Recall	Precision
Equal tokens=POS	60.7	62.5	51.84 & 72.75	61.62 & 64.63
Equal tokens=NEG	61.7	64.1	69.02 & 59.98	63.65 & 64.65

We can see that use of Harvard dictionary indeed gives a better performance in terms of reliability of performance when compared with Random forest and SVM (Radial Kernel). But when compared with Naïve Bayes and KNN it does not give better performance in terms of accuracy.

(d) Next, use Wordnet and Senti-Wordnet to determine the sentiment polarity of the reviews. Provide a description of how this process works – i.e. how WordNet and Senti-Wordnet operates (there are various resources on the web on this; please cite your references, and please remember that we cannot copy-paste without quotes...! Ask Sid if you have any questions about this). Use the Senti-WordNet sentiment scores to classify the reviews. What performance do you find? How does performance in (a) - (d) above compare? What can you conclude? Does the smaller list of 'sentiment' words provide similar performance levels as models built on a broader set of terms? How does the performance of Senti-WordNet compare? Does it help to include Senti-WordNet scores in a model with sentiment-words or with the broader set of terms? Which is your 'best' model?

"WordNet is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. WordNet interlinks not just word forms—strings of letters—but specific senses of words. As a result, words that are found in close proximity to one another in the network are semantically disambiguated. WordNet labels the semantic relations among words, whereas the groupings of words in a thesaurus does not follow any explicit pattern other than meaning similarity".

"SENTIWORDNET is the result of the automatic annotation of all the synsets of WORDNET according to the notions of "positivity", "negativity", and "neutrality". Each synset s is associated to three numerical scores Pos(s), Neg(s), and Obj(s) which indicate how positive, negative, and "objective" (i.e., neutral) the terms contained in the synset are. Different senses of the same term may thus have different opinion-related properties.".

We will first provide wordnet 3.0 as a source to extract words & then match the sentiments & provide sentiments score for each word.

The process hierarchy will be: -

- Tokenize
- Filter stop words
- Filter words by length
- Transform cases
- Open Wordnet
- Stem- Wordnet
- Extract sentiment

Split data: - This operator will help us splitting the data in the segments we require.
Generate attribute: - With help of this operator we will create an attribute "PREDICTION" & set a logic "if (sentiment>0, "1", "0"). This logic will help us for predictions based on number of tokens.
MAP: - This operator will help us converting all (0→neg) & (1→pos) in PREDICTION attribute. To match the format of previous label for prediction.

Set Role: - This operator will help us setting the attribute "PREDICTION" which we generated earlier as prediction. This will help us create confusion matrix.

Performance (Classification): - This operator will help us evaluate performance of training & Validation data sets.

With number of attributes =**19860**

	Training data	Validation data	Recall	Precision
Sentiment 0=POS	57.2	62.7	89.22 & 35.1	58.86 & 75.77
Sentiment 0=NEG	57.2	62.7	89.22 & 35.1	58.86 & 75.77

After using wordnet and senti word-net we get the above performance, so when we compare this performance with our best model we obtained in the 3(a), we observe that wordnet do not do a very good job in making predictions accurately when compared with Naïve bayes. The accuracy over validation data of Naïve bayes.

Note:-

- We also tried using pruning parameter for HARVARD & WORDNET approach. It seems that, pruning doesn't help improve the performance but it can help reduce the size of document term matrix, which will help reduce process time and money involved

- We also tried applying different models (i.e. k-nn, SVM, Naïve bayes, etc,) over document term matrix which we got from WORDET & HARVARD operator output. All models are giving almost same performances and it is less than that of actual HARVARD & WORDNET model.

  Training set performance= 60.23 %
  Validation set performance = 56.64 %

Best Model:-

NaiveBayes	Training Performance	Validation Performance	Recall Neg	Recall Pos	Precision Negative	Precision Positive
Binary Term Occurrence	83.80	75.10	71.60	78.60	76.99	73.46

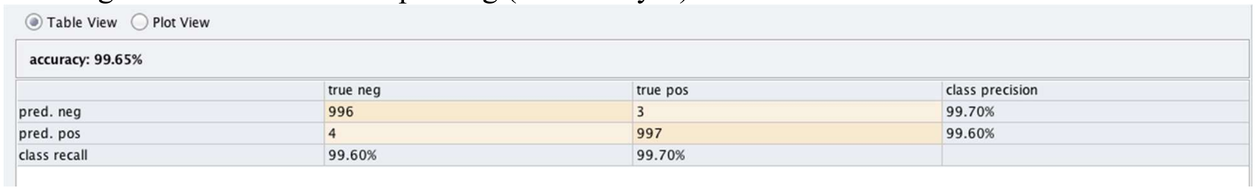
The smaller set of words give better performance than broader set of words. Boarder set of words will increase the complexity of the model increasing the problem of overfit. Our best model is Naïve bayes built with following parameter:
Binary Term occurrence
Pruning 5%-95%
Tokenization
Filter Stopwords
Filter by length
Transform
Stemming

We have chosen this model by taking into account the accuracy, reliability and stability of the model. For accuracy we have included precision and recall precision of the models too. Although the performance from SVM (dot kernel) is comparable to Naïve Bayes when it comes to accuracy but the recall and precision is better in Naïve Bayes.

Appendix:

Term frequency

Training Performance without pruning (Naïve Bayes)

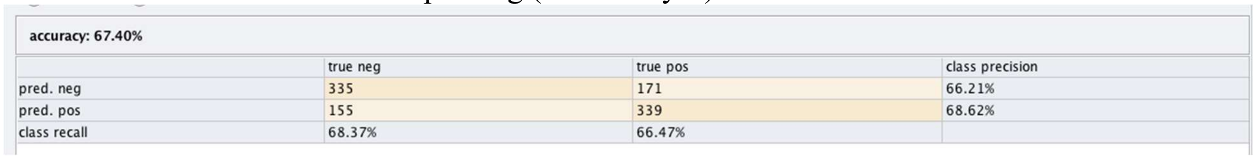


[✓ Table View] Plot View

accuracy: 99.65%

	true neg	true pos	class precision
pred. neg	996	3	99.70%
pred. pos	4	997	99.60%
class recall	99.60%	99.70%	

Validation Performance without pruning (Naïve Bayes)

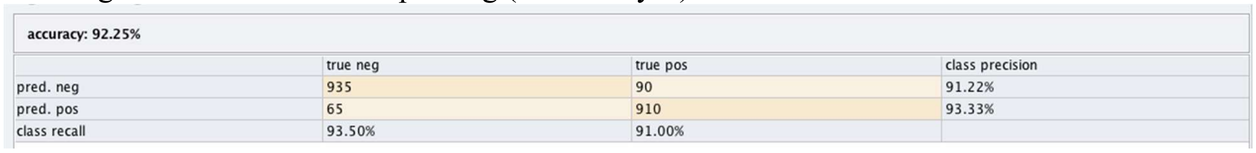


accuracy: 67.40%

	true neg	true pos	class precision
pred. neg	335	171	66.21%
pred. pos	155	339	68.62%
class recall	68.37%	66.47%	

Binary term occurrences:-

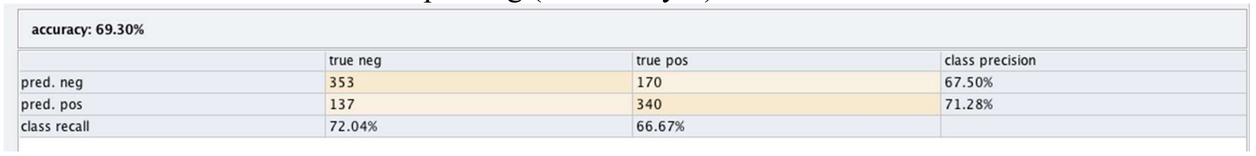
Training Performance without pruning (Naïve Bayes)



accuracy: 92.25%

	true neg	true pos	class precision
pred. neg	935	90	91.22%
pred. pos	65	910	93.33%
class recall	93.50%	91.00%	

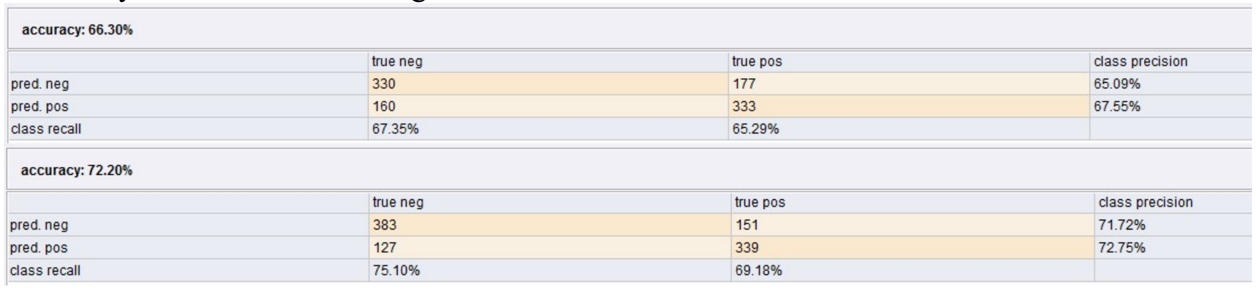
Validation Performance without pruning (Naïve Bayes)



accuracy: 69.30%

	true neg	true pos	class precision
pred. neg	353	170	67.50%
pred. pos	137	340	71.28%
class recall	72.04%	66.67%	

Naïve bayes model with POS tags



accuracy: 66.30%

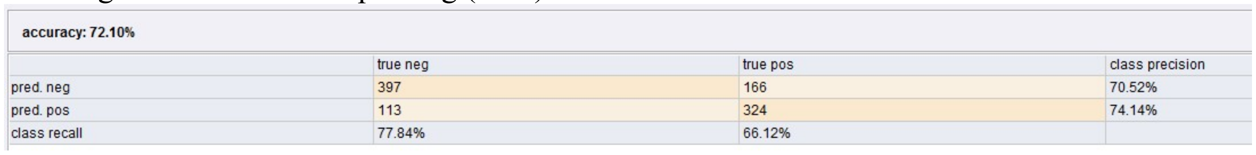
	true neg	true pos	class precision
pred. neg	330	177	65.09%
pred. pos	160	333	67.55%
class recall	67.35%	65.29%	

accuracy: 72.20%

	true neg	true pos	class precision
pred. neg	383	151	71.72%
pred. pos	127	339	72.75%
class recall	75.10%	69.18%	

TF-IDF

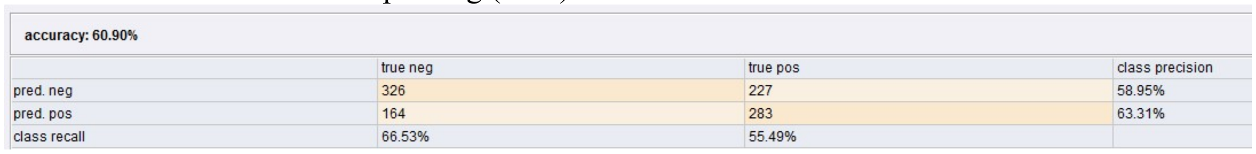
Training Performance with pruning (k-nn)



accuracy: 72.10%

	true neg	true pos	class precision
pred. neg	397	166	70.52%
pred. pos	113	324	74.14%
class recall	77.84%	66.12%	

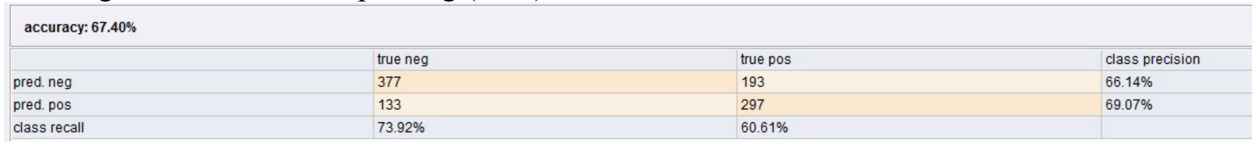
Validation Performance with pruning (k-nn)



accuracy: 60.90%

	true neg	true pos	class precision
pred. neg	326	227	58.95%
pred. pos	164	283	63.31%
class recall	66.53%	55.49%	

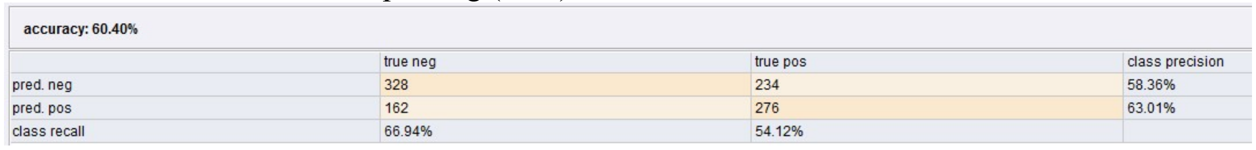
Term Frequency
Training Performance with pruning (k-nn)



accuracy: 67.40%

	true neg	true pos	class precision
pred. neg	377	193	66.14%
pred. pos	133	297	69.07%
class recall	73.92%	60.61%	

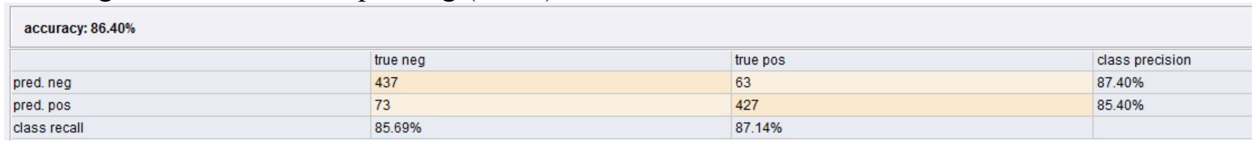
Validation Performance with pruning (k-nn)



accuracy: 60.40%

	true neg	true pos	class precision
pred. neg	328	234	58.36%
pred. pos	162	276	63.01%
class recall	66.94%	54.12%	

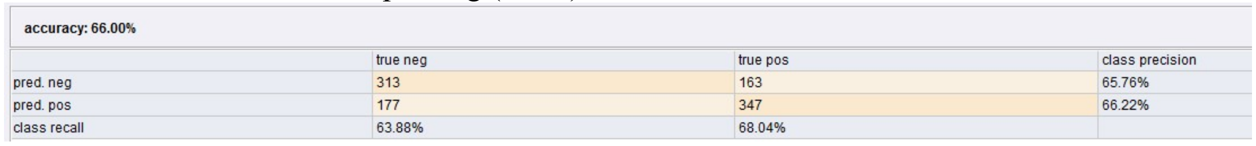
Term Frequency
Training Performance with pruning (SVM)



accuracy: 86.40%

	true neg	true pos	class precision
pred. neg	437	63	87.40%
pred. pos	73	427	85.40%
class recall	85.69%	87.14%	

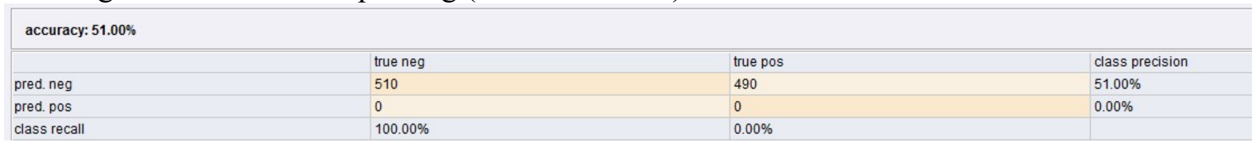
Validation Performance with pruning (SVM)



accuracy: 66.00%

	true neg	true pos	class precision
pred. neg	313	163	65.76%
pred. pos	177	347	66.22%
class recall	63.88%	68.04%	

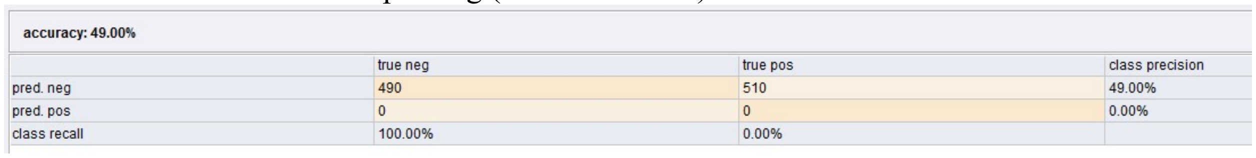
Term Frequency
Training Performance with pruning (Random Forest)



accuracy: 51.00%

	true neg	true pos	class precision
pred. neg	510	490	51.00%
pred. pos	0	0	0.00%
class recall	100.00%	0.00%	

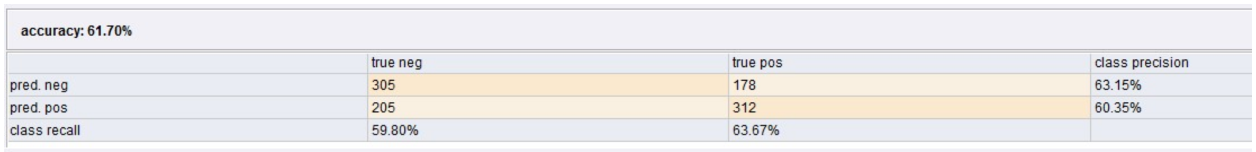
Validation Performance with pruning (Random Forest)



accuracy: 49.00%

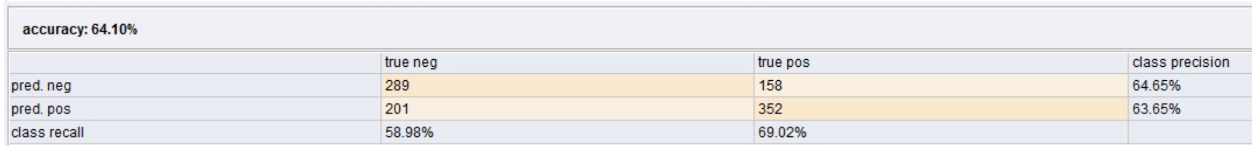
	true neg	true pos	class precision
pred. neg	490	510	49.00%
pred. pos	0	0	0.00%
class recall	100.00%	0.00%	

HARVARD with equal tokens as negative
Training & Validation sets:-



accuracy: 61.70%

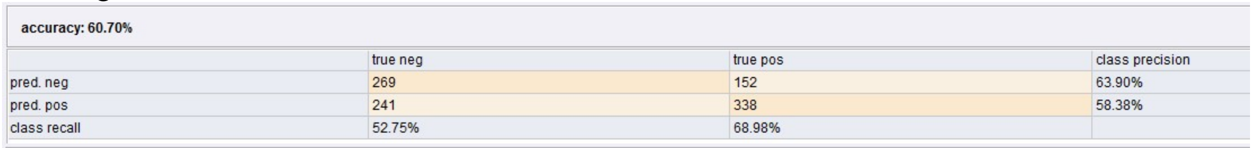
	true neg	true pos	class precision
pred. neg	305	178	63.15%
pred. pos	205	312	60.35%
class recall	59.80%	63.67%	



accuracy: 64.10%

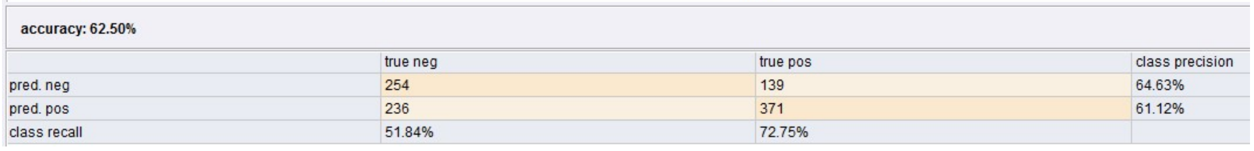
	true neg	true pos	class precision
pred. neg	289	158	64.65%
pred. pos	201	352	63.65%
class recall	58.98%	69.02%	

HARVARD with equal tokens as positive
Training & Validation sets:-



accuracy: 60.70%

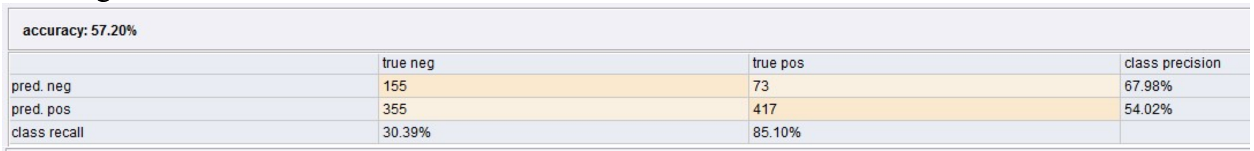
	true neg	true pos	class precision
pred. neg	269	152	63.90%
pred. pos	241	338	58.38%
class recall	52.75%	68.98%	



accuracy: 62.50%

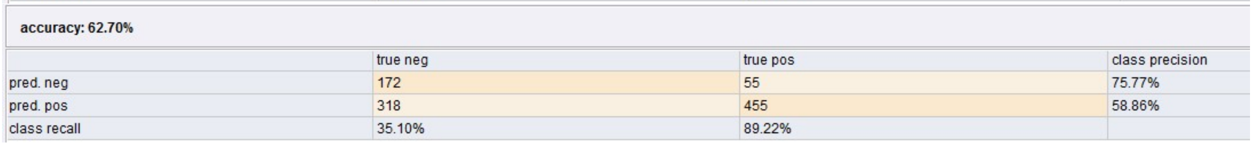
	true neg	true pos	class precision
pred. neg	254	139	64.63%
pred. pos	236	371	61.12%
class recall	51.84%	72.75%	

Wordnet with "O" sentiment as negative
Training & Validation sets:-



accuracy: 57.20%

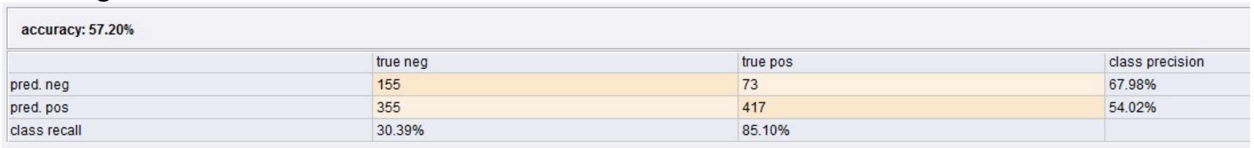
	true neg	true pos	class precision
pred. neg	155	73	67.98%
pred. pos	355	417	54.02%
class recall	30.39%	85.10%	



accuracy: 62.70%

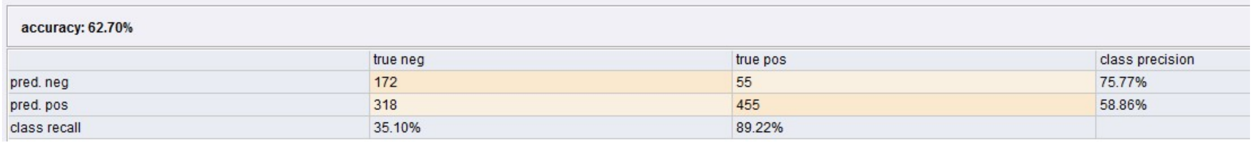
	true neg	true pos	class precision
pred. neg	172	55	75.77%
pred. pos	318	455	58.86%
class recall	35.10%	89.22%	

Wordnet with "O" sentiment as positive
Training & Validation sets:-



accuracy: 57.20%

	true neg	true pos	class precision
pred. neg	155	73	67.98%
pred. pos	355	417	54.02%
class recall	30.39%	85.10%	



accuracy: 62.70%

	true neg	true pos	class precision
pred. neg	172	55	75.77%
pred. pos	318	455	58.86%
class recall	35.10%	89.22%	