LIS 590TXL Final Project

Emotion detection from text using different supervised learning approaches

---

1. Introduction

With the rapid growth of Internet, people now not only communicate with each other by speech but also interact in textual form. The large collections of documents/texts can provide various knowledge and information and many fields such as Natural Language Processing (NLP), Affective Computing have a keen interest in these collections of texts. And one of the growing areas of interest for textual analysis is the detection of emotion. Emotion detection has a great application in different domains such as e-mail (Mohammad and Yang, 2011), suicide prevention [Desmet, 2013], microblog [Roberts, 2012] by analyzing users' emotion through his/her text (Eg: Tweet, blog, notes) to understand their feelings about particular topics, programs, services, products or even their characteristics.

Emotion detection has been implemented in various studies using different machine learning approaches including both supervised and unsupervised learning. Those studies inspire me to find out which is the best and most appropriate approach for us to detect emotions in text. Hence, in the final project, I would like to examine different methods of supervised learning classification on several datasets to find out the most plausible method for emotion detection.

2. Problem statements

My project attempts to answer two questions:

a. Compare the emotion classification performance over 3 different datasets of different topics (fairy tale stories, news/headlines and people responses over various situations), using different supervised methods including Naive Bayes, Logistic Regression, SVM and Decision Tree. Which method gives the best performance over all datasets?

b. Can Dimensionality Reduction (SVD) help in improving the classification results before applying classifier?

Also an extra experiment I would like to implement in the project is to find out the emotions expressed in the stories people posted on Human of New York using the best classification I get from my comparison.

3. Related Work

3.1. Emotion models:

There are two popular models used to represent and measure emotions including emotional categories and emotional dimensions [Canales, 2014]. The categorical model divides emotions into distinct emotion classes. One of the most influential models among emotional categories is Ekman's basic emotion model which includes 6 basics emotions: anger, disgust, fear, happiness, sadness and surprise [Ekman, 1999]. Dimensional model in contrast represents emotions in a quantitative form using multidimensional scaling such as 2-dimentions (valence and arousal) or

3 dimensions (valence, arousal and dominance). Despite the two emotional models, categorical model is the most commonly used due to its simplicity and familiarity [Kim, 2011]

3.2. Computational approach for emotion detection

Machine learning techniques have been widely used for emotion detection including two approaches - supervised and unsupervised learning.

Supervised learning approaches are based on a labelled training data to train the data for emotion detection. By analyzing the features of training dataset, it determines the mapping function which is used to predict the emotion classes of required test data. Various supervised machine learning classification techniques have been applied to automatically recognize emotion in text, such as Naïve Bayesian, Support Vector Machines (SVM), KNN, Decision Tree... However, the training dataset requires a large corpus/collection of texts and the process of labeling with emotional classes for such large collection is time consuming which becomes one of the most disadvantages of supervised learning approaches. Besides, feature selection is significant and important to achieve good performance in text classification and to get a good feature set that works well with specific classification method is not an easy task [Kim, 2011].

In contrast to supervised learning approaches, unsupervised learning does not require labeled data but use only the predicted variables to learn about hidden structure patterns in the unlabeled data to build models for emotion classification [Kim, 2011].

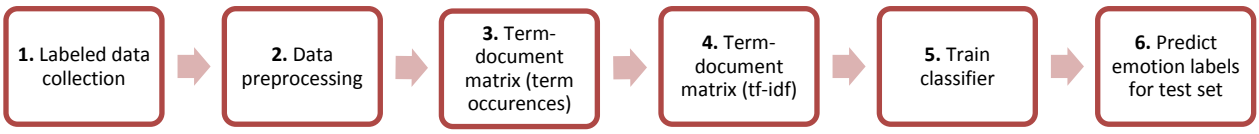
Despite the listed disadvantages of supervised learning approach, it is more commonly used in emotion detection due to its better results than unsupervised learning [Kim, 2011].

In a study of detecting emotions in Twitter Messages by Hasan 2014, they compared the accuracy of several supervised learning algorithms such as SVM, Naive Bayes, KNN, and Decision Tree for classifying the moods of Twitter messages and were able to achieve high classification accuracy of above 90%.

Motivated by this study, in the final project, I also want to compare the accuracy among different supervised learning algorithms (Naïve Bayes, Logistic Regression, SVM, Decision Tree) over three emotion-labeled datasets (Cecilia's Affected Data, SemEval 2007 and ISEAR)

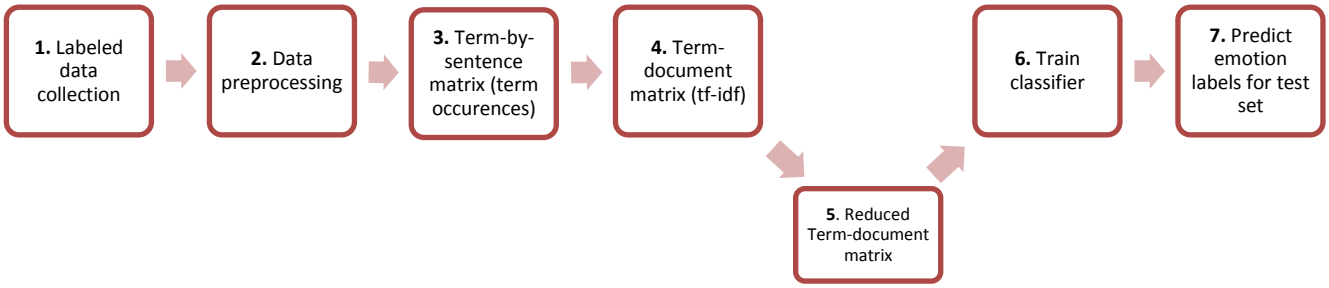
It is also advised that in order to perform machine learning on text documents, we first need to convert the text content into numerical feature vectors (http://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html). Hence, for feature selection, I use term frequency/inversed document frequency (tf-idf) to build the relationships between a set of documents and the terms mentioned in these documents. Based on this feature, I can train a classifier to predict the category of emotion of test data.

The overview of categorical emotion classification system is described as below:



However, the feature vector in the term-document matrix stores a lot of 0 values due to a large number of vocabularies available in the corpus (more than 10k) but just a few of them are available in our texts. This might cause computational problems. To reduce the computational time as well as noises in data, we can use dimensional reduction techniques before applying supervised classifier. One of the earliest and common techniques to transform our high-dimensional term-document matrices into lower dimensionality is the unsupervised approach - Latent Semantic Latent Analysis (LSA). Using LSA, we can reduce the dimension of feature vectors of textual data based on Singular Vector Decomposition (SVD) – a reliable method of

matrix decomposition. This technique was used in the study of Calvo, R. A., & Mac Kim, S. (2013) about Emotions in Text for categorical models.
Categorical emotion classification system with dimensional reduction LSA is shown as following:



4. Data

There are 3 datasets I use for the project: SemEval 2007, ISEAR and Cecilia's Affected Data

4.1. SemEval 2007 (http://web.eecs.umich.edu/~mihalcea/downloads.html)

This is the "Affective Text" from the SemEval 2007 task. It contains 1250 news headlines from newspaper and news websites such as such as Google news, CNN; and is labeled into six basic emotion classes - anger, disgust, fear, joy, sadness and surprise. Each sentence (specified by an id from 1 to 1250) can have multiple tags corresponding to multiple emotions the sentence can express with scoring for intensity from 0 to 100.
Dataset format:

- "Document ID – Document Content" is stored in *affectivetext_*.xml* file
  Eg:

```
<corpus task="affective text">
<instance id="500">Test to predict breast cancer relapse is
approved</instance>
<instance id="501">Two Hussein allies are hanged, Iraqi official
says</instance>
<instance id="502">Sights and sounds from CES</instance>
<instance id="503">Schuey sees Ferrari unveil new car</instance>
…
</corpus task>
```

- "Document ID – Emotion Details" is stored in the emotion annotations file *affectivetext_*.emotions.gold* and follows the format: [*id] [anger] [disgust] [fear] [joy] [sadness] [surprise]*
  Eg:

```
500 0 0 15 38 9 11
501 24 26 16 13 38 5
502 0 0 0 17 0 4
503 0 0 46 0 31
```

To store this dataset in our system, I use two dataframe.

- One dataframe stores the document id – document details mapping with two columns **id** and **content**
  Eg:

id	content
500	Test to predict breast cancer relapse is approved
501	Two Hussein allies are hanged, Iraqi official says

- The other dataframe stores the document id – emotion label with two columns id and sentiment.

As anger and disgust are very close expressions, I will sum them up into one group “Anger-Aisgust”

Besides, to simplify the dataset, I assign the document to emotion with highest score.

For example: `501 24 26 16 13 38 5`

I consider document #501 expresses emotion corresponding to the highest score 38 which is sadness.

4.2. ISEAR (http://emotion-research.net/toolbox/toolboxdatabase.2006-10-13.2581092615)

This dataset is collected by a large group of psychologist all over the world during the 1990s for the ISEAR project. The data was based on student responds (both psychologists and non-psychologists) on different situations they had experienced for all of 7 major emotions (joy, fear, anger, sadness, disgust, shame, and guilt). The final data set thus contained reports on seven emotions each by close to 3000 respondents in 37 countries on all 5 continents.

The data is already saved in csv format *ISEARdata.csv* with 2 columns sentiment and content. We can load it directly using pandas.read_csv().

joy	[On days when I feel close to my partner and other friends. Ã¡ When I feel at peace with myself and also experience a close Ã¡ contact with people whom I regard greatly.]
fear	Every time I imagine that someone I love or I could contact a Ã¡ serious illness, even death.
anger	When I had been obviously unjustly treated and had no possibility Ã¡ of elucidating this.

Also I drop records tagged with uncommon emotions “shame” and “guilt” as well as combine “anger” and “disgust” into “anger-disgust”.

4.3. The Affect data by Cecilia Ovesdotter Alm

(http://people.rc.rit.edu/~coagla/affectdata/index.html)

The dataset was collected by Cecilia Ovesdotter Alm for her dissertation research based on the Fairy Tale corpus of approximately 185 children stories by Grimms’, Andersen’s and B.Potter’s. The documents in dataset were annotated with 5 basic emotions Angry-Disgusted (code: 2), Fearful (code: 3), Happy (code: 4), Sad (code: 6), and Surprised (code: 7)

The dataset consists of 3 files *GrimmsAll4labsagree.txt, hcand460All4labsagree.txt and Potter167All4labsagree.txt* with format:

[story name]

[sentence-id-in-story]@[label-code]@[sentence].

For example:

```
agree-sent/106_the_poor_millers_boy_and_the_cat.agree
8@7@He looked around on every side and exclaimed, "Oh, heavens, where am I?"
9@6@Then he got up and clambered out of the cave, went into the forest, and
thought, "Here I am quite alone and deserted, how shall I obtain a horse now?"
agree-sent/114_the_cunning_little_tailor.agree
17@4@The little tailor did not let himself be frightened away, but was quite
delighted, and said, "Boldly ventured is half won."
24@2@"Eh!" thought he, "what a stupid blockhead I am!
```

To load the data, I initialize an empty dataframe, then load each line of the file, split them using delimiter ‘@’ and save into a temporary dataframe with 3 columns – **id, sentiment_code** and **content**; finally, append the temp dataframe to the original dataframe.

With the dataframe containing all data from dataset, I further drop the records having no sentiment_code (it’s the story name line), then map the sentiment_code to its corresponding

sentiment and store in a new column of the dataframe **"sentiment"** (to make the sentiment name consistent in all 3 datasets, I convert the name for this dataset a bit such as Angry-Disgusted -> anger-disgust, fearful -> fear, happy -> joy, sad -> sadness, surprised -> surprise)

Further data cleaning is required for all 3 datasets as there are several meaningless special characters which is done in function ***contentCleaning(df)***. This function requires a dataframe as parameter.
Basically, this function will convert sentence into lowercase form and check whether the first and last characters in each token of sentence contain any special characters; if yes, remove those characters until no special characters at first and last positions in a token. Stoplist and lemmatization is also applied for tokens. Although punctuations such as exclamation could help express strong emotion (could be extremely happy, surprise, angry…), it requires more work to analyze this feature; hence, I ignore this feature for now and focus on word only in this project.

```
#Function for data cleaning for all datasets
def contentCleaning(df):
    df=df[['content','sentiment']].drop_duplicates()
    filtered_content=df['content']
    filtered_content=[row.lower() for row in filtered_content]
    for i in range(0,len(filtered_content)):
        temp=""
        for w in filtered_content[i].split():
            while (len(w)>0 and (not re.match("[a-z]",w[len(w)-1]))):
                w=w.replace(w[len(w)-1],"")
            while (len(w)>0 and (not re.match("[a-z]",w[0]))):
                w=w.replace(w[0],"")
            if (w.find("http")!=-1):
                w=""
            if w not in stoplist:
                w=lem.lemmatize(w)
                temp=temp+" "+w
            filtered_content[i]=temp
    df['refined_content']=filtered_content
    return df
```

Overall, we have the summary of sentiment distribution in the 3 datasets with their dataframe notations as following:

sentiment	Celicia's Affect Data (df1)	SemEval 2007 (df2)	ISEAR (df3)
anger-disgust	217	211	2131
fear	166	166	1071
joy	444	438	1064
sadness	264	240	1038
surprise	114	220	0

In the final project, I will use 80% of the dataset for training and 20% for testing.

5. Experiments and Findings

5.1. Experiments:

There are 3 experiments implemented in the project.

a. ***<u>Experiment 1:</u> Compare the classification performance using 4 different supervised learning methods: Naïve Bayes, Logistic Regression, SVM and Decision Tree over 3 datasets.***

For each dataset, I use 4/5 data for training and 1/5 for testing. I create 4 models to fit the training dataset. The process of vectorizer (bag of word creation) => transformer (term-document matrix using tfidf transformation) => classifier is implemented by building a Pipeline. 4 Pipelines are created corresponding to 4 different classifiers.

Example of a classifier pipeline using Naïve Bayes classification as following:

```
text_clf= Pipeline([('vect', CountVectorizer()),
          ('tfidf', TfidfTransformer(norm='l2',use_idf=True,smooth_idf=True,
          sublinear_tf=True)),
          ('naive_clf', MultinomialNB())
          ])
```

In the project, I use a function to fit model which has 3 input parameters:

- dataset to use (df) in Dataframe format,
- classification number (clf_num) in integer to specify which classifier to apply and
- joblib filename (.pkl) to store fitted classification object so that we can load it easily in the future.

```
def fitModel(data,clf_number,modelname):
    train=int(len(data)/5*4)
    train_set=data['refined_content'].iloc[0:train]
    train_label=data['sentiment'].iloc[0:train]
    if clf_number==1: #using Naive Bayes classification
        text_clf= Pipeline([('vect', CountVectorizer()),
                            ('tfidf', TfidfTransformer(norm='l2', use_idf=True, smooth_idf=True, sublinear_tf=True)),
                            ('naive_clf', MultinomialNB())
                           ])
    elif clf_number==2: #using Logistic Regression
        text_clf= Pipeline([('vect', CountVectorizer()),
                            ('tfidf', TfidfTransformer(norm='l2', use_idf=True, smooth_idf=True, sublinear_tf=True)),
                            ('logit_clf', LogisticRegression())
                           ])
    elif clf_number==3: #using SVM classification
        text_clf= Pipeline([('vect', CountVectorizer()),
                            ('tfidf', TfidfTransformer(norm='l2', use_idf=True, smooth_idf=True, sublinear_tf=True)),
                            ('svm_clf', SGDClassifier(loss='hinge', penalty='l2',alpha=1e-3, n_iter=5, random_state=42))
                           ])
    elif clf_number==4: #using Decision Tree
        text_clf= Pipeline([('vect', CountVectorizer()),
                            ('tfidf', TfidfTransformer(norm='l2', use_idf=True, smooth_idf=True, sublinear_tf=True)),
                            ('tree_clf', DecisionTreeClassifier(random_state=0))
                           ])
    else:
        print("Please enter a valid classification number(clf_num) (1:Naive Bayes,2: Logistic Regression,3:SVM or 4:Decision Tree
        return
    _ =text_clf.fit(train_set,train_label)
    _ =joblib.dump(text_clf, modelname)
```

To fit model using training set from dataset 1 (df1) and Logistic Regression, we can call the function as: `fitModel(df1,2,"df1_logit_2.pkl")`

Finally, to get the classification results, simply load classifier from the saved joblib and use that classifier to predict the result of test data, compare with the correct labels to get the accuracy. Besides the normal classifier accuracy, we also evaluate the classification results based on metrics classification report and confusion matrix for details.

```
#job_list=["df1_naive_1.pkl","df1_logit_2.pkl","df1_svm_3.pkl","df1_tree_4.pkl"]
#job_list=["df2_naive_1.pkl","df2_logit_2.pkl","df2_svm_3.pkl","df2_tree_4.pkl"]
job_list=["df3_naive_1.pkl","df3_logit_2.pkl","df3_svm_3.pkl","df3_tree_4.pkl"]
data=df3
train=int(len(data)/5*4)
test_set=data['refined_content'].iloc[train:]
test_label=data['sentiment'].iloc[train:]
for j in job_list:
    clf = joblib.load(j)
    predicted = clf.predict(test_set)
    print(j)
    print("Accuracy score= ",clf.score(test_set,test_label))
    print("Metrics classification report:\n",metrics.classification_report(test_label, predicted, target_names=clf.classes_))
    print("Confusion matrix:\n",metrics.confusion_matrix(test_label, predicted))
    print("--------------------------------\n")
```

b. *Experiment 2:* *Examine whether performing dimension reduction (SVD) on term-document matrix before classification can improve classification accuracy.*

To reduce dimensions of feature vector in the term document matrix, I use TruncatedSVD. I just add this into original Pipeline before applying supervised classifier. Also due to SVM gave us the best results in experiment 1, I also use this classifier only in the 2nd experiment. The new Pipeline becomes:

```
svd_clf = Pipeline([('vect', CountVectorizer()),
          ('tfidf', TfidfTransformer(norm='l2', use_idf=True,
          smooth_idf=True, sublinear_tf=True)),
          ('svd', TruncatedSVD(n_components=<# of Dimensions to reduce>)),
          ('svm_clf', SGDClassifier(loss='hinge', penalty='l2',alpha=1e-3,
          n_iter=5, random_state=42))])
```

The `n_components` parameter in TruncatedSVD is the number of dimensions to reduce. In this experiment, I use different values for n_components (from 100 to 1500) to see which value could give the best performance.

```
accuracy_list=[]
for i in range(100,1500,100):
    svd_clf = Pipeline([('vect', CountVectorizer()),
                        ('tfidf', TfidfTransformer(norm='l2', use_idf=True, smooth_idf=True, sublinear_tf=True)),
                        ('svd', TruncatedSVD(n_components=i)),
                        ('svm_clf', SGDClassifier(loss='hinge', penalty='l2',alpha=1e-3, n_iter=5, random_state=42))])
    _ =svd_clf.fit(train_set,train_label)
    predicted = svd_clf.predict(test_set)
    accuracy_list.append(np.mean(predicted == test_label) )
```

c. *Experiment 3:* *Apply emotion classification model to detect emotion in Human of New York dataset.*

Although we should not apply a classification model which is trained from a dataset coming from a specific population on another dataset from different population due to the difference in sampling methods and characteristics of two populations, I find that the ISEAR dataset and Human of New York dataset also have some similarities as both focus on feelings of different people.

Hence, in this 3rd experiment, I would like to use the best classification model of ISEAR dataset to analyze the emotions in people's stories in Human of New York dataset (details of dataset collection was given in Topic Modelling assignment https://courses.ischool.illinois.edu/mod/forum/discuss.php?d=373995

5.2. Findings:

a. *Experiment 1:*

The classification results in term of Precision, Recall and F1-score for each emotion across the 3 datasets are described in Table 1. The highest scores of precision, recall and F1 are also highlighted for each emotion class.

In Cecilia's Affect data, in general, SVM gives the best results for anger-disgust, Logistic Regression gave best results for fear, joy and sadness, Decision Tree gives the best results for surprise. In SemEval 2007, SVM gives the best results for all emotions among all classifiers. Moreover, in these 2 datasets, joy is the emotion having best classification results compared to other emotions and it is also the emotion with largest number of sentences in both datasets (as well as in training dataset). Surprise and fear have very poor classification results which should be due to their small amount of data in the datasets.

In ISEAR dataset, majority of the data expresses anger-disgust emotion. Anger-disgust is also the emotion having best classification results. Comparing the 4 classification method, Logistic Regression and SVM give best results for all emotions in general.

Emotion	Classification Method	Cecilia's Affect data			SemEval 2007			ISEAR		
		Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score
anger-disgust	Naïve Bayes	**1**	0.06	0.11	0.35	0.26	0.3	0.58	**0.98**	0.73
	Logistic Regression	1	0.16	0.28	**0.5**	0.22	0.31	0.65	0.95	**0.77**
	SVM	0.71	**0.48**	**0.57**	0.26	**0.41**	**0.32**	0.65	0.95	**0.77**
	Decision Tree	1	0.03	0.06	0.24	0.3	0.26	**0.72**	0.73	0.72
fear	Naïve Bayes	**1**	0.1	0.18	0.4	0.07	0.12	**0.91**	0.44	0.59
	Logistic Regression	0.73	**0.32**	**0.44**	0.25	0.04	0.06	0.88	0.61	0.72
	SVM	0.7	0.27	0.39	**0.3**	**0.26**	**0.28**	0.87	0.64	**0.74**
	Decision Tree	0.35	1	0.52	0.28	0.26	0.27	0.67	**0.65**	0.66
joy	Naïve Bayes	0.36	**0.97**	0.53	0.42	0.91	0.57	0.73	0.56	0.63
	Logistic Regression	0.48	0.86	**0.62**	0.41	**0.95**	0.57	**0.76**	0.64	**0.69**
	SVM	**0.54**	0.51	0.52	**0.55**	0.8	**0.65**	0.74	0.63	0.68
	Decision Tree	0.38	0.72	0.5	0.43	0.61	0.5	0.57	**0.66**	0.61
sadness	Naïve Bayes	0.5	0.43	0.46	0.58	0.2	0.3	0.92	0.34	0.5
	Logistic Regression	**0.54**	0.54	**0.54**	**0.65**	0.24	0.35	0.93	0.47	**0.63**
	SVM	0.4	**0.61**	0.48	0.59	**0.42**	**0.49**	**0.94**	0.46	0.62
	Decision Tree	0.37	0.57	0.45	0.52	0.29	0.37	0.62	**0.52**	0.56
surprise	Naïve Bayes	0	0	0	0.5	0.08	0.13	-	-	-
	Logistic Regression	0	0	0	0.5	0.04	0.07	-	-	-
	SVM	**0.5**	0.07	0.12	**0.5**	**0.15**	**0.23**	-	-	-
	Decision Tree	0.3	**0.1**	**0.15**	0.31	0.19	0.24	-	-	-

Table 1: Precision, Recall and F1-score

In term of accuracy score (described in Table 2), SVM gives best classification results across 3 datasets.

Classification Method	Accuracy Score		
	Cecilia's Affect data	SemEval 2007	ISEAR
Naïve Bayes	0.3776	0.4275	0.6626
Logistic Regression	0.4191	0.4314	0.7257
SVM	**0.5228**	**0.4824**	**0.7267**
Decision Tree	0.4523	0.3843	0.6579

Table 2: Accuracy Score

We also can describe the classification performance using the average F1-score measure. The average F1-score weights every category/emotion equally, regardless of how many sentences are assigned to each category. Using this measurement, we can control the bias of the imbalanced data distribution in each dataset (Calvo, R. A., & Mac Kim, S., 2013). The average F1-scores using different classifiers over 3 datasets are shown in Table 3. From the results, we can see that SVM gives the best performance results across all datasets.

Classifier	**F1 Score**		
	Cecilia's Affect data	SemEval 2007	ISEAR
Naïve Bayes	0.25	0.35	0.64
Logistic Regression	0.33	0.34	0.72
SVM	**0.49**	**0.45**	**0.72**
Decision Tree	0.43	0.37	0.66

Table 3: Average F1-score

b. Experiment 2:

Figure 1 shows accuracy score of SVM classification on 3 datasets when using different n_components values of TruncatedSVD.

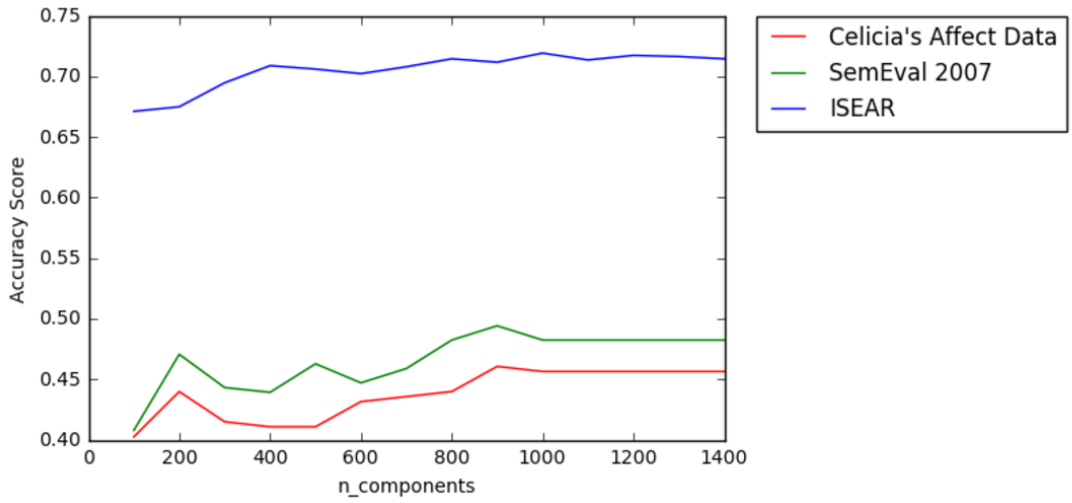


Figure 1: Accuracy Score using different n_components values

Compare to the accuracy score without using SVD to reduce dimensions (given in Table 2), I did not see any improvement in accuracy score after performing dimension reduction. However, considering that the difference was not large (Eg: 0.4606(SVD) vs 0.5228(w/o SVD) for df1, 0.4941(SVD) vs 0.4824 (w/o SVD) for df2, 0.7191(SVD) vs. 0.7267(w/o SVD) for df3)

and converting into fewer dimensions could reduce computational time and storage, we can apply dimensional reduction (such as SVD) before applying supervised classification.

c. ***Experiment 3:***

Using SVM classifier model of ISEAR dataset to test on Human of New York dataset, the emotion distribution for Human of NY's stories is illustrated in Table 4 and Figure 2; assume that each story is classified to only one emotion.

We see that most of the stories (139/192) express anger-disgust emotions. This result might not be reasonable as each story contains multiple sentences and each sentence could express different emotion. Hence, a story can include several emotions instead of just only one.

sentiment	Story Count
anger-disgust	139
fear	17
joy	21
sadness	15

Table 4: sentiment distribution over stories in Human of NY

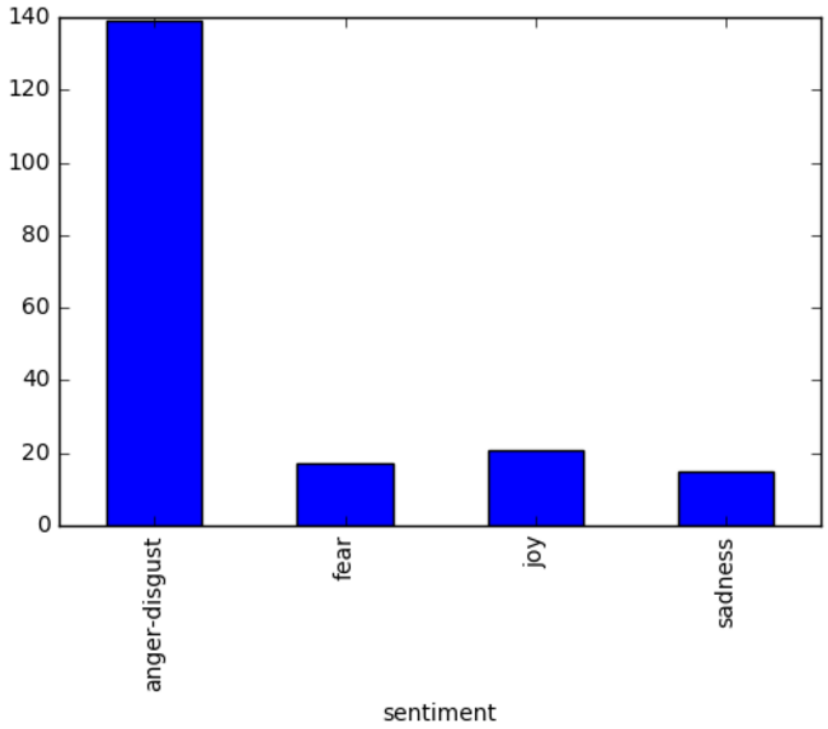


Figure 2: Emotion distribution in Human of New York's stories

To make the data more reasonable, each story is splited into sentences using separator "." and the classifier is applied to sentence instead of whole story.

We get the emotion distribution group by story and emotion class in Table 5.

		content	refined_content
storyno	sentiment		
1	anger-disgust	7	7
	fear	1	1
	joy	1	1
2	anger-disgust	4	4
	fear	1	1
3	anger-disgust	7	7
4	anger-disgust	23	23
	fear	4	4
	joy	1	1
	sadness	2	2
...	...	...	...
187	anger-disgust	6	6
	fear	2	2
	sadness	1	1
188	anger-disgust	2	2
189	anger-disgust	11	11
	joy	3	3
	sadness	3	3
190	anger-disgust	11	11
	fear	1	1
191	anger-disgust	9	9
	fear	5	5
	joy	1	1
192	anger-disgust	1	1

Table 5: Emotion distribution group by story and emotion class

Figure 3 shows the new emotion distribution over stories in Human of New York. Only 2 of the stories do not express anger-disgust emotion (story numbers 20 and 94) as following:

```
test_df2[test_df2['storyno'].isin([20,94])]
```

	content	storyno	sentiment	refined_content
251	“He fell in love with me because I used to hav...	20	joy	fell love used huge as
1316	“If I think back, I get depressed	94	sadness	think back get depressed
1317	If I think ahead, I get afraid	94	fear	think ahead get afraid

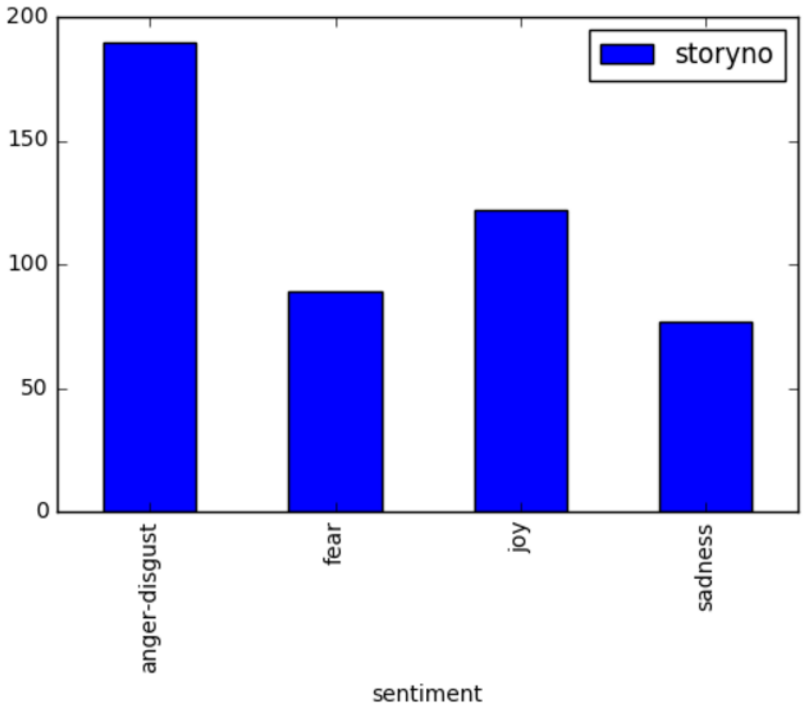


Figure 3: New emotion distribution

I would like to use Venn diagram to show the count of stories having overlapped emotions; however, matplotlib_venn package only supports up to 3-circle venn diagrams. As 190/192 stories include anger-disgust emotion (almost all stories), I just draw the Venn diagram to show the intersection of fear, joy and sadness in Figure 4.

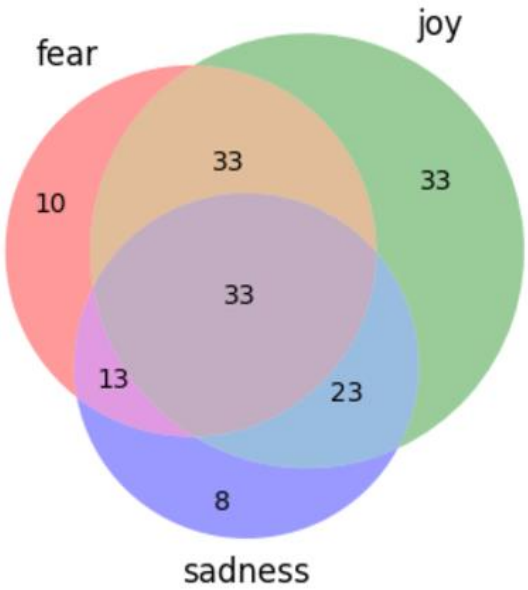


Figure 4. Overlapped stories of fear, joy and sadness emotions

(Note that most stories do include anger-disgust emotion which is not shown in this diagram)

Table 6 provides spot checks for 10 random sentences in the dataset.

content	storyno	sentiment (predicted)	refined_content	Sentiment Evaluation
I was working as a security guard and my children were not suffering	82	anger-disgust	working security guard child suffering	FALSE
Now we've spent the whole day together	84	joy	we've spent whole day together	TRUE
She would never respond when we tried to correct her	157	anger-disgust	would never respond tried correct	TRUE
And eleven hamsters	154	anger-disgust	eleven hamster	FALSE
"Ten years ago I lost my job at a hotel	159	sadness	ten year ago lost job hotel	TRUE
I take the medicine because it's part of the deal for me having my own apartment, but I'm not mentally ill	18	anger-disgust	take medicine it's part deal apartment i'm mentally ill	TRUE
She was bleeding internally and had a blood clot in her leg	173	anger-disgust	bleeding internally blood clot leg	TRUE
For awhile I was so depressed that I couldn't leave my room	128	sadness	awhile depressed couldn't leave room	TRUE
But I'm not mentally ill	18	anger-disgust	i'm mentally ill	TRUE
I had to carry it up the hill	191	anger-disgust	carry hill	TRUE
' All my friends forgot about me	186	anger-disgust	friend forgot	TRUE
I was going to midweek services	4	fear	going midweek service	FALSE
He makes me feel like a princess	11	anger-disgust	make feel like princess	FALSE
I always thought the relationship between a mother and a child was about giving and receiving orders	184	sadness	always thought relationship mother child giving receiving order	TRUE
We don't know exactly what happened	100	anger-disgust	don't know exactly happened	TRUE

Table 6. Spot checks for 10 random sentences

6. Conclusions and Next Steps:

From the experiment results in the project, we can see that SVM classification performs quite well when working with text. Moreover, the distribution of classes in dataset also takes an important part impacting the classification performance. From the project, we do not see a noticeable improvement of classification accuracy when applying dimension reduction method (SVD); however, it can be a trade-off with the reduction of computational expenses as well as controlling bias of the classification model. For future works, we can explore and extract different features (besides using bag of words) which would be useful for detect emotions such as punctuations (exclamation marks), negative words (not, don't, no,…), domain-specific dictionary for emotions instead of using all words in dataset, emoticons like the study of Hasan 2014.

Regarding emotion detection for Human of New York's stories, in this project, we cannot verify its accuracy (just random spot checks). However, in the future, we can build bigger dataset for Human of NY with its own class labels and fit model from that dataset to detect emotion for other test set from same population, it would be more appropriate and accurate.

References:

[1] Canales, Lea and Patricio Martínez-Barco. "Emotion Detection from text: A Survey." (2014).

[2] Paul Ekman. 1999. Basic emotions. In Handbook of cognition and emotion, pages 45–60. Virginia Francisco and Pablo Gerv´as. 2013. EmoTag: An Approach to Automated Mark-Up of Emotions in Texts. Computational Intelligence, 29(4):680–721.

[3] Calvo, Rafael A., and Sunghwan Mac Kim. "Emotions in text: dimensional and categorical models." Computational Intelligence 29, no. 3 (2013): 527-543.

[4] Hasan, Maryam, Elke Rundensteiner, and Emmanuel Agu. "Emotex: Detecting emotions in twitter messages." Academy of Science and Engineering (ASE) (2014).

[5] Roberts, Kirk, Michael A. Roach, Joseph Johnson, Josh Guthrie and Sanda M. Harabagiu. "EmpaTweet: Annotating and Detecting Emotions on Twitter." LREC (2012).

[6] Desmet, Bart, and Véronique Hoste. "Emotion detection in suicide notes." Expert Systems with Applications 40, no. 16 (2013): 6351-6358.

[7] Sunghwan Mac Kim. 2011. Recognising Emotions and Sentiments in Text. Ph.D. thesis, University of Sydney.