Sentiment Analysis and Text Mining for Hilary Clinton's Emails and Twitter US Airline Sentiment

Master Project
April, 2*nd*. Spring 2018

Final Project

Submitted to
Division of Computer Science and Engineering
School of Electrical Engineering and Computer Science
Louisiana State University

Committee
Dr. Jianhua Chen(Chair), Dr. Evangelos Triantaphyllou, Dr. Konstantin Busch

By
Jumao Yuan
jyuan4@lsu.edu

ACKNOWLEDGEMENTS

I would like to thank Dr. Jianhua Chen, my major professor, for her patience and guidance and also Dr. Evangelos Triantaphyllow, and Dr. Konstantin Busch, my committee members, for their support and time. The concepts that I have learned in each of their courses have helped me to successfully complete this project. I would also like to thank Jie Chen and Guoji Xu, my classmates in Computer Science for their great help in my courses and project. Last but not the least, I would like to thank my parents and my friends for their love and encouragement all the time.

ABSTRACT

Natural language processing (NLP) for sentimental analysis plays an important role in text mining analysis. This project aims to apply NLP and sentimental analysis techniques in two datasets, Hillary Clinton's emails during her stay as the United States secretary of state and Twitter US airline sentiment. Due to Hillary Clinton's controversy over the use of personal email accounts on non-government servers in that period, the State Department released her emails to public. Data scientists try to unravel how the content of Clinton's emails affect US emotion and behaviors with other countries around the world. The second dataset was scraped from Twitter in February of 2015 and contributors were asked to first classify positive, negative and neural tweets followed by categorizing negative reasons such as "late flight" or "rude service".

In Hillary Clinton's emails, since the categorical variable "sentiment" is not given, we manually labeled sentiments for around 250 comments with three levels of sentiments, happy, unhappy and neutral. Bag of words and different vectorizers, 1-gram, n-grams, df-idf have been explored during NLP. Meanwhile, conventional machine learning methods, Naive Bayes(NB), Support Vector Machine(SVM) and Logistic Regression(LR) were used for the small dataset. However, for large data modeling, the conventional machine learning method may be not a good fit. To explore sentiment analysis in large dataset, deep learning methods Long Short Term Memory(LSTM), a special method of Recursive Neural Network(RNN) was studied for Twitter US airline sentiment.

In the results, we found RNN methods in larger training data performs a higher accuracy than classical machine learning models. Different vertorizers like 1-gram, 2-grams or N-grams where N is even bigger and tf-idf generate different results in different models. Even though our datasets are quite unbalanced, by using NLP and sentimental analysis tools, we still obtained acceptable predicted categorical variable for these two datasets.

Contents

1 Introduction

Sentiment analysis (SA), called as opinion mining, is a discipline to use NLP, text analysis and computational linguistics to identify and extract information of people's opinions, feelings and behaviors in source materials [1, 2], "Why do the customers choose another companies?". We may guess, like price, customer servers, advertisements, competition, etc. Here, we want to know some key words, such as tacky design, inferior customer service and so forth. However, it seems not easy to survey customers who chose other companies. Instead, we could consider SA for customers' comments on Blogs, amazon, tweets, facebook etc. meanwhile, SA has broad applications in other areas, such as political science, marketing and even psychology study.

NLP (Natural Language Processing) is a primary tool to implement sentimental analysis. NLP presents an interaction between human and computer [3]. NLP indicates the ability that a computer understands human language, product of the cooperation among disciplines, statistics, artificial intelligence and linguistic. With machine learning techniques, modern NLP improved its superior performance. Compared to hand-produced principles, machine learning algorithms on NLP enable robust, accurate and general results [3]. Major tasks for NLP, include sentence detection, part-of-speech (POS) tagging, parsing, entity extraction and words disambiguation. In this project, we focus on sentence tokenization, POS tagging and syntactic parsing. Sentence tokenization is to split sentences in a given text based on English grammars and punctuations. Generally, periods, commas, and other punctuation marks indicate sentence boundaries. POS, known as lexical category, is defined as a linguistic category of words. Meanwhile, different languages have different lexical categories. In English, there are eight basic lexical categories, noun, pronoun, adjective, verb, adverb, preposition, conjunction and interjection [4]. The goal of POS tagging is to identify lexical category for each word in a given text. Parsing, known as grammatical analysis, is aimed to produce a parse tree for a given sentence.

In our datasets, we skipped two steps "POS" and "Words disambiguous" to build simple models for sentiment analysis.

In terms of NLP Fig. 1, we mainly use OpenNLP [1] as syntactical parser. Also, there are other well-known syntactical parsers: Stanford Parsers, Link Parsers and Minipar [5]. In this project, NLTK package in Python [2] and Standford CoreNLP toolkit in R [3] were used to help syntactic parsing and sentimental analysis.

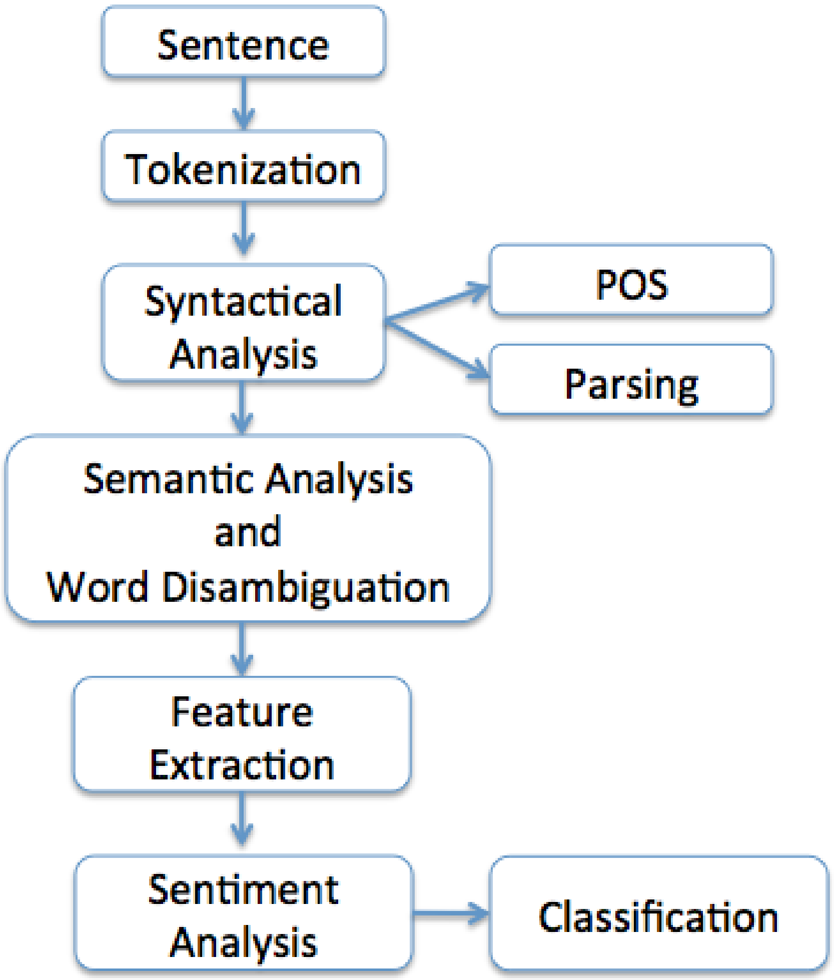


Figure 1: A simple pipeline architecture for NLP in sentimental analysis system. This system reads data from sentences, and then use tokenization to split sentences, which can be called as Syntactic Parsing. Next, we classified each sentence with sentimental analysis from word library.

[1]`https://opennlp.apache.org/`
[2]`https://pypi.python.org/pypi/nltk`
[3]`https://cran.r-project.org/web/packages/NLP/index.html`

2 Data Description

In our project, there are two datasets called email data and Twitter data. Email data is a very raw data, composed of email text without labeled sentiment; Twitter data includes both text comments and labeled sentiment which is download form Kaggle.

2.1 Email Data

Throughout 2015, Hillary Clinton has been embroiled in controversy over the use of personal email accounts on non-government servers during her stay as the United States Secretary of State. Some political experts and opponents maintain that Clinton's use of personal email accounts to conduct Secretary of State affairs is in violation of protocols and federal laws that ensure appropriate record-keeping of government activity. Hillary's campaign has provided their own four sentence summary of her email use here [4].

There have been a number of Freedom of Information lawsuits filed over the State Department's failure to fully release the emails sent and received on Clinton's private accounts. On Monday, August 31, the State Department released nearly 7,000 pages of Clinton's heavily redacted emails (its biggest release of emails to date).

So far, Hillary Clinton has released 7,945 emails in response to a FOIA request.

Name	Count
Emails	7945
Persons	513
Aliases	850
Email Receivers	9306

The documents were released by the State Department as PDFs. It has released some of Hilary Clinton's email from the time when she was secretary of state. The department reviewed more than 50,000 pages of Clinton's messages and released them on a monthly schedule, concluding Feb. 29, 2016. We've cleaned and normalized the released documents for public analysis. This dataset

[4]`https://github.com/benhamner/hillary-clinton-emails`

is not meant to express any particular political affiliation or intent, but only personal interest for data analysis to uncover Hilary's political landscape.

Raw PDF to SQLite

We transformed Hilary Clinton's emails released through the FOIA request from raw PDF documents to CSV files and a SQLite database [4], making it easier to understand and analyze the documenets.

Fields list [4]:

- ID - unique identifier for internal reference
- MetadataSubject - Email SUBJECT field (from the FOIA metadata)
- MetadataTo - Email TO field (from the FOIA metadata)
- MetadataFrom - Email FROM field (from the FOIA metadata)
- ExtractedSubject - Email SUBJECT field (extracted from the PDF)
- ExtractedTo - Email TO field (extracted from the PDF)
- ExtractedFrom - Email FROM field (extracted from the PDF)
- ExtractedBodyText - Attempt to only pull out the text in the body that the email sender wrote (extracted from the PDF)
- RawText - Raw email text (extracted from the PDF)
- Name - person's name
- Alias - text in the From/To email fields that refers to the person
- PersonId - person that the alias refers to

The SQLite database contains emails, persons, aliases and email receivers with their corresponding field. Our next step focuses on improving from/to address extraction mechanisms, normalize various email address representations to people and improve the BodyText extraction. Here we used Python3 with pandas, arrow and numpy packages to handle the raw PDFs data [5]. here is a shortcut email from WSJ website:

[5] `http://graphics.wsj.com/hillary-clinton-email-documents/`

RELEASE IN FULL

From: Mills, Cheryl D <MillsCD@state.gov>
Sent: Friday, February 1, 2013 7:17 AM
To: H
Subject: Fw: Police: Suicide bombing at US Embassy, 2 dead (AP)

From: OpsNewsTicker@state.gov [mailto:OpsNewsTicker@state.gov]
Sent: Friday, February 01, 2013 07:05 AM
To: DS Command Center; NEWS-EUR; PAFOGroup2@state.gov <PAFOGroup2@state.gov>; NEWS-Mahogany
Subject: Police: Suicide bombing at US Embassy, 2 dead (AP)

ANKARA (AP) - A police official says a suicide bomber has detonated an explosive device at the entrance of the U.S. Embassy in the Turkish capital and at least two people are dead.

An Associated Press journalist on Friday saw a body in the street in front of an embassy side entrance. The bomb appeared to have exploded inside the security checkpoint at the entrance of the embassy.

Private NTV television said two security guards at the entrance were killed.

The police official spoke on condition of anonymity in line with government rules. The phones were not being answered at the embassy.

NewsTickers alert senior Department officials to breaking news. This item appears as it did in its original publication and does not contain analysis or commentary by Department sources.

In our sentiment analysis study, we only have interest in sentiment classifiers as well as bodytext which include metadata subject and extracted bodytext.

Since there is no labeled sentiment classifiers in the raw emails, we randomly picked up around 250 subjects and manually labeled them as happy, unhappy or neutral followed by categorizing rules. I worked on sentiment labeling with my labmate Dr. Kio Omoefe from Chemistry Department. First, we labeled separately for 250 emails and then compare our labels. There are about 9 labels we have disagreement. We discussed and discarded 7 ambiguous emails. Finally we reached an agreement for 107 happy, 42 unhappy and 94 neutral sentiment-labeled emails.

Category	Count
Happy	107
Unhappy	42
Neutral	94

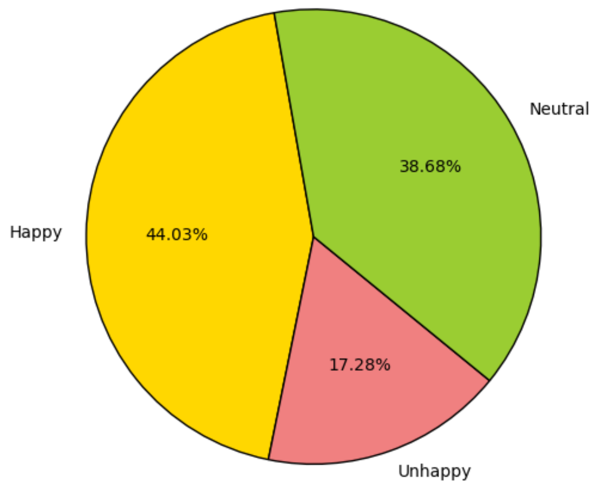


Figure 2: Pie plot for total percentage of email sentiment

2.2 Twitter Data

Our second dataset was downloaded from Kaggle competition [6]. The Twitter data was scraped from February of 2015 and already classified as positive, negative and neutral tweets provided by

[6]https://www.kaggle.com/crowdflower/twitter-airline-sentiment/data

contributors followed by categorizing negative reasons. The data was stored in SQLite database for further analysis. The labeled data contains 14640 tweets from 7700 users, large enough for deep learning exploration. The original data variables include "airline sentiment", "airline sentiment confidence", "negative reason", "negative reason confidence", "name of airline", "name of twitter", "text","user time zone" etc. For sentiment analysis, we subtract interested two columns, "airline sentiment" and "text" for machine learning modeling. About 62.7% of the reported sentiments were negative as shown in the pie plot below.

Category	Count
Negative	9179
Positive	2363
Neutral	3097

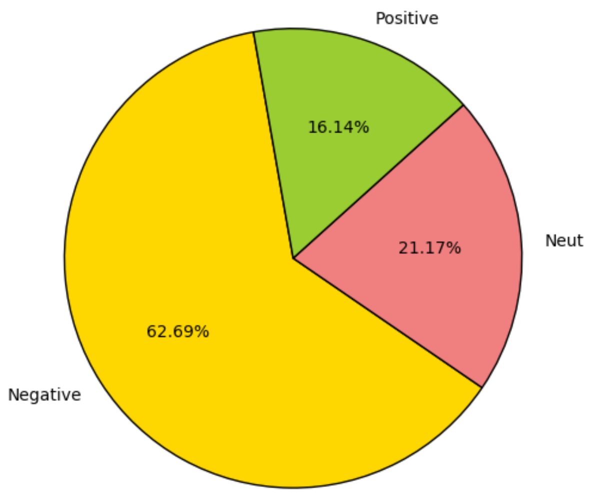


Figure 3: Pie plot for total percentage of airline sentiment

3 Text Pre-preprocessing

With very raw and complicated datasets. We need figure out which variables we need for analysis in the following steps. For email data, we extracted "bodytext" and "subject" and merged these two columns as a single column. Next, manually select 250 sentences and labeled sentiment categories as happy: 1, unhappy: 2 and neutral: 3.

For Twitter US airline sentiment data, the sentiment is already labeled by volunteers. Just download the training data and testing data from Kaggle competition website and ready for use.

The cleaned datasets are quite unbalanced, but no a big effect for text mining in the next step.

For given sentences or documents, words pre-processing is necessary before vectorization. Essentially, there are four steps.

- Tokenization

  Tokenization is the first step during indexing process, splitting text into words with spaces, punctuation signs, cases and so on into standardized words. This process makes the indexing relavant and preserve the source information that is rendered.

- Stemming

  Stemming is necessary to find stems of each word in context. The process depends on a suffix dictionary and make it possible to extract stems after words analysis. In Python NLTK package, it computes the most relevant stems from the grammar and syntax rules of languages.

- Lemmatization

  As an advanced analysis process where each word is computed from a dictionary of inflected forms, helpful to reduce inflectional forms of words such as plurals, conjugation etc.

- Stop Words

  To reduce word vector size and save computation time, empty words deletion is quite important in vectorization. The stop words we manually defined are

  ```
  'the', 'a', 'an', 'and', 'but', 'if', 'or', 'because', 'as', 'what', 'which',
  'this', 'that', 'these', 'those', 'then',
  'just', 'so', 'than', 'such', 'both', 'through', 'about', 'for', 'is', 'of',
  'while', 'during', 'to', 'what', 'which',  'is', 'If', 'while', 'this',  'at',  'i'
  ```

4 Methodology and Learning Algorithms

4.1 SQLite Database

SQLite [7] an open source database system written in programming language C by D. Richard Hipp with a high reliability and full featured database engine. [6, 7]. The well-known features of SQLite include consistent and atomic transactions even system crashing happened, no administration needed, a complete database storage, user friendly API, large storage of database, fast manipulation etc. There are approximately 80 functions in SQLite API. Nevertheless, only several functions are usually necessary for queries and function call. SQL compiler of tokenizer, parser and code generator are fast and easy to be implemented. Due to the prominent features, SQLite embedded system denotes a satisfied performance to database applications.

4.2 Feature Extraction

The bag-of-words model is used in NLP and information retrieval (IR) as a simplifying representation [8, 9]. In this model, a text in a sentence or a document is represented as the bag of words without considering grammar and word sequences.

For example, two sentences described below

```
'All my cats in a row',
'When my cat sits down, she looks like a Furby toy!',
```

A list is created based on the two strings above.

```
'all': 0, 'cat': 1, 'cats': 2, 'down': 3, 'furby': 4, 'in': 5, 'like': 6, 'looks': 7,
'my': 8, 'row': 9, 'she': 10, 'sits': 11, 'toy': 12, 'when': 13
```

The list contains 14 unique words in the vocabulary. If we express the texts as numeric vectors, we would get,

[7] `https://www.sqlite.org/`

```
‘All my cats in a row’ = [1 0 1 0 0 1 0 0 1 1 0 0 0 0]
‘When my cat sits down, she looks like a Furby toy!’, =
[0 1 0 1 1 0 1 1 1 0 1 1 1 1]
```

In most cases, term frequency–inverse document frequency (TF-IDF) is usually used to implement bags-of-words model, which is a numerical statistic to reflect how important a word is to a document in a corpus [10, 11]. The counts of a term that occurs in a document is defined as term frequency. TF-IDF is successfully demonstrated for preposition filtering in text classification.

Typically, the tf-idf weight include two parts, the normalized Term Frequency (TF), which is the frequency of a word in a document divided by the total number of words in that document and the Inverse Document Frequency (IDF), the logarithm of the number of the documents in the corpus divided by the number of documents including the terms.

- TF: Term Frequency mathematically defined as TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).
- IDF: Inverse Document Frequency, indicates the importance of a word in the document, represented as IDF(t) = log_e(Total number of documents / Number of documents with term t in it).

For example, when input four sentences

```
"The sky is blue."
"The sun is bright today.",
"The sun in the sky is bright."
"We can see the shining sun, the bright sun."
```

IF term frequency will be calculated for each word.

Terms	blue	bright	can	see	shining	sky	sun	today
doc1	1	0	0	0	0	1	0	0

```
doc2  0      1     0   0    0      0   1   1

doc3  0      1     0   0    0      1   1   0

doc4  0      1     1   1    1      0   2   0
```

Then use IDF complete math formula to calculate the term frequency matrix and obtain the IDF weight.

```
blue    bright       can       see   shining       sky       sun
0.6931472 0.0000000 0.6931472 0.6931472 0.6931472 0.2876821 0.0000000
today
0.6931472
```

Now, we have our matrix with term frequency and IDF weight, it's ready to calculate the full tf-idf weight as a sparse matrix.

```
Docs      blue bright       can       see   shining       sky sun today
1 0.9236103      0 0.0000000 0.0000000 0.0000000 0.3833329   0     0
2 0.0000000      0 0.0000000 0.0000000 0.0000000 0.0000000   0     1
3 0.0000000      0 0.0000000 0.0000000 0.0000000 1.0000000   0     0
4 0.0000000      0 0.5773503 0.5773503 0.5773503 0.0000000   0     0
```

For comparison, 1-gram and N-gram models also were studied to explore the effect of choosing different vectorizers. If the words in text are correlated, we can pick up N-grams but larger N would decrease performance. Generally, larger N suggests requiring a larger training set since the patterns tend to be more sparsely distributed across documents. While TF-IDF focuses more on correlation between each single word and the sentence.

For example, for given sentences,

```
How are you?
I am fine, and you?
```

The bigram and trigram would be,

```
Bigram: [('Hi', 'How'), ('How', 'are'), ('are', 'you'), ('you', '?'), ('?', 'i'),
('i', 'am'), ('am', 'fine'), ('fine', 'and'), ('and', 'you')]
```

```
Trigram: [('Hi', 'How', 'are'), ('How', 'are', 'you'), ('are', 'you', '?'),
('you', '?', 'i'), ('?', 'i', 'am'), ('i', 'am', 'fine'), ('am', 'fine', 'and'),
('fine', 'and', 'you')]
```

4.3 Logistic Regression

Technically, Logistic Regression(LR) is a discriminative classifier, referring to a classifier that classifies observations into two classes. There are typically three types of LR, binary(Pass/Fail), Multi(Cats, Dogs, Sheep) and Ordinal(Low, Medium, High). For binary LR, in order to map predicted values to probabilities, sigmoid function will be used. The function maps any real value into digit 0 and 1. In machine learning, sigmoid is used to map predictions to probabilities.

$$S(z) = \frac{1}{1 + e^(- z)} \tag{1}$$

where $S(z)$ is the output of digit 0 and 1, z is the input function and e is base of natural log. However, if the prediction function is non-linear, mean squared error is no longer available, instead, cost function also known as log loss will be used in this case.

4.4 Naive Bayes

The naive bayes(NB) classifier is based on Bayes' theorem under the assumption of independence between predictors. Instead of complicated iterative parameter estimation, NB model is easy to build. Typically, Bayes theorem provides a way to calculate the posterior probability $P(c|x)$ from class prior probability $P(c)$, predictor prior probability $P(x)$ and likelihood $P(x|c)$ where the function is

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \tag{2}$$

Namely the Naive Bayes assumption equation which the attributes are conditionally independent given its class states as,

$$P(c|X) = P(x_1|c)xP(x_2|c)x...xP(x_n|c) \times P(c) \tag{3}$$

4.5 Support Vector Machine

Typically, four tuning parameters are quite important in SVM.

- Kernel

  The learning of the hyperplane in linear SVM is to use some linear algebra. For linear kernel, the equation for prediction for a new input using the dot product between input x^T and another input vector y plus an optional constant parameter c. A linear kernel is often equivalent to a non kernel counterpart.

$$K(x, y) = x^T y + c \tag{4}$$

  where the coefficients B_0 and a_i must be estimated from training data through learning algorithms. Nevertheless, for the degree-d polynomial kernel, it is defined as,

$$K(x, y) = (X^T y + c)^d \tag{5}$$

  where x and y are input vectors, which of features computed from training or testing samples. c is a non-negative trade-off parameter. Especially when $c = 0$, this kernel can be treated as homogeneous. Meanwhile, Gaussian kernel(also called radial basis function kernel) is popular among kernel functions, which is defined as,

$$K(x, x') = exp(-\frac{||x - x'||^2}{2\sigma^2}) \tag{6}$$

  where x and x' are two samples represented as input feature vectors. $||x - x'||^2$ can be recognized as squared Euclidean distance and σ is a free parameter.

Alternatively, if we introduce a new parameter where $\gamma = \frac{1}{2\sigma^2}$, the Gaussian kernel function could also be implemented using the updated equation,

$$K(x, y) = exp(-\gamma||x - y||^2) \tag{7}$$

The exponential kernel is closely related to the Gaussian kernel and also belongs to a radial basis function kernel, which is,

$$K(x, y) = exp(-\frac{||x - y||}{2\sigma^2}) \tag{8}$$

Moreover, there are some other kernels such as laplacian kernel, anova kernel, sigmoid kernel, quadratic kernel etc.

- Regularization

  In Python's sklearn library, the regularization parameter is often termed as C parameter to tell the SVM optimization the percentage to avoid misclassifying training samples.

- Gamma

  The gamma parameter indicates the influence of a single training sample. Low value means far and high value means close from plausible seperation line.

- Margin

  A good margin is where the separation is large for both classes, which allows the points to be in their respective classes without crossing to other class.

4.6 RNN (LSTM) in Deep Learning

LSTM and Naive Bayes (NB) are both classifiers to learn how the articles are split into those categories and then be able to classify new articles on it's own.

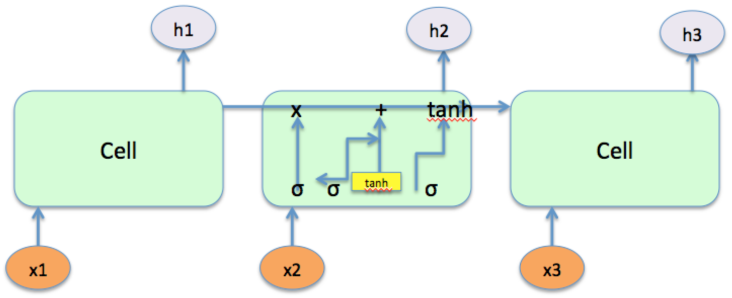
In order to use this classifier for text analysis, we usually pre-process the text (bag of words + tf-idf) so that we can transform it into vectors containing numerical values. These vectors serve as an input to the NB model. The classifier assumes that our feature – the attributes of the vectors we produced are independent of one another. When this assumption holds, it is a very strong classifier that requires very little data to work. NB belongs to a category of models called generative. This means that during training, NB tries to find out how the data was generated in the first place. It essentially tries to figure out the underlying distribution that produced the examples we input to the model; while LSTMs are networks that read our data sequentially, while keeping a "memory" of what they have read previously. These are really useful when dealing with text because of the correlation words have between them. On the other hand RNN is a discriminative model. It tries to figure out what the differences are between our positive and negative examples, in order to perform the classification. RNNs most of the time are trained on dedicated GPUs, which compute a lot faster than CPUs.

In the last few years, RNNs has achieved incredible success when applying to speech recognition, image captioning, language modeling etc. However, in practice, conventional RNNs seems not to be able to learn the problem of long-term dependencies. LSTMs – Long Short Term Memory networks – are a special kind of RNN, capable of learning long-term dependencies. This deep learning algorithm was firstly introduced by Hochreiter & Schmidhuber in 1997 [12] and were popularly used by many researchers. As we all know, all RNNs have the form of a chain of repeating modules of NN. In standard RNNs, this repeating module has only a simple structure, a single tanh layer.

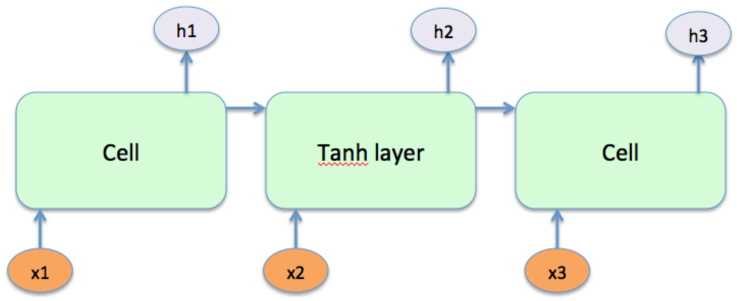
In the diagram above, each line carries an vector from output of one node to input of another node. The first step in LSTM is to decide the abandoned information from the cell state by a sigmoid layer called "forget gate layer". There are two inputs h_{t-1} and x_t and outputs a binary number 0 or 1 for output parameter C_{t-1} in the cell state, where 0 represents "forget the layer" while 1 represents "keep the layer". The neural function is

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{9}$$

The second step is to decide the new information to be stored in the cell state. Two parts here, input gate layer decides which values need be updated and a tanh layer creates a vector of new



(a)



(b)

Figure 4: (a) A repeating module in an LSTM with four interacting layers and (b) A repeating module in a standard RNN with a single layer

parameter $\tilde{C}_t$ added to the cell state.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)\tilde{C}_t = tanh(W_C[h_{t-1}, x_t] + b_C) \tag{10}$$

Then, we need combine these two parts to create an update to the state.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{11}$$

Finally, we need decide the output value in the cell state.

$$O_t = \sigma(W_o[h_{t-1}, x_t] + b_o)h_t = o_t * tanh(C_t) \tag{12}$$

Because RNNs are considered a deep learning algorithm, they have implementations in all major deep learning libraries:

- TensorFlow: Most popular DL library at the moment. Published and maintained by google.
- theano: Similar library to tf, older, published by the University of Montreal.
- keras: Wrapper for tf and theano. Much easier. What I suggest you use if you ever want to implement RNNs.
- caffe: DL library published by UC Berkeley. Has python API.

4.7 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) was put forward by Blei et al. [13, 14] and widely used as a statistical model for topic exploration. As for LDA model, the process is to generate N (a parameter) topics based on the frequency of a term from a given document set.

LDA assumes the following conditions for each document **w** in a corpus **D** [15] ,

- Choose $N \sim Poisson(\epsilon)$
- Choose $\theta \sim Dir(\alpha)$.
- For each of N words W_n, choose a topic $Z_n \sim Multinomial(\theta)$ and choose a word W_n from a multinomial probability condition $p(W_n|Z_n, \beta)$.

A k-dimensional Dirichlet random variable θ in (k-1) vector has probability density,

$$p(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \theta_1^{\alpha_1 - 1} ... \theta_k^{\alpha_k - 1} \tag{13}$$

where α is a positive k-vector and $\Gamma(x)$ is the Gamma function. Unlike a simple Dirichlet-multinomial clustering model, LDA involves a two-level model for a corpus. In general, LDA is a powerful toolbox for topic study in text mining as well as other concentrations.

5 Implementation

Based on the algorithm described above and two datasets at hand, it's quite important to know how to implement in scripts. Python is a powerful and efficient programming language in machine learning and deep learning analysis.

Python sqlite3 and NLTK (Natural Language Toolkit) packages are mainly used in programming. NLTK is an incredible library in Python to handle text process and words analysis. For example, for the sentence "The staff in the Apple store is nice." First, build a corpus via split the sentence into word by word separately without any punctuations, so as a result, the sentence is split into "The", "staff", "in", "the", "apple", "store", "is", "nice". As we know, there are eight parts of speech in grammar, verb, nouns, adverbs, pronouns, adjectives, prepositions, conjunctions and interjections. For example, for above words, we can generate,

```
[({'The'}, ['DT']),
 ({'staff'}, ['NN']),
 ({'in'}, ['IN']),
 ({'Apple'}, ['NN']),
 ({'store'}, ['NN']),
 ({'is'}, ['IN']),
 ({'nice'}, ['positive', 'JJ']),]
```

and the packages in NLTK we used in our project are listed below.

```
import nltk
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer
from nltk.tokenize import word_tokenize
from nltk.stem.porter import PorterStemmer
```

To classify a word as negative or possible depends on word dictionaries as standard. Typically, Naive Bayes Classifier, KNN or other machine learning methods are available in Python nltk.classify package.

Logistic regression, naive bayes and SVM can be implemented in Python sklearn packages.

```
from sklearn.linear_model import LogisticRgression
from sklearn.svm import linearSVC
from sklearn.naive_bayes import MultinomialNB
```

Also N-grams and TF-IDF are available in sklearn pckages. While for deep learning algorithm RNN LSTM, sklearn is obviously not enough. Keras, a high-level neural networks API, capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or MXNet, is designed to enable fast experimentation with deep neural network. Mostly, we use "keras.layers" to define tunning parameters.

```
from keras.datasets import imdb
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers.embeddings import Embedding
from keras.layers import Dropout
from keras.preprocessing import sequence
```

6 Results

After scripting in Python with NLTK, Sklearn and Keras packages, we generated results related to vectorizer analysis, different learning algorithms analysis and tunning parameters exploration.

6.1 1-gram, n-grams and tf-idf vectorizers for email data

In order to explore the correlation between each single word in sentences and each sentence in document corpus, n-grams (n=1,2,3...) and tf-idf were studied for vectorizer selection in SVM model.

Name	word-Vector-Size	Accuracy
1-gram	3121	0.9180
2-gram	6233	0.5738
3-gram	6379	0.4426
tf-idf	300	0.9016

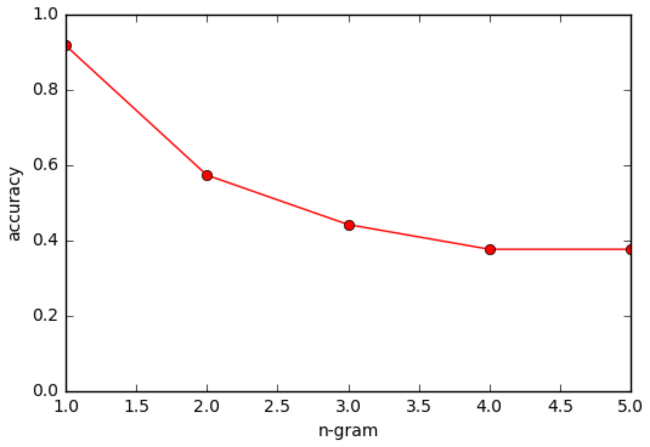


Figure 5: Accuracy in SVM model versus N-grams where N=1,2,3,...

As we can see from the figure above, when N increases in N-gram vectorizer, the accuracy of prediction decreases sharply in the beginning and tend to flat when N is larger than 4. In the next step of machine learning modeling, 1-gram or tf-idf will be used as a count vector.

6.2 Conventional ML models comparison for email data

We randomly split ∼250 small labeled data into 70% training data and 30% testing data for learning algorithms. And for each algorithm, we iterated ∼50 times to get average accuracy.

Here, we explored 3 conventional Machine Learning(ML) models for Hilary Clinton's emails dataset, Naive Bayes(NB), Logistic Regression(LR) and Support vector Machine(SVM), which are popularly used in text classification problems. In the evaluation of predictions, the confusion matrix indicates SVM performs the best.

Models	Accuracy
NB	0.7869
LR	0.8852
SVM	0.9016

Predict / Actual–**NB**	Happy	Unhappy	Neutral
Happy	24	2	1
Unhappy	2	6	4
Neutral	4	0	18

Predict / Actual–**LR**	Happy	Unhappy	Neutral
Happy	23	3	1
Unhappy	1	9	2
Neutral	0	0	22

Predict / Actual–**SVM**	Happy	Unhappy	Neutral
Happy	25	2	0
Unhappy	2	8	2
Neutral	0	0	22

6.3 Topic Modeling with LDA for email data

Curiously, what Hilary Clinton talked about in her back and forth emails with other officers, we build topic modeling with Latent Dirichlet Allocation(LDA) in Python. The data we used for topic modeling is the whole ∼8000 emails instead of small labeled data since sentiment category is not required in topic modeling. Here, we defined 3 topics

- Topic 1: world life want team schedule yes week foreign ll report
- Topic 2: state did time got ve people talk way day family
- Topic 3: thank need com fyi sid day way tomorrow pm sent

The three topics might be overlapped with each other but we still could find some useful information in the emails.

Number	Topic	key-words
1	foreign policy	world, foreign
2	daily life	state, family
3	date	day, tomorrow, pm

According to the table, it makes sense in Hilary Clinton's emails talking foreign policies, her daily life with other contacts and activities every day.

6.4 LSTM model for Twitter data

Since Twitter training data (∼15000) is much larger than email training data, we chose K-fold cross validation to K different subsets. We defined K=10 and obtained K subset data. Same as email data, to guarantee a relatively high accuracy, we iterated ∼50 times and get average value of accuracy.

```
KFold(n_splits=10, random_state=None, shuffle=False)
TRAIN: [ 1464  1465  1466 ... 14637 14638 14639] TEST: [   0    1    2 ... 1461 1462 1463]
TRAIN: [    0     1     2 ... 14637 14638 14639] TEST: [1464 1465 1466 ... 2925 2926 2927]
```

```
TRAIN: [    0     1     2 ... 14637 14638 14639] TEST: [2928 2929 2930 ... 4389 4390 4391]
TRAIN: [    0     1     2 ... 14637 14638 14639] TEST: [4392 4393 4394 ... 5853 5854 5855]
TRAIN: [    0     1     2 ... 14637 14638 14639] TEST: [5856 5857 5858 ... 7317 7318 7319]
TRAIN: [    0     1     2 ... 14637 14638 14639] TEST: [7320 7321 7322 ... 8781 8782 8783]
TRAIN: [    0     1     2 ... 14637 14638 14639] TEST: [ 8784  8785  8786 ... 10245 10246
TRAIN: [    0     1     2 ... 14637 14638 14639] TEST: [10248 10249 10250 ... 11709 11710
TRAIN: [    0     1     2 ... 14637 14638 14639] TEST: [11712 11713 11714 ... 13173 13174
TRAIN: [    0     1     2 ... 13173 13174 13175] TEST: [13176 13177 13178 ... 14637 14638
```

The parameters need to be adjusted when complement LSTM model with Python Keras.

- lstm_out. The number of memory units in the LSTM layer.
- number of epochs. Epoch represents one forward pass and one backward pass of all the training examples.
- batch_size. The number of training examples in one forward/backward pass. The higher the batch size, the more memory space you'll need.
- hidden nodes. This is the number of neurons of the LSTM. If you have a higher number, the network gets more powerful. However, the number of parameters to learn also rises. This means it needs more time to train the network.
- input_dim. The dimensions of your features/embeddings. For example, a vector representation of the words in the sentence.
- learning_rate. Indicates, how much the weights are updated per batch.
- dropout_value. To reduce over fitting, the dropout layer just randomly takes a portion of the possible network connections. This value is the percentage of the considered network connections per epoch/batch.
- verbose. By setting verbose 0,1 or 2. 0 will show nothing(silent); 1 will show an animated progress bar and 2 will just mention the number of epoch.

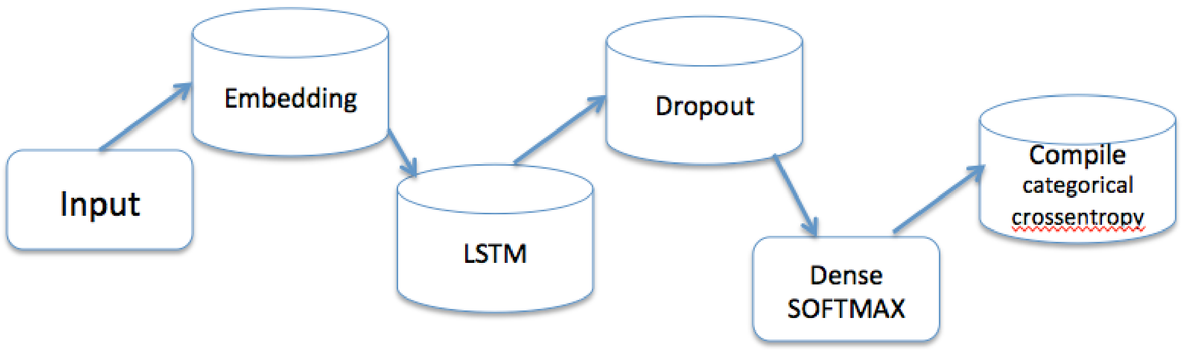


Figure 6: A schema of a simple neural network for LSTM model

After our first run, we found the parameter "epochs" would affect the performance of loss and accuracy. Here, we set number of epochs=10 and calculate loss/accuracy for each epochs value.

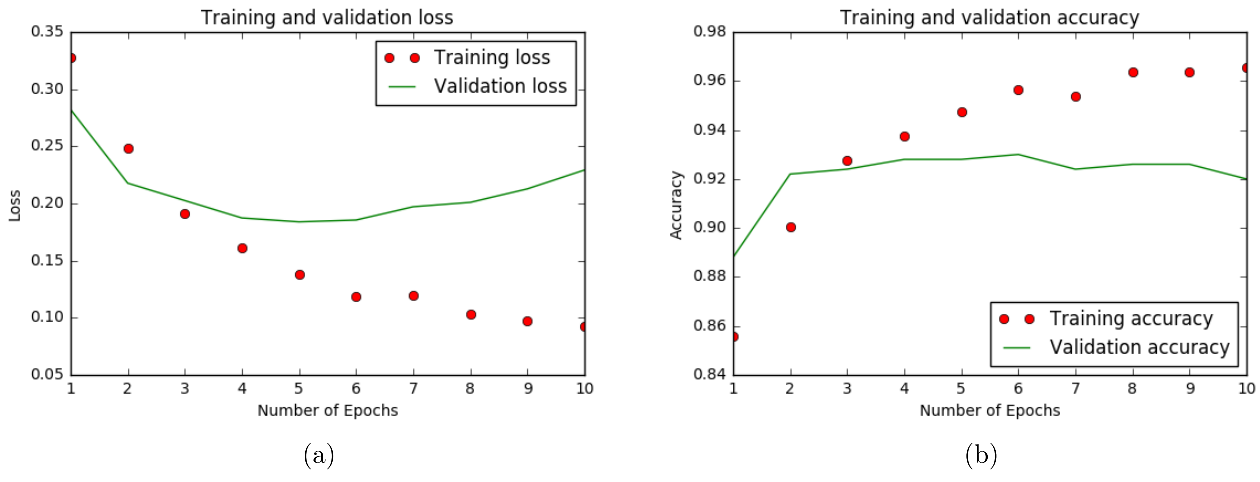


Figure 7: Training and validation (a)loss and (b)accuracy when "epochs" value changes

Methods	Accuracy
LSTM(random split)	92%
LSTM(10-fold CV)	85%
NB	74.5%
SVM	74.4%

From the figures above, the parameter epochs indeed affects the performance of model prediction. We have figured out the highest accuracy of prediction is around 94% when epochs =

3 where training loss and validation loss are almost equal. With a simple LSTM network and few epochs (equal to 3 performs the best), the result shows a higher accuracy compared to Naive Bayes classifiers.

7 Conclusions

In our study of sentiment analysis for two unbalanced datasets with conventional machine learning models and deep learning methods, we found the superior advantage of neural network in large data processing. Even though neural network is known as its complexity, with a simple network and few parameters, neural network could perform better or as well as conventional machine learning models. While for small dataset, conventional machine learning methods such as SVM, Naive Bayes and Logistic Regression still play an important role in classification problems.

Bibliography

[1] Vinay Kumar Jain, Shishir Kumar, and Steven Lawrence Fernandes. Extraction of emotions from multilingual text using intelligent text processing and computational linguistics. *Journal of Computational Science*, 21(Supplement C):316–326, 2017.

[2] Shiliang Sun, Chen Luo, and Junyu Chen. A review of natural language processing techniques for opinion mining systems. *Information Fusion*, 36(Supplement C):10–25, 2017.

[3] Olivier Pietquin. *Chapter 4 - Natural Language and Dialogue Processing*, pages 63–92. Academic Press, Oxford, 2010.

[4] Hongyuan Gao, Erin J. Aiello Bowles, David Carrell, and Diana S. M. Buist. Using natural language processing to extract mammographic findings. *Journal of Biomedical Informatics*, 54(Supplement C):77–84, 2015.

[5] Mazhar Ali Dootio and Asim Imdad Wagan. Syntactic parsing and supervised analysis of Sindhi text. *Journal of King Saud University - Computer and Information Sciences*, 2017.

[6] Jihong Zhang and Xiaoquan Chen. Research and Design of Embedded Wireless Meal Ordering System Based on SQLite. *Physics Procedia*, 25(Supplement C):583–587, 2012.

[7] Ali Khwaldeh, Amani Tahat, Jordi Marti, and Mofleh Tahat. Atomic Data Mining Numerical Methods, Source Code SQlite with Python. *Procedia - Social and Behavioral Sciences*, 73(Supplement C):232–239, 2013.

[8] Fernanda B. Silva, Rafael de O. Werneck, Siome Goldenstein, Salvatore Tabbone, and Ricardo da S. Torres. Graph-based bag-of-words for classification. *Pattern Recognition*, 74(Supplement C):266–285, 2018.

[9] Alexander Richard and Juergen Gall. A bag-of-words equivalent recurrent neural network for action recognition. *Computer Vision and Image Understanding*, 156(Supplement C):79–91, 2017.

[10] Bruno Trstenjak, Sasa Mikac, and Dzenana Donko. KNN with TF-IDF based Framework for Text Categorization. *Procedia Engineering*, 69(Supplement C):1356–1364, 2014.

[11] Wen Zhang, Taketoshi Yoshida, and Xijin Tang. A comparative study of TF*IDF, LSI and multi-words for text classification. *Expert Systems with Applications*, 38(3):2758–2765, 2011.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[13] Johan Jonasson. Slow mixing for Latent Dirichlet Allocation. *Statistics & Probability Letters*, 129(Supplement C):96–100, 2017.

[14] Joshua Charles Campbell, Abram Hindle, and Eleni Stroulia. *Chapter 6 - Latent Dirichlet Allocation: Extracting Topics from Software Engineering Data*, pages 139–159. Morgan Kaufmann, Boston, 2015.

[15] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.