Sentiment Analysis on Email Archives using Deep Learning (Advised by Sudheendra Hangal)

Abstract

I have attempted to use Deep Learning as a feature in SVM to classify email archives into positive or negative categories. I have primarily focused on the email body of users' inboxes. The concept of Deep Learning has been used at each sentence level - which includes labels for every syntactically plausible phrase in thousands of sentences, allowing us to train and evaluate compositional models. This in turn, is applied to SVM as a feature in order to reduce the error.

1. Introduction

Sentiment Analysis is a key topic and has become very instrumental in a large majority of fields such as social media, human resources within corporations, customer relationship management, amongst others. Manning and Socher have successfully applied Deep Learning to movie reviews – here we had certainty that the content of these reviews will contain opinions and words that will most definitely point us in the direction of ascertaining a positive or negative sentiment. I have leveraged RNTN Deep Learning as a feature in SVM. The end goal is to try to utilize it in conjunction with the algorithms we learned in class to achieve the least possible error in the classification of emails into positive or negative. I have done so by analyzing email body of users' and then extracting some sort of positive or negative emotion from it. MUSE (Memories USing Email), a system that combines data mining techniques and an interactive interface to help users browse a long-term email archive. MUSE analyzes the contents of the archive and generates a set of cues that help to spark users' memories: communication activity with inferred social groups, a summary of recurring named entities, occurrence of sentimental words, and image attachments.

2. Dataset

MUSE is installed and run locally from each user's machine; each user specifies one or more sources of email to MUSE, including online POP/IMAP servers or mbox format files stored on a local file system. The user can select folders to analyze from each source and optionally apply filters by date range, or tell MUSE to only analyze their email messages. In the test dataset, I had access to approximately 15,000 public emails of Sarah Palin. Out of which, 5,260 emails have been labeled as positive or negative by MUSE. This data was already labeled and made available for testing. Out of these – 2,277 were negative and positive was 2,465. Greif category accounted for 142 emails, Anger was 376.

Upon running MUSE, the email data is successfully categorized into following categories:

- anger
- congratulations
- family
- festive
- grief
- life event
- love
- medical
- memories
- milestones
- negEmo
- posEmo
- racy
- religion
- superlative
- surprise
- vacations

MUSE inherently maintains its own lexicon and simply applies the bag of words model for classification. It has a lexicon for each of the categories listed above. During classification, it is simply performing a search query for each word of the lexicon on the email body. If a successful match is found, then it classifies the email into that specific category.

3. Multinomial Naïve Bayes

As a first step, I ran Naïve Bayes on the same dataset that MUSE ran on. I was able to do using a simple set up in Matlab. To classify our email messages, I used a multinomial Naive Bayes model. Further, I have used simple cross validation. I have chosen to split the data as 70% as training data (*trainData*) and remainder 30% as test data (*testData*). As we know, one drawback to this approach is wastage of data – in our case; the wastage is about 30%.

3.1 Methodology

I used the test data set for Sarah Palin's emails already provided. The first step was extracting the email body from the given dataset. Metadata associated with the email such as sent date, email headers (from, to, received by, return path etc.) had to also be removed and only email body was taken into consideration. The data was then preprocessed in order to help with the classification effort and accuracy.

3.2 Data Pre-processing

The email body had to be preprocessed in order to help make the algorithm more accurate and also run efficiently, to some extent.

Stop word removal: stop words such as 'the', 'and' etc. have been removed since they provide no useful information about the overall sentiment of the email. Thus, they will not help us determine whether an email is positive or negative.

Removal of non-words: I have removed numbers and punctuation. All white spaces (tabs, newlines, and spaces) have all been trimmed to a single space character.

3.3 Feature Representation

I've taken the uniform-row approach. Representing the features in the way, we are able to have uniform rows whose lengths equal the size of the dictionary. In essence, my feature count is about 58,402 which is basically the size of the vocabulary.

3.4 Results for Naïve Bayes

Here, I used 0-1 misclassification error. We have established that the error rate is about 0.3.

After a quick analysis, I was seeing a lot of duplicate data – same email categorized as positive and negative. Thus, I ran Naïve Bayes again and tried to be more aggressive with the smoothing but was not getting good results as the error rate was still approximately 0.3054 with smoothing parameter set to 1. When I set it to 1.5, the error was 0.2972. As I increased the value of the smoothing parameter, I found the error rate was not decreasing further.

4. Establishing ground truth

In an effort to establish ground truth, I manually labeled all of 5,260 the test emails and categorized them into two main categories of positive or negative emotion. Thus, I encapsulated some of the existing MUSE categories into these two categories as well. Below are the categories I considered:

Negative – negEmo, anger, grief

Positive - posEmo

My main motivation behind choosing these categories was due to the obvious nature of these emotions. As in, other categories such as marriage, life events were a bit ambiguous and were mostly falling into the neutral classification. With anger and grief categories, the strong words used in these categories clearly eliminate the possibility of these emails being neutral.

Upon manually analyzing the results of MUSE categorization, I found there to be a lot of duplicates in classification. For example, an email containing the phrase '*This is a state problem'* was categorized correctly as a negative email due to the word '*problem'*. The same email was categorized as positive due to the word '*proud'*. (2.txt)

I also found the classification to be incorrect in a lot cases. For example, there was a clearly negative and sarcastic email including the phrase '*Sarah Pailn MUST BE KILLED'*, yet, this email has been labeled by MUSE as positive. This is most likely due to the presence of the word '*proud'*. (60.txt)

Thus, it was easily identified there is most certainly room for improvement upon the existing classification model to reduce error.

It can be calculated that MUSE inherently generates an error of about 0.6693[1]. I calculated this error by counting how many of the total MUSE files were not present in the ground truth files for Positive and Negative categories (as established above)

5. Rerun Naïve Bayes and SVM

Once I had correctly labeled data, I re ran Naïve Bayes and SVM on this data. This time, I used the scikit as I wanted to leverage it's built in methods and features such as cross validation. I have used Multinomial Naïve Bayes and SVC with linear kernel. Here again, I used 0-1 misclassification error. I used *trainData* to generate our training and cross validation errors. Using 3 fold cross validation in Grid Search (in scikit), we were able to identify optimized parameter values for model selection.

Using the bag of words model, I found the following:

LinearSVC C [1-10]	Misclassification Cross Validation Error

[1] [Note: this error is calculated based on the logic explained in this report. This error is not derived from any published data on MUSE and hence should not be quoted.]

	= 0.3027 (Least)
SVM with linear kernel C[1-10]	Misclassification Cross Validation Error = 0.275 (Least)
Naïve Bayes	Misclassification Cross Validation Error = 0.2970

6. Deep Learning

Thus, we look into Deep Learning algorithm in an effort to get better results. The goal of deep learning is to explore how computers can take advantage of data to develop features and representations appropriate for complex interpretation tasks. This model works best on individual phrases/sentences, thus it shows great results for analyzing tweets. Our analysis of emails falls in the middle of analysis of large documents and analysis of phrases. I have used the actual email body and broken it down into unique sentences. I have then applied the RNTN Deep Learning algorithm to each sentence – this gives us a rating for each sentence in the email body and we can represent the data as follows:

File Name	Very Negative	Negative	Neutral	Very Positive	Positive
12008.txt	0	0	3	4	0
12010.txt	0	2	1	4	0

Thus, the overall positive or negative score of an email is calculated based on adding all the columns above. In order to improve upon these results, I also then took into account the first 3 sentences and the last 3 sentences of the email. If the initial result is neutral, then we calculate the rating for the first and last 3 sentences. The resulting rating is then assigned to the email. Applying this methodology, I found that the result set still did not improve.

As I saw, there is a large amount of data for individual sentences; I was faced with the problem of aggregating this data and in turn classifying a whole email as positive or negative. Thus I decided to use SVM and implement RNTN deep learning as a feature in SVM. I ran it against the same dataset that we ran Naïve Bayes on. The 0-1 misclassification cross validation error was 0.2949.

Model	**Parameter Value**	**Misclassification Cross Validation Error**
SVC with Linear kernel	C [0.2 to 10.2] with 70 increments	0.3027
SVC with RBF kernel	Gamma [0.1, 0.01, 0.001, 0.0001]	0.2933
Polynomial	C [0.2 to 10.2] with 70 increments	0.3341
LinearSVC	C [0.2 to 10.2] with 70 increments	0.2949

Thus, upon analysis, I found that I was seeing a high bias problem as my data was getting under fitted. Thus, I started exploring more features.

7. Analysis

As we know, the bag of words model is a simple classification disregarding grammar and even word order but accounting for word frequencies. This model works great when we have documents with large amounts of text. Emails do not necessarily classify as large size documents and thus our accuracy applying simple bag of is not very high in section 3 and upon re-run in section 5 the 0-1 misclassification error was reduced to 0.27.

As we see from the results thus far, the error % is fairly high and we need to reduce it down. Application of RNTN Deep Learning algorithm as an SVM feature was not sufficient to improve the results for better accuracy. One conjecture I could draw was that the inaccurate results could be due to the fact that Deep Learning is meant to be applied individual sentences and hence does not to provide reasonable results on larger text such as emails.

Thus, I decided to utilize email meta data, along with Deep Learning as features in SVM algorithm. The meta data features were primarily Day, month, year of email, MUSE lexicon usage. With this approach, I was able to reduce my features from 58,402 to 7. Thus, I was able to capture the sentence level analysis of deep learning and bag of words model used in MUSE.

8. Sentence Sentiment Features, Bag of Words, LDA and more features

I have applied linear SVM to our dataset and implemented the following features:

- Day of email receipt [Mon/Tue/Wed/Thur/Fri/Sat/Sun]
- Month of email receipt [Jan/Feb/Mar/Apr/May/Jun/Jul/Aug/Sep/Oct/Nov/Dec]
- Year of email receipt [2007-2011]
- MUSE positive lexicon [posEmo]
- MUSE negative lexicon [negEmo, anger, grief]
- Deep learning positive sentiment
- Deep learning negative sentiment
- LDA

Day / Month / Year – changing these features seemed to have a significant impact on the result set. It was observed that the number of emails (positive and negative) was significantly increased as the announcement of VP ticket came in.

MUSE (positive/negative) lexicon - we were able to leverage the already established lexicon in MUSE that is maintained by emotion. For the purposes, of our classification – I have used negEmo, anger and grief to represent a negative sentiment. For positive sentiment, I have used MUSE's posEmo.

Deep learning (positive/negative) sentiment – this was applied at each sentence of the email and it seemed to have a big impact on the overall result set.

LDA – We know, given a set of documents, LDA tries to learn the latent topics underlying the set. It represents each document as a mixture of topics (generated from a Dirichlet distribution), each of which emits words with a certain probability. Upon running this on my email dataset, I was able to extract 30 topics from this dataset. Further, I was also able to establish a mapping of all the documents by topic weights. I then used this as a binary feature - which will simply identify whether the email represents a specific topic or not. I found that this feature helped decrease the error a fair amount. Prior to it, I was getting an error of 0.1997 and with this feature in play I got 0.1962. Thus, it was a great boost.

Model	**Parameter Value**	**Misclassification Cross Validation Error**
SVC with Linear kernel	C [0.2 to 10.2] with 70 increments	0.2308
SVC with RBF kernel	Gamma [0.1, 0.01, 0.001, 0.0001]	0.1962
Polynomial	C [0.2 to 10.2] with 70 increments	0.1972
LinearSVC	C [0.2 to 10.2] with 70 increments	0.2268

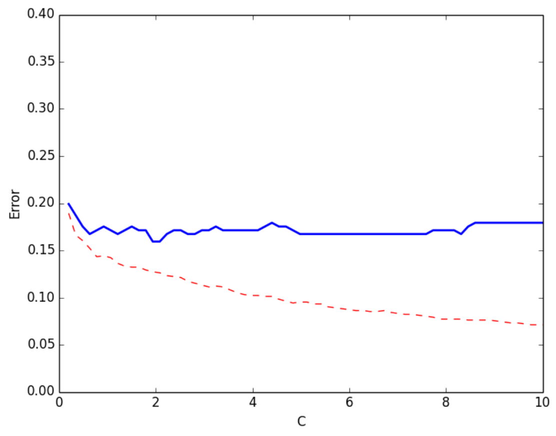
9. Cross Validation / Grid Search and Regularization

I used *trainData* to generate our training and cross validation errors. Using 3 fold cross validation in Grid Search (in scikit), we were able to identify optimized parameter values for model selection. Here, I used 0-1 misclassification error. The optimized parameters are listed below:

linearSVC	C=0.4898	Least Cross Validation Error = 0.2268
SVC with RBF kernel	C=5.1275 ; Gamma = 0	Least Cross Validation Error = 0.1962
SVC with linear kernel	C= 6.7217	Least Cross Validation Error = 0.2308
Polynomial	C=8.7 ; Degree = 2	Least Cross Validation Error =0. 1972

Analyzing the different parameters from grid search, I then validated the parameter options by calculating the training error and validation error on the *trainData*. In this case, I chose the RBF kernel with SVC which seems to be performing the best. I also observed that RBF kernel was giving the lowest error with Gamma set to 0. Next, I tried to establish whether I have a bias or variance problem by analyzing the values of the training and cross validation errors.

By plotting the cross validation error and training error against C, we can infer that our optimized parameter values are indeed correct (for SVC with RBF kernel)



Here, the blue line represents the training error while the red dotted line represents the test error for SVC with RBF kernel.

We had set aside the remainder 30% data to report the generalized testing error. The final error reported is the generalized testing which is calculated by using the optimized parameters identified above and analyzing the training and cross validation error. I found that the generalized error was varying between 0.2235 to 0.2500 for SVC with RBF kernel. For linearSVC, after analysis of cross validation and training errors, we updated the value of C from 0.4898 to 2.6. The final generalization error was 0.2433

10. Summary

Thus, it can be seen that applying SVM in conjunction with RNTN deep learning (along with the features in section 8) as a feature, we were able to get very good results. Our error was reduced significantly and the classification was fairly accurate. I believe this is a great use of RNTN Deep Learning algorithm in the context of sentiment analysis of emails. It has helped us improve upon MUSE's existing bag of words model and its results while at the same time leveraging MUSE's current lexicon. We started out with MUSE misclassification error of about 0.6693 and we ended with a final error of approximately 0.224 thus improving MUSE results significantly.

11. Conclusion and Next Steps

Thus, we can conclude that a mixture of models yields the best results in the context of sentiment analysis on MUSE. Using LDA helped for topic modeling significantly helped reducing the error**.** Now that we have applied RNTN Deep Learning successfully using the sub categories of Anger and Grief, we should extend this further to the other categories of MUSE as well. Currently, we have done a high level sentiment analysis for positive or negative. Future extension would be possible if we had more labeled data for each category. Further, we can use LDA to learn topics that correlated to MUSE categories.

References

http://nlp.stanford.edu/sentiment/

http://nlp.stanford.edu/~socherr/EMNLP2013_RNTN.pdf

http://mobisocial.stanford.edu/muse/muse-papers.html

http://suif.stanford.edu/~hangal/hangal-thesis.pdf

http://nlp.stanford.edu/IR-book/html/htmledition/support-vector-machines-the-linearly-separable-case-1.html

http://scikit-learn.org/stable/modules/cross_validation.html

http://scikit-learn.org/stable/modules/svm.html#classification

http://nlp.stanford.edu/downloads/tmt/tmt-0.4/

https://github.com/echen/sarah-palin-lda

Acknowledgements

I would like thank Sudheendra Hangal for introducing me to MUSE , providing all the necessary source code and his valuable input in the project. I have used the Deep Learning algorithm developed by Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng and Christopher Potts.