Detection of Email Impersonation

Pooja Lakshmi Narayan

December 10, 2017

1 Introduction

The use of emails in our day-to-day communication has become the most preferred way of communication in the present times. With most of our professional and personal communication happening electronically, including those that are sensitive and confidential, it becomes very important to provide a secure email infrastructure for the users to trust the content of the emails as well as the authenticity of the persons authoring them.

The advancement in cryptography and network security has built in the necessary trust in the email systems. However, its infrastructure is still vulnerable to hijacking attempts due to a number of human factors such as using a simple password, theft of personal devices and so on, which would give the attacker a chance to control the personal email account of a user and send emails impersonating the user with a malicious intent.

Every human being has a unique style of writing which would extend to the authorship of emails as well. In the case of the user account being hijacked by an attacker, the style of the messages will be presumably different from the actual user.

In this project, we explore the possibility of detecting authorship of an email by training a neural network over the authors previous messages. The trained model is used to determine whether a given test(or new) email is authored by the same person or not.

Our preliminary results on this task seems promising. We observe 47% improvement over a baseline model of 10% accuracy.

2 Approach

The setting for the task is the following:

A user's inbox receives emails from multiple authors, say,

$$a_1, a_2, \dots a_k$$

.

Let the emails authored by the a_i be denoted by the set $\{E_{a_i}\}$. The system builds a model for this classification task using Recurrent Neural Network(RNN) to be able to classify emails authored by them. Essentially, we have:

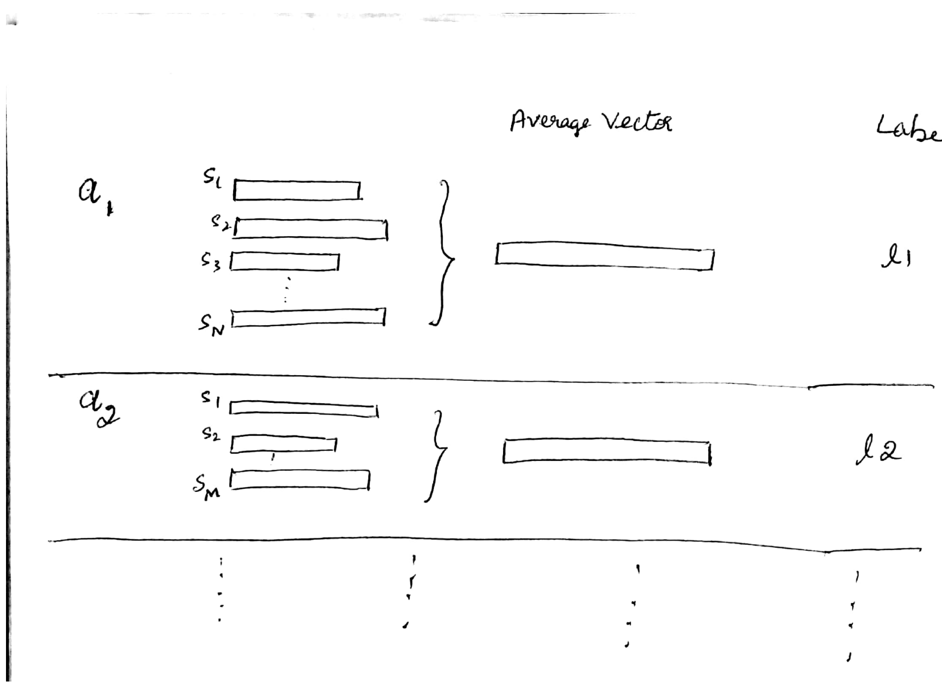

Figure 1: RNN model for two sample authors- a_1 and a_2

Figure 1 The sample authors- a_1 and a_2 have sentences of different lengths. The author1(a_1) has sentences $s_1 \dots s_n$. The author2(a_2) has sentences $s_1 \dots s_m$. A pre-trained embedding for each of the words in the sentence are taken. Running an fully connected RNN forward pass on the sentences to get a hidden representation of each sentence. Then we take an average vector of the hidden representation for each of the authors and its passed through an RNN again, along with projecting it to the size of the labels and then propogating the loss backwards.

Now, for a given email E during test, we perform a (k)-way classification task, to determine whether the email was authored by one of the k-authors in the inbox list of the person. We compare the output of our classifier to the meta-data present in the email header to detemine the accuracy of our classifier.

Implementation details

The platform used to implement the author impersonation system is DyNet.

Before getting into the model details we would like to discuss about data and the pre-processing step. The Enron corpus(link : `https://www.cs.cmu.edu/~enron/`) has emails of the whole organization and to use it suitably for my task, had to write a parser to get the email body and remove the meta-data. In some emails, the previous threads also need to be clipped to get clean data for training my model. We used regular expression to filter out contents other than body, since standard email parsers in the python package did not work well. The data cleaning itself was a challenge in itself, still I do see some noise in the data. But we would like it keep as it as, since during test time as well there would be noise!

The two main modules of the model are `train` and `test`. The first part of the code is reading data from the files, whose filename is the author of the email _ the number specifying the writing sample frequency as extracted from the raw Enron corpus. We are usign a NLTK sentence splitter to get all the sentences of the email and then succesively applying a word splitter on the sentence to get the words of a sentence. Then we convert each of the words to indexes and load pre-trained embeddings for the words. The same is also applied to labels, they are converted to indexes. Once the preprocessing is done, each sentence of this author of the email is passed through a forward_pass function to get its hidden representation. Then a list of hidden representations of the sentences of an email are averaged to get a hidden representation. This then serves as the one which is projected to the size of the label, that goes to the loss function with the true label. The `train` method takes number of epocs as a parameter, the train loss is noticed to be decreasing with each epoc.

To evaluate the model, we implemented a 10-fold cross validation. At each fold, some data is held out for testing and the rest is used for training. There is one more helper method filter_freq_email which clips the data when not many writing samples of that author are not available in the whole inbox. Basically ignoring less frequent author email data, since its hard to generalize with that. We have hardcoded this number to be 10 most frequent author emails that "taylor-m" receives, It can be customized. Finally an evaluate method compares with the gold label and gives the percentage accuracy. The overall average accuracy across folds is also reported.

3 Experiments

Dataset used

ENRON corpus - The coupus has 150 inboxes, of 150 different people working in the organization. Each inbox of a user has multiple folders for sent, inbox, saved and specific forders personal to each user. For my experiment here I am considering a specific user "taylor-m"'s inbox, since he has relatively more number of emails. Having a closer look at the number of different people from whom he gets emails, gives 1370 different classes to classify and around 11,000 writing samples. The number looks big, but emails can be very short and sometimes common emails which may be written by different people appear. Eg:- common classes: Enron.Payroll@enron.com, announcements, American Airlines Net SAAver, dmummery@milbank.com, etc. We have to remove them as they will only confuse the classifier. In our setting inbox- taylor-m/ is used by filtering top 10, gives total 3437 emails which is used as train and test using cross validation.

Results

The metric used for the results is percentage *Accuracy*, which is calculated as the total number of predictions that match the gold label by total number of samples. And since its a percentage, its multiplied by 100.

Average over 10 folds = 58.55 %

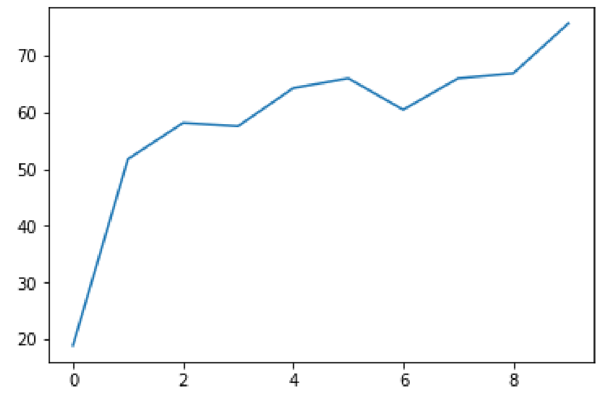

Figure 2: Plot which shows the accuracy for 10folds

Figure 2 The performance shows an upward trend across folds, but that is just coincidence. Each time a new model is created and saved, the model is evaluated on the data that is held out by loading the saved file. Compared to the baseline the accuracy for this data is good, since we have 10 labels, the baseline accuracy is 10%.

Observations

Error Analysis

The confusion matrix can be found below:

```
array([[  0,  16,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0, 173,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,  24,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,  27,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,  14,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,  31,   0,   0,   0,   2,   0,   0,   0,   0],
       [  0,  17,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,  17,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,  14,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   6,   0,   0,   0,   0,   0,   0,   0,   0]])
```

Since the majority class in the training data, out of 3000 writing samples, 1600 are from one specific author (ark.taylor@enron.com), hence the predictions are favouring that class.

It can be seen that not all classes are prediction as 1, on 6^{th} column there is a 2. Hence the model is learning something but not enough to do well on less data classes, such as ones with 100 writing samples.

Since we are taking an average of all the sentences of the email, its hidden representation surely has some authorship information but some of it are lost. Hence concatenating them would mostly capture things like discourse features, but this would increase the train time immensely.

Though trainng time is huge, the time to predict if a given email is authored by the same person or not is almost instantaneous!

4 Conclusion and Future Work

In the error analysis we observed that, with classes of less data, the model makes more mistakes. Hence domain adaptation is one area where we see a potential scope for future work. The idea is that, if we train our model on other authorship attribution tasks which has more data, such authorship of books and save it. Later further train the model using the email data available, then the model might learn faster! The important features for this task would have been learnt, its like getting a warm start instead of cold start.

5 Related Work

The work related to email authorship detection :

1. `http://homepages.inf.ed.ac.uk/ballison/pdf/LREC_final.pdf` – Authorship Attribution of E-Mail: Comparing Classifiers Over a New Corpus for Evaluation

- describes a new dataset which is a subset of enron for authorship identification task. - can be used as reference to filter the enron corpus for our problem - But here compared to our approach more data is used, all the data from sent items of each user is used for training which makes around 2000 writing samples of each person, as opposed to 150 writing samples in our approach.

2. Authorship Attribution of Micro-Messages (IMPT paper):

- `https://homes.cs.washington.edu/~roysch/papers/twitter_authorship/twitter_authorship_slides.pdf` – slides - http://www.aclweb.org/anthology/D13-1193 – paper link - done for short msgs like twitter. very relevant for email which are short - uses some SVM style learning algo. need to investigate further - see usage of k-signatures

3. Mining E-mail Content for Author Identification Forensics (IMPT paper)

- http://math.arizona.edu/ hzhang/math574m/Read/emailAuthorIdentify.pdf - authorship identification on email messages . - again uses SVM for training

4. Authorship identification (more general problem) using NeuralNets: https://web.stanford.edu/class/cs22 – looks like a class project. But nevertheless might contain ideas to be incorporated into email authorship identification

5. Recent paper by Ray Mooney on authorship identification using CNN: https://arxiv.org/pdf/1709.02271. (they use some novels and imdb reviews datasets not email)

6. Paper by Sebastian Ruder on authorship identification using NeuralNets on Email data! (Character-level and Multi-channel Convolutional Neural Networks for Large-scale Authorship Attribution (https://arxiv.org/pdf/1609.06686.pdf) – one of the datasets is enron email dataset.

- But here lots of data is used, its kind of simulated data, not working on real data inboxes, hence accuracy is 10points higher.