Email recipient recommendation

Ayoub Louati Mehdi Kchouk Evann Courdier Xingmiao chen

March 19, 2017

1 Introduction

The goal of this data challenge was to predict potential recipients of an email given past data about the exchanges between users. More precisely, our task was to predict 10 potential recipients for a given input mail.

This is a relevant task as it can allow us to suggest some recipients to the user when he is typing an email. Moreover, we can also warn a user when he is about to send an email to an unlikely recipient and therefore prevent time consuming mistakes and information leakage.

We developed different ranking algorithms to perform this task. We are going to present two of them. The first one is based on neural networks, which we fed in different ways. The second one is a weak-classifier based classification mainly inspired from papers cited further below.

2 First method

Firstly, we tried a Neural Network based method. We first preprocessed the text by removing punctuation, stop-words, and numbers. We then stemmed and tokenized it. We finally removed very frequent words (those that appear in more than 10% of emails, like "hello" or "enron") and that are not discriminative. From this basis, we can play on different aspects to build relevant features.

We can choose different ways of doing word embedding and then sentence embedding, use different ways of ranking and weighting weak classifiers, and use different classification techniques. We will now describe two NN models that gave us our first results.

We learned a word embedding from the email bodies using Word2Vec, and then took the mean of the words of each email as a body embedding. We also tried to use google-news word embedding, without notable improvements. Then we tried to learn one NN classifier per sender, by setting as output the k classes corresponding to the k known recipients of this sender, and feeding the network in two different ways. First, we fed it directly with the embedding of the email. This gave us practically the same results as the frequencies, which hints that the email embedding alone was not really relevant, and the network rather learnt the most frequent recipients. In a second time, we fed it with the result of the scalar product of the email to classify with each email sent by the sender. This second input gave us slightly better results but it was still a very little improvement from the baseline. Considering the little amount of data to train each sender's network, we decided to try another method which was not based on NN.

3 Second method

3.1 Feature engineering

Based on the data pre-processing (as described above) and the following papers [SC15] and [CC07], we selected a list of features that can be categorized into three groups: textual features, temporal features and greeting features.

Textual feature We tried to construct a discriminant feature based on the textual contents of the messages. This feature is based on the K-nearest neighbors algorithm. In fact, for a given message m_u from a user u we start by looking for the n most similar messages in the training set (n: is the number of similar messages to select). To attribute a score of similarity, we started

by deriving the TF-IDF representation for each message in the list of outgoing messages of the user u. Then, we normalized the vector to length 1. Once we have the TF-IDF representation for each message we build the centroid vector by summing up all the TF-IDF vectors representing the messages sent by u. Finally, similarity measure is performed using the cosine distance between the centroid vector and the TF-IDF representation of m_u.
Using the n top similar messages selected from the training set, we computed the weight of each recipient r_i by summing up all the similarities scores for messages in which r_i was included as recipient.

Temporal features In addition to the textual feature, we extracted other frequency features. These features aims at capturing the most common exchanges between users. Consequently using these features we can detect sub communities that exchanges high number of mails and allow us to improve our recipients prediction.
Outgoing Message Percentage (OMP): This feature measures the percentage of emails sent by sender u to recipient r with respect to all messages sent by u. Let us denote the set of messages sent by c_1 to c_2 by: $\mathcal{M}(c_1 -> c_2)$. Then we have:

$$OMP = \frac{|\mathcal{M}(u -> r)|}{|\mathcal{M}(u -> *)|} \tag{1}$$

Where * denotes all the recipients.
Incoming Message Percentage (IMP): In a similar way to the first feature, we define the incoming message percentage that measures the percentage of messages sent to u by r, with respect to all messages sent to u.

$$IMP = \frac{|\mathcal{M}(r -> u)|}{|\mathcal{M}(* -> u)|} \tag{2}$$

Recency: The last temporal feature is used to capture information in latest sent messages. Similarly to the above features, recency is defined as the normalized sent frequency of all users in the training set using only the last m messages (m : is the number of last messages to use). It also appears that this only feature was enough to get significant improvement from the baseline (about 5%)

Greeting feature When we started pre-processing mails in the training set, we noticed the presence of names of recipients in a high number of mails. Moreover, this presence is mostly in the beginning of the mail. Based on this observation and referring to the paper [SC15] we created the greeting feature as follows: We take the beginning of the mail (the first 30 characters) then we split these characters to get names. If the name in the header of the mail corresponds to the name of the considered recipient then the feature is equal to 1 and it is equal to 0 otherwise.

3.2 Model tuning and comparison

In order to prepare our features, we started by creating a directed graph in which nodes represents all users and edges are oriented from senders to recipients and weighted with the number of exchanged emails between the two considered users. The directed graph made it easier and quicker to calculate the OMP and IMP features since it's very well optimized.
Once we built our features, we performed the classification procedure to obtain the best predictions for a given mail.
As presented in the previous section, the different features presented a limited number of parameters which are: the number of similar messages to use in ranking recipients with KNN algorithm and the number of recent messages to use when calculating the recency feature.
We started by using the same classifier and varying the value of parameters to analyze the parameter sensitivity. To do so, we started by evaluating performance for the values cited in the paper that we referred to. Then we tried new values in order to see the effect of each parameter.
When analyzing the results, we noticed that using a higher number of similar messages (n = 50 compared to n = 30 in the paper) gave better results, however for higher values of this parameter the performance decreases. Consequently, we selected $n = 50$ which is the best value for this parameter.
Concerning the number of latest messages to use in calculating recency feature, we tried different values cited in the paper (30, 50, 100) and we concluded that the best value is the highest one.

Consequently, we selected the value 100 for this parameter (m = 100).
Selection of parameters were made using the same classifier in order to focus on the effect of each parameter. In a second time, we tried to vary the classifier that we're using and we analyzed the different results. We used several classifiers among which: random forest classifier, SVC, Logistic Regression and other ones. We got the best results using the Logistic Regression classifier.

4 Conclusion

In the second model that gave us the best results, we addressed the problem as a multi-class multi-label classification task, in which every email address in the user's address book is a possible class to be assigned to the current message. Though this model succeeded in getting good results, it is still far from a perfect recipient prediction. The model is based on feature engineering. We tried to combine two types of features: textual features extracted from message contents and network information based features describing the frequency of exchanges between users and the headers of mails. The model can still be improved by defining new features that can help to better identify recipients.
This last model allowed us to rank $\mathbf{6}^{th}$ on the public leaderboard with 0.4157 accuracy, and $\mathbf{9}^{th}$ on the private one with 0.4017.

References

[CC07] Vitor R Carvalho and William Cohen. Recommending recipients in the enron email corpus. *Machine Learning*, 2007.

[SC15] Zvi Sofershtein and Sara Cohen. Predicting email recipients. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, pages 761–764. ACM, 2015.