Email recipient recommendation - Team Bisous

Olivier Chancé, Sophia Lazraq, Benjamin Battino, Ilana Lascar

March 19, 2017

1 Feature Engineering

1.1 Cleaning the data

- 201 dates have a bad formatting, the year is 0001 and 0002. We correct this by assuming it is actually 2001 and 2002.
- 120 emails have a bad formatting (do not contain @ or do not contain a TLD). The majority of those are actually broken in two parts, and thus can be easily reconstructed.
- For certain features we needed to clean and tokenize email bodies. For each one of them, the following steps where involved: expand contractions (e.g "doesn't" becomes "does not"), lemmatization, lowerization, special character and stop words removal and finally tokenization.

1.2 Features motivation and intuition

The features we used are from the paper *Predicting Email Recipients*, Zvi Sofershtein, Sara Cohen.

For each email, and for each contact in the address book of its sender, we contruct:

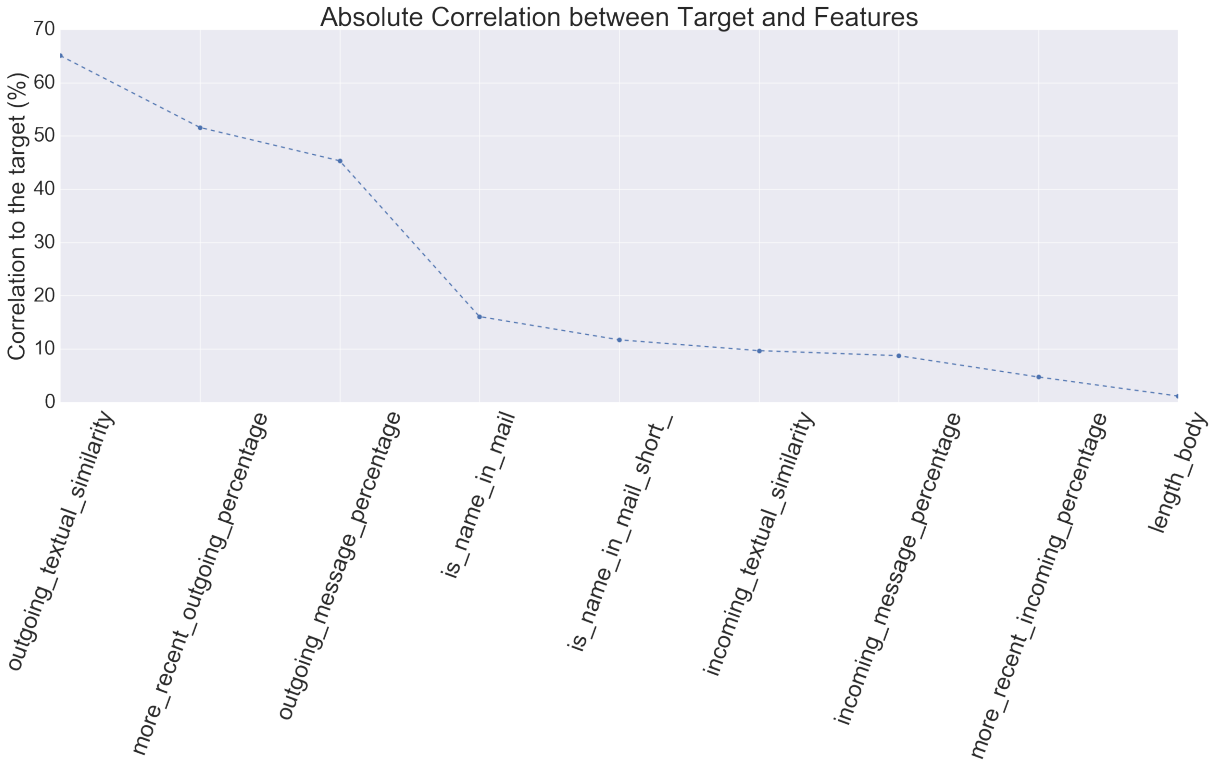
- **Outgoing Message Percentage:** Percentage of messages sent by u to c at time t with respect to all messages written by u.
- **Incoming Message Percentage:** Percentage of messages sent to u by c at time t with respect to all messages written by u.
- **More Recent Outgoing Percentage:** Percentage of messages sent by u at time t since he last emailed c.
- **More Recent Incoming Percentage:** Percentage of messages sent to u at time t since she last received an email from c.
- **Outgoing Textual Similarity:** Given a message m, we search the 30 most similar messages sent by u using BM25 similarity measure. For a given contact c, the feature is 1 if includes a message sent to c, -1 otherwise.
- **Incoming Textual Similarity:** Given a message m, we search the 30 most similar messages sent to u using BM25 similarity measure. For a given contact c, the feature is 1 if includes a message sent by c, -1 otherwise.
- **Is name in mail:** Check if the words present in the mail (often a name) is present in the body.
- **Is name in beginning of mail:** To avoid considering the signature.

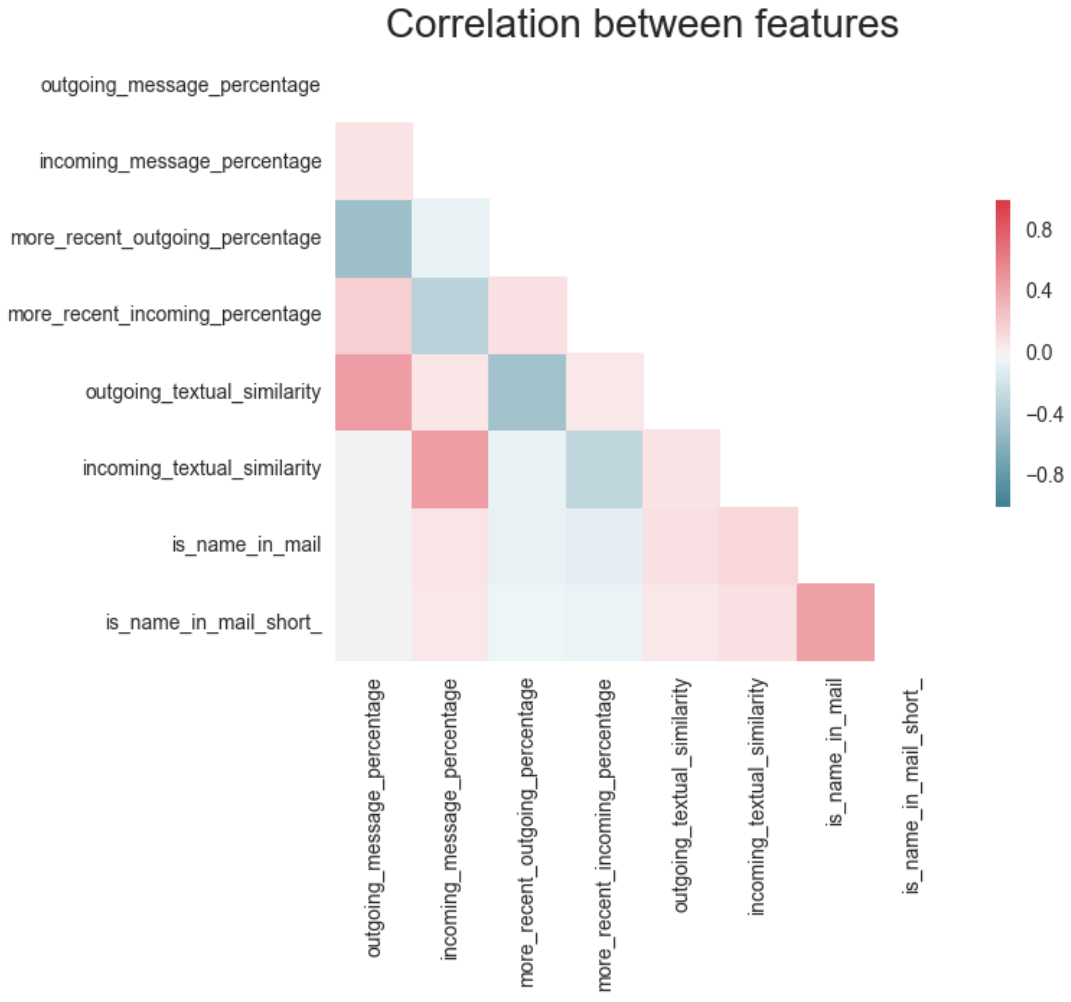
We set the target at $+1$ if the email was actually sent to the contact and to -1 otherwise. For each mail, we took all positive samples and randomly selected the same amount of negative sample to reduce the dataset size and to keep a balanced train dataset.

1.3 Experimentations

We notice that the feature the most correlated with the target is *outgoing textual similarity*(about 66%), followed by *more recent outgoing percentage*(52%) and *outgoing message percentage* (45%). The other features are less correlated with the target.

Analyzing the correlation between features, we notice that the 3 features most correlated with the target are also correlated together, but not more than 40%).


Absolute Correlation between Target and Features
Correlation to the target (%)
70
60
50
40
30
20
10
0
outgoing_textual_similarity
more_recent_outgoing_percentage
outgoing_message_percentage
is_name_in_mail
is_name_in_mail_short_
incoming_textual_similarity
incoming_message_percentage
more_recent_incoming_percentage
length_body


Correlation between features
outgoing_message_percentage
incoming_message_percentage
more_recent_outgoing_percentage
more_recent_incoming_percentage
outgoing_textual_similarity
incoming_textual_similarity
is_name_in_mail
is_name_in_mail_short_
0.8
0.4
0.0
−0.4
−0.8

2 Model tuning and comparison

Our five test sets are composed by the last 10, 20, 30, 40, and 50 last emails sent by each sender, taking the real recipients as positive and the other contacts present in the address book of the sender as negative samples. This allowed us to have a good estimation of the MAP score.

2.1 K-Nearest Neighbors

Another method that we tried was based on K-Nearest neighbors algorithm described by Yang Liu. For each message m_u from a user u that he addressed to a set of recipients $R(m_u) = [r_1, ..., r_{|R(m_u)|}]$, we compute its 10 most similar messages in the training set. In our case, in order to compute the similarity, we used the cosine distance between the text of two normalized TF-IDF vectors. Then, with these 10 most similar messages selected from the training set, we found the weight of each recipient r_i aaccording to the sum of similarity scores of the neighboring messages in which r_i was one of the recipients. Once all the |R| recipients in the given message are ranked according to this method, we selected the one with the lowest score.

Given our feature engineering, this algorithm performs the worst. It results in a mean average precision @10 of 0.27, which is far behind the other methods explained below.

2.2 TfIdf-Centroid

This method consists in deriving the message's TF-IDF vector representation $\vec{m_u}$ from its textual contents. This is done for all messages m_u sent by user u in the training set. The vector representation is then normalized to legth 1. We compute the centroid using the following formula (the same notions as mentionned in K-N neighbors): $\vec{centroid}(r_i) = \sum_{m_u | r_i \in R(m_u)} \vec{m_u}$. When testing, we compute cosine similarities between the message's TF-IDF vector representation and the adress-book |AB(u)| centroid vectors related to the given user u. Similarly to the previous method, the recipients are ranked according to the cosine similarity scores. We get a MAP @10 of 0.31.

2.3 Classification algorithms

We tried several classification models, with a result in probability, that allowed us to rank the potential recipients and to keep only the 10 most likely. XGBoost gave the best score among the classification algorithms.

Model	Precision
XGBoost	0.39
Random Forrest	0.37
Extra Trees	0.37
Logistic Regression	0.38

Table 1: Precision for classification algorithms

2.4 Neural Network

By cross validation, a neural network with one hidden layer, 100 hidden units and a L2-regularization parameter equal to 0.0001 gave promising results. That's the learning classifier we decided to use to train our model and that yield our score on the leaderboard. The training set was composed of all recipients that indeed received a mail. And as the classifier has to learn how to recognize a recipient from someone that is not, we included for each email some adresses that were not part of the recipients. In fact, for each email, the same quantity of positive samples and negative ones - randomly selected - were included in the training set. In the test set, all recipients that appeared in the sender's address book were included.

2.5 Conclusion

Neural Network and Xgboost perform the best in contrast with K-N neighbors and Centroid algorithms. We eventually used Neural Network (tunned by cross-validation) to get our score.

References

- Zvi Sofershtein, Sara Cohen. *Predicting Email Recipients*
- Yiming Yang, Xin Liu. *A re-examination of text categorization methods*