Kaggle in class:

Master Data Science/MVA data competition 2017

Report By : TeamX
MEZZI NARJES & OUMOUSS EL MEHDI
Master 2 Data & Knowledge (Télécom Paristech)

Data Challenge 2017

Problem:
Email recipient recommendation systems are systems capable of suggesting the recipients of an email while it is composed.This is done based on the current content of the mail as well as network information (e.g. number of mails sent to a certain recipient) and previous recipients. These systems are a valuable addition to professionals in the corporate world, as it helps preventing misunderstandings caused by forgetting to add a recipient, information leakage, etc. Where the need to build effective email recipient recommendation systems.

Goal: Based on the available data set, develop a system capable of recommending a list of 10 recipients ranked by decreasing order of relevance.

DataSet:
To implement a solution to this problem, we had to start from somewhere, and there is no better place to start from than the data at disposal. The fact that the domain word 'enron' came out in the majority of the emails, let us wonder what is this 'enron' about, to find out that behind this data there is a whole long story of fraud and bankruptcy of the Enron Corporation. The **Enron Email Corpus** is a large dataset, consisting of emails generated Enron's employees.

Why this dataset is very important ? "This data is valuable; to my knowledge it is the only substantial collection of "real" email that is public" [3]

Solution:
After searching about the Enron Email Corpus, we read several articles and papers. One of the papers that we appreciated the most was [1], thanks to its simplicity and the fact that it is pretty well explained. Next, we explain some of the approaches treated in this paper.

Approach 1. Tfldf-Centroid

As explained in [1], this technique is based on cosine similarity between two TF-IDF vector-based representations of the email messages.

- **Training Phase -Feature Engineering-**

For every message in the training set sent by a user (sender):
- Using the mail's body, we get its TF-IDF vector representation.
- We normalize the vector.

For every recipient in the Address Book |AB| of the user:
- We build a TF-IDF centroid vector, which represents the sum of the normalized TF-IDF vectors of all messages that were sent from the sender to a recipient.

For the TF-IDF representation , we used Scikit-learn's TfidfVectorizer that enables also to remove stop words, as well as ignore words with frequency that exceeds a certain amount which helps reduce a little bit the dimensionality of the message's' body.

- **Testing Phase**

For each message in the test set:

- Get the TF-IDF vector representation of the message
- Compute the cosine similarity between this representation and the |AB(u)| Centroid vectors.
- Finally, Rank the |AB| recipients based on the cosine similarity scores .

<u>Approach 2.</u> K-Nearest-Neighbors

In this second approach, we applied the Knn-30 method presented in [1]. But due to computational constraints, we considered only the 20 nearest neighbors for each email. To implement the method we proceeded in the following way :

- Created for each sender a csv file which contains its emails in the training set.
- For each sender :
    - From the created file, build a corpus.
    - Retrieved the email we wish to predict its recipients from the test set.
    - Added the email from the test set to the corpus.
    - Calculated the TF-IDF vectors of the different emails.
    - Calculated the similarity between the test email and the training emails using the cosine distance.
    - Chose the 20 emails the most similar to the test email.
    - Pick all the recipients of the 20 chosen emails and calculate the similarity score of each one. (the similarity score is the sum of the cosine distance between the test email and the training emails to which the recipient belong)
    - Chose the 10 recipients having the greatest similarity score.
- Add all the result to a commun csv file.

<u>Results:</u>

Having the Mean Average Precision @10 as a metric of evaluation, we got a precision of 0.18 for the Tfidf-Centroid and 0.21 for the KNN. The second approach performed slightly better. We think this is thanks to the fact that there is two layers of similarity calculation and also that in the KNN approach we first get the nearest emails in matter of similarity then we operate on the recipients belong to these emails. However, both approaches remained weak compared to the given baseline, but more importantly compared to machine learning methods. And this is obviously due to the fact that these two approaches :

- Didn't take the 'date' into consideration
- Neglected many network-based features (see section 3.2 in [1])

Proposition:

We tried also to attack this problem from a machine learning perspective, by looking to the prediction of recipients as Multi-label classification problem.

We did follow the next steps, which included some data preparation, especially :

- Merging the two given datasets (training-set and training_info based on the 'mid', same for Tfidf-Centroid, just for convenience).

- Apply some transformation (use of dictionaries as mappings between the emails and given ids, in order to deal with numerical data rather than text)
- Use of tfidfVectorizer to get a numerical representation of the message body.
- Once the data was ready, we applied some machine learning algorithms (OneVsRestClassifier + LinearSVC/SGDClassifier, KNN, RandomForest)

N.B: Constrained by limited computational power at disposal, the ML experiments were conducted on small subset of the data. Also, the results lacked a sense of ranking, which we thought can be added ulteriorly to pick the top 10 predictions.

DC17 Takes off:

Working on this Data Challenge was of great importance to us, in the sense that we learnt a lot in matter of :

- Text processing (NLP; data preparation and cleaning).
- Different approaches to take on the same problematic.
- How different people can use the same data, yet draw different insights and deploy it on different applications.

References:

1. Vitor R. Carvalho, William W. Cohen, 'Recommending Recipients in the Enron Email Corpus' http://www.cs.cmu.edu/~wcohen/postscript/cc-predict-submitted.pdf
2. Scikit-learn http://scikit-learn.org/stable/index.html
(feature_extraction, , ...)
3. Enron Email Dataset, https://www.cs.cmu.edu/~./enron/

For more readings !

- **Investigating Enron's email corpus: The trail of Tim Belden (Neo4j)**

https://linkurio.us/investigating-the-enron-email-dataset/
Neo4j a graph database management system, applied to the Enron Email Corpus.

- **Intelligent Email: Aiding Users with AI**

http://dirichlet.net/pdf/dredze08intelligent.pdf
Artificial Intelligence application to Emails (summary keyword generation, **reply prediction** and attachment prediction)

- **Intelligent Email: Reply and Attachment Prediction**

https://www.cs.jhu.edu/~mdredze/publications/dredze_intelligent_email_iui08.pdf

- **EnronData - Research**

https://enrondata.readthedocs.io/en/latest/references/research/
Some research studies conducted on the Enron Email Dataset.

- **Network Analysis with the Enron Email Corpus**

http://ww2.amstat.org/publications/jse/v23n2/hardin.pdf
Study relationships in a network by applying centrality measures.

- **SNAP : Stanford Network Analysis Project** (Python)

https://snap.stanford.edu/snappy/index.html
https://snap.stanford.edu/data/email-Enron.html
A general purpose network analysis and graph mining library.

- **Towards Hierarchical Email Recipient Predictions**

https://www.youtube.com/watch?v=r1wVAz3opSE
Presentation of paper, 8th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing (IEEE CollaborateCom 2012)