Genpact Email Classification

The solution will be based on PredictionIO. There will be multiple instances, one per client, each client having their own data. The number of categories in the classification and the categories themselves may vary by client.

The classifications are: AP processing errors, Audit request, Bank queries, Bank rejections, BMG, CBCP escalation, CBCP request, Claim status, Concur issues, Contract Related, Credit Card Maintenance, FYI, GL requests/ PC uplifts, Invoice Status, Invoices for scan, JBA/SMF Maintenance, Manual payment request, Matching Report, Missing documents/ supporting documentation, Non AP documents Non PO Invoices (coding information, VAT), OIR, Open Item reports, Other, out of office messages, Payment confirmation, Payment reminder, Payments (MPR, urgent payment requests, BMG), PO related, Process updates, Proof of payment, Recall, Two Way match report, Urgent payment request, Vendor master data, Vendor statement, Web Ex Trainings.

Report

PredictionIO is an open source machine learning server. The company behind PredictionIO was acquired by Salesforce.

PredictionIO has some readily available templates for text classification. The official template will act as the base and will have to be customized for Genpact. The template uses Scala, but the customization may be done in another language such as Java.

A web service will be created (as part of the customization of the PredictionIO classification template) for use by the Genpact clients.

PredictionIO will use any of PostgreSQL, MySQL, or Elasticsearch/HBase.

Software

Name: PredictionIO

Version: v0.9.6 (latest)

Estimated annual cost: None, since PredictionIO is open source.

Link to licensing information: https://docs.prediction.io

PredictionIO is licensed under the Apache License, Version 2.0. See https://github.com/PredictionIO/PredictionIO/blob/master/LICENSE.txt for the full license text.

Diagrams

Relationships between Salesforce Clients, Web Services (part of the solution), and the PredictionIO instances:

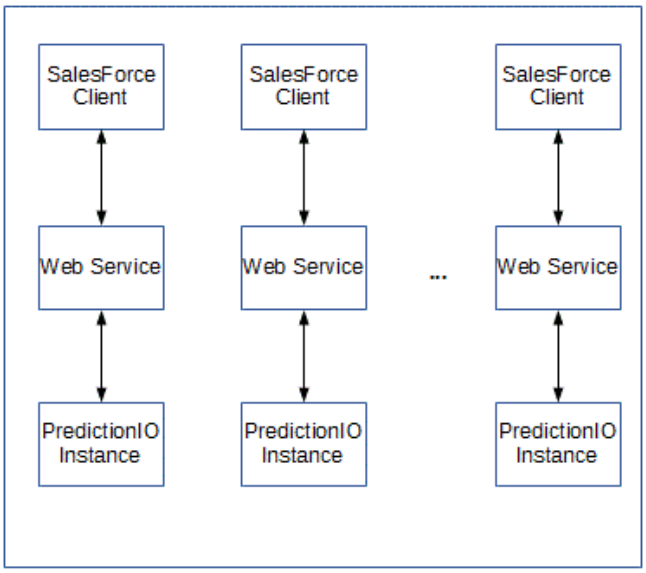
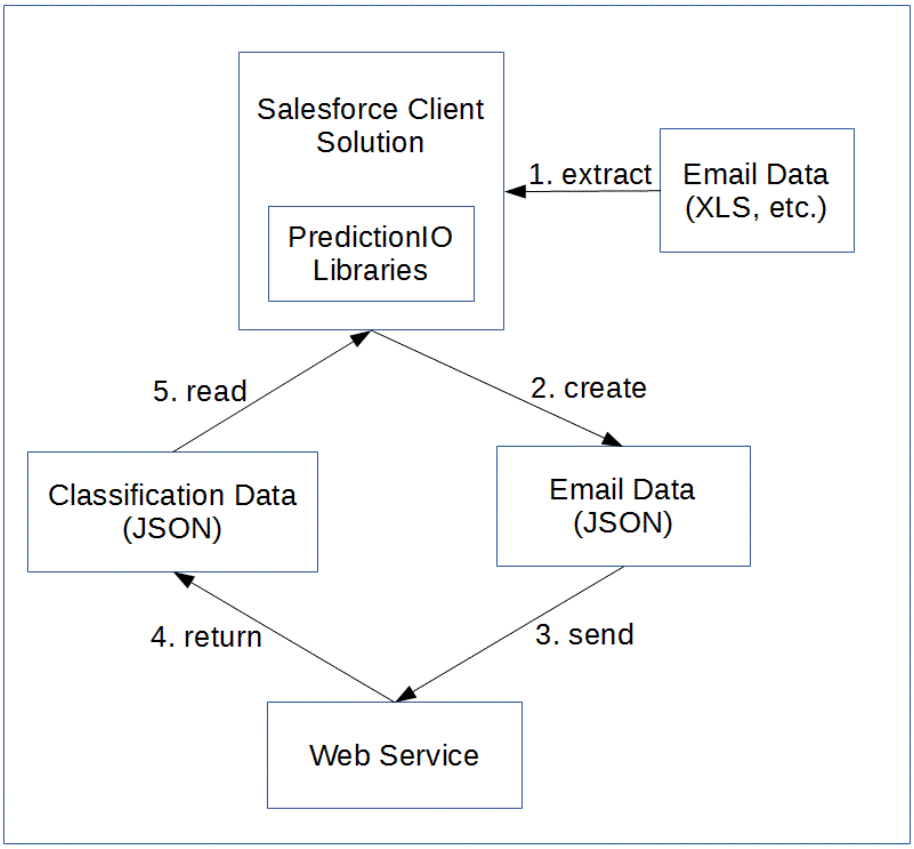


Diagram to show the Extraction Process and use of the Web Service:



Algorithms

Name: *General Flow*

Supervised Learning will be used to train the engine to classify emails.

General Pseudo-code:

1. Install and Run PredictionIO
2. Customize the Engine using the PredictionIO template
3. Data Collection and Training (feed the engine with combinations of email attributes with the expected email classification)
4. Deploy the Engine as a Web Service
5. Use the Engine by Sending combinations of email attributes and getting the classification

Step 3 will have to be repeated if re-training the engine.

Name: *Supervised Learning Using Naive Bayes*

This is the default algorithm used by the PredictionIO template. Naive Bayes is commonly used for classifying emails on whether they are spam or not. In our case we will be using this algorithm for classification into different categories.

The Naive Bayes-based engine will predict the email classification based on email properties: "Sender Email", "To Email", "Subject", and "Html Body".

Naive Bayes Classification uses the existence and frequency of certain words in the text to classify the whole email into one of the several categories, taking into consideration that certain keywords are most probable to fall into certain categories.

PredictionIO mentions in their blog that they have more than 8000 developers and 400 apps using their platform. Being part of Salesforce, their solution will be used extensively with SalesforceIQ.

Name: *Extraction and Use of the ID, System, Other ID, Sender Email, Count, To Email, and Subject*

Since we will be using the web services exposed by the installation of our customized PredictionIO solutions, we need to convert incoming data to JSON. If we are given Excel files containing the email data, extraction into JSON should just be straightforward using an Excel library such as Apache POI (if using Java).

Name: *Handling New Categories*

Introduction of new Categories will require that our engine be re-trained.

Name: *Handling New Variables*

Introduction of new Variables will require that our engine be re-trained.

Name: *Handling Classification Errors and Incorporating Feedback*

Errors in classification will require that our engine be re-trained. The erroneously classified email should be used as part of the training set, along with the correct classification to expect.

Name: *Handling Unknown Classifications*

If our engine cannot come up with the classification for certain instances of data, this will require that our engine be re-trained. The unclassified email should be used as part of the training set, along with the correct classification to expect.

Other Information

Scalability:

Since we will be using different instances for different clients, our solution must be more scalable as compared to using a monolithic solution to be shared by all clients. Should a single instance be not enough to service a certain client, we can scale our solution like how we would scale any web service. We can use a load balancing solution which uses several instances of a web service.

Degree of Scalability:

Since our solution is tiered and not monolithic, we can scale parts of the solution and not just do a wholesale replication of all components when we need to scale. If we are running low on storage we can scale only the backing store used (PostgreSQL/MySQL/Elasticsearch/HBase) and leave the other components as is. If we just need more responders to requests we just add load-balanced services behind our load balancer and leave the other components as is (such as the backing stores).

Advantages and Disadvantages of Naive Bayes Classification:

- Performs well in avoiding false positives. Not just because an email contains the word “Invoice” will it be automatically classified as Invoice if it should really be classified in another category.
- Bayesian poisoning is used by spammers in their attempt to evade classification of their emails as spam, but it is probably unlikely that Genpact customers will use this technique to deliberately have their emails classified in the wrong category.

Accuracy:

Naive Bayes by itself can be very accurate in classification after training but several measures can still be done to maintain high accuracy:

1) train with a substantially large set of samples
2) deal with rare words
3) use additional heuristics such as ignoring certain words (the, a, an) and/or use a combination of words when classifying and not just individual words.

References:

Quick-start document on how to install, train, and use a classification engine: https://docs.prediction.io/templates/classification/quickstart

Using a different algorithm instead of Naive Bayes with PredictionIO: https://docs.prediction.io/templates/classification/add-algorithm/

Naive Bayes classifier wiki: https://en.wikipedia.org/wiki/Naive_Bayes_classifier

Naive Bayes spam filtering wiki: https://en.wikipedia.org/wiki/Naive_Bayes_spam_filtering