GenPact Email Classification Ideation Contest

Document version: 2.0 (post- checkpoint feedback). NOTE: Please evaluate just this version, not the first one.

Topcoder member: **ged**
Submission March 22, 2016

Introduction

Propose a system to allow extracting important data from incoming emails sent to GenPact and classifying them based on contents into a known set of categories (A process currently done manually).

Other requirements (from contest descriptions and forums answers):

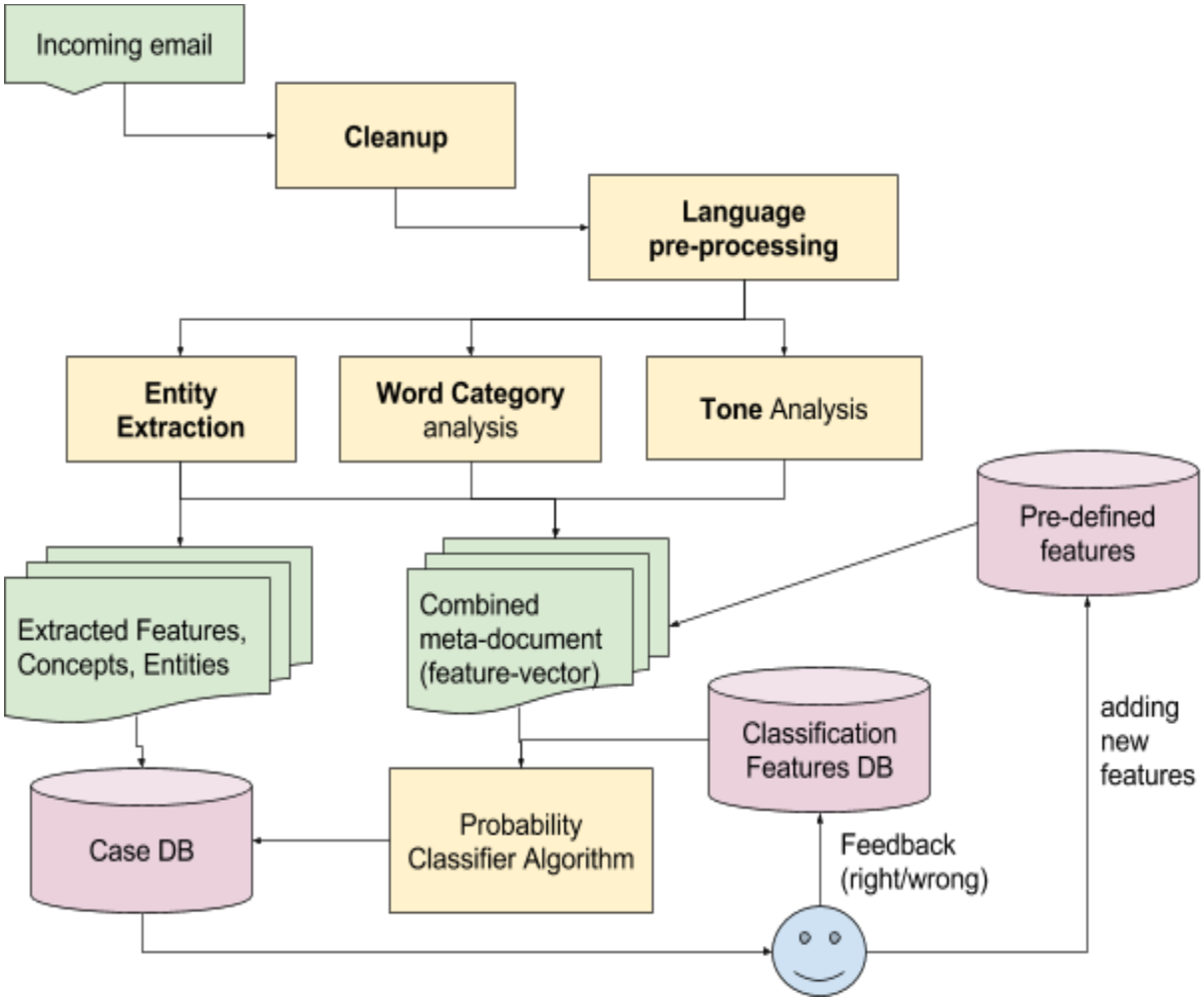
- Use of Salesforce as preferred platform
- Multi-tenancy, or at least the ability to reproduce the solution to different customers
- There will exist different email classifications
- The system must be able to "learn" or receive feedback
- Some cost estimate of the solution needs to be provided

The sample classification contains the following classes: AP processing errors, Audir 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.

Solution

Design

The following describes a summary of the classification task. Note, this is largely independent of an implementation and how to fit in different architectures - that goes to later sections.



The process described flows as follows:

- An incoming email is accepted by the system. This triggers the process.
- The email (and thread) is first cleaned up. This removes HTML and rich text formatting, and also headers and footers like salutations and signatures (This would generate noise to classifiers and all further processing).
- Some language pre-processing is applied. At the very least, a language identification service[1] needs to run to determine which is the language of the message. If the later processing does not support this language natively, then it may be necessary to do a machine-translation[2] -- although this approach (autmatic translation) may not be optimal.
- What follows is a series of analyses extracting features from the document (message):

[1] Watson Language Translation: http://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/language-translation/api/v2/#identify Can identify languages as well as translate.

[2] Again - Watson Language Translation can be used, or Google Translate APIs (and there are dozens of providers)

    - An Entity Extractor[3] will analyze the document to discover **concepts, entities, topics, relationships**. This is all automated so each feature comes with a cofidence - used later on as another score or probability.
      For example, an email may be classified as "Finance", "Invoices", "Payments" (which may not correspond 1:1 to client-assigned categories, but will help the actual classifier).
    - We will also run a language (dictionary) analysis such as LIWC[4] to extract word **categories**.
      For example, certain percentage of words are about the "money" category, or using "inclusive" or "3rd-person" words, etc. This is, again, input for later stages.
    - A third processor may (optionally) analyze Tones or Emotions in the message.[5]
      Usage of this is conditional to the client -- whether they find useful to know if a message is highly emotional (e.g. an overdue payment) or very tentative or highly social…
      Actual features that can be extracted, and how they map to the client's domain, is to be discussed -- this is why this is mentioned as optional and feedback from the client is welcome.
      The output would be, again, certain scoring for each of the tones or emotions (or in general, categories) analyzed.
- Part of the output of the analyzers is a series of data which may be useful in the information extraction mentioned in the contest -- but there is no much detail about it. As the analyzers can extract entities such as "123-456-7890 of type PhoneNumber" or "Mr. Doe of type Person" or "tomorrow of type Date", they can serve as input for the last module -- when a case is created/filed.
- The other part of the output of the "analyzers" above is a series of scores -- one can think of a "feature vector" which dozens (or hundreds) of category => weight (or score) mappings.
- The feature vectors are analyzed with probabilistic algorithms. They will use an existing dataset (from a training stage, classified manually) to assign probabilities for the document to belong to each of the known categories.
  An obvious input is, then, the training set -- which should come from far more than the 300+ documents provided as example in the contest.
  Another input can be certain known categories marked as the client as determining one category with high probability.
  The implementation for this stage is open, but we could use Bayesian classifiers, or simply run regression tests and come with an equation based on the N categories (a

---

[3] For example AlchemyLanguage http://www.alchemyapi.com/products/demo/alchemylanguage

[4] LIWC - Linguistic Inquiry and Word Count by Pennebaker : http://liwc.wpengine.com/
This is the most widely known, there are alternative implementations too.

[5] For Tones, see Tone Analyzer
http://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/tone-analyzer.html (with alternatives) and AlchemyLanguage for Emotions.

linear regression in its simplest form, but more complex models can be used). Further experimentation would be necessary to find which method works best. But this is partly independent of the design of the overall system.

- The cassification output ends up stored as a Case in the CRM or flagging the email with a tag in the email client, etc. Additionally the metadata (features extracted) can be stored in the same CRM or offered to the agent when processing the case, for easier filling of form (e.g. invoice number offered if type==Invoice, it if was extracted, or customer master data pre-populated with data from the message).

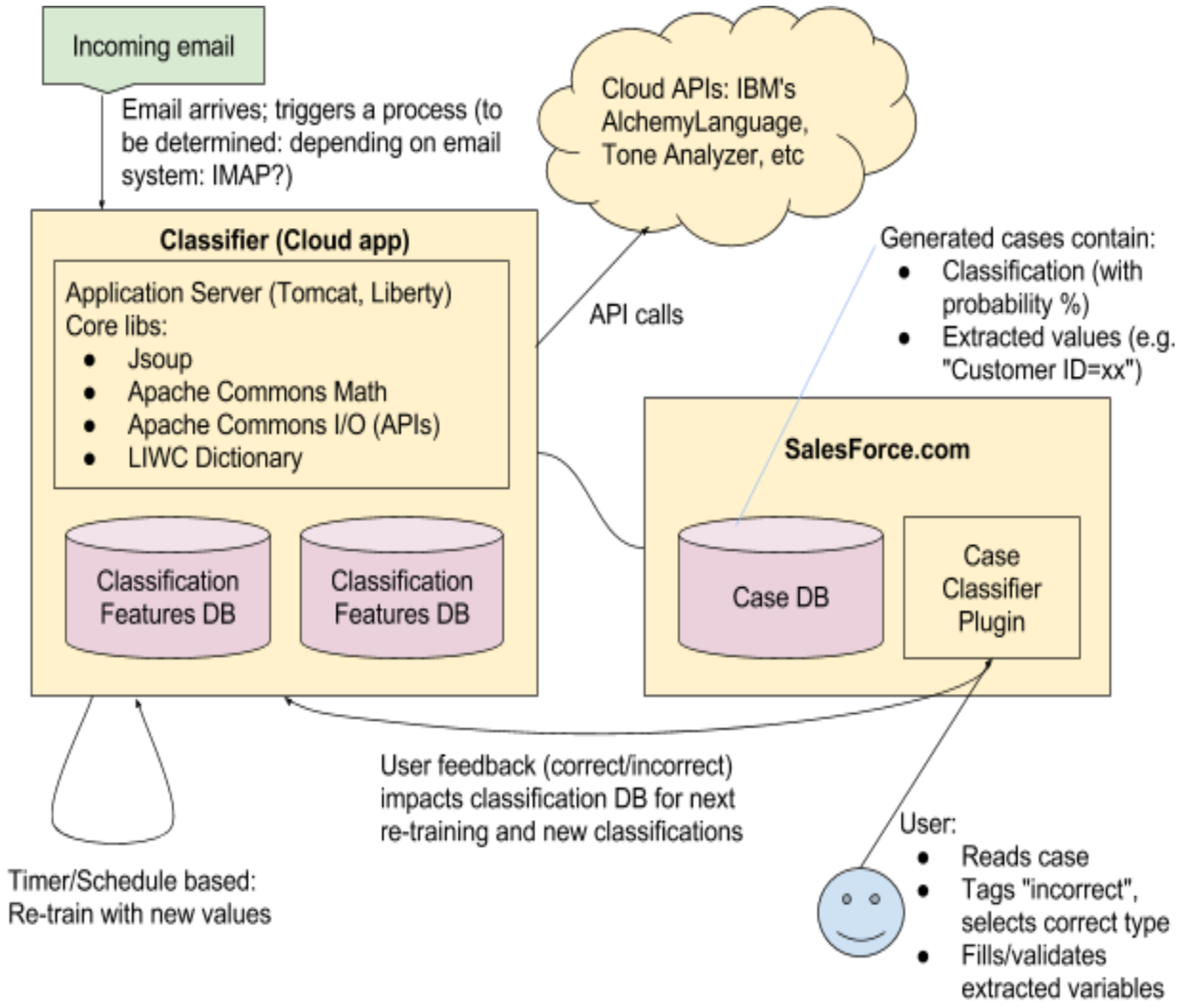
The outlined steps are sufficient for classification for a static system. Additionally, the users when using the system will be able to:

- Flag messages (cases) as correct/incorrect from the classification. This goes to the features database and serves as input for further training iterations of the system.
- Identify unique features as 'highly important' to classify a message. Specific phrases, or features (like an "invoice number") that map or most likely (with high probability) map to specific class.

Architecture

The previous section describes the core of the idea -- ie. how we can train, classify and extract variables from the emails and convert them into cases.

What follows is a proposed architecture for this system.



So the "classifier system" itself is basically what was described in the previous section. It will be implemented in Java and run in a web application container such as Apache Tomcat 7.x or IBM Websphere Liberty runtime. This can run in any hosting platform, and it may depend on the skills and IT platforms already in use by the client -- but two recommended platforms could be:

- Amazon web services
- IBM Bluemix (for its tight integration to IBM's Watson APIs). In this case, Websphere Liberty is natively supported so it is easier than Tomcat, although the latter is still feasible.

This web application will run the following components:

- Crawl (or receive updates from) the email system, retrieve new emails
- Run the core algorithms: cleaning text etc.
- Call the text processing APIs (feature exaction etc.)
- Run the actual classifiers
- Run linear regressions when re-training the model

- Call SalesForce APIs to store the email as new case -- with all the derived metadata including extracted variables and the actual assigned "class".
- It also serves part of the UI for the Case viewer plugin (such as buttons for re-classify, give user input)

Added features

Variable Extraction.

Note that the original contest statement was fuzzy about "extracting" values from the input emails. However, given the processing done and the information available with the proposed algorithms, it is also possible to extract values such as names, emails, telephone numbers, case IDs or numbers, etc. They can be added as metadata to fields in the SalesForce cases, and just be left available for the user (for example the user could be suggested those values when filling a case form) or, when the information is clear and confidence is high, field values can be added themselves.

Incorporating feedback

The proposed system is not intended to be static. While it should be trained with an original dataset, it will "learn" from users feedback to improve classification accuracy.
The system will periodically re-train the models, doing the same regressions proposed for the original training, except it will have more data over time.
The proposal is to have the user "flag" cases a being classified correctly or incorrectly. Then , the extracted features of an email marked 'correct' will be given more weight than other features just found with higher correlations to the inferred classes. Alternatively, when a user flags something as 'incorrect' this will also have a strong weight in the training, just with a negative correlation (This is standard for training a probabilistic system).

Implementation Details

Software

The following describes the software to be incorporated in the solution.
Note that some (or all) components may have alternatives; some of them will be described in this document, generally accompanied with pros/cons and a preferred choice.

First about the platform -- the contest only specifies SalesForce. It is not clear if the software must run in the SalesForce CRM or any of the SalesForce solutions (e.g. cloud). Is is also not clear if, even if working with the CRM, the system can run externally and feed SalesForce or must be a SalesForce.com plugin. In any case, I feel the following architecture is generic

enough to fit in any of the scenarios mentioned, and can be refined to a second iteration after the interim evaluation.

Technologies and products to use

Note: For pricing, see further below.

- Cleanup / preprocessing: OpenSource libraries like JSoup (Java), and some custom algorithms/heuristics (e.g. to remove signatures). These are free. Includes:
    - Either Apache Tomcat 7.0.x (Apache Foundation, Apache License)
      Or Websphere Liberty Profile by IBM.
    - Jsoup 1.8.3 [6] by Johnathan Hedley. MIT License.
      For parsing HTML
    - Apache Commons Math 3.x (stable 3.6) by Apache Foundation. Apache license.
      To implement linear regression and Bayesian filters.
    - Apache Commos IO 2.4 (stable) by Apache Foundation. Apache license.
      For calling external APIs
    - LIWC 2015 by Pennebaker Conglomerates Inc. Licensing is custom; ask for Commercial license.
- Watson Language Translation APIs, by IBM. *Cloud service (get latest version)*
  http://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/doc/language-translation/
  Language identification and/or translation.
- AlchemyLanguage APIs by Alchemy (an IBM Company). *Cloud service (get latest version)*
  https://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/alchemy-language.html
  For entities and concepts exaction
- Watson Tone Analyzer by IBM. *Cloud service (get latest version)*
  http://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/tone-analyzer.html
  For measuring Tone
- Actual classification will use well known techniques such as Bayesian filters
  https://en.wikipedia.org/wiki/Naive_Bayes_spam_filtering
  The implementation is simple and there are plenty of academic sources with ideas, and as-is licensed code snippets.
- For storage, a simple database will suffice, either relational or object-based (json) or non-relational.
- The service itself can run as a webserver with APIs in some cloud environment, and be invoked from SalesForce plugins, or live within SalesForce itself.

[6] http://jsoup.org/download

Performance Considerations

As explained above , processing each email involves a series of steps, including some algorithms and invoking some API calls.
All tested APIs above work well under a second per API call for moderate texts. And the algorithms (like html cleanup or the actual classification) are simple and there exist very efficient implementations. So processing each email can take a few seconds, let us say well under 5 seconds to be classified (and most likely about 1sec if calls are parallelized).
In forums feedback it was stated "not in the order of milliseconds" [7] so this timing should be sufficiently good.

Multitenancy

It was suggested in the statement that the solution must support multiple customers, or be deployed to different environments. I do not see the need of specifying this on the environment. All the models above can contain classifiers and data tailored to each client, not interfering with each other. So a single system, single database, etc. would be able to act as multi-tenant and working with each of the classifiers customized to each customer. This is my recommended solution.

However, there can also be multiple instances to support multiple customers, if this is a fixed requirement (e.g. if for compliance reasons data cannot be stored in same hosts as other clients' data, or cannot live in certain countries/regions/datacenters).

Pricing

There are two parts in the cost model. One is to license and run a monthly service (hosting, platform-as-a-service apps, database storage, etc.). Exaggerating costs, let us assume $300/month.

Some software like LIWC requires purchasing a one-time license , which can be about $89 - not adding this to the calculation as it is not significant [8]

The second part of the model is about processing emails - as some of the solutions proposed are api-based. Most of the non-free solutions are APIs, cloud-based - with the benefit of being upgraded continuously and not having to maintain them -- and based on a pay-as-you-go model.

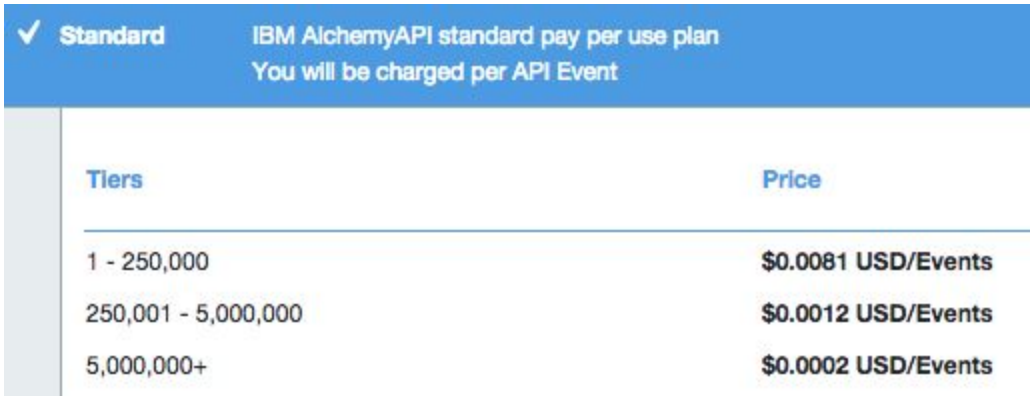
---

[7] http://apps.topcoder.com/forums/?module=Thread&threadID=879137&start=0
[8] http://liwc.wpengine.com/

The IBM Watson solutions below are offered on Bluemix (the PaasS IBM offering) but they can be used outside Bluemix as just APIs, so I include the raw API costs (excluding apps cost which would not be used).
For estimation purposes, we will assume an average email is 200 words long.

These are some APIs costs. Note that there are mutliple vendors and alternatives

- **Language Translation**: Free first 1M characters (200,000 words); then $0.02 for each 1000 chars (~200 words).
  Cost: 2 API calls / message, about **$0.04 / email**.
  http://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/language-translation.html#pricing-block
- **AlchemyLanguage**
  Depends on the system usage. Assuming <250,000 emails processed per month (worst, most expensive tier), this is **$0.0081 / email** (this runs all feature extraction, concepts, etc. in the same API call).



  http://www.alchemyapi.com/products/pricing -- URL takes you to https://console.ng.bluemix.net/catalog/services/alchemyapi (which requires login -- price snapshots taken March 16 2016)
- **ToneAnalyzer**
  This service is in Beta, free of charge. It is expected to go GA soon. Given it is a Watson service, let us use an educated guess of **$0.04 / email** as Language Translation.

There was no accurate information about sizing, but in the forums it was suggested "from 100s to 1000s on a daily basis" [9]. Let us assume 1,000 emails/day. Cost of email from APIs perspective is about $0,09. This brings $90 / day, or $2,700 / month.

So the total monthly cost of operating the system would be about $3,000 / month for a processing load of 1,000 daily emails.

[9] http://apps.topcoder.com/forums/?module=Thread&threadID=879137&start=0

Evaluation

Some of the technologies proposed are not well known (relatively new to the market) so I am including a preview of what they can do in the form of a Proof of Concept.
Remember that it is not expected that these services will classify the messages themselves -- rather, they will create some categories to serve as input to the actual classifier

Here a cleaned up version of message in row #11 of the sample spreadsheet:

> *Dear CustomerPlease find attached your account statement as of 32/20/2025.We look forward to receiving your payment in accordance with the agreed payment terms as stated on your invoice. If you have any questions regarding your invoice please contact us at custserv@bocaaabbb.*

I am showing what some of the proposed services can extract from there.

AlchemyLanguage

Demo: http://www.alchemyapi.com/products/demo/alchemylanguage

Taxonomy:

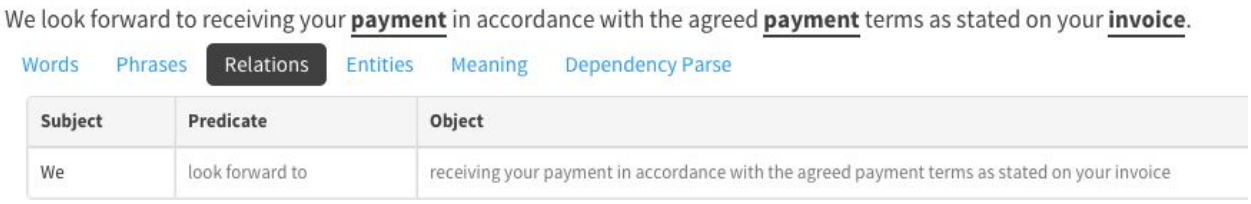
Label	Score	Confident?
/finance/accounting and auditing/billing and invoicing	0.593598	
/finance/personal finance/lending/credit cards	0.373357	no
/finance	0.213071	no

Keywords:

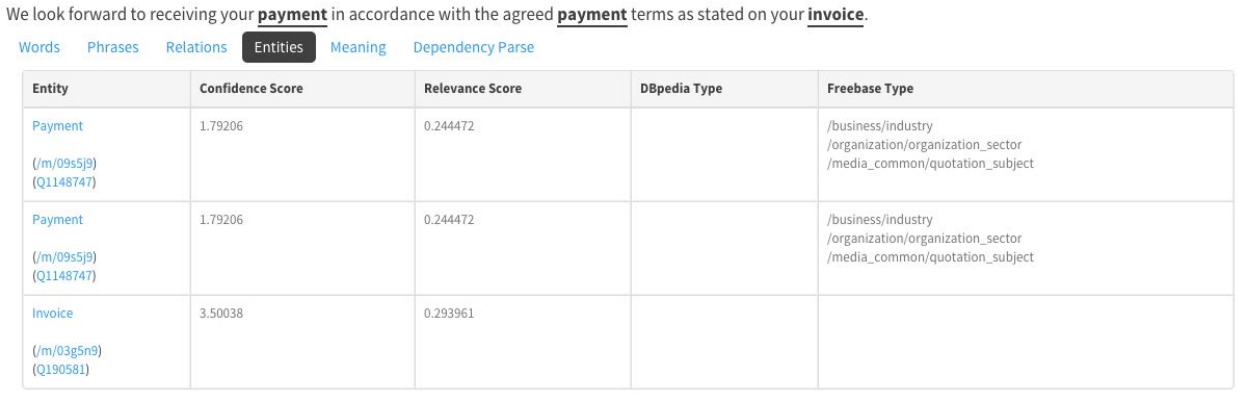
Keyword	Relevance
agreed payment terms	0.907599
invoice please contact	0.705278
Dear CustomerPlease	0.671021
account statement	0.597848
accordance	0.423721
questions	0.344932
custserv	0.34407
bocaaabbb	0.343855

TextRazor (competitor - alternative to Alchemy)

Demo: https://www.textrazor.com/demo
Relations:



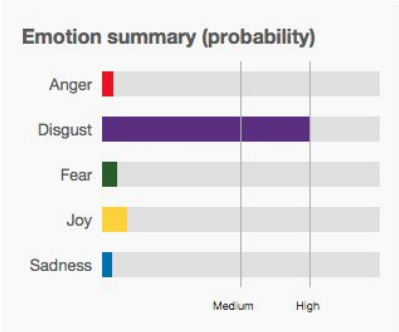
Entities:

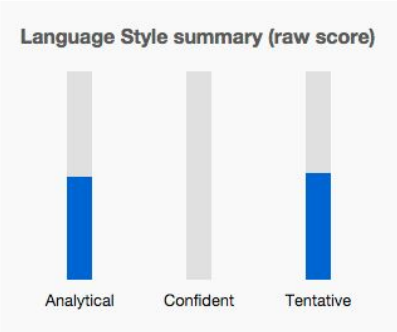


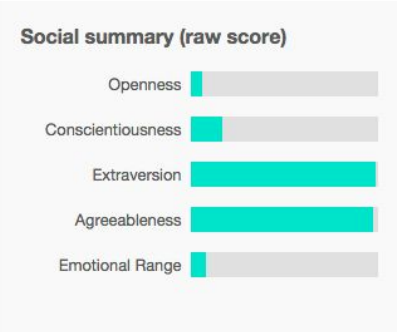
Topics:

Watson Tone Analyzer

Demo: https://tone-analyzer-demo.mybluemix.net/







Jsoup

While this is not really a new tool, it is depicted here to show what it does well: flexible parsing of HTML and (optionally, but specific to us) plain text extraction.
This is a Java library. Code would be something like: cleanContent = Jsoup.parse(html).text();
Online demo: http://try.jsoup.org/