# Project Description and Work Plan

### Repository

https://gitlab.eeecs.qub.ac.uk/40180175/distributed\_nlp\_emails

### Problem Statement

Email management can take up a significant amount of times for individuals in the workplace, who often find themselves bombarded with emails constantly. Attempting to identify which emails are worth opening and reading can be cumbersome and slow. This project describes a number of machine learning models which can be used to aid the management of emails, and improve general efficiency of a task many of us are required to perform numerous times each day.

#### References

[1] [The social economy: Unlocking value and productivity through social technologies](https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/the-social-economy)

### 

### Solution Description

Creation of a pipeline to process bulk email files (.eml).

These emails will be provided by Proofpoint, an email security company with whom I’ve been working for over the last year. All emails are required to be scrubbed of any personally identifiable information (PII) before being processed, in order to comply with regulations. All PII which is identified will either be removed or replaced with generated alternatives.

This pipeline will consist of: \* Categorization of emails into topics such as ‘Announcements’, ‘Business’, ‘E-Commerce’, ‘Social Media’, etc. \* Identification of emails requiring a response, while identifying the degree of urgency required, such as high, normal or low urgency. \* Summarization of emails to an optimal length while maintaining relevance to matter at hand.

An API will serve the trained model, so the application can be offered in real-time at the email scanning stage. The model may also be served directory in the browser.

A basic email client GUI will be created to showcase the change in workflow when managing emails with this mined information.

### Overview of Core Technologies and Infrastructure

* Apache Spark: Distributed parallel processing framework
* Docker: Containerized builds.
* Kubernetes: Orchestrator of containerized instances.
* Kubernetes Operations (KOPS): Tool to ease the creation of clusters.
* Amazon Web Services: Cloud computing facilitator of hardware.
* Apache Parquet: CSV like file-type will partitioning capabilities.
* Google Snappy: Compression algorithm suitable for Parquet.
* Amazon Simple Storage Service (S3): Distributed data store.
* Terraform: Reproducible infrastructure as code.
* Helm: Kubernetes application provisioning library.
* Apache Airflow: Jobs scheduler and pipeline orchestrator through use of directed acyclic graphs (DAGs) of execution.
* Jenkins: Enable continuous integration and continuous delivery.
* TensorFlow: Machine learning library with emphasis on Deep Learning (DL).
* Horovod: Distributed DL framework compatible with TensorFlow and Spark.
* React: JavaScript UI library.

### 

### Core Languages

* Python
* JavaScript/TypeScript

### 

### De-identification tool

Using non-domain specific pre-trained models, create a pipeline to consume raw email files, identify personally identifiable information (PII) and replace this information with alternative ‘fake’ data.

The emails will be sourced from S3. In terms of the amount of emails, my aim is to work in the multi-million scale.

For entity recognition, I’ll likely be using spaCy and one of it’s pre-trained models, specifically the en\_core\_web\_lg model, which has been trained using the OntoNotes corpus.

Here’s a basic example of spaCy identifying entities.

By no means is this a perfect solution, as spaCy has clearly missed some ‘personal’ data. But with the use of Faker, we can obscure the data with generated alternatives, making it far harder to identify an individual with this email. If the quality of the fakes were enriched, this effect would be amplified. Faker seems to only concatenate names together and call this a valid company name, which isn’t fantastic.

As part of the mail de-identification, the emails components will be parsed, and elements extracted. The output of the process will be an Apache Parquet file.

These are the emails which will be used to train the forthcoming models.

#### Acceptance Criteria

* Approval by Proofpoint’s data controller on the legality side of things.

#### References

* [1] [spaCy Named Entity Recognition](https://spacy.io/usage/linguistic-features#named-entities)
* [2] [Faker Providers](https://faker.readthedocs.io/en/master/providers.html)

### 

### Categorization of emails

Standard classification of unstructured text into categories. Since we won’t know the categories ahead of time, this will encompass topic detection as well. Overall a popular topic within NLP, specifically around medical text and social media.

#### References

* [1] [Online Learning for Latent Dirichlet Allocation](https://papers.nips.cc/paper/3902-online-learning-for-latent-dirichlet-allocation.pdf)
* [2] [Email Classification with Machine Learning and Word Embeddings for Improved Customer Support](http://www.diva-portal.org/smash/get/diva2:1189491/FULLTEXT01.pdf)

### Identification of actionable emails

More domain specific challenge and likely to be the trickiest. Work done on this topic for emails specifically, intertwined with intent understanding in text. Once identified, attempt to sort by priority/urgency (low, normal, high). Perhaps the number of requests in text should be useful.

#### References

* [1] [Detecting Emails Containing Requests for Action](https://www.aclweb.org/anthology/N10-1142.pdf)
* [2] [Classifying Action Items for Semantic Email](https://pdfs.semanticscholar.org/beed/b0bac9657fe61dd3910c411aa45b49e57f96.pdf)
* [3] [Extracting Tasks from Emails: first challenges](https://medium.com/@rodrigo_23805/extracting-tasks-from-emails-first-challenges-86e7fbbf4672)
* [4] [Context-Aware Intent Identification in Email Conversations](https://www.microsoft.com/en-us/research/uploads/prod/2019/05/Wang_SIGIR19.pdf)
* [5] [Characterizing and Predicting Enterprise Email Reply Behavior](https://cseweb.ucsd.edu/classes/fa17/cse291-b/reading/sigir17a_email.pdf)

### Summarization of emails

#### Overview

Another more general goal of text modelling. Unlikely to cause too much trouble. The problem at hand can be viewed as unstructured text once again, and hence is fitting for unsupervised learning. Will likely be phrase based, as opposed to word based for the categorization.

#### References

* [1] [A Survey of Unstructured Text Summarization Techniques](https://pdfs.semanticscholar.org/9064/f2a72907bdc78116ff07f551a0b2302ebcfc.pdf)
* [2] [Email Summarization-Extracting Main Content from the Mail](http://www.ijircce.com/upload/2015/october/141_Email.pdf)
* [3] [Text Summarization Techniques: A Brief Survey](https://pdfs.semanticscholar.org/12a5/0024da4b1ad71ddab2fb68785dc56c2e540f.pdf)

### Model Serving

TensorFlow has fairly extensive guides on serving their trained models. Ideally model will be served as an API (likely created using Python), and hosted as part of Kubernetes cluster. An alternative approach offered by TensorFlow is serving models through JavaScript, which may integrate well with the email client component.

#### References

* [1] [Serving Models](https://www.tensorflow.org/tfx/guide/serving)
* [2] [TensorFlow.js](https://www.tensorflow.org/js)
* [3] [TensorFlow on Spark (Yahoo)](https://github.com/yahoo/TensorFlowOnSpark/blob/master/test/test_pipeline.py)
* [3] [Horovod (Baidu + Uber initiative)](https://eng.uber.com/horovod/)
* [4] [Petastorm](https://github.com/uber/petastorm)
* [5] [Multi Worker Mirrored Strategy (Experimental)](https://www.tensorflow.org/guide/distributed_training#multiworkermirroredstrategy)
* [6] [Horovod vs CARS in 2018](https://www.logicalclocks.com/blog/goodbye-horovod-hello-collectiveallreduce)
* [7] [Deep learning with Horovod and Spark using GPUs and Docker containers](https://conferences.oreilly.com/artificial-intelligence/ai-eu/public/schedule/detail/78122)

### Email Client

Creation of a progressive web app (PWA) / single page application (SPA) using React with TypeScript to view emails while displaying the capabilities of the above models and showcasing the change in workflow (hopefully more efficient).

For example \* Automatically creating folders per topic. \* Allow searching per discovered topic. \* Have flag for urgency - can view actionable mails only \* Instead of showing a truncated version of the email body, show the summarized version when showcasing all mails.

Attempt to emulate GMail viewing capabilities, but worth noting this will only be used to view mail, will not be a fully fledged email client.

### Deadlines

|  |  |  |  |
| --- | --- | --- | --- |
| Start | Length | Finish | Goal |
| 11/10/19 | 1 Week(s) | 25/10/19 | De-identification tool complete |
| 25/10/19 | 2 Week(s) | 08/11/19 | Classification task working locally |
| 08/11/19 | 1 Week(s) | 15/11/19 | Creation of packages for deployment |
| 15/11/19 | 2 Week(s) | 29/11/19 | Scripts for automated creation of cluster |
| 29/11/19 | 1 Week(s) | 06/11/19 | Manual working classification in a multi-node Spark cluster |
| 06/11/19 | 1 Week(s) | 13/12/19 | Completion of interim report |
| 13/12/19 | 2 Week(s) | 27/12/19 | Automated job execution using Airflow |
| 27/12/19 | 2 Week(s) | 10/01/20 | Summarization task complete |
| 10/01/20 | 3 Week(s) | 31/01/20 | Actionable task complete |
| 31/01/20 | 2 Week(s) | 13/02/20 | Serving of models |
| 13/02/20 | 3 Week(s) | 03/03/20 | Creation of React email client |
| 03/04/20 | 3 Week(s) | 24/04/20 | Completion of dissertation and project |