BECCA: Bulk Email Content Classifier & Analyzer

Steve Trush, Shrestha Mohanty, Shannon Hamilton

December 2016
Info 256: Applied Natural Language Processing
Professor Marti Hearst
UC Berkeley, School of Information

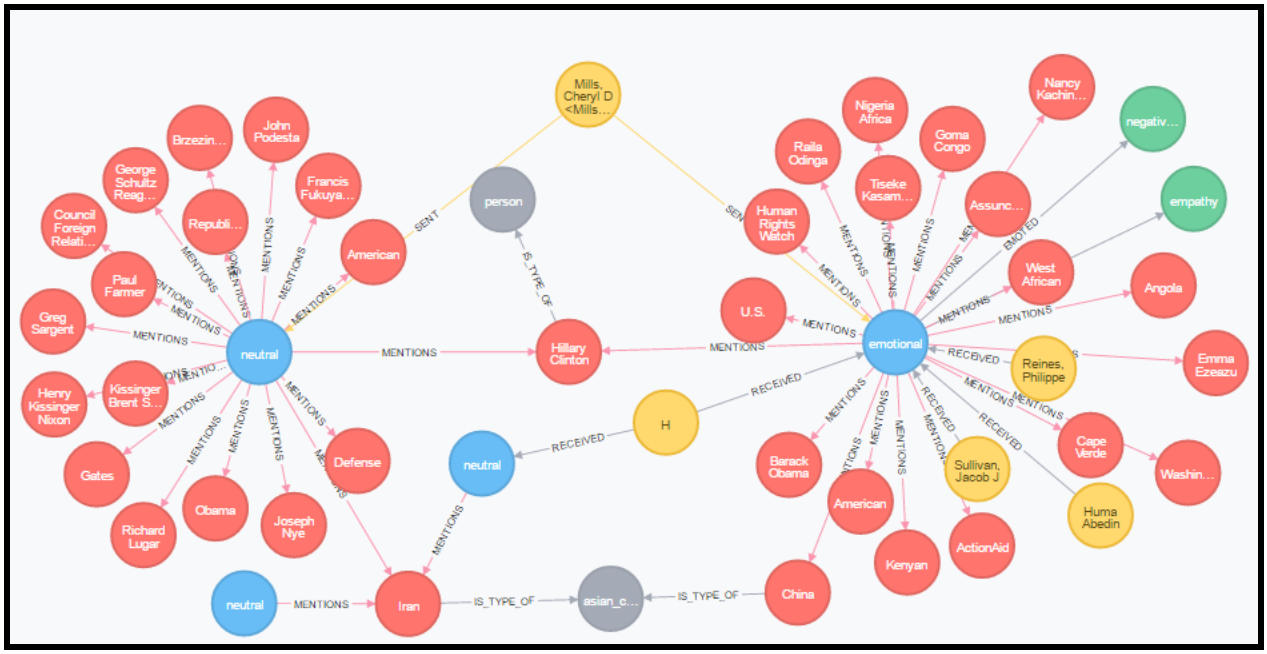


Table of Contents

Introduction

In the last decade, as funding of reporting by traditional media has dwindled, a number of new actors - non-profit organizations involved in various forms of investigative journalism - have emerged in the news industry with the aim of promoting openness, transparency and civic engagement. From the Pentagon Papers and Enron to the more recent Hillary Clinton private server scandal and the Democratic National Committee hacking, mass email collection releases and leaks have become a common form of exposing and challenging corruption. In this digital age, the ability to view, study and use this information is growing ever more important to the media, government, and public. Yet because email collections are composed of complex chains of conversations written in informal and abbreviated speech and lack a 'gold standard', analyzing the content of these collections can prove difficult.

For our final project, we used natural language processing and machine learning techniques on Hillary Clinton's State-released emails to extract important people, topics and tone and visualize them in a graph relational database (Neo4J).

Other bulk email collection tools like MIT Media Lab's Immersion platform (https://immersion.media.mit.edu/) have been used to effectively vizualize the meta-data of the emails - even using Clinton-related email collections (https://clinton.media.mit.edu/clinton). However, these tools do not allow users to explore the content of the email beyond their superficial metadata, especially not by content or emotional categories. While metadata analysis can reveal interesting relationships between email users, the ability to explore a network of email by content or topic would allow users to answer questions related to 'What is so interesting?' without sifting through thousands of emails manually.

Related Work

The recent growth of Internet usage has created large amounts of online, user generated content across social media, blogs, emails, and forums. Much of this content utilizes an informal style of writing that often disregards typical grammatical rules and uses a mixture of abbreviations and context dependent terms. Because our dataset primarily contains informal language, we reviewed three papers to understand the challenges and opportunities when applying NLP techniques to this type of text: Chang and Sung's (2013) paper on applying name entity recognition to informal text, Li et al's (2015) article about extracting entities from business emails, and Medlock's (2008) paper on investigating classification accuracy techniques for informal text.

Additionally, since its release in 2002, the Enron Email dataset has been a popular source of study for many NLP researchers. Their research served as a helpful cornerstone to our keyword and classification strategy. We found Hardin et al's (2015) paper on network analysis in the Enron email corpus to be a useful resource, as well as a blog post by Tribolet (2016) about using Neo4J, our graphical database of choice, to explore Enron's emails.

We also reviewed a number of articles that explored sentiment and tone classification in informal text. Much of our strategy for classifying emotional tone leans on the framework described in Das and Bandyopadhyay's (2010) paper on identifying emotional expressions

in informal text. We have also chosen to integrate VADER (for Valence Aware Dictionary for sEntiment Reasoning), a model for general sentiment analysis after reading a paper released by VADER's creators, Hutto and Gilbert (2014). Zac Stewart's (2014) blog post about pipelines and FeatureUnions was also critical to helping us build our custom transformers in our classifier. While we did not explore power dynamics directly, we found papers by Bramsen et al (2011) and Danescu-Niculescu-Mizil et al (2012) to be relevant and related resources should we pursue further work.

Lastly, we reviewed Jason Brownlee's blog post to improve our classification model given the imbalance of neutral and emotional emails in our dataset.

Project Goals

As outlined in our project proposal, our project goals and measurements of success consisted of the following:

1. Classifying emails according to sentiment / tone
    a. **Measurement of success**: At least 70% classification accuracy for pre-tagged selection of emails.

2. Extract keywords and tone from an email
    a. **Measurement of success**: We will have our keywords and tones peer-reviewed by 10 peers. They will be evaluated on a scale from 1 to 5 (1 being keyphrases and tones that are unrepresentative of the email, 5 being very representative). Success is an average score of 3 out of 5.

3. Build a graph database from the extracted / categorized data
    a. **Measurement of success:** A graph database is created.

4. Design the organizing system to support future work such as advanced entity extraction and resolution
    a. **Measurement of success:** An organizing system is outlined.

Methodology

Data

Where did the data come from?

We secured our data from Kaggle (link here). In August 2016, the State Department released nearly 7,000 pages of Clinton's emails. The documents were released by the State Department as PDFs. The owners of the dataset cleaned and normalized the released documents and hosted them for public analysis on Kaggle. Here is the link to the GitHub repository that creates the data we used.

The data file was 14MB and contained approximately 7900 emails in both SQLlite and CSV file formats.

How was the data processed?

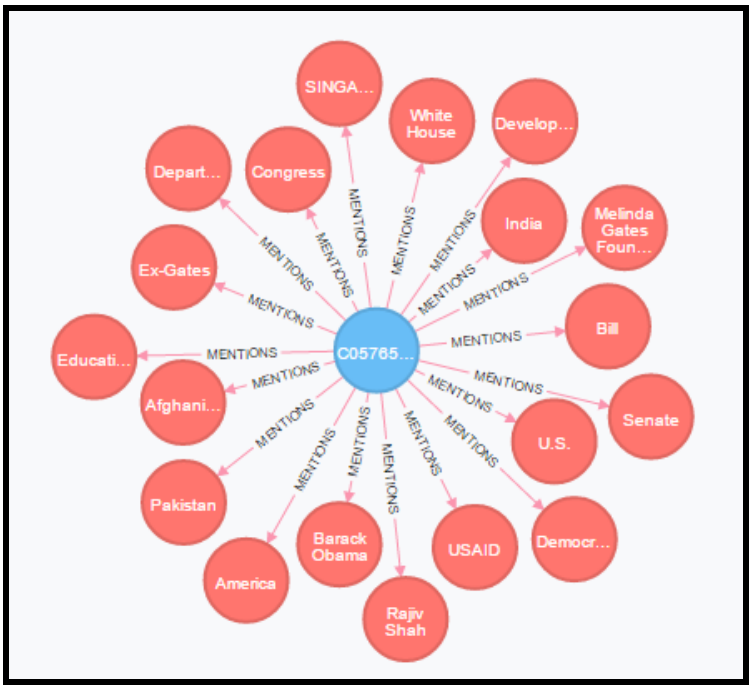
Before processing and analyzing the data, we hand-tagged 1000 emails for emotion. Originally, inspired by the categories created for the Enron released email dataset, we tagged emails across 14 categories of emotional tone. We found though that the data was imbalanced toward no tone 5:1. Because of this, we normalized our email tags to be binary - 0 for no tone and 1 for a significant degree of (any) emotional tone. Since deciding whether an email is emotionally toned or not is a rather subjective process, we coded several dozen emails together as a team to enhance intercoder reliability. When making this judgement, we took into account the amount of emotionally expressive language across the entire email chain in a threaded email.

We then created training, development and test sets. Because of the inherent imbalance of our dataset, we distributed emotional emails across our training, dev and test sets.

Keyword Extraction Algorithm

Overview

The first step to build our email graph was the summarization of the email contents. Our concept was to extract the most important words or phrases from an email. Deciding what is "'key" in a body of text is an inherently subjective process, so we decided on the following characteristics of a key phrase: (1) proper noun phrases; (2) among the twenty most frequent proper noun phrases within an email; and (3) not a substring of another noun phrase in the same email. The focus on proper noun phrases biases the extracted phrases towards people, places, and organizations mentioned within the emails. After reviewing successful approaches for entity extraction in business-related emails (Li, Sen, and Zaman, 2015), we performed a process of (1) preprocessing and tokenization, (2) Named Entity Recognition, and (3) phrase de-duplication. Additionally, we performed categorization of the key phrases using hypernyms and extracted categories of emotion or affect words.



Tokenization

For each email, we removed the email header information, excluding the subject, as we did

not want to extract phrases with the sender and receiver information. The data was also cleaned of the markup text from the State Department release. After this preprocessing, we tokenized each by sentence using NLTK's built-in sent_tokenize and word_tokenize functions then used filters to remove stop words, other irrelevant (for our purposes) language such "Subject", "FW", "RE", "Attachment" and numbers.

Named Entity Recognition

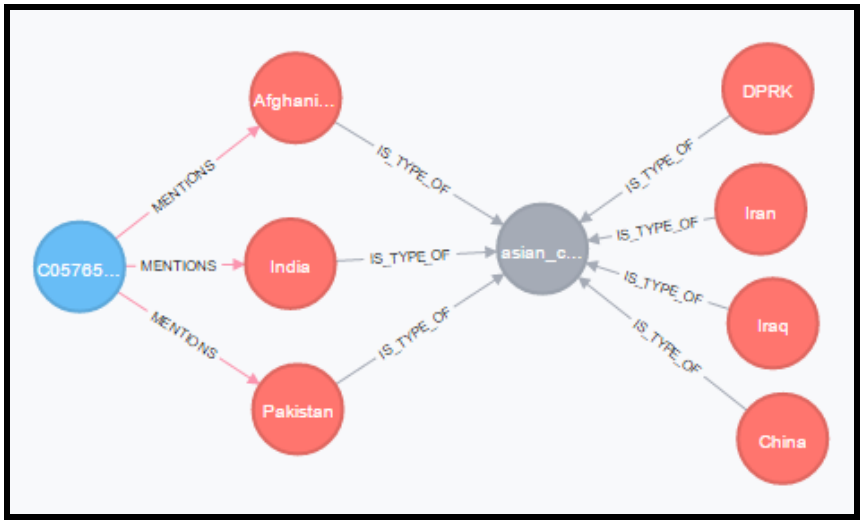
Our initial attempt at named entity recognition (NER) used NLTK's built-in ne_chunk() tagger, however, upon visual inspection of the results, we saw that the extracted entities were very heavily influenced by capitalization with many common nouns, those starting with a capital letter or containing all capital letters, incorrectly identified as named entities. In an informal text such as an email, capitalization rules may not be followed as regularly as in a formal text. After searching for more robust NER platforms, we incorporated the Stanford Named Entity Recognizer into our system (Finkel, Grenager, and Manning, 2005). Iteratively adjusting our pre-processing and tokenization, the CoNLL 2003 four class model that includes a 'MISC' entity class in addition to the 'Location', 'Person', and 'Organization' seemed to consistently produce keyphrases that matched our desired criteria.

De-duplication

A simple algorithm limits the number of key phrases with repetitive contents. After normalizing for capitalization, we eliminate any key phrase that is completely contained within a longer key phrase. For example, if our NER returns a set of entities such as {'President Bill Clinton','Bill Clinton','Bill'}, our algorithm will only retain 'President Bill Clinton'. After completing this process for all phrases in an individual email, the twenty most common phrases are contained. It is possible that no key phrases are returned by this process, reflecting an email that is heavily redacted or contains no recognized proper noun phrases.

Categorization by Hypernym

Building on the four classes given from Stanford NER (Person, Location, Organization, Misc.), we decided to categorize the extracted key phrases by higher-level concepts using WordNet (Fellbaum 1998). WordNet is included with NLTK as nltk.corpus.wordnet. This categorization would allow users of our system to not only search by words specified in an email, but also search by types of words mentioned in the emails. For instance, a user would be able search for 'Asian Country' and find all emails that mention a nation in Asia instead of having to

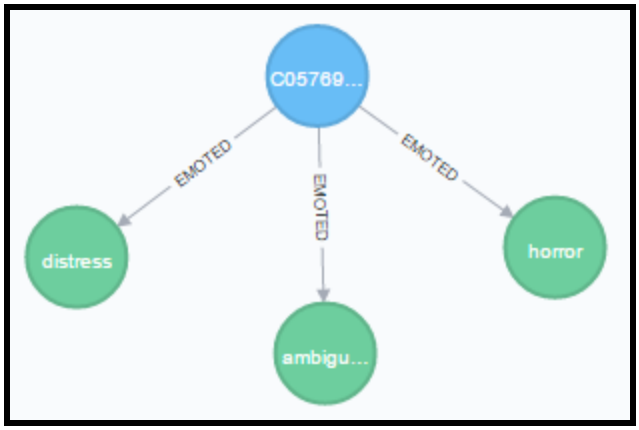


enumerate ‘Afghanistan’, ‘India’, ‘Iraq’, ‘China’, etc. when searching the database.

We accomplished this by looking up each phrase for a corresponding synset in WordNet, choosing a synset, and then finding a normal hypernym or instance hypernym for that synset. If a hypernym was not found in WordNet, we defaulted to using the initial NER class result for that phrase’s category. We opted to choosing the first synset as, similar to traditional dictionary listings, the first synset usually corresponds to the most common usage for the given word. This approach could be improved using word sense disambiguation in the future, but was beyond the scope of this project.

Emotion Categories using WordNet Affect

To give the user additional description resources to categorize the emails in our system, we decided to extract a list of emotional categories contained in the body of each email. For this task, we used WordNet Affect (“an extension of WordNet Domains, including a subset of synsets suitable to represent affective concepts correlated with affective words”). To use WordNet Affect across all parts of speech (not simply noun phrases like our key phrases), after the initial preprocessing and tokenization process described above, we tagged our emails using a backoff tagger trained with NLTK’s NPS chat corpus (nltk.corpus.nps_chat.tagged_posts).



We used Clement Michard’s python module (https://github.com/clemtoy/WNAffect) to integrate WordNet Affect into our project. By looking up every noun, verb, and adjective in the WordNet Affect domain, we build a distinct set of emotional categories (or affective labels) from the returned synsets.

Classification Algorithm

Tokenization + POS Tagging

We began by defining a tokenizer and stemmer which return a set of stems in a text. Sentences are tokenized first, then words using NLTK’s built in sent_tokenize and word_tokenize. Then words with numbers, legal & stop words, and markup lines are removed. The tokens are then stemmed using NLTK’s SnowballStemmer in preparation for part-of-speech tagging. Given the nature of our text, we then trained and evaluated a n-gram back off tagger using the NPS Chat Corpus as our training and test set. We achieved 85% part-of-speech tagging accuracy on our test set.

Emotion Categories from WordNet Affect

We then fed our tagged tokens, all nouns, verbs and adjectives, into WordNet Affect, an extension of WordNet Domains mentioned previously. This allowed us to build a set of emotional categories from the returned emotion synsets.

Feature Selection + Engineering

Given the challenges of our unbalanced dataset, we defined a number of custom transformers to help our classifier classify emotional from neutral emails.

- **CountVectorizer + TfidfTransformer:** We first used CountVectorizer to count our tokens and represent them as vectors. We then used TfidifTransformer to calculate TF-IDF values from the word counts across the corpus. This allowed us to create our tokens into a concise representation of feature values.

- **LengthTransformer**: This feature is a logarithmic transformation of the total number of characters in an email. After becoming more familiar with our dataset, we felt that the length of the email could be a useful feature, especially as strong emotional content in longer emails could easily be dominated by the volume of emotionally-neutral content in the same text.

- **PuncTransformer**: The feature is the number of expressive punctuation (ex: ?, !, ...) and intensifiers (ex: “very”, “so much”) per line of text. We believe emotional emails are more likely to have expressive punctuation and intensifiers than neutral emails. The scores are adjusted by a logarithmic function and the emails are stripped of news article forwards. We chose to strip forwarded articles because we often saw emails only with forwarded news articles and without any accompanying personal annotation or message. We felt it was important to differentiate emotional words found in news articles from those found in interpersonal exchange.

- **SentimentTransformer**: The feature is a score of 0 to 3 based on the degree of positive and negative sentiment measured by the polarity scorer of NLTK’s SentimentIntensity Analyzer. We believe emotional emails are more likely to have a greater degree of sentiment than neutral emails.

- **EmoTransformer**: The feature is the number of emotions in an email per line of text. We believe that emotional emails will have a great number of emotions per line of text than neutral emails. The scores are adjusted by a logarithmic function and the emails are stripped of news article forwards.

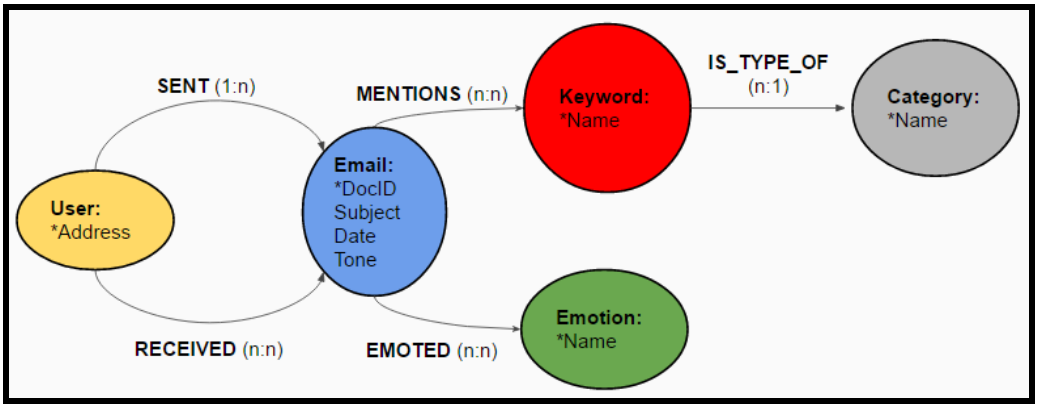
Classification

Given our number of transformers, we chose to implement a pipeline using scikit-learn’s pipeline module. The pipeline module allows us to chain transformers together in a single unit. Before doing so, hoping to find the best combination of transformers, classifier, and their related parameters, we used a simple grid search from scikit-learn’s GridSearchCV module. We used the outputs of that grid search to inform our final parameter selection. We experimented with a number of classifiers - Random Forest, LinearSVC, SGDClassifier, Voting Classifier, Logistic Regression, MultinomialNB.

Next we ran our pipeline. In our pipeline, we ran our five transformers and our classifier. We settled on Linear SVC given our test set accuracy scores and findings from literature. We also tried precision/recall adjustments in our classifiers. We felt comfortable with more false positives (emails being incorrectly classified as having an emotional tone), but wanted most emotional emails classified correctly and felt this was the best way to ensure this.

Creating the Graph

We chose to use Neo4j since the Neo4j graph database and its Cypher query language is conducive for efficiently creating, querying, and storing large network graphs that resemble our model. Queries finding the shortest path between nodes or filtering nodes by number of incoming or outgoing edges are possible and simple to implement. Another benefit is Neo4j's built-in console viewer which lets users easily visualize and explore our data. While one team member had prior experience with the database, the two other team members quickly learned how to use the platform. Using Nigel Small's py2neo toolkit, we added nodes (User, Email, Keyword, Emotion, Category) and relationships according to the below schema:



A breakdown of the properties of each node:

- **User {address}**: from original dataset, a sender / receiver's email address or contact listing
- **Email {DocID}:** from original dataset, a unique identifier for each email
- **Email {Subject}:** from the original dataset, the subject line of the email
- **Email {Date}:** from the original dataset}
- **Email {Tone}:** either 'emotional' or 'neutral' according to our classifier
- **Keyword {Name}:** keyphrase extraction from email body
- **Category {Name}:** hypernym lookup using WordNet or NER class
- **Emotion {Name}:** emotion/affective labels using WordNet Affect

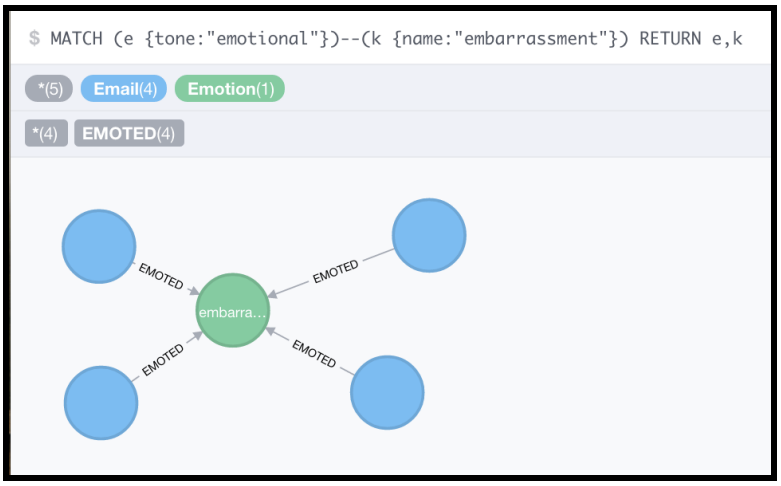
Of note, whenever a node is added to the graph, Neo4J assigns a unique identification number to each node. For our purposes when adding a new email to the graph, we ignore this primary key and search for existing nodes based on the Address for User nodes, DocID for Email nodes, and Name properties for Keyword, Emotion, and Category nodes. If we find any matches, our program will simply add relationships between the existing nodes with nodes containing new data.

Accessing the Data

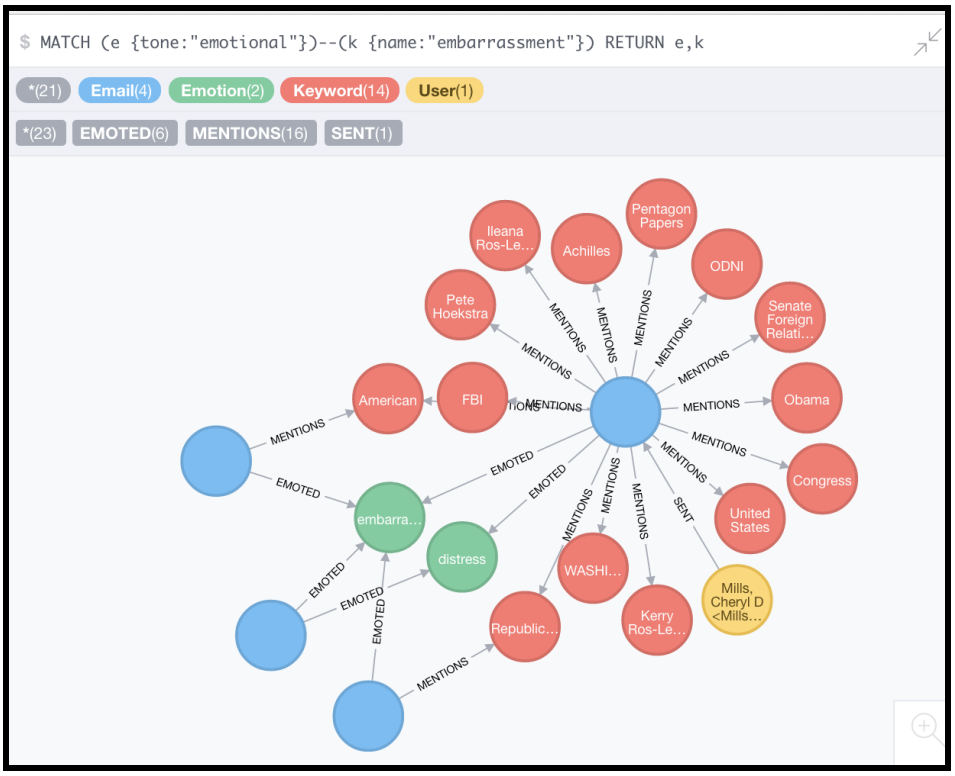
Neo4J's Cypher query language (https://neo4j.com/developer/cypher-query-language/) enables users of the resulting graph dataset to explore the data through expressive queries.

For example, to access ‘emotional’ tone emails that emote ‘embarrassment’, the below query entered in the Neo4J console viewer returns the resulting graph:

```
MATCH (e {tone:"emotional"})--(k {name:"embarrassment"})
RETURN e,k
```

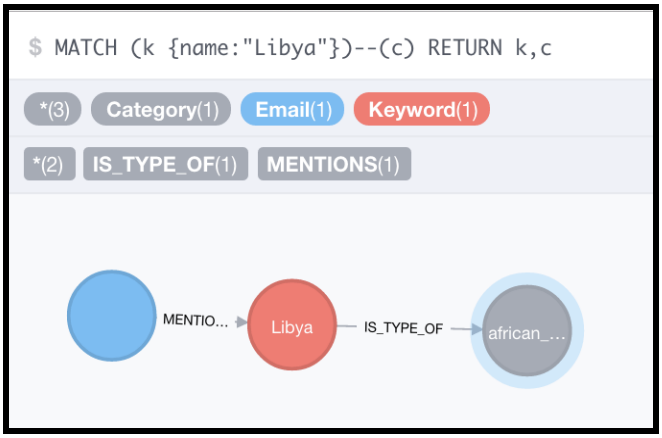


Four emails (represented by blue nodes) are returned that emote “embarrassment.” If a user were to “expand” one of the emails, the below graph is revealed, displaying that email’s extracted keywords (red nodes), emotions emoted (green nodes) and its associated senders and/or receivers (yellow nodes):

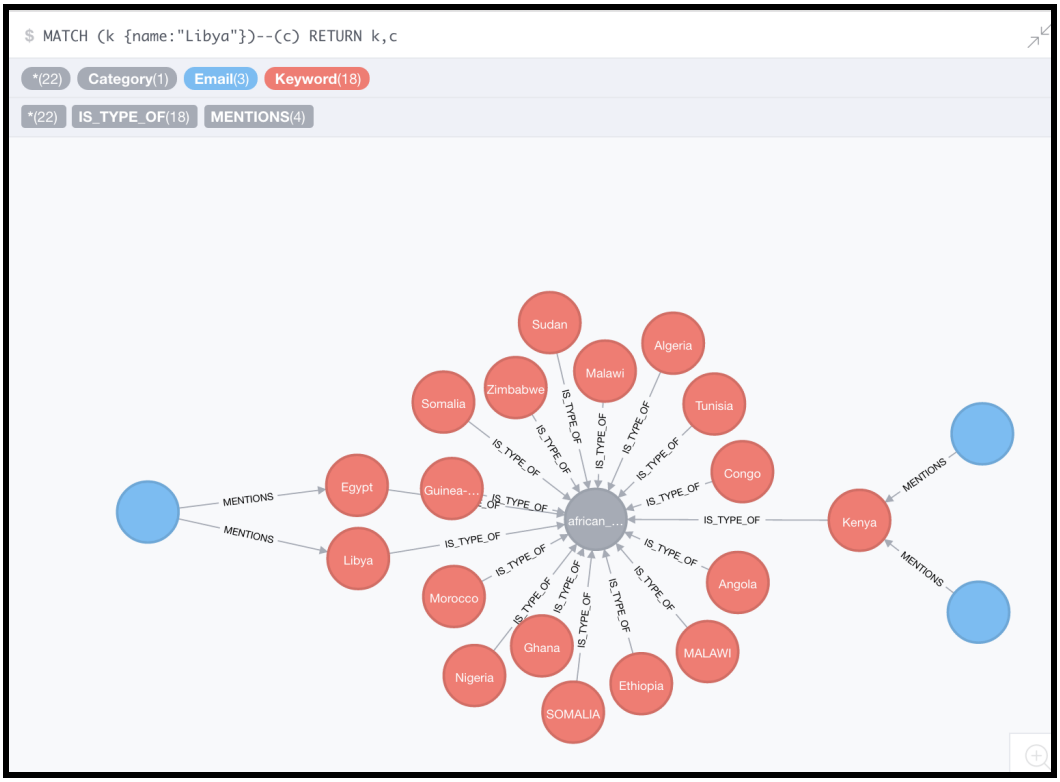


In another example, to access emails that mention “Libya”, the below query entered in the Neo4J console viewer returns the graph:

```
MATCH (k {name: “Libya”})--(c)
RETURN k,c
```



The above graph shows that of the 1000 emails queried, one email is returned that contains the keyword “Libya.” Interestingly though, because Libya is associated with the instance hypernym category, “african_countries”, a user can then “expand” that Category node to see all other African countries mentioned in the dataset. In the below graph, we’ve also expanded “Kenya” and find two emails (represented by blue nodes) that contain “Kenya” as a keyword.



Results + Evaluation

Referring back to project goals and measurements of success:

1. Classifying emails according to sentiment / tone: **Success**

   We achieved 79% classification accuracy upon running our algorithm on our test set - above our goal of 70%. We've included the confusion matrices for both our dev and test sets below.

   a. Size of training set: 797
   Size of dev set: 98
   Overall Score: 78.57%, (77.63% Neutral correct, 81.81% Emotional correct)

	Neutral (Predicted)	Emotional (Predicted)	All
Neutral (True)	59	17	76
Emotional (True)	4	18	22
All	63	35	98

   b. Size of test set: 100
   Overall Score: 79.00%, (80.77% Neutral correct, 72.72% Emotional correct)

	Neutral (Predicted)	Emotional (Predicted)	All
Neutral (True)	63	15	78
Emotional (True)	6	16	22
All	69	31	100

2. Extract keywords and phrases from an email: **Success**

   We had 10 peers review our keywords for representativeness of the email. Keywords were evaluated on a scale from 1 to 5 (1 being keyphrases were unrepresentative of the email, 5 being very representative). We achieved an average score of 3.73 out of 5 for keywords and an average score of 3.80 out of 5 for emotions - both above our goal of 3 out of 5.

3. Build a graph database from the extracted / categorized data: **Success**

   We created a dynamic and queryable graph database using Neo4J that visualizes senders and recipients of email, each email's keywords and their associated hyponyms, and the emotion, if any, contained in the email.

4. Design the organizing system to support future work such as advanced entity extraction and resolution: **Success**

   Our graph database enables interactions to search and access the user, keyword and emotion entities, the system's description resources that describe the email document resources. The graph can be used immediately to analyze the State Department-released dataset, but, importantly, Neo4J's architecture and supporting python libraries allow for future developers to

manipulate the data (add, merge, remove, or split nodes or relationships). Even using a simple edit distance algorithm, “User” nodes could be resolved to be the same user given similarities in email address. Additionally, the system can scale to add more meaningful relationships between nodes such as relationships extracted between Keywords in an email (“Hillary”--[Went To]--”The White House”). Finally, our modularized code separates the tasks of extracting keywords, classifying of emotion, and building the graph to give a future developer the ability to focus on a single natural language processing task.

Future Work

If we were to continue on this project, there are a number of areas we believe would be interesting to pursue for future work:

- **Develop front-end interface** to enable easier query of emails and to view the exact email on the interface.
- **Improve entity disambiguation** (user email address, label matching): This would then allow user nodes with similar email addresses to be linked to one another for increased searchability.
- **Word Sense Disambiguation**: This would help to ensure that the chosen synsets and hypernyms fit within the context of the email.
- **Keyword matching or similarity.** By integrating this into our algorithm, keywords that share morphemes, for instance, could then be linked or merged together in the graph.
- **Evaluate on other bulk email collections**: Given the growing need to explore large datasets, it would be interesting to see how our algorithm applies to other bulk email collections, and then evaluate and improve our algorithm to fit other collections.

Conclusion

Our work on this project has explored how computers can effectively pull keywords and emotions out of very large email datasets. More largely, our work has shown how natural language processing tasks can be used as a resource to explore large datasets of informal language. The challenges of working with text that eschews orthographic conventions in capitalization and punctuation forced us to explore techniques for tokenization, keyword extraction, and classification tasks. The domain specifics of email-based communication - its varied speech acts, various perspectives of multiple senders, forwards and quoting of other expository or narrative text, demonstrate the need for algorithms, like ours, to grow in robustness so that we might more efficiently understand and explore these bulk datasets.

Team Roles

We first split the three main parts of the project identified as keyword extraction, classification and Neo4J among ourselves starting with one iPython notebook and then building off the notebook incrementally. Each of us then went about helping and refining all the tasks saving each version as a new notebook so that we could revert or trace the changes as needed. The table below summarizes the contribution of our team.

WORK	**TEAM MEMBER CONTRIBUTIONS**
Project scoping and proposal	Shannon, Steve, Shrestha
Hand labeling	Shannon, Steve, Shrestha
Keyword extraction and tokenization	Steve
Classification and correction of imbalanced data	Shannon, Steve, Shrestha
Neo4J	Steve
Presentation and final write up	Shannon, Steve, Shrestha

Supporting Files

In "BECCA with database.zip":

- **BECCA_Classification_FINAL.ipynb** (our code to classify emails and build the database)
- **emailgraph.py** (Our module to extract keywords / categories and create graph)
- **emotion.py and wnaffect.py** (supporting modules from Clement Michard's github)
- **graphdb.zip (**compressed Neo4J graph database for 999 handtagged emails)
- Folder "**hillary-clinton-emails"** contains:
    - **Labeler2.py** (our code used to handtag 999 emails)
    - **Emails.csv** (the original 7900 emails)
    - **tagged-mail** (folder with hand-tagged emails in CSV format)

Additional dependencies (not included):

- Neo4J Community Edition
- Scikit-learn
- NLTK
- WordNet Affect (WordNet Domains)
- py2neo
- Stanford NER Jar (requires Java)

Additional file:

- **NLP: HIllary Emails** (Google Sheet with evaluation survey results)

References

Li, Juan, Sen, Souvik, and Zaman, Nazia. 2015. Entity Extraction from Business Emails. International Journal of Information Technology and Computer Science (IJITCS), vol.7, no.9, 15-22.

Finkel, Jenny R., Grenager, Trond, and Manning, Christopher. 2005. Incorporating Nonlocal Information into Information Extraction Systems by Gibbs Sampling. In Proceedings of the Association for Computational Linguistics, ACL '05, 363–370.

Fellbaum, Christiane. 1998. WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press.

Princeton University "About WordNet." WordNet. Princeton University. 2010. <http://wordnet.princeton.edu>

Strapparava, C. and Valitutti, A. 2004. Wordnet-affect: an affective extension of wordnet. In Proceedings of the 4th International Conference on Language Resources and Evaluation. Lisbon.

Bentivogli, Luisa, Forner, Pamela, Magnini, Bernardo and Pianta, Emanuele. 2004. Revising WordNet Domains Hierarchy: Semantics, Coverage, and Balancing, in COLING 2004 Workshop on "Multilingual Linguistic Resources", Geneva, Switzerland, August 28, 2004, 101-108.