



BCIS 5140- Artificial Intelligence in Business

Final Report

Email Classification

Mudita Garg

Table of Contents

Abstract	
Introduction	
About the client	
Business Problem	
Project Goal	
Value Proposition	
Requirements	
Project Cost	
Algorithm	
Outlook Architecture	
Prototype	
Conclusion	

Abstract

Personal and business users prefer to use email as one of the crucial sources of communication. The usage and importance of email grow exponentially despite the prevalence of alternate means such as social networking applications, electronic messages, etc. As the volume of business-critical emails continues to grow, the need to automate the management of emails increases for several reasons, such as multi-folder categorization, user-relevant priority emails classification, and others. This project mainly focuses on defining the emails according to the user-relevant labels and classifying the emails in the right category.

Introduction

For hundreds of years, humans used to store and organize objects for easy retrieval. Nowadays, there are machines to do the same job. Likewise, organizing emails in folders is necessary for easy access. Presently, billions of people use emails for both personal and official purposes. It is one of the most common and efficient means to communicate with one another and there are no signs that this means of communication will soon lose its popularity. By 2022, the number of emails exchanged would reach 333 billion suggested by some reports. An email box is often rife with many different messages ranging from high priority to low priority and other subscriptions clutter. To not lose sight of the high priority of the mails, it's important to organize emails in a proper category. In this digital era - it will be best to have an automation system for organizing emails - a classification algorithm to process the email and classify them into appropriate folders by user relevance.

About the Client

University of North Texas (UNT) is a public research university that consists of 14 colleges and schools in Denton, Texas. UNT has another two campuses in Dallas and Frisco. Currently, the G.Brint Ryan college of business has approximately 8000 students and is growing exponentially. It has five different departments like Accounting, Finance, Insurance, Real Estate & Law (FIREL), Management, Information Technology & Decision Sciences (ITDS), and Marketing & Logistics. In this project, we are considering the emails received for the graduate admission process. It is a two-tiered process. The first step is to apply on the website and the second step is to submit secondary documents directly to the specific department typically via email. The documents received are tracked via a secondary database (Microsoft Access) and must be matched to the admission application and transcripts after the Graduate Admission Processing team has completed their tasks. UNT uses outlook because Microsoft security and privacy policies are better complying with FERPA and other laws.

Business Problem

Managing emails is one of the essential tasks in any organization. The time-sensitive emails should be categorized in a way that they should be dealt with first rather than prioritizing them in the order they are received. Nowadays, our inbox is cluttered with a variety of emails like newsletters and other promotional offers which hinders the high-priority emails, and as a result, the employees find it difficult to meet their deadlines. Self-sorting & labeling is stressful work.

The outlook has an existing feature of customizing the labels and classifying the emails which is a tedious job as they must think about all types of emails. The employees working in the organization must manually enter the keywords for labeling and they have to be in a specific

sequence for the classification to work. Students seeking admission are from around the globe, where English is not their first language. Their way of composing emails might differ from the standard way in the USA. Thus, we need a system that can help the employees working in the Graduate Admission office to overcome those issues. Is there a way we can automate and self-organize the emails?

Do we need an AI Solution?

It is important to remember that ML is not a solution for every type of problem. There are certain cases where robust solutions can be developed without using ML techniques.

Use AI for the following situations:

Coding rules is difficult: Many human tasks cannot be solved using a simple rule-based solution. Many factors could influence the answer. When rules depend on too many factors and many of these rules overlap or need to be tuned very finely, it soon becomes difficult for a human to accurately code the rules.

Scaling is difficult: Manually recognizing a few hundred emails and deciding whether they are high priority or what label they will go into would not be an issue. However, this task becomes tedious for millions of emails. AI solutions are effective at handling large-scale problems.

Proposed solution

NLP is a subfield of AI to handle textual data. It allows machines to manipulate human natural languages. Since the knowledge of labels is unknown thus unsupervised machine learning algorithms will be used. To perform the updated classification, there are different algorithms to achieve that. The incoming emails are considered as input that is then preprocessed to remove

special characters, punctuations, stop words, etc. The email will be classified based on predefined labels used in the training dataset. As this is an automated process, there will be an extra label to capture all the emails which don't fall into any of the predefined categories.

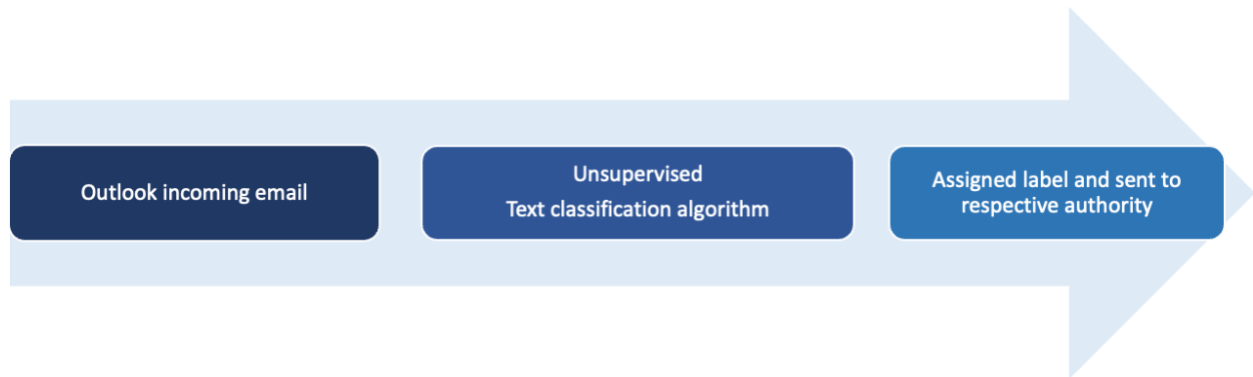


Fig 1: Proposed solution

Assumptions

1. Emails received by the admission office are non-Spam emails.
2. Incoming email is composed in the English language.
3. Ingested email is in CSV format.
4. Presuming no missing and inaccurate information is provided in the email.

Requirements

Universities will consider email as a form of exemplifying interest in getting admission. Moreover, universities receive emails from different parts of the world, so the text language will not be the same, but the purpose of the email might be similar. The email will contain the sender address, receiver address, subject, body, date. In this project, only the body/content of the email for classification is considered.

Email ingestion is the process of reading emails from outlook and processing them. The email can be imported in different formats such as JSON, XML, CSV. We are assuming that the ingested

email is in CSV format. To use the same model in different formats, only minor changes are needed in the model.

Gathering student admission data is tedious as it is confidential hence a similar dataset of Enron Corporation is used for this project. The Enron email is an unsupervised dataset containing approximately 500,000 emails generated by employees of the corporation. It was obtained by the Federal Energy Regulatory Commission during its investigation. Since the dataset contains all the spam and non-spam emails, this project mainly focuses on non-spam emails. As the emails are free-flowing text, it has a lot of noise that needs to be cleaned before building the model. The dataset has emails that are in text format and the computer doesn't understand the text structure hence we would need to convert them to the binary format which the machine understands. To do that we will be using a count vectorizer to transform data into feature vectors which then can be used to build an NLP model.

Goal

Working in an employee-centric institution is one of the tedious jobs where we need a lot of manpower to write nonautomated emails specific to the received emails. However, segregating, and labeling emails will help employees or customer support teams to answer user queries as soon as possible based on the priorities of the email. Most of the organizations that thrive for robust and quicker responses are incorporating email labeling using machine learning algorithms to help employees to respond and meet their deadlines as quickly as they can. Our goal is to categorize emails in a way that is relevant to the employee so that they can prioritize their work.

Scope

Scope of the project is to ease the tedious job of the admission department where they need to work on thousands of time-sensitive emails. Even though the scope of this project is specific to admission organizations but not limited, as it can be easily incorporated into a variety of organizations with a little tweak to the functionality of our machine learning algorithm. We will be using different combinations of classification and clustering algorithms to find out relativity in the training emails and will perform model evaluations on test data to finalize the deployment model.

Value Proposition

One of the largest barriers to the ongoing success of AI implementations within organizations is the increasing costs of maintaining those solutions over time. Maintenance is required not only for the continued use of ML models but also to ensure the outputs of those models remain accurate and useful over time.

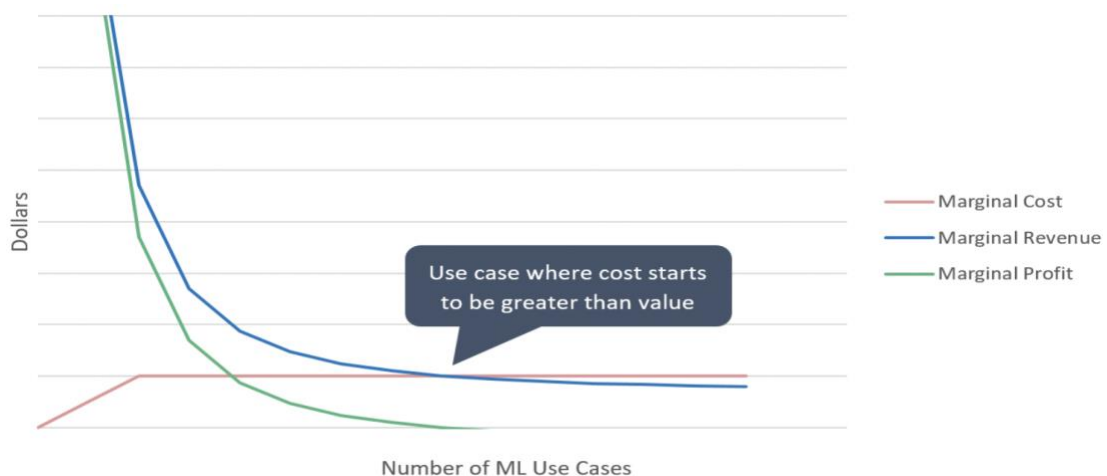


Fig 2: Cost with Number of ML use cases

Source of Image - <https://www.wwt.com/article/getting-started-with-mlops-the-value-proposition>

MLOps is a pipeline offering organizations the capabilities needed to develop, deploy, monitor, and reproduce secure solutions. As depicted below, the MLOps pipeline aids in the efficient management of ML models, and thus offers a framework to quickly production and scale data science units.

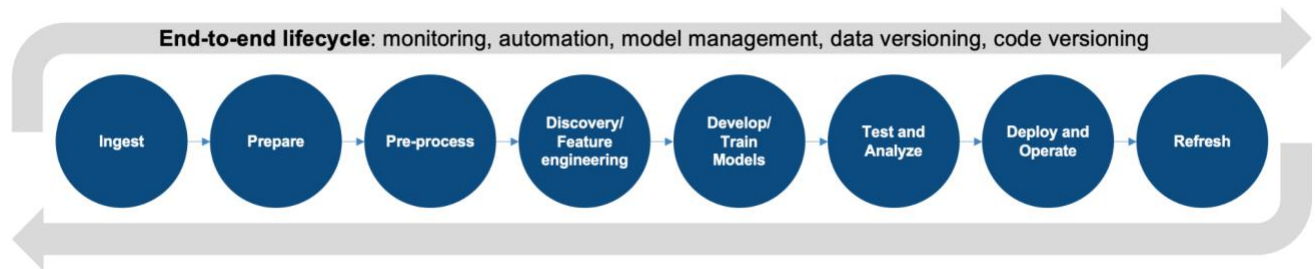


Fig 3: End to End lifecycle

Source of Image - <https://www.wwt.com/article/getting-started-with-mlops-the-value-proposition>

Project Cost

The project cost estimation comprises of below factors:

1. **Update/change Cost:** There will be a need to update the labels generated in the future based on the organization's needs.
2. **Software/Hardware Cost:** The software used to build and deploy the model is an important factor in estimating project cost.
3. **Maintenance Cost:** The model should be trained regularly to adapt to the changes.
4. **Data Cost:** 96% of organizations run into the problem of training data quality and quantity.



Fig 4: Data Cost of the project

Source of the Image-Dimensional Research study illustrating the most common issues companies facing when it comes to data

5. **Research cost:** Initial feasibility study if it can be solved by AI or not?
6. **Production cost:** It includes the infrastructure (cloud, storage), integration (API, Pipelines), and maintenance cost.
7. **Opportunity Cost:** Accessing the return of investment of ML as compared to the other traditional alternatives.
8. **Workforce cost:** Number of people and hours they are utilizing on this project.

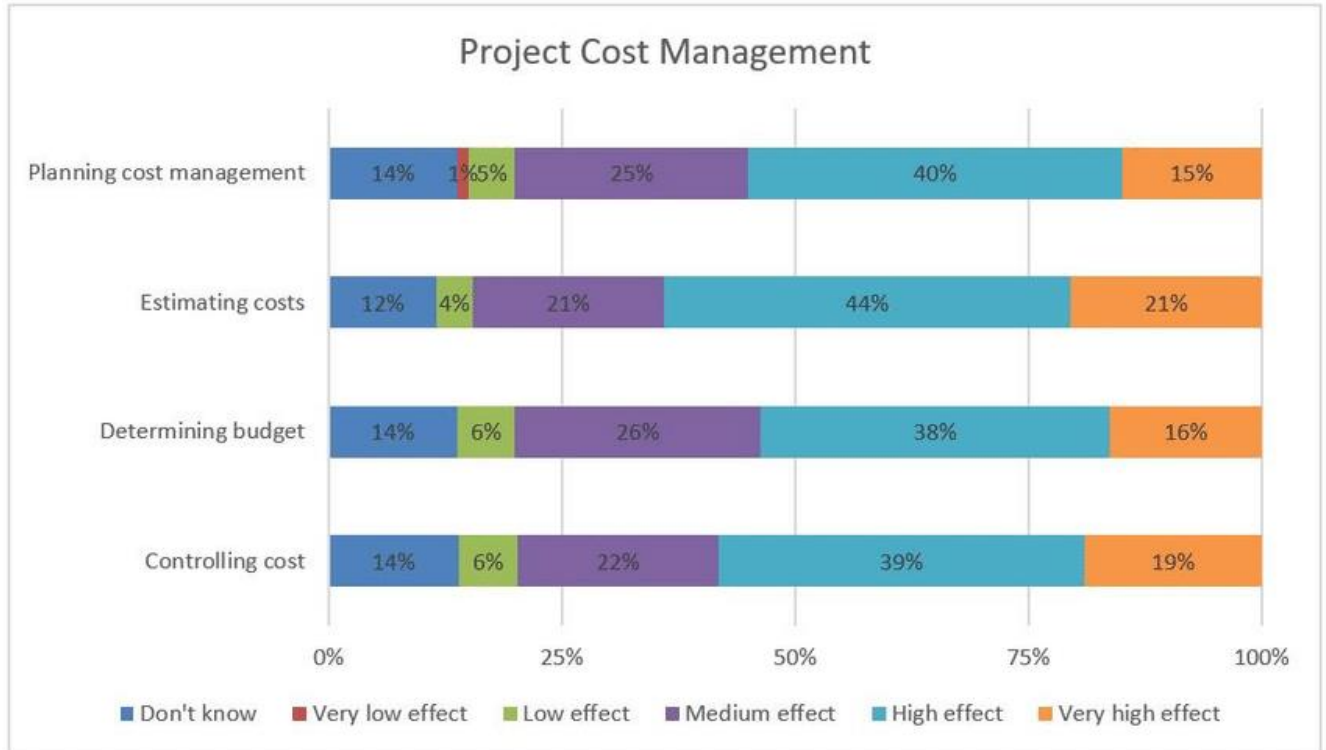


Fig 5: Project Cost Management

Source of the image: https://www.researchgate.net/figure/AI-effect-on-processes-of-the-Project-Cost-Management_fig3_349528473

Algorithm

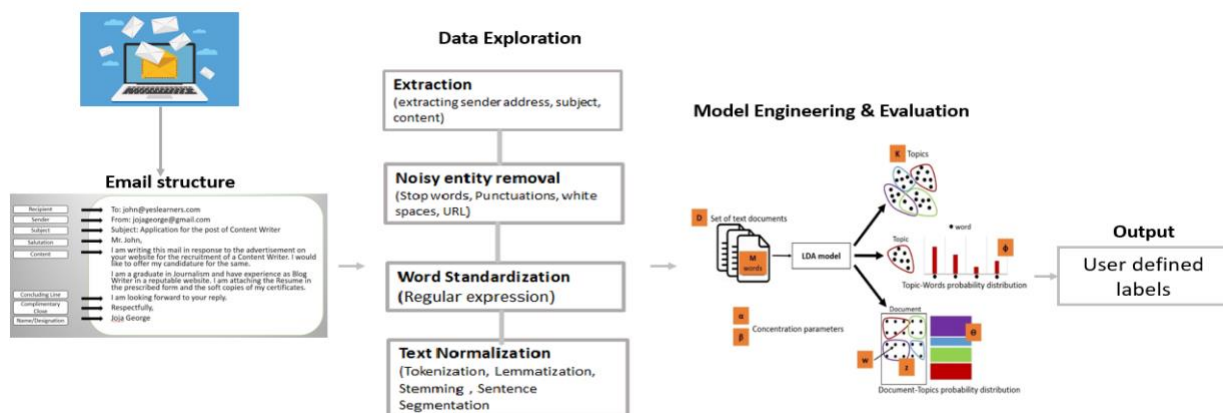


Fig 6: Machine learning pipeline for the training dataset

Data preprocessing

Data Cleaning is one of the most important steps when building a Natural Language Processing model. Text data is one of the most unstructured forms of available data and when it comes to dealing with human language then it's too complex to process. Text data contains noise in various forms like emoticons, punctuations, text in different forms. There are many algorithms and libraries to deal with text cleaning.

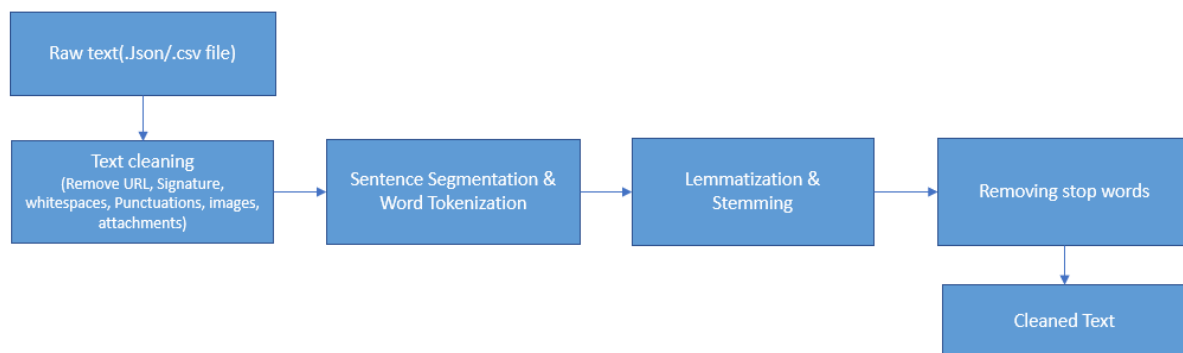


Fig 7: Data preprocessing pipeline

Implementing text preprocessing

Below is a sample of one of the emails in the dataset.

```
[ 'Subject: enron methanol ; meter # : 988291 this is a follow up to the note i '
'gave you on monday , 4 / 3 / 00 { preliminary flow data provided by daren } '
'. please override pop s daily volume { presently zero } to reflect daily '
'activity you can obtain from gas control . this change is needed asap for '
'economics purposes .' ]
```

As you can see in the sample email, there are different kinds of noise in the text.

- First, the email values in the data frame are converted into a list.
- Removing all the new line characters and single quotes using the re (regular expressions operations) library in python.

Tokenization

Tokenization is the process of breaking up a sequence of strings into pieces such as words, keywords, phrases, symbols, and other elements called tokens. Tokens can be unigrams (one word), bigrams (two words), phrases, or even whole sentences.

To perform tokenization, the Gensim library in python is used on the dataset.

```
[[ 'subject', 'enron', 'methanol', 'meter', 'this', 'is', 'follow', 'up', 'to', 'the', 'note', 'gave', 'you', 'on', 'monday', 'preliminary', 'flow', 'data', 'provided', 'by', 'daren', 'please', 'override', 'pop', 'daily', 'volume', 'presently', 'zero', 'to', 'reflect', 'daily', 'activity', 'you', 'can', 'obtain', 'from', 'gas', 'control', 'this', 'change', 'is', 'needed', 'asap', 'for', 'economics', 'purposes' ]]
```

Stemming and Lemmatization

Stemming is the process of reducing a word to its word stem that affixes to suffixes and prefixes or the roots of words known as a lemma. The advantage of stemming is to reduce the total number of unique words in the dictionary. It helps in generating better topics in the end. SpaCy library in python is used for this process.

```
[ 'methanol meter follow note give flow datum provide override volume reflect activity can obtain gas control change need asap economic purpose', 'see attached file' ]
```

Clustering

Working with a large dataset of emails and checking each email is a tedious task. Hence using clustering to group the same type of emails is a better choice. The clustering algorithm determines the intrinsic grouping of the unlabeled data present. In this algorithm, there are no specific requirements for good clustering. Moreover, it makes some assumptions about the similarity of points, and each assumption results in a different cluster.

Topic Modeling

The goal of the analytics industry is to extract "Information" from data. It's tough to get relevant and required information with the expanding volume of data in recent years, most of which is unstructured. However, technology has created some strong tools for mining through data and retrieving the information that we want. Topic Modelling is one such technique in the realm of text mining. It is an unsupervised machine learning technique that analyzes a series of documents, finds word and phrase patterns in them, and automatically clusters word groups and related expressions that best characterize the series of documents. A document can belong to several topics, similar to how each data point in fuzzy clustering (soft clustering) belongs to multiple clusters. Topic Models are extremely effective for document clustering, structuring big blocks of textual data, retrieving information from unstructured text, and selecting features. Topic models, for example, are being used by the New York Times to improve its user-article recommendation engines. Topic models are used by a wide range of experts in the recruiting industry to extract latent elements from job descriptions and map them to the relevant candidates. They're being used to manage massive amounts of data including emails, consumer reviews, and social media profiles.

Why Topic Modeling?

Topic modeling enables massive electronic archives to be automatically organized, understood, searched, and summarized.

It can aid in the following areas:

- Learning about the collection's hidden themes.
- Organizing the documents into the topics that have been discovered.
- Organize, summarize, and search the documents using the classification.

For example, let's say a document belongs to the topic's status, information and permission. So if a user queries "information permission", they might find the above-mentioned document relevant because it covers those topics (among other topics). We can figure its relevance concerning the query without even going through the entire document.

Therefore, by annotating the document, based on the topics predicted by the modeling method, we can optimize our search process.

The most popular modeling techniques are LSA, LDA. Both models are based on the same assumptions:

1. Every document consists of a mixture of topics.
2. Every topic consists of a collection of words.

Moreover, they have the same input that is Bag of words in matrix format. However, LSA focuses on reducing the matrix dimensions and LDA is most used for solving topic modeling problems. Hence, we chose to use LDA for this project.

Difference between LSA and LDA.

<i>Topic Model</i>	<i>Characteristics</i>	<i>Advantages</i>	<i>Disadvantages</i>
<i>LSA</i>	<i>Analyze document words with the same meaning.</i>	<i>Error reduction by dimension reduction.</i>	<i>Does not give well-defined probabilities.</i>
<i>LDA</i>	<i>Words are grouped into topics and one word can exist in more than one topic.</i>	<i>Works well with large corpus.</i>	<i>Several topics are set in advance.</i>

		<p><i>Does not face overfitting problem like in PLSA</i></p> <p><i>Noise reduction is possible by dimension reduction</i></p>	
--	--	---	--

Source: http://www.digitalxplore.org/up_proc/pdf/268-148653117584-89.pdf

Latent Dirichlet Allocation (LDA)

The (LDA) algorithm is an unsupervised learning algorithm that tries to classify a set of observations into a range of discrete categories. The most common application of LDA is to find a user-defined number of topics shared by documents in a text corpus. In the algorithm, observations are documents, features are the words in the documents, and the categories are the topics. The topics are not defined upfront because the approach is unsupervised, and they are not guaranteed to coincide with how a human would intuitively organize documents. Each document's topics are learned as a probability distribution over the words that appear in it. Each document is described as a combination of topics.

Consider a collection of graduate admission emails and we want to sort them into topics. LDA takes a geometrical approach. If we have three topics such as update, information, permission then it builds a triangle where the corners are the topics and then it puts the emails/documents in the triangle in such a way that emails/documents are close to the corner of the topic they belong to. The document around the edge between update and permission is the one that has content related to topic update and permission. Likewise, topics are described by the distribution of words.

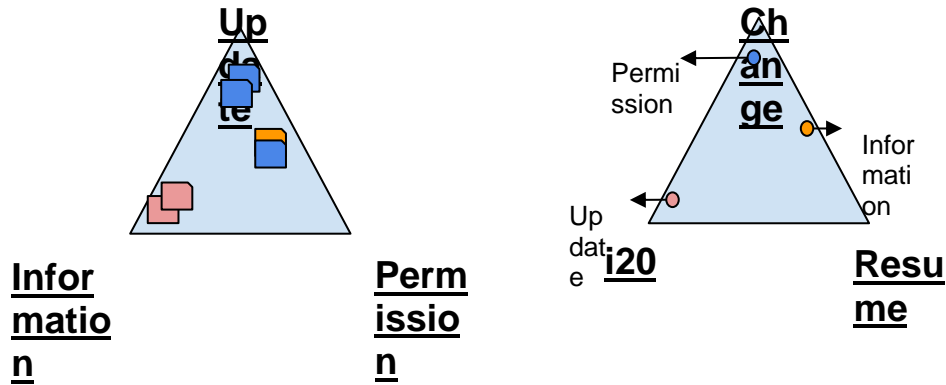


Fig 8: Emails are described by the distribution of topics and Topics are described by the distribution of words.

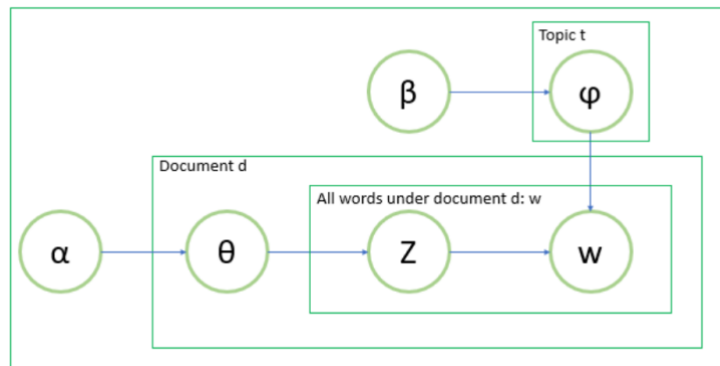


Fig 9:LDA Blueprint

Source:<https://towardsdatascience.com/2-latent-methods-for-dimension-reduction-and-topic-modeling-20ff6d7d547>

Hyper Parameters

1. Alpha (document-topic density factor): controls per-document topic distribution.
2. Beta (topic-word density factor): is responsible for per word per topic distribution.
3. The number of topics “t”: defines the expected number of topics in the corpus of the documents. It is usually assigned a value based on domain knowledge.

A high alpha value means that every document is likely to contain a mixture of all the topics and not just any single topic specifically and a Low Alpha value means that a document is more likely to be represented by just a few of the topics. A high beta value means that each topic is likely to

contain a mixture of most of the words and Low beta value means that a topic may contain a mixture of just a few of the words. To summarize the alpha and beta values, the high alpha will make documents appear more like each other and the high beta will make topics to appear more like each other.

```
# Build LDA Model
lda_model = LatentDirichletAllocation(n_components=20,          # Number of topics
                                     max_iter=10,
# Max Learning iterations
                                     learning_method='online',
                                     random_state=100,
# Random state
                                     batch_size=128,
# n docs in each learning iter
                                     evaluate_every = -1,
# compute perplexity every n iters, default: Don't
                                     n_jobs = -1,
# Use all available CPUs
                                     )
```

Fig 10: LDA initialization with hyperparameters

Other Parameters:

- Theta: is the distribution of topics for each document i .
- Phi: is the distribution of words for each topic(t).
- Learning Decay: it's a technique for training algorithms, it helps in finding local minima.
- Learning method: Two options are available - Batch and Online. We are using the 'online' update as it will be much faster than the batch update.
- Batch Size: Number of documents to use in each EM iteration.
- Max Iteration: The maximum number of passes over the training data.
- N Jobs: The number of jobs to use in the E-step.

Dirichlet Distribution

There are two Dirichlet distributions:

1. **Documents-Topic:** it associates documents with the corresponding topics
2. **Topic-words:** it associates topics with corresponding words

The probability of any document to appear is given by the following formula:

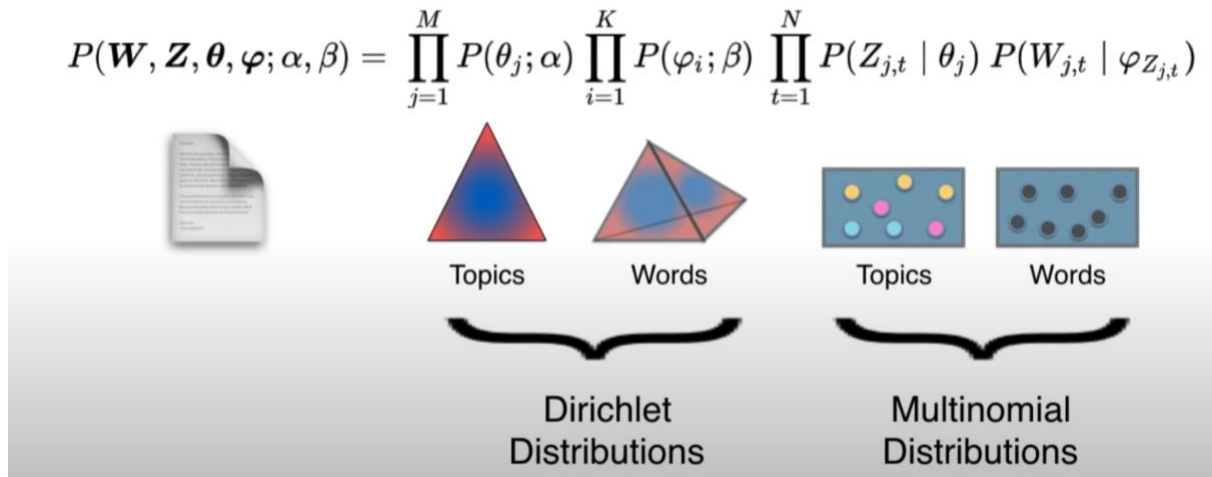
$$P(\mathbf{W}, \mathbf{Z}, \boldsymbol{\theta}, \boldsymbol{\varphi}; \alpha, \beta) = \prod_{j=1}^M P(\theta_j; \alpha) \prod_{i=1}^K P(\varphi_i; \beta) \prod_{t=1}^N P(Z_{j,t} | \theta_j) P(W_{j,t} | \varphi_{Z_{j,t}})$$


Fig 11: Probability of a document

Source: <https://www.youtube.com/watch?v=T05t-SqKArY&t=682s>

LDA considers that documents are made up of words and that helps to determine the topics. Further, it assigns each word in a document to a separate topic to map the document to a list of topics. Assume that we have five documents(D), each of which contains the N number words present in them (ordered by frequency of occurrence).

```
Doc1: word1, word3, word5, word45, word11, word 62, word88 ...
Doc2: word9, word77, word31, word58, word83, word 92, word49 ...
Doc3: word44, word18, word52, word36, word64, word 11, word20 ...
Doc4: word85, word62, word19, word4, word30, word 94, word67 ...
Doc5: word19, word53, word74, word79, word45, word 39, word54 ...
```

Fig 12: “D” documents with “N” number of words.

Source: <https://towardsdatascience.com/latent-dirichlet-allocation-lda-9d1cd064ffa2>

We want to know the words that belong to a topic. As shown in the below table, each row represents a topic and each column represents a different word in the corpus. Each cell represents the probability of a word belonging to a topic. It's worth noting that LDA ignores the sequence in which words appear and syntactic information. It considers documents to be nothing more than a collection of words or a bag of words.

	Word 1 (i20)	Word2 (allow)	Word3 (time)	...
Topic 1 (Update)	0.7	0.45	0.1	
Topic 2(Information)	0.02	0.05	0.08	
Topic 3(Permission)	0.1	0.25	0.1	

Fig 13: Probability estimates for topic assignment to words.

Once the probabilities have been calculated, you can either select the top 'r' probabilities of words or specify a probability threshold and select just the words whose probabilities are greater than or equal to the threshold number. For instance, if words are “i20”, “fees”, “application” then they can correspond to Topic “update”.

	ability	absence	abuse	accept	acceptance	access	accommodate	accomplish	accord	accordance	...
Topic0	0.050004	0.050001	0.050001	5.434719	0.050004	0.050001	0.050001	4.295969	5.959740	0.050001	...
Topic1	0.050001	0.050002	0.050001	0.050010	0.050001	0.050001	0.050001	0.050020	37.756156	0.050001	...
Topic2	0.050002	0.050001	0.050001	0.050002	0.050002	17.428038	0.050015	0.050002	0.050001	0.050001	...
Topic3	0.050198	0.050010	0.050002	17.416676	9.325391	185.235897	0.050002	0.050044	5.036986	0.050034	...
Topic4	0.050001	24.191156	0.050001	0.050214	0.050001	0.050023	0.050003	0.050003	0.050001	0.050001	...

Fig:14 Probability estimates for topic assignment to words for Enron data.

Scoring

Model evaluation is necessary to know the topic quality. The topic should be coherent, understandable, and serve the organization's needs. The accuracy of the model cannot be tested manually as it is a time-consuming process.

In this project, quantitative metrics such as likelihood and perplexity scores are used.

Perplexity is also known as likelihood and the logarithm of likelihood is calculated to determine the accuracy. 80% of the training set to train the model and 20% test set for prediction is used. The perplexity score tells how well the model will perform in the prediction. The model has lower perplexity and higher log-likelihood. Therefore, the model turns out to be good. The model accuracy is improved by changing the values of the hyperparameters.

Scoring Output

```
Log Likelihood: -1332184.4892775281
Perplexity: 496.22683334617335
{'batch_size': 128,
 'doc_topic_prior': None,
 'evaluate_every': -1,
 'learning_decay': 0.7,
 'learning_method': 'online',
 'learning_offset': 10.0,
 'max_doc_update_iter': 100,
 'max_iter': 10,
 'mean_change_tol': 0.001,
 'n_components': 20,
 'n_jobs': -1,
 'perp_tol': 0.1,
 'random_state': 100,
 'topic_word_prior': None,
 'total_samples': 1000000.0,
 'verbose': 0}
```

Predicting a new email

When the incoming email is passed as input, the preprocessing and transformations will be performed on the content of that email before predicting the topic. After running that, the email will be moved to its respective label. We must configure the system to forward that email to the respective employee in the organization for further process.

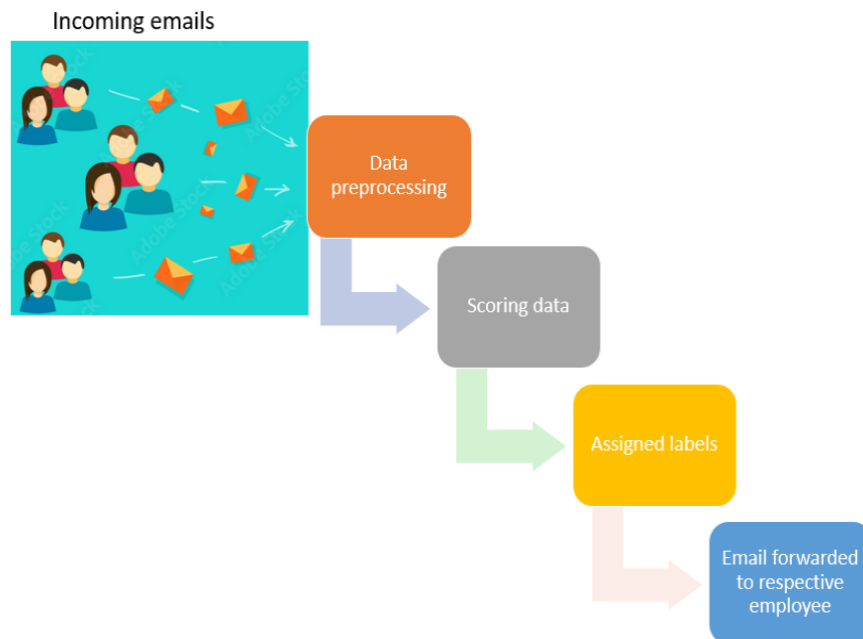


Fig 15: Predicting the incoming email

Data Visualization

The following visualizations are created to get better insights from the dataset.

Word cloud

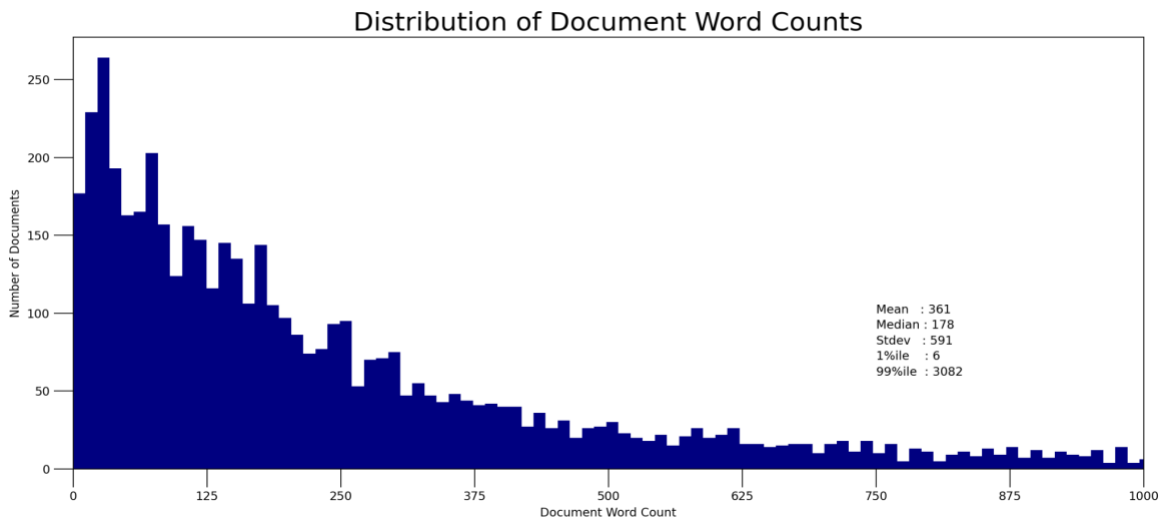
The word cloud shows the words which are frequently used in the dataset. In this visualization one can see that need, purchase, gas, deal, time, information words are more used. It makes sense as

the dataset is about a company that deals in gas thus has all the relevant discussion about purchasing and requiring information on a company or a deal.



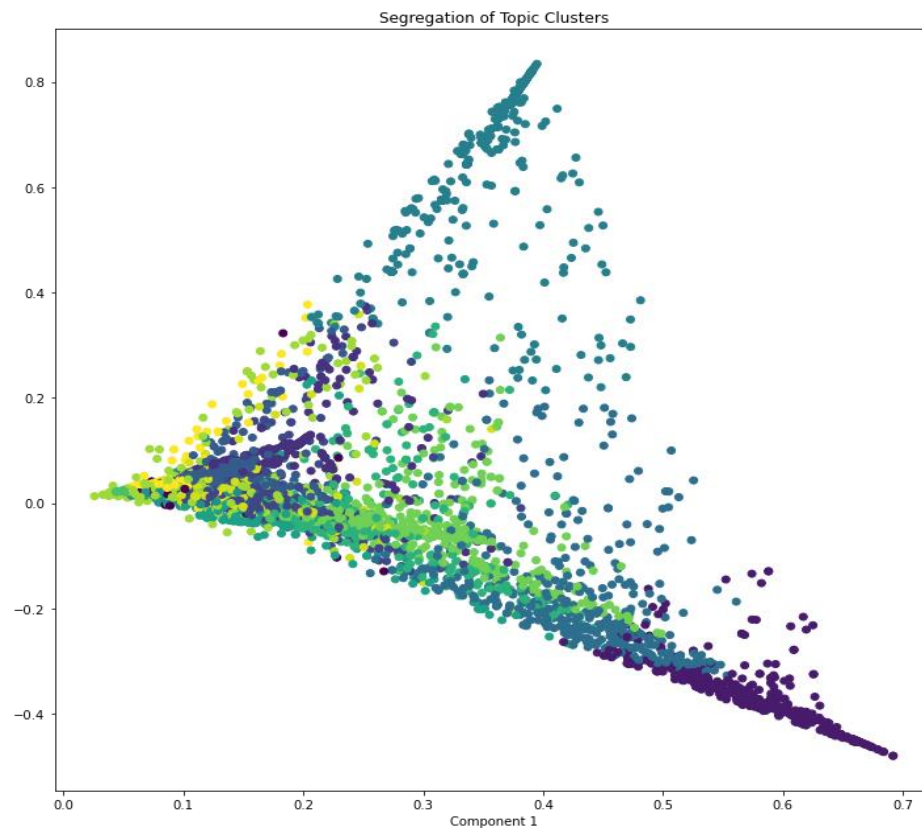
Distribution of Document Word Counts

The documents may contain words from different topics. This visualization shows the distribution of words referring to the significance of it in the document. The mean, median, and stddev show that the occurrence of these words follows a stochastic approach.



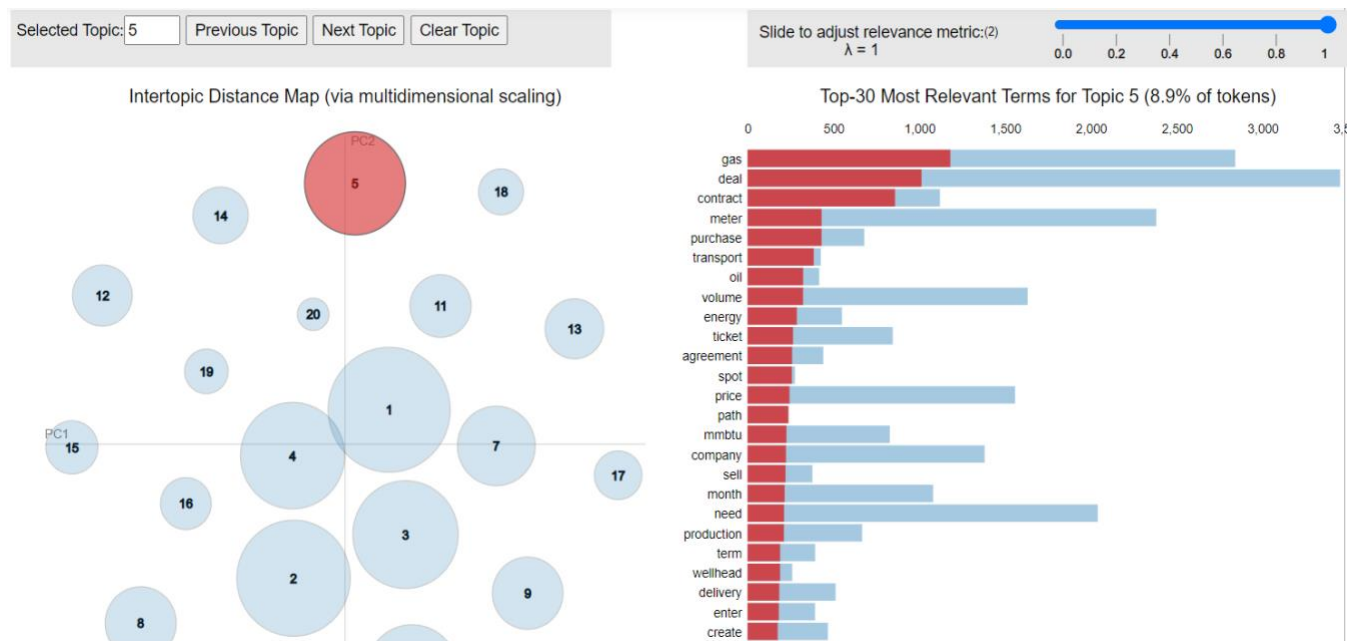
Segregation of topic clusters:

The K Means clusters are constructed to find the documents which share a similar topic. 15 different clusters are created and the singular value decomposition model is built using two components. The below visualization shows the topic cluster segregation.



Relevant terms for the topic

This is an interactive visualization chart that is used to understand the topics better and their relationships with other topics. When the topic on the left side is selected, the 30 most relevant words in that topic will be displayed as a bar chart on the right side. . The lambda is used as a relevance metric. If the lambda value is close to 1, it shows the words which are occurring frequently.



Solution Architecture

UNT GA Label Add-In Architecture

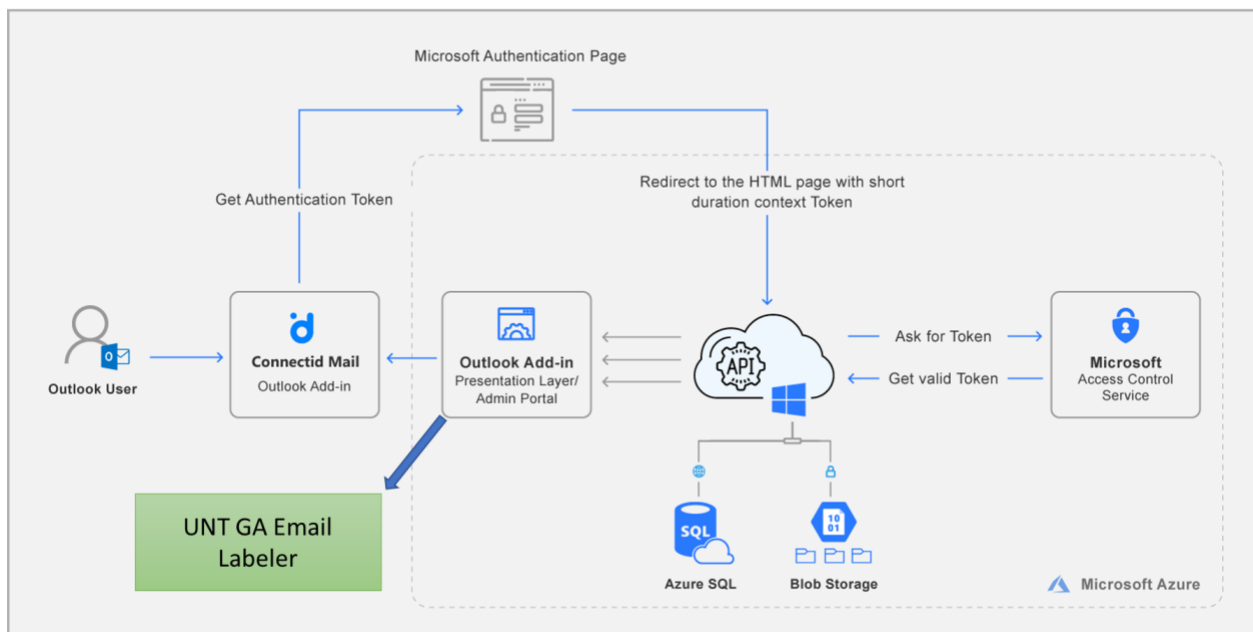


Fig 12: <https://business.connectid.io/docs?product=mail&type=architecture&lang=en-US>

Outlook Architecture for external Add-ins

Outlook add-ins are third-party resources that can be integrated and built into Outlook using a web-based platform. And it also provides premium add-in features for enterprise-grade service subscribers. There are three main features in Outlook add-ins, and they are as follows:

- Computers (Outlook on Windows and Mac), web (Microsoft 365 and Outlook.com), and mobile all use the same add-in and business logic.
- Outlook add-ins are made up of two parts: a manifest that describes how the add-in interacts with Outlook (for example, a button or a task pane), and JavaScript/HTML code that defines the add-user in interface and business logic.
- End-users and administrators can get Outlook add-ins through AppSource or sideload them.

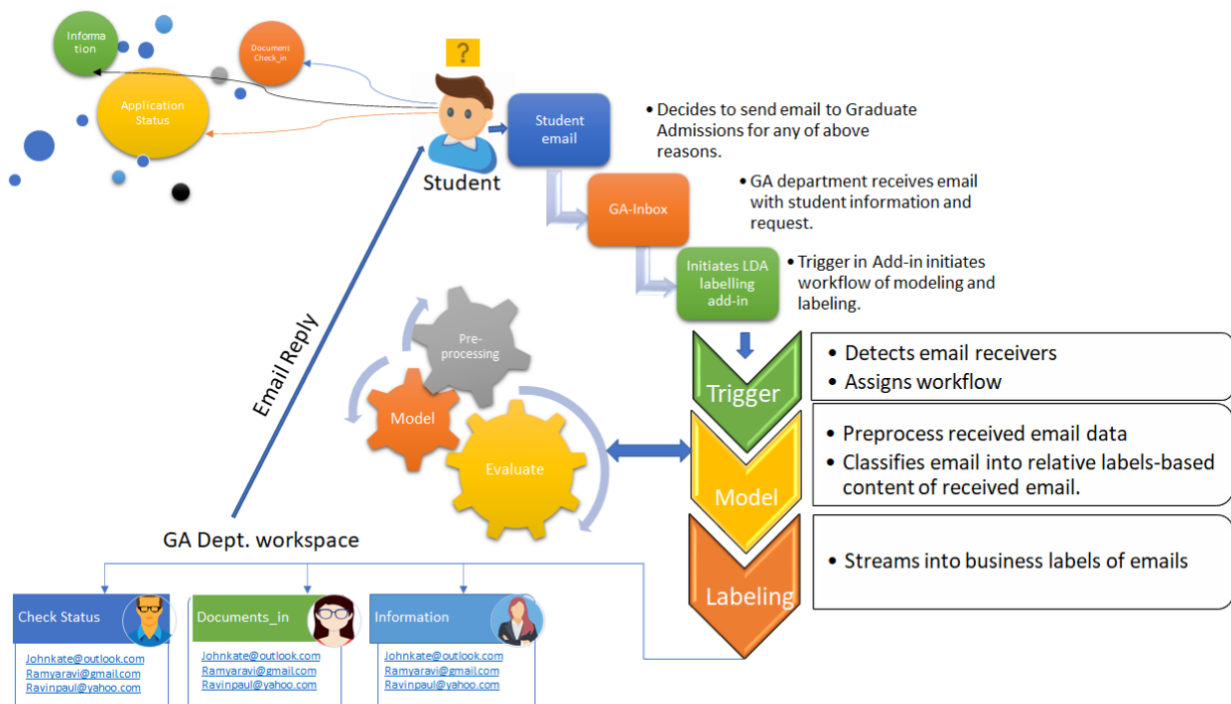
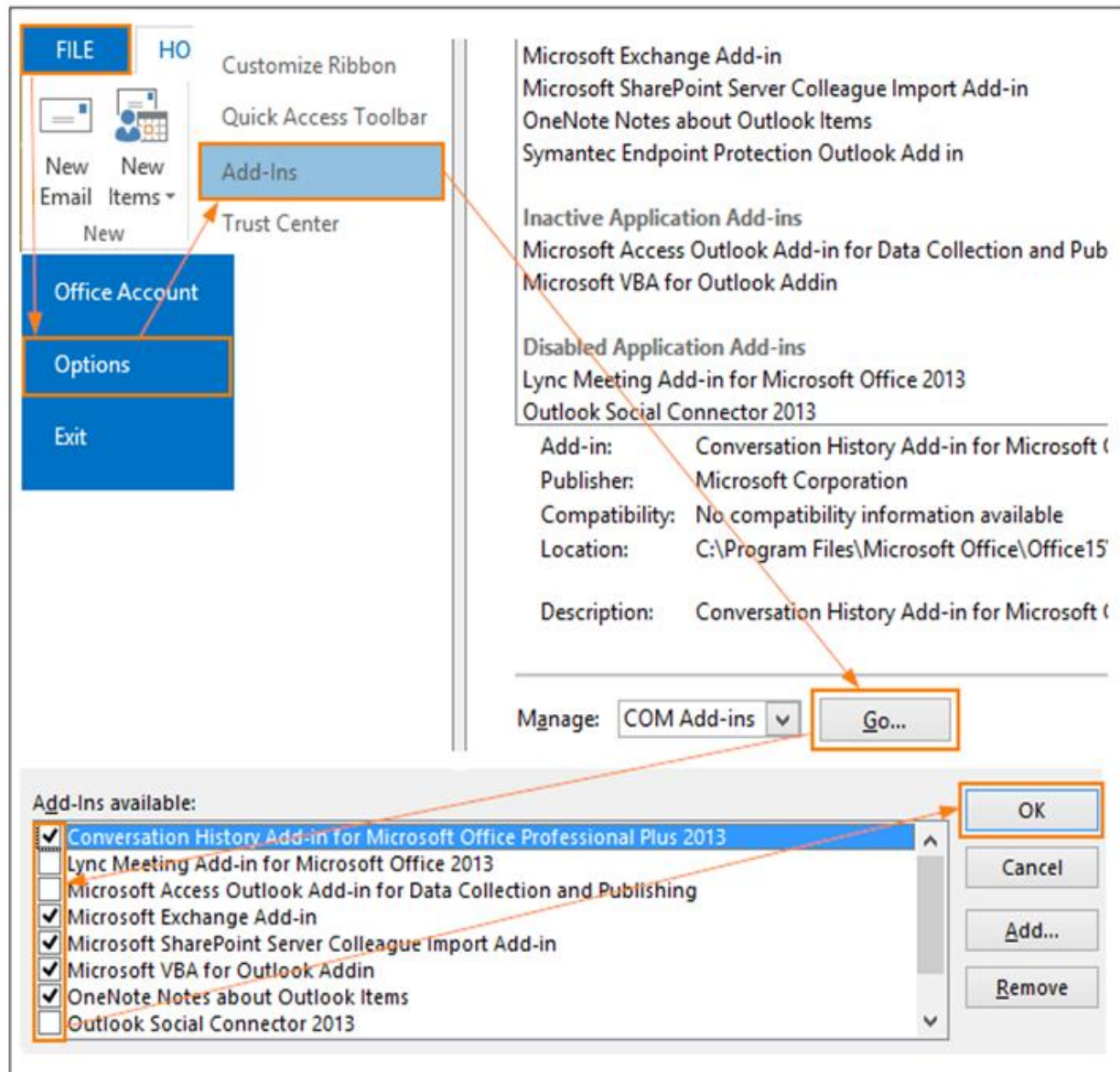


Fig 13: UNT Email labeling Outlook Add-In Architecture

Email messages, meeting requests, answers and cancellations, and appointments are among the

Outlook objects that allow add-ins. Each Outlook add-in specifies the context in which it operates, such as the categories of objects and whether the user is reading or writing a message.



Outlook add-ins are activated when the user inbox receives non-spam(ham) emails, once add-ins functionality triggers its workflow initiates all the processes which are meant to define the flow of the demonstrated process and analyzes email content, and manage receiving as well as output of service. Data flow of email labeling service is handled through outlook internal storage service

notably any backend database or allocated Azure server etc. Once the working process of labeling is completed then all the labeled and unlabeled emails stream into target labels or miscellaneous labels for Graduate Admission (GA) department business-specific. Where Graduate Admission department employees will work on labeled emails and revert to specific students with appropriate content.

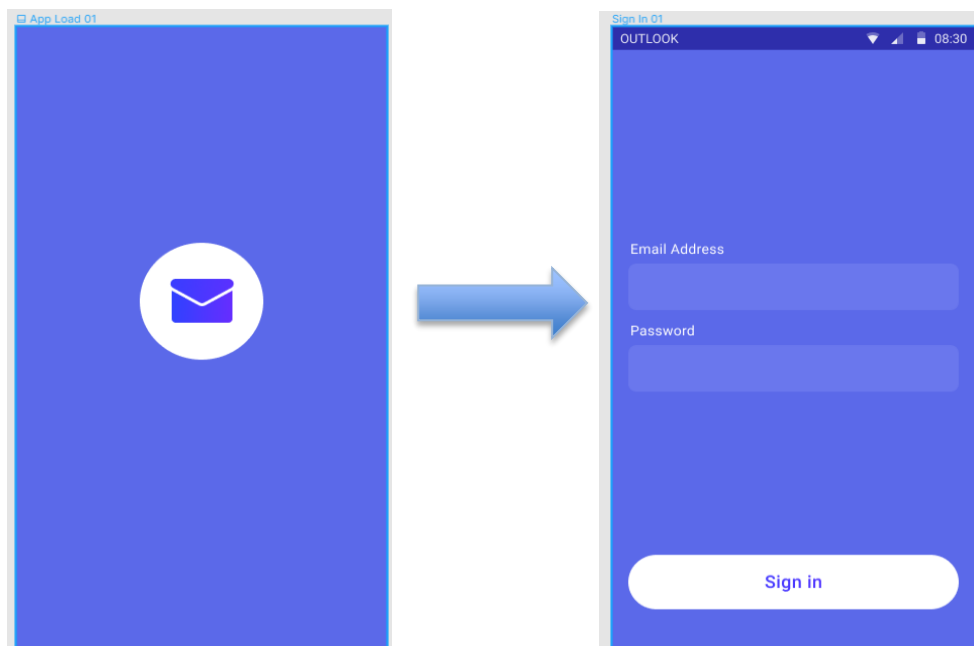
Prototype

The prototype is created for the outlook app on mobile phones. It will work in the desktop application as well. Figma web-based tool is used for creating the prototype.

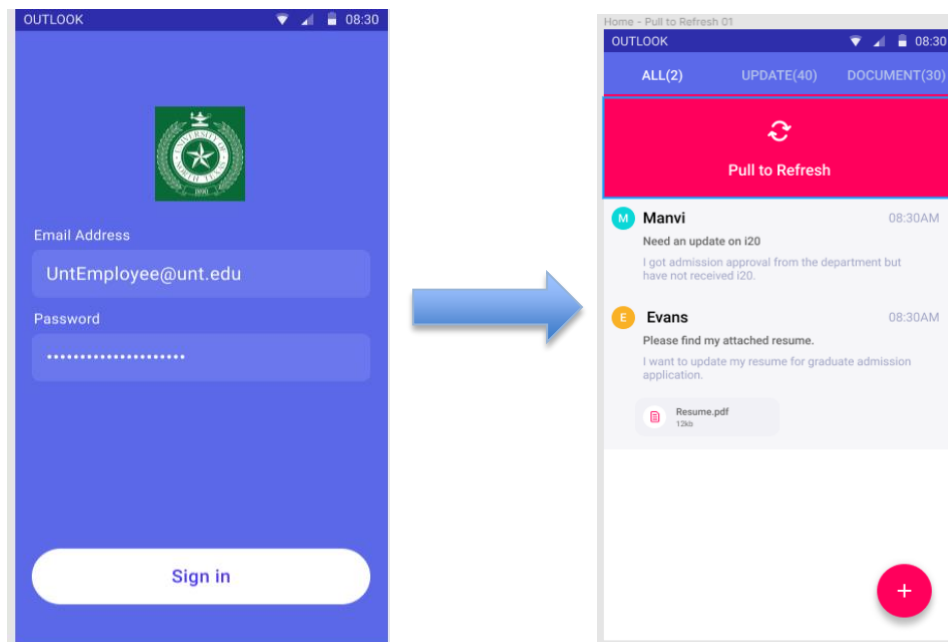
The prototype is demonstrating the workflow of how a new email will move from inbox to the respective label in Outlook:

Following are the frames of the Prototype:

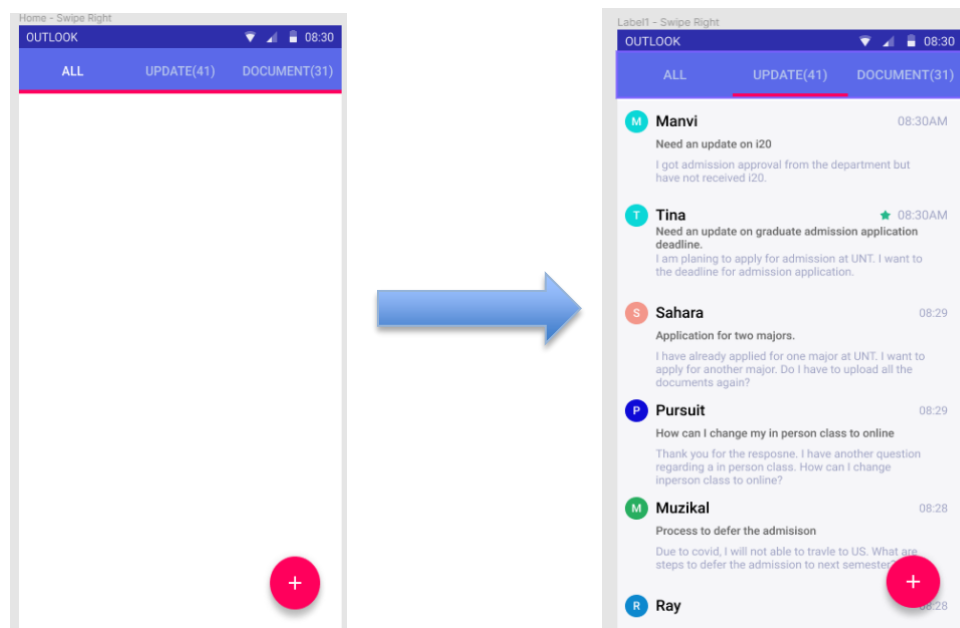
1. App load page and Sign In page (Here employees will enter email address and password and then click on Sign In)



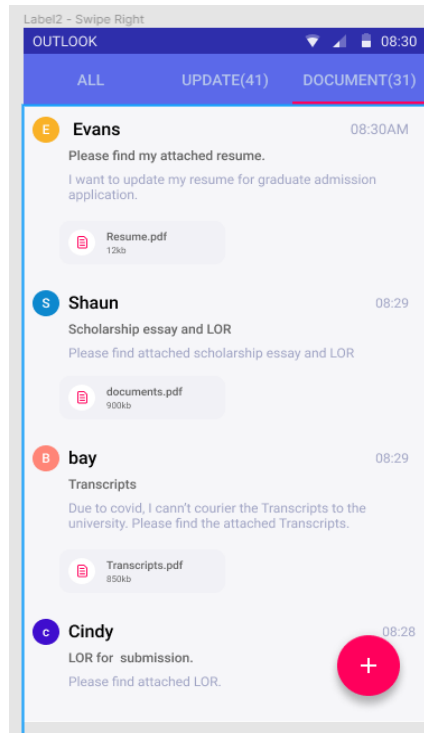
2. Email Home page: New emails are under the all label and within 300 ms they will move to the respective label.



3. Home page after refresh: The new emails are moved to the respective labels. The updated label has the emails where students are requesting updates/information from the graduate admission department.



4. The Document label has the emails where students have sent some attached documents (resume/transcripts) to the department.



Conclusion

Outlook add-ins are the programs that help in automating tasks when reading or creating messages. This project introduced the addition of automatic robust labeling outlook add-in for the Graduate Admission Department of Business College at the University of North Texas. The Graduate Admission department receives the bulk of emails every day, it is an obvious tedious task to read each email, segregate it to corresponding departments and respond to the students within a short period. Updates of the emails are time-sensitive. Hence, the concept of labeling the emails has been put forth to save time and reduce the manual work for the department. As part of this project, LDA (Latent Dirichlet Allocation) algorithm has been used to segregate the emails by reading the message body and assigning them to the corresponding departments. Once the department receives

the email, they respond to the students' queries more quickly than earlier. In the future, email servers or applications should include different types of predefined folders. It should allow users to add new folders that can be user-defined or automatically defined.

References

1. <https://docs.aws.amazon.com/sagemaker/latest/dg/lda.html>
2. <https://www.mygreatlearning.com/blog/understanding-latent-dirichlet-allocation/>
3. <https://towardsdatascience.com/latent-dirichlet-allocation-lda-9d1cd064ffa2>
4. https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation
5. <https://medium.datadriveninvestor.com/nlp-with-lda-analyzing-topics-in-the-enron-email-dataset-20326b7ae36f>
6. <https://business.connectid.io/docs?product=mail&type=architecture&lang=en-US>
7. <https://www.getmailbird.com/email-overload-survey/>
8. <https://medium.com/nanonets/topic-modeling-with-lsa-psla-lda-and-lda2vec-555ff65b0b05>
9. <https://par.nsf.gov/servlets/purl/10055536>
10. <https://towardsdatascience.com/2-latent-methods-for-dimension-reduction-and-topic-modeling-20ff6d7d547>

Appendix

The code of the project is as follows:

Data preprocessing

1. Text cleaning

```
#converting text column to list
data=df.text.values.tolist()
```

```
# Remove new line characters
data = [re.sub(r'\s+', ' ', sent) for sent in data]
```

```
# Remove distracting single quotes
data = [re.sub(r"'", "", sent) for sent in data]
```

```
#printing email in row 1
pprint(data[:1])
```

```
['Subject: enron methanol ; meter # : 988291 this is a follow up to the note i '
'gave you on monday , 4 / 3 / 00 { preliminary flow data provided by daren } '
'. please override pop s daily volume { presently zero } to reflect daily '
'activity you can obtain from gas control . this change is needed asap for '
'economics purposes .']
```

2. Tokenization

#Tokenization is the act of breaking up a sequence of strings into pieces such as words, keywords, phrases, symbols and other elements called tokens.(some characters like punctuation marks are discarded.)

```
def sent_to_words(sentences):
    for sentence in sentences:
        yield(gensim.utils.simple_preprocess(str(sentence), deacc=True)) # deacc=True removes punctuations
data_words = list(sent_to_words(data))
print(data_words[:1])
```

```
[['subject', 'enron', 'methanol', 'meter', 'this', 'is', 'follow', 'up', 'to', 'the', 'note', 'gave', 'you', 'on', 'monday',
'preliminary', 'flow', 'data', 'provided', 'by', 'daren', 'please', 'override', 'pop', 'daily', 'volume', 'presently', 'zer
o', 'to', 'reflect', 'daily', 'activity', 'you', 'can', 'obtain', 'from', 'gas', 'control', 'this', 'change', 'is', 'neede
d', 'asap', 'for', 'economics', 'purposes']]
```


3. Stemming and Lemmatization

```
'''Stemming is the process of reducing a word to its word stem that affixes to suffixes and
prefixes or to the roots of words known as a lemma.
Lemmatization, unlike Stemming, reduces the inflected words properly ensuring that the root word belongs to the language'''

def lemmatization(texts, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']): # 'NOUN', 'ADJ', 'VERB', 'ADV'
    texts_out = []
    for sent in texts:
        doc = nlp(" ".join(sent))
        texts_out.append(" ".join([token.lemma_ if token.lemma_ not in ['-PRON-'] else '' for token in doc if token.pos_ in allowed_postags]))
    return texts_out
```

```
#en_core_web_sm is a small English pipeline trained on written web text (blogs, news, comments), that includes vocabulary,
#syntax and entities.
sp = spacy.load("en_core_web_sm")
```

```
# Initialize spacy 'en' model, keeping only tagger component (for efficiency)
# Run in terminal: python -m spacy download en
nlp = spacy.load("en_core_web_sm", disable=['parser', 'ner'])
# Do Lemmatization keeping only Noun, Adj, Verb, Adverb
data_lemmatized = lemmatization(data_words, allowed_postags=['NOUN', 'VERB']) #select noun and verb
print(data_lemmatized[:2])
```

```
['methanol meter follow note give flow datum provide override volume reflect activity can obtain gas control change need asa
p economic purpose', 'see attach file']
```

4. Remove stop words and Count vectorizer

```
'''LDA topic model algorithm requires a document word matrix as the main input.
Configured the CountVectorizer to consider words that has occurred at least 10 times (min_df), remove built-in
english stopwords, convert all words to lowercase, and a word can contain numbers and alphabets of at least length
3 in order to be qualified as a word'''
vectorizer = CountVectorizer(analyzer='word',
                             min_df=10,
# minimum reqd occurrences of a word
                             stop_words='english',
# remove stop words
                             lowercase=True,
# convert all words to lowercase
                             token_pattern='[a-zA-Z0-9]{3,}',
# num chars > 3
                             # max_features=50000,
# max number of unique words
                             )
data_vectorized = vectorizer.fit_transform(data_lemmatized)
```

Model

Building LDA model

```
# Build LDA Model
lda_model = LatentDirichletAllocation(n_components=20,           # Number of topics
                                     max_iter=10,
# Max learning iterations
                                     learning_method='online',
                                     random_state=100,
# Random state
                                     batch_size=128,
# n docs in each learning iter
                                     evaluate_every = -1,
# compute perplexity every n iters, default: Don't
                                     n_jobs = -1,
# Use all available CPUs
                                     )
lda_output = lda_model.fit_transform(data_vectorized)
print(lda_model) # Model attributes

LatentDirichletAllocation(learning_method='online', n_components=20, n_jobs=-1,
                          random_state=100)
```

```
#Found best hyperparamters values and used in the model
LatentDirichletAllocation(batch_size=128, doc_topic_prior=None,
                          evaluate_every=-1, learning_decay=0.7,
                          learning_method='online', learning_offset=10.0,
                          max_doc_update_iter=100, max_iter=10, mean_change_tol=0.001,
                          n_components=10, n_jobs=-1, perp_tol=0.1,
                          random_state=100, topic_word_prior=None,
                          total_samples=1000000.0, verbose=0)

LatentDirichletAllocation(learning_method='online', n_jobs=-1, random_state=100)
```

Model evaluation

```
# Log Likelihood: Higher the better
print("Log Likelihood: ", lda_model.score(data_vectorized))
# Perplexity: Lower the better. Perplexity = exp(-1. * log-Likelihood per word)
print("Perplexity: ", lda_model.perplexity(data_vectorized))
# See model parameters
pprint(lda_model.get_params())
```

```
Log Likelihood: -718685.926504486
Perplexity: 355.111797436037
{'batch_size': 128,
 'doc_topic_prior': None,
 'evaluate_every': -1,
 'learning_decay': 0.7,
 'learning_method': 'online',
 'learning_offset': 10.0,
 'max_doc_update_iter': 100,
 'max_iter': 10,
 'mean_change_tol': 0.001,
 'n_components': 20,
 'n_jobs': -1,
 'perp_tol': 0.1,
 'random_state': 100,
 'topic_word_prior': None,
 'total_samples': 1000000.0,
 'verbose': 0}
```

Finding dominant topic

```
# Show top n keywords for each topic
def show_topics(vectorizer=vectorizer, lda_model=lda_model, n_words=20):
    keywords = np.array(vectorizer.get_feature_names())
    topic_keywords = []
    for topic_weights in lda_model.components_:
        top_keyword_locs = (-topic_weights).argsort()[:n_words]
        topic_keywords.append(keywords.take(top_keyword_locs))
    return topic_keywords
topic_keywords = show_topics(vectorizer=vectorizer, lda_model=lda_model, n_words=15)
# Topic - Keywords Dataframe
df_topic_keywords = pd.DataFrame(topic_keywords)
df_topic_keywords.columns = ['Word '+str(i) for i in range(df_topic_keywords.shape[1])]
df_topic_keywords.index = ['Topic '+str(i) for i in range(df_topic_keywords.shape[0])]
df_topic_keywords
```

```
/Users/muditagarg/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function get_feature_names is deprecated; get_feature_names is deprecated in 1.0 and will be removed in 1.2. Please use get_feature_names_out instead.
```

```
warnings.warn(msg, category=FutureWarning)
```

	Word 0	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7	Word 8	Word 9	Word 10	Word 11	Word
Topic 0	gas	day	volume	nomination	supply	nominate	month	thank	forward	production	flow	shut	fol
Topic 1	com	message	send	mail	email	http	address	service	www	internet	web	visit	hotn
Topic 2	location	list	stock	corporation	monitor	locate	reflect	station	quote	floor	department	communicate	cre
Topic 3	meter	deal	volume	thank	need	flow	day	know	gas	let	contract	month	alloc
Topic 4	phone	say	person	know	thank	people	number	hope	fax	love	tell	tax	fc

Assigning labels to the topics

```
Topics = ["Purchase","Exigency","Contact/Support","Requisition","Notification",
          "Health/wellness", "Basic Details", "Order processing", "Graphic Design",
          "Delivery details","Personal details","Allocation","Frontend","Software",
          "Stock information","Operation","Transaction","Process","Time period","Travelling"]
df_topic_keywords["Topics"]=Topics
df_topic_keywords
```

Word 2	Word 3	Word 4	Word 5	Word 6	Word 7	Word 8	Word 9	Word 10	Word 11	Word 12	Word 13	Word 14	Topics
contract	meter	purchase	transport	oil	volume	energy	ticket	agreement	spot	price	path	mmbtu	Purchase
volume	thank	gas	day	daren	subject	flow	know	sale	need	let	farmer	ticket	Exigency
contact	mail	link	message	list	availability	sale	email	reply	line	owner	copy	send	Contact/Support
use	number	report	employee	process	question	note	request	provide	list	team	receive	survey	Requisition
outage	time	sit	training	attend	server	koch	subject	environment	begin	location	impact	room	Notification
rescription	med	viagra	health	pain	cornhusker	man	doctor	medication	woman	need	stop	weight	Health/wellness

Predicting Incoming email

```
# Define function to predict topic for a given text document.
nlp = spacy.load('en_core_web_sm', disable=['parser', 'ner'])
def predict_topic(text, nlp=nlp):
    global sent_to_words
    global lemmatization
    # Step 1: Clean with simple_preprocess
    mytext_2 = list(sent_to_words(text))
    # Step 2: Lemmatize
    mytext_3 = lemmatization(mytext_2, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV'])
    # Step 3: Vectorize transform
    mytext_4 = vectorizer.transform(mytext_3)
    # Step 4: LDA Transform
    topic_probability_scores = lda_model.transform(mytext_4)
    topic = df_topic_keywords.iloc[np.argmax(topic_probability_scores), 1:14].values.tolist()

    # Step 5: Infer Topic
    infer_topic = df_topic_keywords.iloc[np.argmax(topic_probability_scores), -1]

    #topic_guess = df_topic_keywords.iloc[np.argmax(topic_probability_scores), Topics]
    return infer_topic, topic, topic_probability_scores
# Predict the topic
mytext = ["tomorrow, i would like to refine this spreadsheet and try to support some of the population and job growth assumpt
infer_topic, topic, prob_scores = predict_topic(text = mytext)
print(topic)
print(infer_topic)
```

```
['know', 'thank', 'let', 'change', 'subject', 'farmer', 'send', 'daren', 'deal', 'desk', 'message', 'think', 'want']
Process
```