PIGEON: An Interest-Based, Targeted Email Service for Large Organizations

Cissy Chen 2016

Advisor: Robert Fish

Fall 2015 Independent Work

Abstract

Large organizations, like colleges and companies, send a lot of internal advertising emails. These mass advertising emails are necessary, but they cause two pain points for recipients and senders. Recipients end up with cluttered inboxes with irrelevant emails they need to sift through, decreasing productivity; senders suffer from ineffective emails, since mass emailing reduces the chance of any one person reading the email. Pigeon is a two-pronged solution tackling both problems by allowing recipients to subscribe to tags they're interested in receiving emails about and senders to send emails with lists of tags that will only end up in the inboxes of the recipients who are likely to be interested in the email. Experiments suggest that Pigeon increases the relevance and effectiveness of emails that users send/receive, since emails sent through Pigeon yielded a 70% open rate, compared to a 20% open rate when sent through a traditional email list. Pigeon can also suggest tags to senders through applying natural language processing techniques to the email content. Preliminary results show that 32% of tags suggested by Pigeon are highly relevant but Pigeon often misses relevant tags altogether. While I'll describe the product as a whole, I will focus my implementation details and results on my half of the product, the Chrome Extension that solves the sender-side problem.

1 Introduction

Group email, especially on university campuses and companies, is unable to target certain interest groups on a per-email basis. For smaller, closed groups where the owner would like to closely control the membership, email lists may work, but for larger audiences, email lists have become spam-y and cluttered with predominantly advertisement emails that a majority of recipients do not read.

1.1 The Problem

To communicate within large organizations like university campuses and companies, many people use various group emailing services. The vast majority of campuses and companies use some form of group-mailing mechanism. These are used for internal / group-specific interactions, but are more commonly used for widespread advertising within the organization. Specifically, one sample from a Princeton student's inbox indicates that about 87% of campus emails sent on email lists within Princeton are for promotional or advertisement purposes. But this inevitably gives rise to cluttered and spam-filled inboxes, resulting in two pain points. First, cluttered inboxes reduce productivity and create distractions for recipients – people waste about 21 working days a year on unwanted emails (or half a day each week) [12]. Other productivity-harming mechanisms of disorganized emails are detailed in [3], which specifically cites that "the recipient is burdened by the cost of having to process and deal with the email" due to recipients/message mismatch. In fact, 10% of people receive more than 100 emails per day, which is more than one email every 6 minutes during the work day) [4].

Secondly, cluttered inboxes reduce the overall effectiveness of emails sent, because they're less likely to be read, even by people who might otherwise be interested in their content. This is clear on Princeton's campus from tests we've conducted – more than 80% of emails sent to residential college email lists (one of the largest student listservs) are never opened by a majority of recipients.

While the obvious answer might be to just send fewer advertising/promotional emails, that's

impossible in a large organization – if Sally is advertising a science internship opportunity, what else can she do but email each residential college listserv, in an effort to reach all interested parties in the student body? The problem is that Sally doesn't know who exactly consists of the "interest parties", so she sends it to everyone. But this is useless, because contributing to cluttered inboxes means that Sally's email might get lost in the fray and she needs to send 6 different emails in her hope to reach as many interested people as possible. This is equivalent to the classic targeted advertising concept, where targeting only the people who could be interested in the content of ad will decrease costs and reduce "wasted" advertising [7]? But in this case, we're looking to minimize inbox clutter for recipients and the effort necessary to target the right people for senders within a large organization.

Furthermore, although many large organizations theoretically have email lists for specific "interest" groups, they don't lead to interest-specific emails in practice. For example, while the email list "Wilsonwire" contains all of the students in the residential college Wilson, it is used to announce random things completely unrelated to Wilson, like dance auditions. This problem arises because it's still impossible to target certain interest groups on a *per-email* basis, since email lists are static and users can't create a separate list for every email they write.

In short, we have identified a problem caused by the way promotional/advertising emails are sent and received in large organizations, like university campuses and companies. Below, we'll discuss the market for this problem and a more detailed profile of target users.

1.2 Market Size

The larger, general email market is projected to be worth $23B globally by 2018 [11]. But since we are specifically targeting promotional/advertising emails sent within organizations, a more realistic market size is that of email marketing, projected to be worth $6.5B globally by 2018 [11]. This is a significant opportunity. Via future patenting of the technology, solutions to this problem are likely to have good exits, especially considering the potential to be solving a problem that email

giants like Gmail have yet to solve for large organizations.

1.3 Target Users

Our target users are members of large organizations, like university campuses and companies. Specifically, we built our product to serve both senders and recipients of emails by addressing both pain points outlined above.

A typical mass email Sender is Sally, the Marketing Chair of Science Club X at Princeton. She wants to advertise a STEM project award in order to maximize the number of received applications. Currently, she would need to send out the same email to each residential college email list. While she knows she'll be cluttering many people's inboxes and that her email will ultimately end up in the Trash of many disgruntled recipients, she has no other way to know exactly whom to target - where are all of the science/tech lovers out there? She goes through the trouble of finding and asking 5 friends in the other residential colleges and ask them to send the same email.

A typical mass email Recipient is Ryan, a freshman on Wilsonwire, a residential college email list. In his first week of classes alone, he's received hundreds of emails advertising various things - most of which he would never have any interest in. This overwhelming number of emails and the fact that most of them are irrelevant to him renders him less likely to read any one specific email, so Ryan might have missed out on the science project application even though he's completed a pretty exciting science project in the past year.

2 Related Work

There are several existing solutions that make an effort to improve organization-wide email communication and inbox management, which I will detail before discussing why they are suboptimal for solving the identified problem.

2.1 Listservs

Listservs, patented in 1986, are used to create many different email groups within a larger organization [8]. This is what Princeton and many other university campuses use. An owner must edit subscribers and maintain the Listserv, which is especially painful after seniors graduate every year. They are used for categorizing along interest dimensions, but these are static and often too broad or irrelevant for what they're actually used for. Listservs are the most widely used form of email list management among university campuses.

2.2 Personal Inbox Management Apps/Tools

Many individuals try to keep their inboxes clean by using personal inbox management apps like Inbox or tools like Gmail filtering. However, there is a lot of risk in missing relevant information because the sender and the receiver haven't agreed to speak the same language with regards to categorization. For example, I might filter for emails that have the word "Dance" in it because I'm interested in dance, but a dance group auditions email might never actually use the word "dance." Furthermore, it doesn't get to the root of the problem, which is that senders are sending irrelevant emails to uninterested recipients in the first place.

2.3 Slack

Slack is a chat service that many younger, technology-forward companies use that is *not* email. Instead, it is a desktop/mobile app that requires organizations to make Channels, which are interest-based and more dynamic since people create a lot of them without too much overhead. However, the problem with this service is that it isn't at all linked to your email and therefore isn't great for long-form content. Also, building a tool directly into email might yield faster adoption than an entirely new app/service like Slack. Slack has passed 1 million daily active users, with 300K paid users [15].

3 Product Features & Use Cases

With the problem and current solutions in mind, we set out to build Pigeon, an interest-tag-based email service built directly into Gmail that allows for dynamic and customizable targeting of each advertising email, specifically for use in large organizations. We use targeted advertising principles to match recipient characteristics (currently, self-declared interests via tags) to only the emails for which there is a good chance of a click-through. The solution is to intercept each email sent to an email alias like"pigeon@princeton.edu" that contains a list of topic tags, and redirect that email so that only the members of the organization who have tagged themselves as interested in those topics will receive the email. It is important to note that while we focused on widespread advertising emails, we also preserve the functionality that Listservs provide for more conversational communication in private, static groups within an organization. This will be discussed below.

This product consists of two halves, a Chrome Extension interacting with Gmail to serve mass email Senders and a web app to serve mass email Recipients. I worked on the full-stack sender-side product while my partner worked on the full-stack recipient-side product. A demo of the product can be found in the link in Appendix H.

3.1 Chrome Extension for Senders

The Chrome Extension only needs to be downloaded by people needing to send emails - this would most likely eventually consist of all members of the organization. The landing page (i.e. Chrome Extension popup) directs you to your Gmail, where all of the Pigeon mechanisms come in. There will be a "Send via Pigeon" button in your Gmail, which will allow you to compose a message, assign tags manually, ask Pigeon for automatically suggested tags, and send the message. Pigeon will do all of the redirecting from there. The following images show the user flow for sending an email via Pigeon.

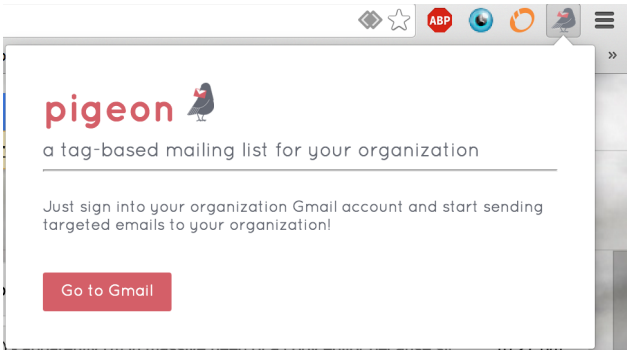


Figure 1: The chrome extension landing page. Button takes user to Gmail account.

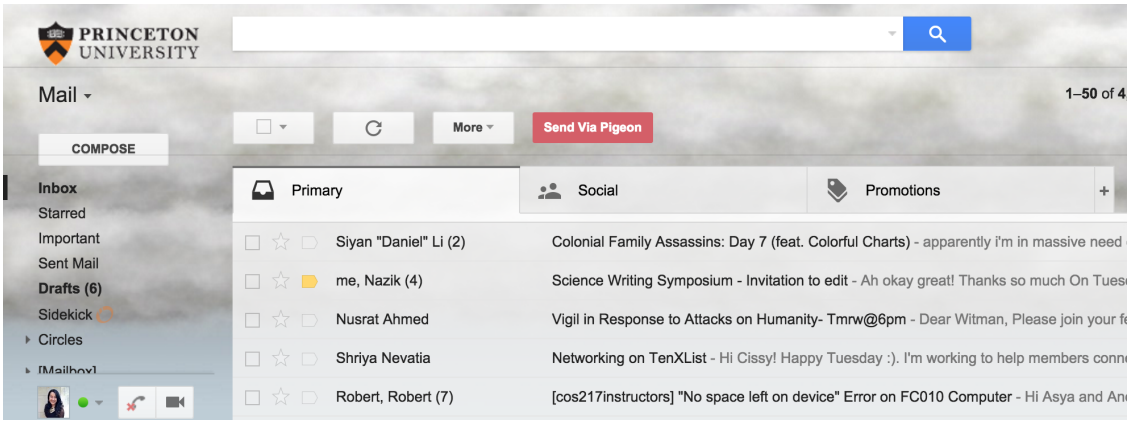


Figure 2: Upon arriving in your Gmail inbox, you'll see a 'Send via Pigeon' button.

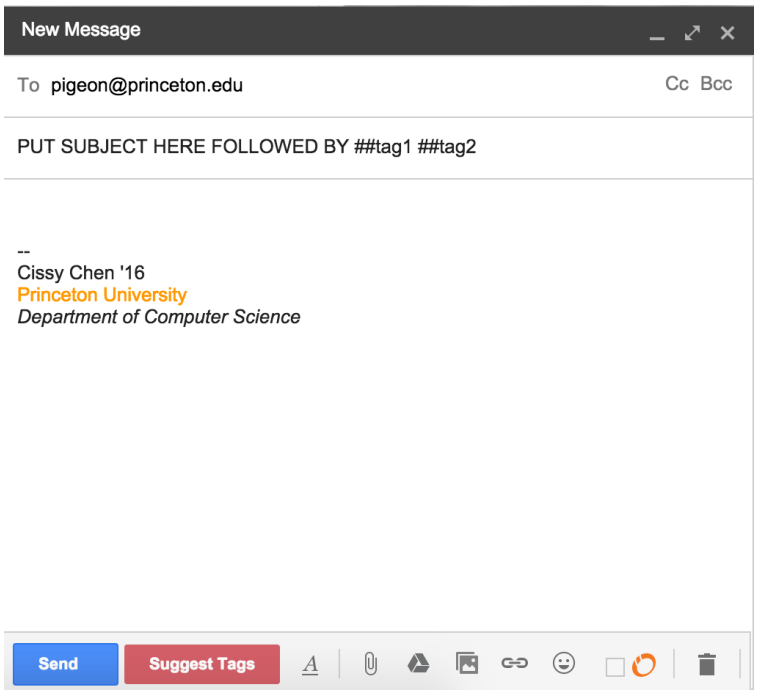


Figure 3: The 'Send via Pigeon' button opens up a compose window with some fields filled in. Tags are listed manually at the end of your subject line (they're removed when the email is actually sent). You can also click 'Suggest Tags' to have Pigeon recommend you a list of tags in your organization that might fit the content of your email via an alert.

This helps Sally, for example, by giving her a way to send an email to only students subscribed to science and engineering tags for her science project application. She doesn't need to send multiple emails, and can be assured that the recipients of her email have a high chance of reading her email due to the relevant content.

3.2 Web App for Recipients

The web app is where all members of the organization need to sign into via their @organization Gmail accounts and indicate their preferences for receiving emails. Each member signs into their organization's page, where they can see the list of all tags in their organization and subscribe/unsubscribe/add tags accordingly. These tags are member-generated for now. These are the

tags that senders will use to categorize their emails.

This helps Ryan, by cleaning up his inbox through getting to the root of the problem. He only receives emails that he has a good chance of being interested in, while allowing him to manually choose the topics he's interested in.

3.3 Key Features

Overall, there are four key factors and features to Pigeon that differentiate it from the existing solutions. First, Pigeon allows the user to easily customize preferences - you can choose which tags interest you so you only get relevant emails in your inbox and you can subscribe/unsubscribe as you wish. This is enabled by the web app. With Listservs, your membership to various groups must be controlled by an administrator and aren't easily changeable.

Secondly, Pigeon is convenient to use - after the Chrome Extension is downloaded, the user can use Pigeon directly from his Gmail inbox and send emails as he normally would, just with some additional features.

Thirdly, Pigeon allows for targeted sending of emails so instead of filtering out relevant emails after they're received, which is a burden on the recipients, it gets to the root of the problem. This is enabled through the Chrome extension and tagging mechanism.

Lastly, Pigeon allows for automatic tagging where Pigeon can suggest tags in your organization that may fit the content of your email. This helps the sender do some of the work of manually tagging.

Additionally, Pigeon is a two-pronged solution - for senders and for recipients of advertising emails. Appendix B shows how Pigeon compares to its competitors.

3.4 Supporting other use cases

Pigeon can still support the use case of non-advertising emails, like a President's email to her club members. Members of Club X can subscribe to a tag "ClubX" on the web app, and the President

simply needs to send an email with just tag “ClubX.” Furthermore, if there is a need for additional security, we can build that in so that users can create ’private’ tags that require a password to subscribe to. Currently this is not supported, since our priority was to support the main use case of promotional emails.

4 Architecture & Implementation

Pigeon contains 2 separate apps, as mentioned, the web app and the Chrome extension. My partner and I worked on the backend and database together, and I was responsible for the Chrome extension for email senders. The Github repository for the web app (frontend and backend) is entitled “pigeon” and the repository for the Chrome extension is entitled “pigeon-chrome” (links can be found in Appendix A).

The architecture for my part of the project is shown below.

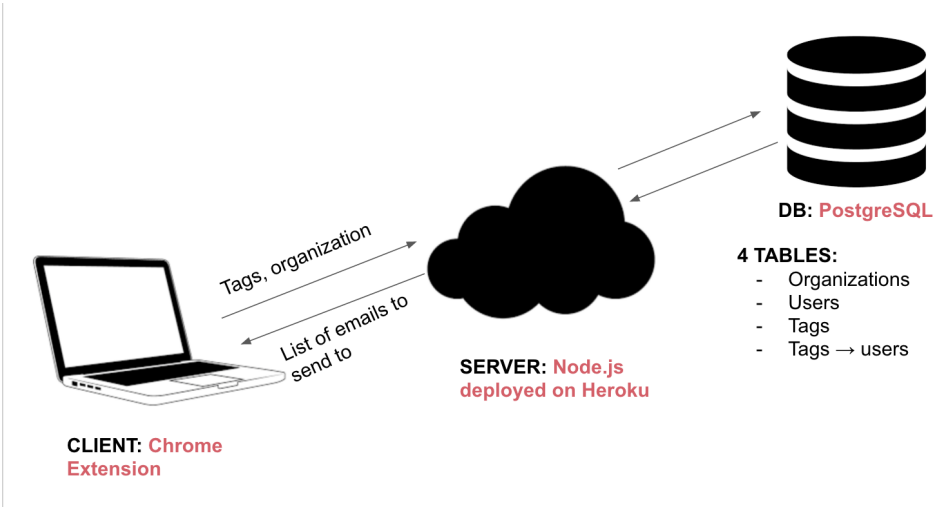


Figure 4: Chrome extension that interacts with a Node.js server deployed on Heroku with a PostgreSQL database.

4.1 Key Ideas

The key ideas for my sender-side Pigeon product are to provide the following functionalities to users:

1. Create a list of targeted recipients based on a list of tags the user inputs
2. Send tagged emails directly from Gmail without the emails indicating that they're using Pigeon
3. Automatically generate tag suggestions for users based on email content.

4.2 Chrome Extension

When building the sender-side product, there were three viable options for the frontend – a web app, a Chrome extension, or a Gmail Gadget. The web app would have required users to sign in, compose a message in a foreign interface, and upon submission of the message, be redirected to their email app to send the actual message. While this would have left a lot of room for improving the UI and customizing it so that users could select tags in a cleaner, clearer manner, this would have been inconsistent with how users normally send emails. Needing to first navigate to a separate web app to send an email through Pigeon is too high of a barrier and would harm the rate of user acquisition. Then, I looked into options that would allow me to directly build Pigeon functionality into Gmail. A Gmail Gadget is a combination of HTML and Javascript components that extend existing Gmail functionality [5]. While this would have allowed me to easily build functionality directly into Gmail, I noticed that some types of gadgets are now deprecated and while they can be tested, they're difficult to launch to other users' computers. So there is a risk for gadgets to be untestable and deprecated, which, coupled with the fact that there is little documentation around these types of apps, made Gadgets a suboptimal choice.

I chose to implement a Chrome extension because it offers more flexibility, documentation support, can be "downloaded" to a user's computer relatively easily, and fulfills Key Idea 2 of sending

emails directly from Gmail. I used javascript content scripts, which are run in the background when the extension is enabled. The Pigeon content scripts are only activated when a user navigates to a Gmail URL. The main external JS libraries I used were jQuery and Gmail.js, an open-source, unofficial, but widely used Javascript library that provides APIs for interacting with existing Gmail elements [6].

The "Send Via Pigeon" toolbar button fills in hard-coded Pigeon information based on the Gmail domain of the user (i.e. @princeton.edu). It also activates a listener for any send message events, using the Gmail.js library. Before a message is sent off to the Gmail servers, Pigeon processes the email tags from the subject line, error checks, and pulls in the correct email addresses to the BCC line. The email is then released to Gmail's servers and sent as normal. A more detailed picture of the logic flow of the implementation of Key Idea 1 is shown in Appendix C.

The key challenge in building this Chrome extension frontend is figuring out how to integrate and build new buttons and Pigeon UI with Gmail's current frontend infrastructure. There are limits to user interface customization due to this interaction and integration with Gmail code. Pigeon works best without other third-party Chrome extensions that affect Gmail enabled - namely, there are known interference problems between Pigeon and the email-scheduling extension, Boomerang, due to incompatibilities of the Gmail.js library with scripts that Boomerang includes.

4.3 Backend Details

The PostgreSQL database schema we use included 4 tables (organizations, tags, users, and tags_users) and we use foreign keys between them to ensure all entries are consistent so that deletion of any tags, users, and organizations would cascade deletions of rows of the appropriate tables.

We use the Express.js web framework for Node to build our server. While my partner created and uses entity models (Tag, Organization, User) for the receiver-side web application, I created the necessary RESTful API endpoints to be called by my Chrome extension. There's a wrapper function in db.js that executes a given SQL query with the given parameters, which I use to write

each of my endpoints. All of the endpoints can be found in api/api_chrome.js. The most notable ones are the /get-union-users-tag-org and /suggest-tags. I will discuss /suggest-tags in a future section. For retrieving the users subscribed to the tags in the email, I had to make the following decision. When Pigeon parses a list of tags "##tag1 ##tag2", I needed to consider what that would mean most intuitively to users – namely, the union or the intersection of the users subscribed to those tags? While ideally, the user would be able to identify which of these he would prefer in the UI, it's difficult to imagine a clean way to indicate exactly what the user wants when there are multiple tags and adding this UI to Gmail is near impossible. Therefore, I interviewed some potential users who overwhelmingly indicated that the union of the users subscribed to the list of tags is the most intuitive, especially when it's topic-based.

Another challenge was calling these endpoints from the Chrome extension, since I needed to deal with cross-origin requests. I solved this in 3 ways. First, I added the server to the permissions line in the Chrome extension's manifest.json file, which is where all important setup information about an extension is found. I also added "Access-Control-Allow-Origin" headers to all POST responses from the server side, so they would be received by the Chrome extension. Another problem was that an OPTIONS request would be sent by the Chrome Extension before sending the actual POST request, which was being denied. Thus, on the server side, I needed to intercept the OPTIONS request and send a 200 status manually to the Chrome extension, so that the real POST request could go through.

4.4 Server-side Login/Sign-up via Google

I also implemented the login via Google flow for the receiver-side web application. We decided that it makes the most sense for users to sign up for Pigeon via Gmail, since they'll be using their Gmails to send emails anyway. To do this, I used Passport.js, an authentication middleware built for Node.js [10]. I created a Google developers project where I registered the Pigeon web application and received the keys necessary to use the Google OAuth 2.0 authentication strategy. Passport's

APIs include serializeUser(), which stores the user email in a session after the initial login authentication, and deserializeUser(), which is called on every authenticated request that gets the user associated with the session. This deals with authenticated sessions in a very clean way. When someone logs in with Google for the first time, we also check whether their domain (@princeton.edu) represents an organization that already exists (if not, we present an error, indicating that an organization must first be initialized with the correct domain). If the user logs in for the first time and they're a part of a valid organization, we create a new user in our system.

4.5 Automatic Tagging Algorithms

This section details the /suggest-tags endpoint, that parses the content in the body of the email being composed and suggests a list of tags that are in the user's organization. This makes progress on Key Idea 3. This is similar to blog post content tagging, which has been studied in both Chun et. al. and Sood et. al.

Chun et. al uses semantic-driven artificial neural networks that learn from how other users tag their content – specifically, to train the ANN, the researchers crawled existing blogs that were manually tagged. A list of keywords were extracted based on statistical methods (including Tf-idf [14]) and were processed for semantic meaning by using WordNet, a linguistic dictionary commonly used for natural language processing techniques. The key innovation of this research was the reliance on collective intelligence in the training and test sets that used information that many bloggers/users tagged collaboratively.

Sood et. al employs unsupervised learning instead, focused on finding similar blog posts via clustering, compiling the list of tags that those blog posts were tagged with, and using statistical methods to suggest the most pertinent ones to the user. Specifically, they utilize a lossless tag compression of the list of all of the tags of all similar blog posts.

The key difference between Pigeon tagging and blog post tagging is that Pigeon tags are often not real words – for example, "cos" has no semantic meaning beyond what Princeton students

assign to it. This means there's essentially a separate "dictionary" of tags to recommend from for every organization on Pigeon - I discuss below how I handle this using search algorithms. However, with these tagging methods as reference, I created by own system for tagging emails for Pigeon by using an NLP processing library to create a list of relevant keywords and concepts, which I then processed based on lexical and semantic similarities into a list of Pigeon tags. There were two sequential parts to my solution.

First, I translate the email text content into a list of associated Keywords and Concepts. To do this, I use the free and well-documented NLP library, AlchemyAPI, that uses statistics, machine learning via deep neural networks, and word taxonomy to process text content [1]. Their neural networks are trained on a huge list of webpages that they crawl to learn semantic structures of various languages. Specifically, I use the Concept Tagging and Keyword Extraction functions provided. Concept Tagging helps extract high-level topics - for example, if the email mentions "Empire State Building" and "Rockefeller Center," it might tag the concept "New York City," even if it's not explicitly mentioned in the email. This happens via crawling webpages about these concepts and using deep neural networks. I collect the top 5 concepts returned by this function for further analysis. Keyword Extraction uses statistics and NLP techniques to find keywords that are directly found in the text and represent the content of the text best - this works well for identifying organization-specific terms, such as "Princeton Innovation," which is the name of a student club on Princeton's campus. Since tags are often organization-specific, keywords might sometimes work better than general concepts. I collect the top 5 keywords, too.

The next step is to translate these list of Keywords and Concepts into any associated tags that exist in the database for the user's organization. This requires me to basically implement a searching mechanism - specifically, to search for the Keywords and Concepts in the names and description of all tags. To do this, I use fuzzy string search to search for Keywords in the names of tags and full-text search to search for Concepts in the descriptions of tags.

In fuzzy string search, Keywords are searched for directly by comparing them with tag names

by calculating string similarity based on the number of trigrams in common. Trigrams are consecutive 3-character substrings of a string and is an accepted and effective way of comparing strings [9]. I experimented with different similarity index thresholds, to accurately distinguish strings that are likely to be actually different from strings that should match but account for different forms of words or typos that might arise. I ultimately chose a similarity index threshold of 0.4, which seems reasonable for catching any typos that may arise and matching multi-word strings. I tended towards a lower threshold to account for the systematic length differences of tag names and Keywords (tag names are often shortened versions (less than 10 characters) of the keywords) and because it is probably better to over-suggest than to under-suggest tags. Some examples of similarity indices: 'Innovation' and 'Innovation website' have a trigram-based similarity index of 0.58, 'musical theater' and 'theater' have an index of 0.5, and 'science policy work' and 'science' have an index of 0.4.

In full-text search, Concepts are searched for in tag descriptions by using the concept of lexemes, which are the smallest meaningful form of an English word, similar to word stems. Specifically, a string of text (like that of the tag description) is translated to a list of lexemes by removing "stop words" (words that have no searchable meaning, like "the", "and", "him", etc.), eliminating casing and tense as well as looking at meaning/synonyms. For example, the text "Try not to become a man of success, but rather try to become a man of value", would be translated to a list of lexemes like {'becom', 'man', 'rather', 'success', 'tri', 'valu'}. Therefore, a list of Concepts includes the word "Entrepreneurs" would still match a tag description that includes the word "entrepreneurship." Therefore, each word in the list of Concepts is lexeme-ified and searched for in the list of lexemes that represents the tag description.

So in the end, the tags that are either returned by the fuzzy string search in tag names or the full-text search in the tag descriptions are returned as a list of suggested tags to the user.

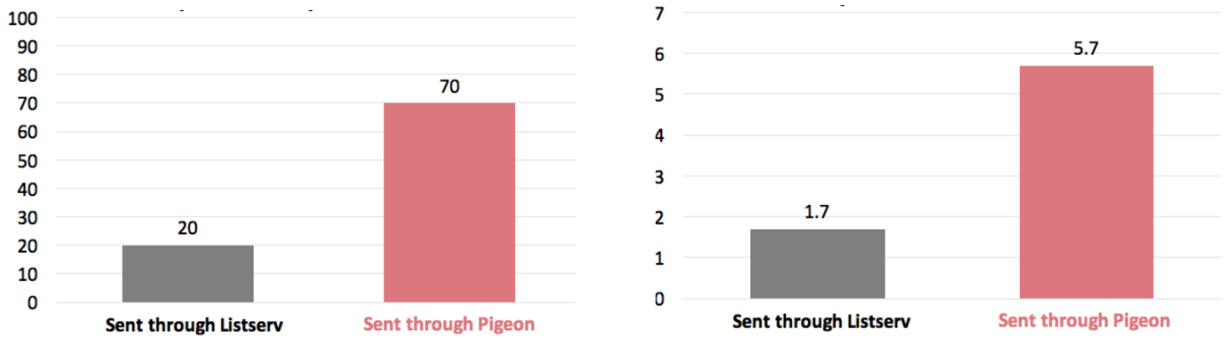
5 User Testing & Evaluation

The success metrics of Pigeon as a product and venture are based on the number of organizations acquired as well as the percentage penetration within each of the organizations. More specifically though, there are three key metrics that influence these high-level success metrics that we tested in our alpha launch at Princeton, through user interviews, surveys, and experiments.

5.1 Usefulness

Our hypothesis was that Pigeon's targeted emails make recipients more likely to read one's email since we're increasing the relevance of emails that users receive. Thus, to test this, we can track the percentage of emails opened when sent through Pigeon versus the control of the percentage of emails opened when sent through the existing solution, a Listserv. Specifically, we composed a real advertising email for a science/tech project award application. We sent that same email through 2 different mediums – via Pigeon with a list of relevant tags, specifically ("##science ##entrepreneurship ##cos ##phy ##mol ##chm ##mat ##orfe ##mae ##ele ##ast ##cbe ##cee ##eeb ##psy"), and via a typical residential college Listserv, Wilsonwire. It is important to note that in both scenarios, the recipients receive *exactly* the same email, including the subject line – there is no indication that an email is sent through Pigeon, so we isolated the email sending means as the independent variable.

We sent these emails with Sidekick, a third-party chrome extension widely used to track whether recipients have opened your email. We tracked both the number of distinct opens of the emails as well as clicks on the application link found in the email. Our results are shown in the graphs below.



(a) % of recipients who opened the email

(b) % of recipients who clicked the link in the email

Figure 5: The same advertising email was sent via a Listserv (Wilsonwire) and via Pigeon. The Listserv contained 752 recipients and Pigeon's algorithm calculated 70 recipients.

There are a couple things to note. First of all, shown by a statistical t-test, these results indicate that Pigeon yielded a statistically significant improvement over Listservs for both metrics of email opens (with a P-value of 0.0001) and link clicks (with a P-value of 0.012679). The calculations are shown in Appendix D. This accounts for the differing sizes of the two samples, given that they are independent, which we have no way of ensuring with our current method of experiment, but that is a comfortable approximation. From this experiment, we can conclude that sending an email through Pigeon (with the appropriate tags) yields a higher rate of email opens and email actions (i.e. link clicks) by the recipients of the emails. This suggests that emails sent through Pigeon are perceived to be more useful/relevant to the recipients due to the tag-based targeting.

However, it is possible that we care more about the raw number (instead of percentage) of these actions – for example, imagine a campus of 100 people and 25 of them are deemed to be interested in a specific email, by Pigeon's tagging mechanism. If I send through Pigeon, I would probably get around 18/25 people to read the email. If I send to all 100 people though, perhaps I'd only get 20 people to read the email, but all 18 of those people who would've read the email sent through Pigeon could very well be included in those 20. So perhaps Pigeon doesn't have a huge impact on getting more interested parties to read the email, since they would've read it anyway. But we

reduced the inbox clutter of the other 80 people who received that irrelevant email by simply not sending it to them. And we have shown here that we can reducing the sheer quantity of emails sent without reducing actual *interested* readership.

5.2 Accuracy

Another assumption is that we'd be able to alleviate some of the manual tagging effort for senders by automatically suggesting some tags based on the email content. To test the accuracy/relevance of these tags, I sent out a survey and received 31 responses. I arbitrarily selected 20 actual advertising emails sent within Princeton email lists (minimizing my bias in composing an email in a way that works with my algorithm) and generated actual suggested tags from my algorithm. 11 of them didn't create any suggested tags and I chose 5 emails to ask survey respondents about. I asked the subjects to select whether or not those tags were relevant given the original emails.

The results were as follows. 71% of the tags generated by Pigeon were deemed to be relevant by more than 50% of survey respondents; 54% of tags were deemed to be relevant by more than 60% of survey respondents; 32% of tags were deemed to be relevant by more than 80% of survey respondents. Conservatively, these results suggest about 32% of tags generated by Pigeon were highly relevant to their respective email contents.

Another key result observed was that about half of the emails (11/20) did not produce any suggested tags. While this might in part be due to our tag database lacking some important topics(i.e. "internships"), the survey responses suggest that there are relevant tags that Pigeon just fails to surface. This is perhaps a larger problem than the fact that Pigeon suggests tags that aren't relevant, since it seems more valuable to the user to over-suggest than to under-suggest. Under-suggesting might look like the tool simply doesn't "do anything" and over-suggesting at least can help us track which tags don't work.

The detailed results, along with the text of the emails and the suggested tags, can be found in Appendix E.

5.3 Usability

Lastly, I did some qualitative usability testing via in-person user interviews, so that I could see how users would use the Chrome Extension in action. I did these user tests throughout the semester, but focused most of them at the end. Specifically, I focused on asking my users whether this method of tagging is too inconvenient for the value it provides, and what the most valuable improvements to the UI would be. Results of the user interviews prompted me change the following UI elements for the Chrome Extension.

Initially, the method for sending a message via Pigeon did not involve a "Send via Pigeon" button - instead, the user just needed to open a compose window independently and fill in the TO field with "pigeon@princeton.edu." The Pigeon activated javascript listeners would always be activated on every opened compose window and would intercept every message sent, checking for this "pigeon@princeton.edu" in the TO field, and only activating the AJAX requests to the server if it existed. However, this led to a number of problems - on the product side, the user wasn't directly reminded or prompted to add the tags at the end of their subject line, and on the technical side, certain javascript dependencies were not loaded on the Gmail page in time for the Pigeon listeners to be activated. This means that the user could feasibly compose a message to pigeon@princeton.edu and send it before Pigeon was fully loaded. Thus, I decided to go through the complexity of embedding a button in the Gmail toolbar that only appeared when everything required for Pigeon was loaded properly. I also attached this button persistently to the toolbar, regardless of page reloads and HTTP requests sent by the page. These changes also restricted where Pigeon's code would be run, ensuring that Pigeon would only be activated on compose windows opened via that button, so Pigeon would have a lower chance of interfering with normal Gmail functionality, as I build out future features.

Secondly, when users are directed to open the chrome extension popup, they often ask whether they needed to sign into their organization-associated gmail account (i.e. "cissyc@princeton.edu"), or if they could just sign into their normal @gmail.com accounts and send Pigeon mail from

there. While they technically won't be able to send messages to any recipients (since their email domain must match what's in our database), the product itself didn't indicate this was the problem. Therefore, I added an alert when the user clicks on the "Send via Pigeon" button and his/her domain isn't in our database. This prevents the user from opening a Pigeon compose window and therefore sending a message through Pigeon if he is logged into the wrong Gmail account.

Thirdly, users wanted better integration of the "Suggest Tags" feature into the normal flow of users adding tags to the subject line. While generally users thought the current method of manually adding a list of tags to the subject line of the email was intuitive enough, they didn't like how the tag suggestions were made, requiring them to manually copy them over to the subject line. In response to this, I tried to make this process smoother, while adhering to Gmail styles and the restrictions of building javascript functionality into Gmail's complex web of code. The final UI allows the user to confirm the tag suggestions, after which the suggested tags will be added to the subject line automatically after a confirm.

6 Business Side Considerations

While building Pigeon, a consumer-facing product, it is necessary to also consider the potential to bring this product to market as a new venture, including possible monetization schemes and go-to-market strategy. This section will discuss these business-side considerations.

6.1 Business Model

Realistically, our initial focus upon launch will likely not be on monetization, even in a freemium way. This is due to our long-term strategy of being acquired by an email management service for organizations, like Gmail, that already allows large organizations to pick their own domain and host their email there. In this case, it is more important that we focus on building out high-impact features, patenting the automatic tagging technology, and acquiring a large, saturated base of users

in each organization that we launch to. Pigeon's success in an organization hinges on widespread adoption, which is unlikely to happen if Pigeon costs money, especially since the alternative is free. However, Pigeon is able to monetize, if necessary, due to one key advantage we have.

Through our tagging database, we can collect a lot of data about how content and tags affect both readership and calls to action within the email (such as "Fill out this form" links). We can recommend to senders what combination of tags to best target their audience and give them valuable data on the number of email opens and more importantly, how many *actions* their emails result in. This information is valuable to student clubs who want to promote tickets to their shows or advertise an event. And we can give student clubs this functionality in a dashboard tool that is available for a small monthly fee. They may be willing to pay this fee since it might directly increase ticket sales to an event, for example, so their cost is offset by the increased revenue they make.

An example of the freemium pricing is shown in Appendix F.

Alternatively, we could charge organizations a flat monthly fee for the Pigeon service - so instead of charging individual Princeton student clubs, we would charge Princeton's Office of Technology (OIT) department. This might make sense so we would only need one initial buy-in from the organization's tech department before launching in an organization – the tech department can set the norm for how mass email is done within the organization, which alleviates some of the effort needed to gain widespread use, which is necessary for Pigeon to be successful in an organization. However, the disadvantage of charging the central IT department is that they may not be as incentivized to pay for this kind of service, especially since they don't currently pay for Listservs and there isn't a clear benefit to them like there is for student clubs trying to advertise, for example. Perhaps Pigeon could lower email storage costs, but this hasn't been quantified in our research. Regardless, we need to consider all players in the ecosystem, understand their various buy-ins, needs, and motivations, before finalizing a monetization strategy.

6.2 Marketing and Strategy

The go-to-market strategy includes an initial launch in Princeton, which is in alpha-stage currently. There are 110 people signed up for Pigeon and we have done some user testing. The purpose of this launch is to get initial feedback on the 3 key features and test our hypotheses that these are useful via the aforementioned metrics. After this initial data, we will work on widespread adoption in Princeton via 2 different means - one, through the OIT department and two, through individual student clubs/existing Listservs. This allows us to gather data on which acquisition strategy works better. Through that, we can expand horizontally to other universities using one of the acquisition strategies. In order to monetize, we will at this point need to build out the dashboard analytics feature for student clubs. Then, we'll focus on vertical acquisition of entire campus communities via paid student ambassadors, so that entire universities will exclusively use Pigeon for their mass email communication needs. Finally after that, we will move into building for companies, which might have a different set of dashboard analytics needs, but the core product should remain valuable for them as a free product. A visual representation of this strategy is shown in Appendix G.

Ultimately, our most realistic exit would be to be acquired by Google's Gmail, since Gmail already has built-in large organization support and market penetration (i.e. most universities and companies use Google Apps for their email hosting).

7 Conclusion & Future Work

Pigeon is a tag-based mailing list system for organizations that provides senders a valuable way to reach their target audience. Mass advertising emails within large organizations, while necessary and useful, cause cluttered inboxes for recipients and ineffective emails for senders. Therefore, Pigeon is a two-pronged solution for both senders and recipients that ultimately leads to cleaner inboxes and more effective emails.

I built the sender-side product, a Chrome Extension that allowed senders to better target their

audiences. It is a fully functional tool, ready to use in Gmail inboxes for Princeton. We have 110 users signed up.

While not extensive, my experiments suggest that Pigeon's targeted emails will make recipients more likely to read an email since we're increasing the relevance of the emails that recipients receive. I found that 70% of recipients opened an email sent through Pigeon whereas only 20% of recipients opened that same email sent through a residential college Listserv.

Furthermore, I implemented a preliminary, but functional version of an automatic tagging system for emails sent via Pigeon that relies on an NLP library called AlchemyAPI and lexical/semantic similarity to extract Pigeon tags that match concepts and keywords in the email. About 32% of these suggested tags were highly relevant to the emails they came from, which is promising. However, more needs to be done to ensure all relevant tags are surfaced, since it seems that there are some emails that produce no suggested tags but that human subjects indicate there are relevant tags for.

7.1 Future Work

Therefore, future work would focus mostly on improving automatic tagging for Pigeon, since this is the most technically challenging and arguably most differentiating key idea. There are two improvements that could be made.

First, a key weakness of the current automatic tagging algorithm is that it uses a final text search of the database for tags that match the concepts/keywords generated using NLP techniques. This final text search relies heavily on good, informational, detailed descriptions of the tags, since that's where we do the searching. For example, if the tag "science" didn't include "research" in the description, it's unlikely it would be suggested for a research-related email. We can solve this in two ways – either prompt the user to make good tag descriptions via the UI or via a network of related tags that the recipient-side web app can produce, or by relying less on the descriptions and text search altogether. This leads to the second possible improvement.

Since relying on translating concepts/keywords to tags is difficult as seen above and Pigeon tags are often not real words (they're strings that mean something to the organization but have no independent semantic meaning), it makes more sense to change the automatic tagging system to use supervised learning instead. That way, we can train a model by observing manually tagged emails, and test the model by tracking which suggested tags users add and which they remove. This would rely on storing users' emails and tags, which is the main reason why I refrained from this option for this iteration of the product – this may lead to privacy issues and require much more data storage. The next steps would be researching how to do this in an efficient (in both memory and time) way especially as we scale, and in a way that alleviates privacy concerns, perhaps via encryption and/or data compression.

Lastly, more work could be done to improve the overall performance of the tool – perhaps fewer database connections should be made and optimizations could be made specifically to the suggest tags button. However, this is mainly part of the larger problem of scaling, since performance issues hasn't manifested itself in a organization with less than 100 tags. However, there are some user-noticed delays when using the suggest tags button.

8 Acknowledgements

I would like to thank my partner, Julia Wang, for working with me on Pigeon. I'd also like to thank Prof. Robert Fish, Yakir, and the entire Thursday *Innovation and Invention* IW seminar who provided valuable feedback on our progress every week. Thanks also to Trello and Google Calendar for helping us manage our tasks and keeping us on track over the semester.

References

[1] *AlchemyAPI - Powering the New AI Economy.* http://www.alchemyapi.com/

[2] Chun, A., & Lee, S. (2007). Automatic Tag Recommendation for the Web 2.0 Blogosphere Using Collaborative Tagging and Hybrid ANN Semantic Structures. *Proceedings of WSEAS, 88-93.*

[3] D'Ambra, J., Van Toorn, C., & Dang, G. (2007). The Negative Aspects of Email and Productivity: Towards Quantification. *ACIS 2007 Proceedings, (72)*. http://aisel.aisnet.org/acis2007/72

[4] *What we learned from analyzing 5,000 email inboxes.* (2014, September 4). http://blog.frontapp.com/2014/09/04/what-we-learned-from-analyzing-5000-email-inboxes/

[5] *Gmail Gadgets*. https://developers.google.com/gmail/gadgets_overview?hl=en

[6] Talwar, K. *Gmail.js - JavaScript API for Gmail*. https://github.com/KartikTalwar/gmail.js

[7] Iyer, G., Soberman, D., & Villas-Boas, J. M. (2005). The targeting of advertising. *Marketing Science, 24(3), 461-476.*

[8] *History of Listserv*. http://www.lsoft.com/corporate/history-listserv.asp.

[9] Owolabi, O., & McGregor, D. R. (1988). Fast approximate string matching. *Software: Practice and Experience, 18(4), 387-393.*

[10] *Passport*. http://passportjs.org/

[11] *Email Market, 2014-2018*. http://www.radicati.com/wp/wp-content/uploads/2014/10/Email-Market-2014-2018-Executive-Summary.pdf

[12] Seeley, M. (2011). *Brilliant email how to win back time and increase your productivity*. Harlow, England: Prentice Hall.

[13] Sood, S., Owsley, S., Hammond, K., & Birnbaum, L. (2007). TagAssist: Automatic Tag Suggestion for Blog Posts. *ICWSM*.

[14] Ramos, J. (2003, December). Using tf-idf to determine word relevance in document queries. *Proceedings of the first instructional conference on machine learning.*

[15] Newton, C. (2015, June 24). *More than a million people are using Slack every day now.* http://www.theverge.com/2015/6/24/8836087/slack-1-million-daily-users

9 Appendix

9.1 Appendix A

Pigeon receiver-side product + landing page: https://pigeonmail.herokuapp.com

Web App (frontend and backend): https://github.com/ccc343/pigeon

Chrome Extension: https://github.com/ccc343/pigeon-chrome

9.2 Appendix B

Pigeon's Competitors

	Works for senders?	Works for recipients?	Dynamic groups (per email)	Accurate topic filtering	Good for long-form content	Integrates with current email inbox
Listservs	✔	✘	✘	✔	✔	✔
Inbox Filtering	✘	✔	✘	✘	✔	✔
Google Inbox/Mailbox	✘	✔	✘	✘	✔	✔
Slack	✔	✔	✔	✔	✘	✘
PIGEON	✔	✔	✔	✔	✔	✔

Figure 6: How Pigeon compares to its competitors on a number of different dimensions.

9.3 Appendix C

Logic of the chrome extension

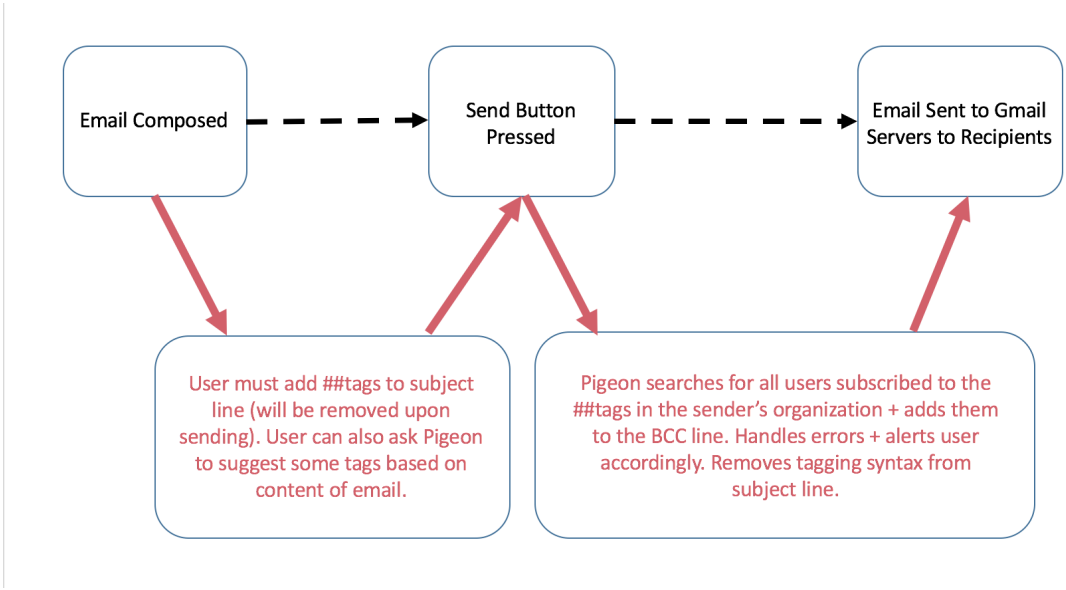


Figure 7: This figure demonstrates the chrome extension implementation. The dotted line indicates the normal flow of an email being composed and sent, whereas the red lines demonstrate what Pigeon does when it intercepts the email.

9.4 Appendix D

Table 1: The same advertising email was sent via a Listserv (Wilsonwire) and via Pigeon. The Listserv contained 752 recipients and Pigeon's algorithm calculated 70 recipients.

	SENT ON LISTSERV	**SENT VIA PIGEON**
Recipients opening email	148/752 (19.7%)	49/70 (70%)
Recipients clicking on link	13/752 (1.7%)	4/70 (5.7%)

Statistical test for % of recipients opening emails sent via Listserv vs. Pigeon:

- Difference between 2 sample proportions = $49/70 - 148/752 = 0.5032$.

- Overall sample proportion = $\frac{(148+49}{(752+70} = 0.2397$.

- Standard error = $\sqrt{0.2397 * (1 - 0.2397) * (1/70 + 1/752)} = 0.05335$.

- Test statistic = $\frac{0.5032}{0.05335} = 9.43$.

This test statistic yields a P-value of much less than 0.05 (closer to .00001), meaning there is a statistically significant difference between the two samples.

Statistical test for % of recipients clicking links sent via Listserv vs. Pigeon:

- Difference between 2 sample proportions = $4/70 - 13/752 = 0.0399$.

- Overall sample proportion = $\frac{(4+13}{(70+752} = 0.02068$.

- Standard error = $\sqrt{0.02068 * (1 - 0.02068) * (1/70 + 1/752)} = 0.0178$.

- Test statistic = $\frac{0.0399}{0.0178} = 2.24$.

This test statistic yields a P-value of less than 0.05 (closer to .012679), meaning there is a statistically significant difference between the two samples.

9.5 Appendix E

The 5 emails and their suggested tags can be found here: https://goo.gl/Mkk4Qs.

The detailed survey results for each of the 5 emails are visualized below:

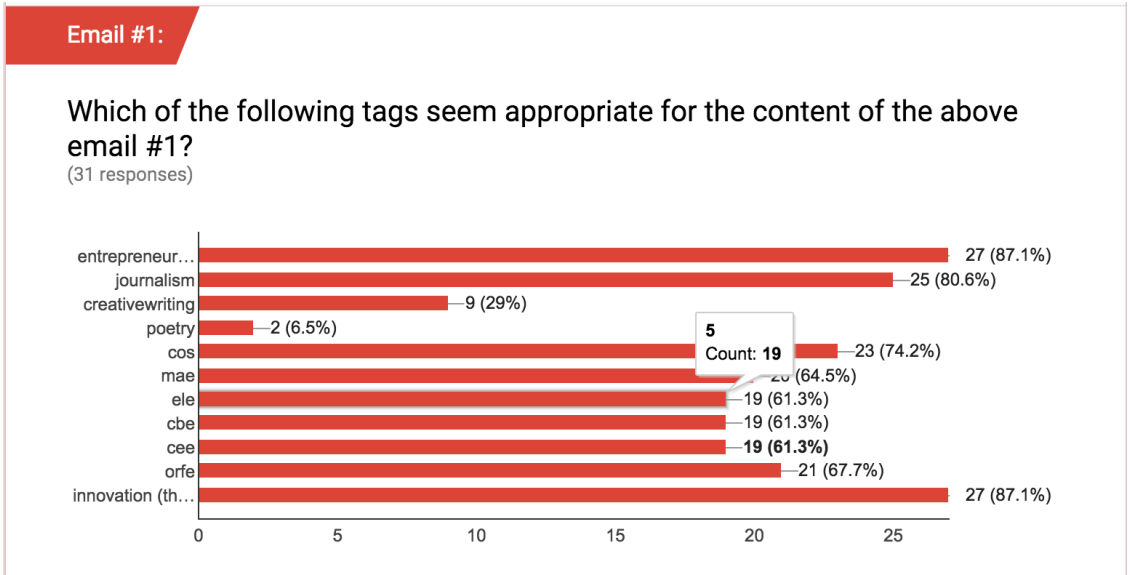


Figure 8: Email 1 survey responses

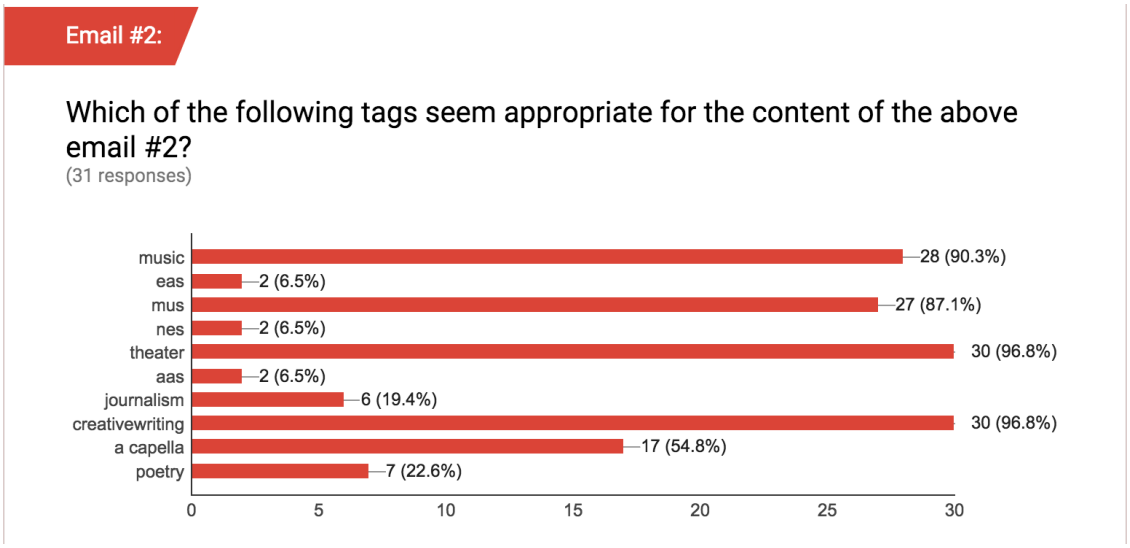


Figure 9: Email 2 survey responses

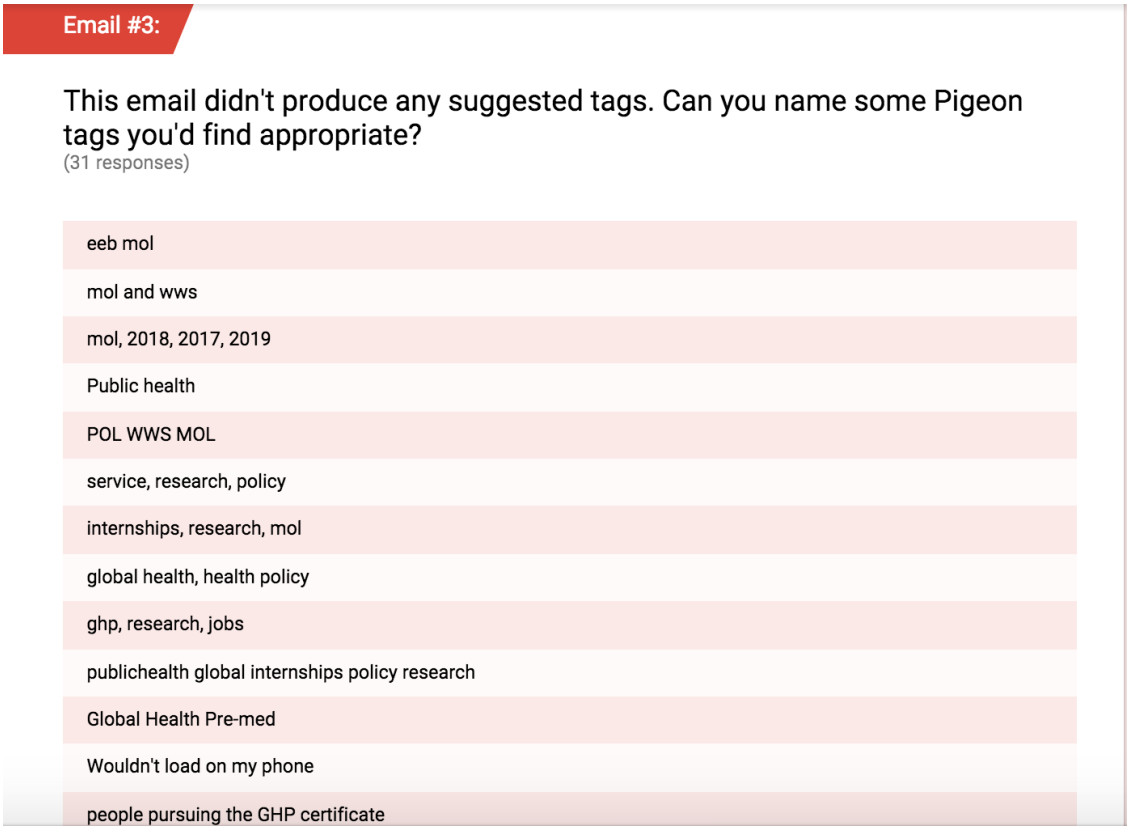


Figure 10: Email 3 survey responses

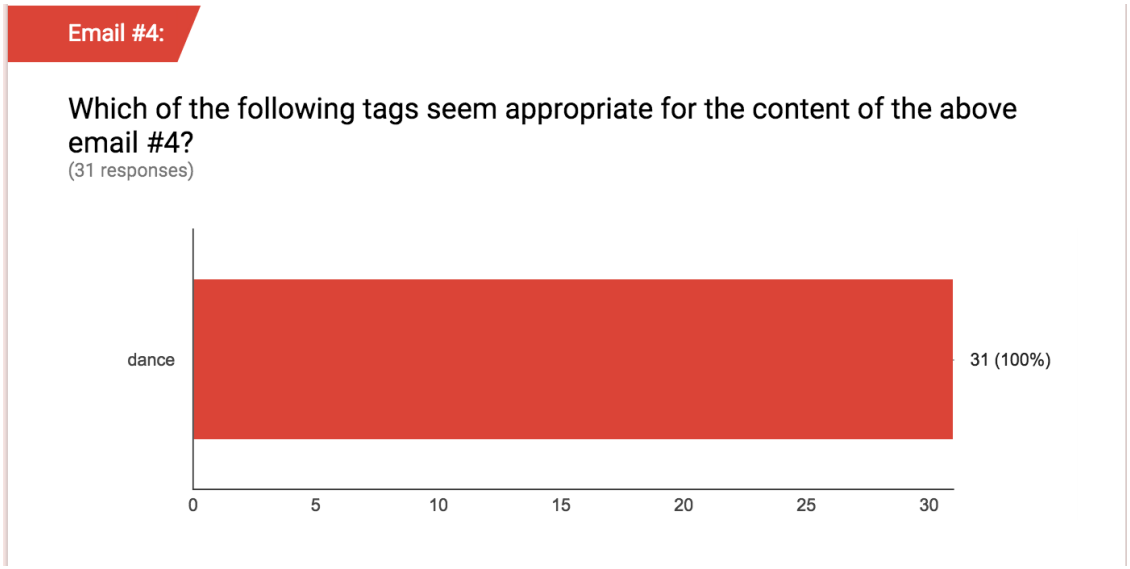


Figure 11: Email 4 survey responses

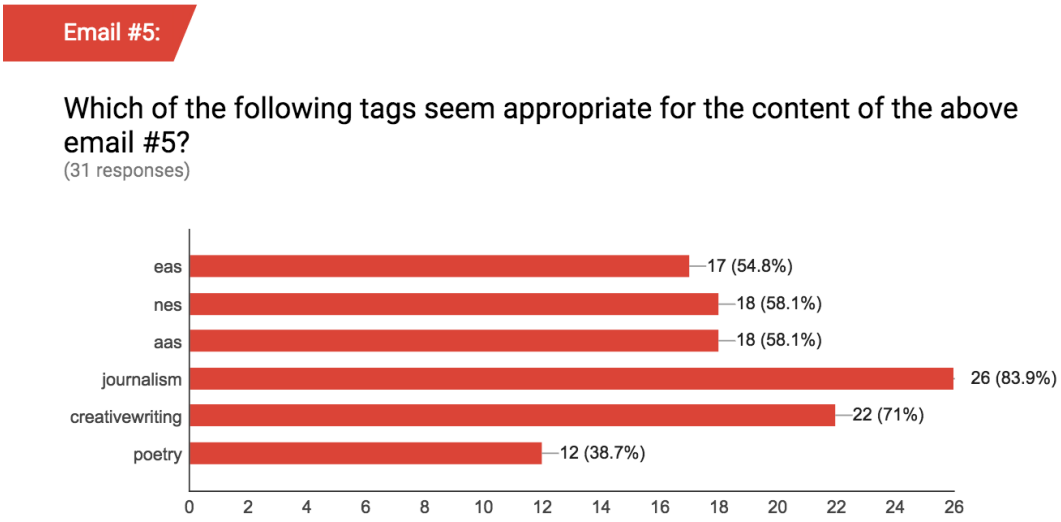


Figure 12: Email 5 survey responses

9.6 Appendix F

Freemium Model

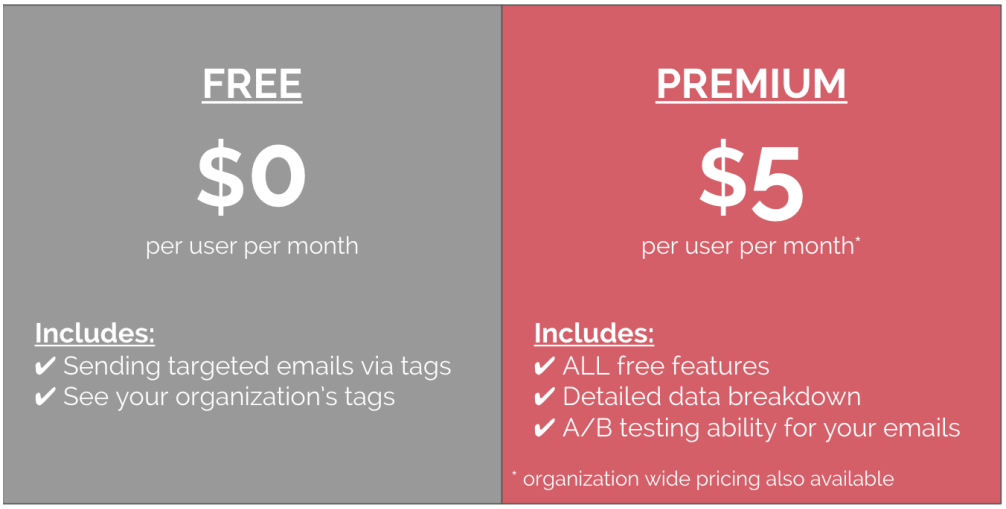


Figure 13: Freemium pricing for individuals (or clubs) within organizations. Paid version provides valuable, actionable analytics in a dashboard.

9.7 Appendix G

Marketing & Strategy

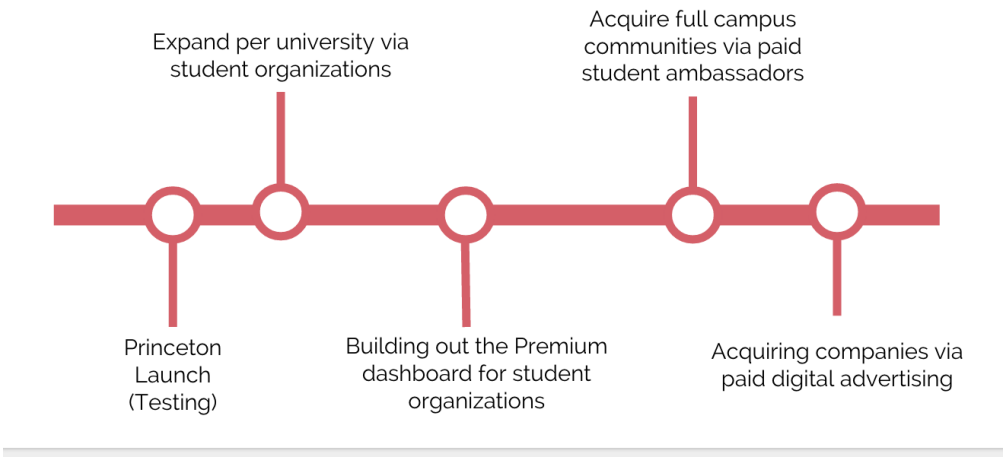


Figure 14: Pigeon's go-to-market strategy: sequential vertical acquisition of university campuses, followed by monetization and the companies user base.

9.8 Appendix H

Final presentation can be found here: https://goo.gl/6O0sEq.