

# Hidden in Plain Sight

## Classifying Emails Using Embedded Image Contents

Navneet Potti\*  
University of Wisconsin - Madison  
nav@cs.wisc.edu

James B. Wendt  
Google  
jwendt@google.com

Qi Zhao  
Google  
zhaqi@google.com

Sandeep Tata  
Google  
tata@google.com

Marc Najork  
Google  
najork@google.com

### ABSTRACT

A vast majority of the emails received by people today are machine-generated by businesses communicating with consumers. While some emails originate as a result of a transaction (e.g., hotel or restaurant reservation confirmations, online purchase receipts, shipping notifications, etc.), a large fraction are commercial emails promoting an offer (a special sale, free shipping, available for a limited time, etc.). The sheer number of these promotional emails makes it difficult for users to read all these emails and decide which ones are actually interesting and actionable. In this paper, we tackle the problem of extracting information from commercial emails promoting an offer to the user. This information enables an email platform to build several new experiences that can unlock the value in these emails without the user having to navigate and read all of them. For instance, we can highlight offers that are expiring soon, or display a notification when there's an unexpired offer from a merchant if your phone recognizes that you are at that merchant's store.

A key challenge in extracting information from such commercial emails is that they are often image-rich and contain very little text. Training a machine learning (ML) model on a rendered image-rich email and applying it to each incoming email can be prohibitively expensive. In this paper, we describe a cost-effective approach for extracting signals from both the text and image content of commercial emails in the context of Gmail, an email platform that serves over a billion users around the world. The key insight is to leverage the template structure of emails, and use off-the-shelf OCR techniques to obtain the text from images to augment the existing text features *offline*. Compared to a text-only approach, we show that we are able to identify 9.12% more email templates corresponding to ~5% more emails being identified as offers. Interestingly, our analysis shows that this 5% improvement in coverage is across the board, irrespective of whether the emails were sent by large merchants or small local merchants, allowing us to deliver an improved experience for everyone.

\*Work done while at Google.

This paper is published under the Creative Commons Attribution-NonCommercial-NoDerivs 4.0 International (CC BY-NC-ND 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.

WWW 2018, April 23–27, 2018, Lyon, France

© 2018 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC BY-NC-ND 4.0 License.

ACM ISBN 978-1-4503-5639-8/18/04.

<https://doi.org/10.1145/3178876.3186167>

### CCS CONCEPTS

• **Information systems** → **Email**; *Wrappers (data mining)*; • **Applied computing** → **Optical character recognition**;

### KEYWORDS

Information extraction; wrapper induction; email

#### ACM Reference Format:

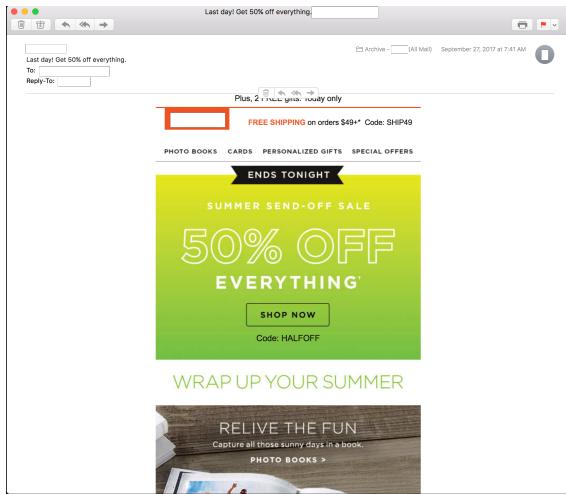
Navneet Potti, James B. Wendt, Qi Zhao, Sandeep Tata, and Marc Najork. 2018. Hidden in Plain Sight: Classifying Emails Using Embedded Image Contents. In *WWW 2018: The 2018 Web Conference, April 23–27, 2018, Lyon, France*. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3178876.3186167>

## 1 INTRODUCTION

Nearly half the world's population uses email today, sending and receiving over 269 billion emails a day [18]. The vast majority (about 90%) of these are machine-generated emails originating from businesses, as email has become the dominant medium for notifications (e.g., purchase receipts, social network updates, financial statements) as well as promotions (e.g., marketing campaigns, newsletters, abandoned shopping cart emails).

Consequently, users are spending a growing fraction of their day managing emails [9], and complaints about *email overload* abound. Researchers have risen to the challenge of improving the efficiency of email management in a number of ways, ranging from improving email search [5] and classification [16], to predicting user actions [10] and auto-suggesting responses [19]. There have also been efforts towards extracting useful information from emails [36] as well as surfacing it through intelligent personal assistants.

A large fraction of consumer emails is made up of promotional offers from businesses. The sheer volume imposes a burden, and users may not be able to read all of these emails. If we are able to extract the relevant details from promotional email, our hypothesis is that we can unlock significant value for our users. For instance, if we know that you recently received a coupon offering a 20% discount from a store and the personal assistant on your phone detects that you are visiting that store, it can notify you of the coupon present in your email so you can take advantage of it even if you hadn't read it and remembered to use it at the store. Simpler experiences such as notifying you when you have a big discount or free shipping available from a store that you often shop at can also unlock more value from promotional emails for our users. Enabling such experiences requires us to first identify that a given commercial email is promoting an offer. Once we are able to identify



**Figure 1: Screenshot of promotional email where the key information is in the image, and is readily visible to a human, but a text-only analysis will not easily yield that the email is about a special sale.**

such emails, we may extract other relevant fields like coupon codes, expiration dates, and offer details.

While information extraction from document corpora such as web pages is a well-studied problem [8, 28], the case of emails continues to be challenging because of privacy, volume, and latency considerations. These issues are rendered considerably worse in the case of promotional emails due to their typically image-heavy designs. Indeed, for many such offer emails, most of the details we would like to extract are only present in the image itself. The screenshot in Figure 1 shows such an email received by one of the authors of the paper containing a large image, but very little text in the email. The information in images is clearly visible to the intended human recipient, but current information extraction techniques based only on text are blind to writing in images. While sophisticated large-scale email campaigns make use of Microdata markup [32] and alt-text attributes [31] in HTML surrounding the image, many small businesses do not. Furthermore, the markup often contains far less detail than the actual image itself. We believe that unlocking the value in promotional emails irrespective of the sender’s sophistication is important to providing a good user experience.

A key constraint that informs our solution is that it needs to be cost-effective at the scale of many billions of emails processed daily. Consequently, a straightforward approach that simply runs the images in each email through an expensive ML model to determine if this is an offer would not be practical for a planet-scale free email service like Gmail.

Our solution leverages template induction techniques [3, 4] that have previously been described in the literature to radically bring down the cost of OCR analysis and extraction from images. We identify images that are commonly included in templates in an offline process, and extract text from them using an off-the-shelf Optical Character Recognition (OCR) engine. We then use the extracted text to augment existing text features used in the text-only ML models. Compared to the naive approach, our OCR analysis as

well as inference using our ML models happens in an offline job that is used to classify a template as an offer. At runtime, an email that matches a known set of templates can cheaply be classified as an offer. While this approach requires us to wait until we discover templates over emails before building extractors, it also produces a significant improvement in precision and recall.

This paper makes the following contributions:

- To our knowledge, this is the first description of the problem of identifying commercial offers in emails by leveraging image content.
- We outline several design choices made to solve this problem in a cost-effective manner: a) we leverage the template-structure of emails to identify image-rich emails, b) we use off-the-shelf OCR engines to extract text rather than train models directly on images or rendered emails, and c) we classify the templates offline so that online cost is limited to simply matching an email to a template rather than running inference on an expensive ML model.
- We present the results of evaluating our approach on emails received by users of Gmail and show that our approach yields 9.12% more templates being identified as offers at the same precision level. These templates could not have been identified as offers using simpler approaches that only leveraged text.

## 2 EMAIL CLUSTERING AND CLASSIFICATION

In this section, we describe the existing framework in which we classify and extract class-specific text fields from emails. In the following section, we describe how we extend this pipeline to incorporate image content.

The existing pipeline consists of three phases, as illustrated in Figure 2. First, we cluster emails together such that the resulting clusters contain documents that are likely instantiated from the same original template. Second, we learn classifiers for various semantically meaningful *verticals* such as purchase receipts, hotel reservations, and offers. Finally, we aggregate the results of email-level classification to learn a template-level classifier. At serving time, as shown in Figure 3, we simply match an email to the template to determine its category.

### 2.1 Template Induction

The vast majority of emails are machine-generated by populating a generic template with user-specific details. For instance, an online retailer may generate shipping confirmation emails by filling out a generic template with the particular products purchased by the user as well as shipping date and address. By associating such emails with the templates they belong to, we can improve the scalability and accuracy of the subsequent classification and extraction phases.

Conventional approaches to document labeling train multiple binary classifiers and apply the learned models to incoming documents online. This would be computationally challenging, given the aforementioned estimated global transmission of 269 billion emails per day. Templates, on the other hand, remove the need for online feature extraction and model inference altogether, by

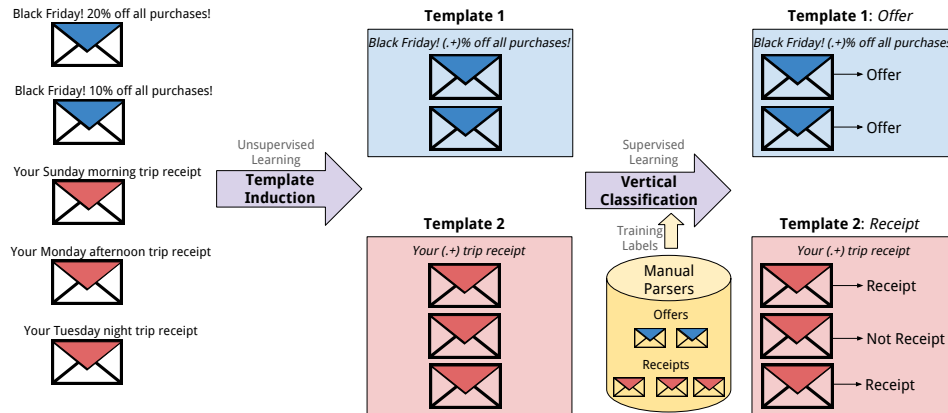


Figure 2: Training pipeline for email vertical classification.

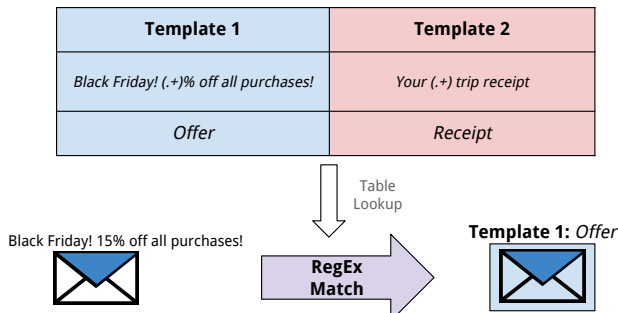


Figure 3: Serving pipeline for email vertical classification.

replacing these functions with a simple lookup for an email’s corresponding template and annotations (e.g. class, extraction rules, etc.). Templates represent thousands to millions of emails, and the cost of inference is amortized across all emails of a given template.

Templates serve to improve the accuracy of email annotations through aggregation. For example, in [34], multiple high-precision low-recall classifiers were applied to all emails in each template cluster. If a majority of emails are labeled as one class, then all emails in the cluster can be labeled as such, correcting occasional errors made by the email-level classifier by leveraging the block-structure imposed by the templates.

Existing techniques for template induction use a variety of similarity measures (either explicitly or implicitly) and range from sender-based clustering followed by subject regular expression deduction [3] to body clustering techniques that compare the underlying HTML structure of the documents [4].

In this work, we employ the techniques of the sender-subject clustering technique mentioned above. Emails are first partitioned by sender. Within each sender cluster, we identify a small set of subject regular expressions by replacing infrequently occurring terms in the subjects with wildcards.

Online, a matching template is found for an email by querying a table keyed by sender email addresses containing lists of subject regular expressions. The sender along with the longest matching subject regular expression that matches the incoming email is the

key to the matching template. For example, the shipping confirmation emails from an online retailer may have subjects matching the regular expression *Your order of .+ has shipped!*, with the wildcard replaced by the specific product(s) purchased. All emails from this sender with a subject that match this regular expression are therefore assigned to this template.

Candidate clusters become templates if they contain emails belonging to at least  $k$  different users, where  $k$  is defined by policy. This approach follows the same objective as the  $k$ -anonymization process presented by Di Castro et al. [11]. This requirement has the dual benefits of increasing the quality of recovered templates as well as meeting our commitment to user privacy.

## 2.2 Vertical Classification

**2.2.1 Training Data.** Our system currently supports a diverse set of verticals, including travel, hotels, bills, receipts, offers and events. For a small number of high-volume sender domains, we manually develop and maintain a set of heuristics in a rule-based classification and information extraction system. While the actual extractions are not the focus of this paper, the manually assigned labels are used as the ground truth labels to train the classifiers for each of the verticals.

While these manual rules (usually comprised of regular expressions and XPath queries) are highly precise, they incur significant development and maintenance overhead. Furthermore, they are brittle since changes to the original template require manual updates to the rules. Our goal is to use these high-precision, low-recall labels to train a high-recall classification system without sacrificing much precision. In other words, given an annotated corpus of selectively labeled [21] emails from high-volume senders, we would like to generalize our classification capability to the “tail” sender domains that have too low a volume to justify manual effort.

We uniformly sample a fixed number of training samples (say, a few hundred) from each template along with their manual labels. In contrast to sampling from all manually labeled emails, this approach has the advantage that the low-volume sender domains are not underrepresented in the training data. We create a test set similarly, ensuring that a template which was chosen for the training set

is excluded to ensure there’s no template-level overlap between the training and test sets. This allows us to verify that the model isn’t simply memorizing common features in the template and can actually generalize to new templates.

**2.2.2 Template-level Classification.** Since an email may belong to multiple verticals (e.g., an air ticket and hotel reservation may be included in the same email), we train a binary probabilistic classifier for each vertical. We choose this instead of a multilabel classifier for practical reasons such as allowing parallel development on different verticals by different engineers. For each template cluster, we use the binary classifiers to obtain the classification probability (scores) for sampled emails from the template. We then aggregate the email-level scores to obtain a template level vertical classification. Once a template is labeled as belonging to a certain vertical, all emails associated with it are also considered to belong to the same vertical (regardless of their individual classification scores). This two-level classifier allows us to build a template-level classifier using both email-level as well as template-level features, without simultaneously using features from all the emails in the template cluster.

Each binary vertical classifier is implemented in TensorFlow [1] as a feed-forward deep neural network using a standard cross-entropy loss function and up to three hidden layers of at most a few hundred units. The exact network architecture and hyperparameter configuration is optimized separately for each vertical using Vizier [14].

**2.2.3 Classifier Features.** The text content of each email is converted into a number of features before being fed into the neural network classifier. We briefly mention a few of these features here. Most features take the form of bags of words, where each word is hashed to obtain an integer value fed to the network along with a weight corresponding to its frequency. We remove stop words and normalize case to reduce feature sparsity. All word features are translated using embeddings, which can be initialized using standard techniques, such as word2vec [24]. We use multiple bag of words features: for words occurring anywhere in the email body text; for words occurring in the subject; for words occurring in a bold or large font; for words occurring in a footer; and for words occurring *near* a date or time in the email. In addition, we also use the counts of various HTML tags (including images, links and scripts) as features. Note that we avoid overfitting to the ground truth labels (which are generally available only for high-volume senders) by excluding features directly related to the sender.

We have also found it helpful to incorporate some template-level features in the classifier. For instance, we add an *open-ratio* feature corresponding to the fraction of all emails clustered to form this template that were opened by users.

**2.2.4 Evaluation.** During model training and hyperparameter tuning, we use the Area under the Curve (AUC) of the Receiver Operating Characteristic (ROC) as our evaluation metric. However, since our objective is to generalize beyond the ground truth labels, we manually assess the accuracy of each trained classifier to obtain its true *precision* and *coverage*.

## 3 IMAGE DATA PIPELINE

The pipeline described in Section 2 is primarily designed to work with text documents. But we have found that a significant fraction of emails, particularly those that contain offers, are image-heavy. Such emails often have all of their useful content in images, making the fields we would like to extract invisible to the existing pipeline. In this section, we describe how we modified our system to add the capability to categorize image-rich emails.

### 3.1 Design Considerations

We begin by laying out a number of desiderata that constrained the system design. We also briefly describe here how we address those considerations, but defer details to the remainder of this section.

- (1) *Respect user privacy.* We extend the  $k$ -anonymity approach to image extraction. An image is considered a “fixed” feature of a template, if and only if its URL is observed by  $k$  unique users within the template cluster, where  $k$  is the anonymity threshold previously discussed. This ensures that we filter out private or user-identifiable images, such as photos.
- (2) *Minimize network load.* Most images in promotional emails are included in the form of URLs that are fetched during HTML rendering. Fetching these images for ahead-of-time analysis as part of the email categorization pipeline can introduce a large network load since images are bandwidth-heavy. Not only does this bandwidth cost add to our expenses as an email service provider, but it also increases the load (and bandwidth cost) experienced by the senders’ servers. However, since constraint (1) restricts our attention to images (identified by URLs) that are sent to a large number of users, we designed our system to only fetch those images just once.
- (3) *Minimize training cost.* Introducing multimedia objects into the email categorization pipeline has the potential to greatly increase the cost of training the machine learning model. For instance, consider the prohibitively expensive (but technically, quite interesting) approach of rendering each email (including images and text, capturing it as a new image and feeding the latter into a convolutional neural network. Instead, we chose to separately convert each image into a set of training features that are then aggregated into email-level features used in the existing pipeline.
- (4) *Minimize cost and latency of serving emails.* Given the enormous volume of emails processed by our system, it is imperative that we set the objective of minimizing per-email serving cost. This goal is complementary to our users’ expectation of real-time access to their emails. In keeping with the scalable design of the existing pipeline, we ensure that there is no machine learning model applied directly per-email. Instead, we match each email with the template to which it belongs and use its cached template-level categorization. This objective also rules out the option of fetching and processing images on a per-email basis, in favor of the template-based offline approach we propose here.
- (5) *Minimize changes to the existing pipeline.* Since our existing pipeline has been in production for years and has been shown to be both robust and scalable, we would like to minimize changes to it. In particular, this meant that we should learn

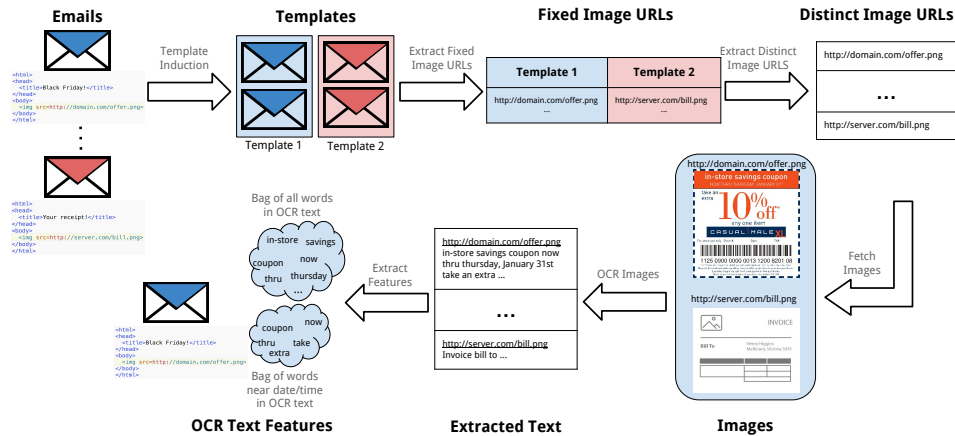


Figure 4: Image pipeline for extracting image OCR features from emails.

from the existing training data sources (which are based on hand-written parsers), rather than building and maintaining a new source such as a manually labeled corpus of image-heavy emails (which would require expensive periodic refreshes to capture new trends). In Section 3.4, we describe how we used an optical character recognition (OCR) engine to convert images into text features that could be incorporated into our text-based email categorization pipeline, and how we used the existing training corpus of text-based hand-written parsers to learn a model containing OCR text features.

### 3.2 Overview

Figure 4 summarizes our image data pipeline. We first leverage our template induction to cluster structurally similar emails together. During this clustering process, we also extract URLs of images that are *fixed*, i.e., common across all unique recipients of the template. Next, we aggregate the distinct image URLs seen across all templates and fetch these images. Finally, we use an OCR engine to extract the text in these images and create features that can be inserted into the classification pipeline of Section 2.

### 3.3 Image URL Extraction and Fetch

After clustering emails into templates, as described in Section 2, we extract the set of unique fixed image URLs that are shared across all emails in the cluster. This is done by building and traversing the document object model (DOM) tree [22] of each email body, and storing the image URL of each `<img>` tag observed, along with fingerprint of the recipient’s ID. After traversing all emails in the cluster, we use the fingerprinted IDs to filter out all image URLs that do not pass the  $k$ -anonymity thresholds.

In addition to preserving privacy, extracting these fixed image URLs also has the advantage of reducing the processing load downstream in the pipeline. Since many templates reuse the same images—a common feature among templates from the same domain—we aggregate over all extracted image URLs to construct a list of distinct image URLs in order to process each only once.

Next, we download all the images using Google’s internal web crawling infrastructure. This framework maximally parallelizes fetches from multiple servers, while adaptively rate-limiting requests to servers that are overloaded. It also handles various failure/error scenarios and automatically retries fetches over a window of several hours. Crucially, it also respects the robots exclusion protocol [26] ensuring that we only download images from servers that permit us to do so. In fact, we find that we consistently discard about a third of the image URLs due to robots exclusion.

We note here that it is possible to be more selective in the decision of which images to fetch. A large fraction of these fixed images are icons (particularly social media icons), buttons (including navigation menus) and unpersonalized tracking pixels. Since these images are typically small in size (both bytes and dimensions) and rarely contain information relevant to our information extraction task, we exclude these images based on their size, either as specified in the HTML markup or in the response to an HTTP HEAD request.

### 3.4 Feature Extraction from Images

**3.4.1 OCR Engine.** We use an optical character recognition (OCR) engine to extract text from the downloaded images. The choice of OCR engine required a careful evaluation of many options considering many different factors, including accuracy, computational expense and language support. While a detailed discussion of this evaluation is beyond the scope of the current work, we briefly remark upon it here.

It is important to note that most images in B2C email campaigns are born-digital, i.e., they are composed of text digitally overlaid on photographs or backgrounds. Consequently, the text detection task here is considerably different from extraction from scanned images of books or Street View images [33]. In fact, an engine that produces high-accuracy results for the latter, “natural” image corpora may not necessarily perform well on our more “synthetic” image corpus. As a further consideration, given the large volume of images on which we want to perform OCR, it is important that the engine makes efficient use of computational resources for each image. Finally, we also wanted to support classification of emails

in multiple languages, including some with non-Latin scripts (such as Japanese, Arabic, or Cyrillic).

We evaluated many OCR engines underlying the Google Cloud Vision [15] service offering, including Tesseract [29], PhotoOCR [6] and Aksara [12]. Based on a small-scale manual evaluation, we found that both Aksara and PhotoOCR had comparable and acceptably low error-rates compared to Tesseract. We ultimately chose to use Aksara based on its broader language support.

An avenue that we have identified for future exploration is training a custom OCR model for our image corpus, using transfer learning from an existing one. This latter option may allow us to explicitly choose a point in the tradeoff space most suited to our needs.

**3.4.2 OCR Text Features.** We convert the OCR text into features in much the same way as the email body text. In particular, we create two bags of words, one containing all the words in the OCR text (across all images in an email) and another containing those words that occur near a date. (The latter feature is based on the intuition that the occurrence of certain phrases, such as “valid through” or “expires on”, directly preceding a date is a strong indicator that the email contains an offer, and that the date in question is the expiration date.) These bags of words are converted to features using the same feature hashing as the email text features.

**3.4.3 Classifier Training.** The OCR text features finally give our classifiers the desired visibility into the content of the images. However, we are now faced with a subtle issue for classifier training: since our ground truth labels originally come from extraction rules that are text-based, they are systematically biased against the image-heavy emails that the new features are designed for. In other words, the training instances where the OCR text features are most helpful in classification and extraction are rarely ever labeled at all in our ground truth labels. This issue manifests itself in smaller effective weights assigned to the OCR text features, since the gradient signal used to update these weights, if present at all, is correlated with the email text features.

We rectify this shortcoming by merging the OCR text features into the corresponding features coming from the email body text (bag of all words in the email body and bag of words near dates, respectively). This technique is predicated on the implicit assumption that the occurrence of a certain word is an equally useful indicator of the class label (both vertical or field), whether that occurrence be in the email body or in an image. For instance, the occurrence of “valid through” followed by a date is an equally strong indicator of an email being an offer, whether that phrase occurred in the email body or in an image.

## 4 EVALUATION

The experiments in this section are designed to support the principal claim that leveraging image content of emails using an off-the-shelf OCR engine and augmenting existing bag-of-words features with the extracted text is an effective way to improve the offer classifier. We show that this approach helps us discover 9.12% more templates than a text-only approach at 90% precision (a threshold required by many applications that consume the predictions of the classifier). We also present somewhat surprising results that

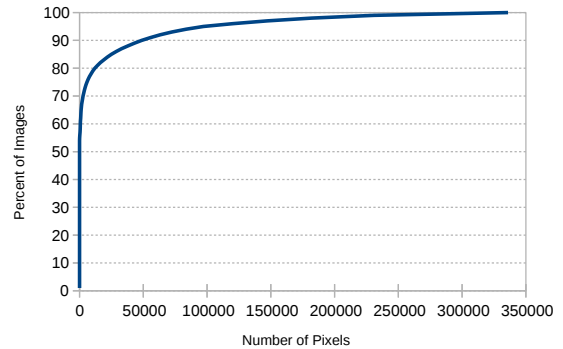


Figure 5: Cumulative distribution of image sizes in email.

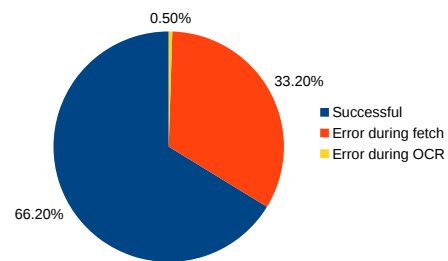


Figure 6: Fraction of images successfully OCRed.

the model trained with OCR features performs slightly better even on templates that are not image-rich. Finally we show that the additional offer templates that we discover cover the full range of templates from small senders to very large senders.

The section is organized as follows: Section 4.1 and 4.2 describe the image feature statistics and experimental setup. Section 4.3 and 4.4 describe how the models are trained and evaluated including the setup of training, validation, and test sets as well as the methodology for human evaluation. Finally, the experimental results are reported and discussed in Section 4.5.

### 4.1 Statistics About Images

Before presenting an evaluation of our system, we discuss some of the statistics we observed about email image data that informed our design choices.

Figure 5 shows the cumulative distribution of the images in a sample of emails by pixel count. Unsurprisingly, a large fraction of the images are small visual elements used to organize the content in the email. Less than 25% of the images are larger than 10K pixels. We avoid crawling and performing OCR on images that are too small to contain recognizable text, thereby saving network and computational resources.

The “alt-text” attribute [31] is empty for 47% of images and contains just a single word for 40%. This indicates that, at least in HTML email, alt-text is often an inadequate description of the image content.

Figure 6 shows the fraction of image URLs (above a minimum size) from which we are able to extract non-empty OCR text. A non-trivial portion is lost to crawling errors, predominantly due to robots exclusions. The experimental results that follow represent a

		FooterText		TextAroundDateTime		
		Y	N	Y	N	
OCR	Y	5.17%	NA	Y	0.11%	0.02%
Text	N	94.83%	0.00%	N	72.13%	27.74%

**Table 1: Co-occurrence of OCR text features and the corresponding email text features. Columns are features from email text and rows are the corresponding features from OCR text. Refer to text for description.**

Training and Validation				Testing
Templates	Positives	Negatives	Ratio	Templates
1.8M	1.4M	14M	90%/10%	950K

**Table 2: Statistics of training, validation and testing dataset. The ratio refers to the ratio between training data size and validation data size.**

lower bound on the amount of signal we are able to extract from image content. As we tackle the error rates in the image acquisition pipeline, we expect the results to improve.

As mentioned in Section 3.4, we merge the OCR text into the existing text features extracted from the email body. The merging can potentially reinforce or enrich the existing features. The co-occurrence between OCR text features and existing text features is captured by the matrix shown in Table 1. Two OCR text features are generated in correspondence to the existing text features, *FooterText* and *TextAroundDateTime*. The first OCR-derived feature is a bag of words of all text extracted from email images. The second OCR-derived feature is a bag of words of OCR text surrounding datetime annotations detected in the images. ‘Y’ and ‘N’ represent whether the feature is present in the training example. It can be seen that for the *FooterText* feature, the corresponding OCR text feature is missing the majority of the time; while for *TextAroundDateTime* feature, the corresponding OCR feature is rarely present. Even still, as we shall see later, the limited extra OCR information boosts the model performance significantly.

## 4.2 Experimental Setup

We split the corpus of templates into 70% *training templates* and 30% *testing templates*. Since the offer classifier is trained at email level, we obtain the emails from the training templates and divide them into *training* and *validation* subsets with the ratio of 90%/10%. The statistics of the resulting datasets are reported in Table 2. The values for positives and negatives refer to number of emails—note that in order to reduce the impact of class imbalance, we downsampled the negatives significantly. Training and validation subsets are used for hyperparameter tuning and final model training after hyperparameters are selected. We use the testing templates to estimate the performance of the model in a real world application scenario.

## 4.3 Model Training

Two models are trained: a *control model* and a *treatment model*. The control model serves as a baseline which does not utilize OCR information, while the treatment model integrates OCR information.

Parameter	Value	Parameter	Value
Batch Size	50	Dropout	0.14
Embedding Dimension	50	Learning Rate	0.043
Layers	2	Layer Size	20

**Table 3: Hyperparameters used for training control and treatment models in the experiment.**

	AUC	1 – AUC
Control	0.9994	0.0006
Treatment	0.9995	0.0005

**Table 4: Model performance on validation dataset. Both models have comparable AUC values, but their differences on 1 – AUC values are more evident.**

Both models use the same features, however, the treatment model merges the OCR text features into existing text features. We adopted a Deep Neural Network (DNN) architecture for both models. The optimal hyperparameters for the control model are determined using Vizier [14], and the same hyperparameters are used for the treatment model. The optimal hyperparameters are shown in Table 3. It is worth mentioning that embeddings are shared across all text features in order to reduce model complexity.

## 4.4 Evaluation Methodology

We first measure the performance of the control and treatment models using the AUC of ROC metric. Results in Table 4 show that the treatment model is superior to the control model. To make the contrast more evident, examine the 1 – AUC values, which represents the the probability for the model to rank a negative example above a positive example. The 1 – AUC of treatment model is roughly  $\frac{0.0006-0.0005}{0.0006} \approx 17\%$  smaller than control model.

Since the labels of our training data are derived from manual parsers, it is likely that they contain certain biases, for example, bias towards high-volume senders and image-poor emails. Thus, performance measured on the validation set might not be representative of how the models perform in the real world where vertical labels are missing for a lot of templates. In order to better measure the performance of these models beyond the limited scope of the manual parsers, we evaluate each model (control and treatment) over a sample of the templates in the testing set and compute the precision and coverage.

As described in Section 2.2.2, we score each template by the average of the binary classification scores of all emails within the template. Thus, the template-level score ranges between 0 and 1. Since our application requires high precision, we set the template-level score threshold to 0.9.<sup>1</sup> We call these candidate templates *score-0.9* templates. For each experimental model we gather their respective set of score-0.9 templates and compare the number of templates (coverage) and the average precision.

We estimate the precision through human assessment on samples drawn from each score-0.9 template set. A score-0.9 template

<sup>1</sup>Ideally, in order to get the maximum recall, we should find the lowest template-level threshold that meets the precision requirement, but this requires significant amount of human assessment effort. In this paper, we set the threshold to 0.9, which in our dataset is a reasonable starting point to obtain the desired precision.

	Templates	Precision	95% CI
Control	67,609	89.24%	[82.37, 96.11]
Treatment	73,772	90.00%	[83.39, 96.61]

**Table 5: Number of score-0.9 templates discovered by control and treatment models and the respective template-level precision. Precision of both models is statistically equivalent, while the treatment model increases template coverage by 9.12%. Note that OCR features are excluded from the dataset for control, and included for treatment.**

is labeled as an offer if from manual inspection, one is able to determine that there’s some promotional offer that might deliver savings for the user. For example, the following patterns are commonly indicative of an offer template:

- Explicit discount rate, e.g. “xx% off”
- Offer expiration date, e.g. “valid through *date*”
- Eligible for free shipping, e.g. “Free shipping for orders over \$x”
- Buy X get Y free

An assessor responds with ‘Yes’ or ‘No’ if there is no ambiguity when judging the template, otherwise ‘NA’, and the responses are excluded from the precision computation.

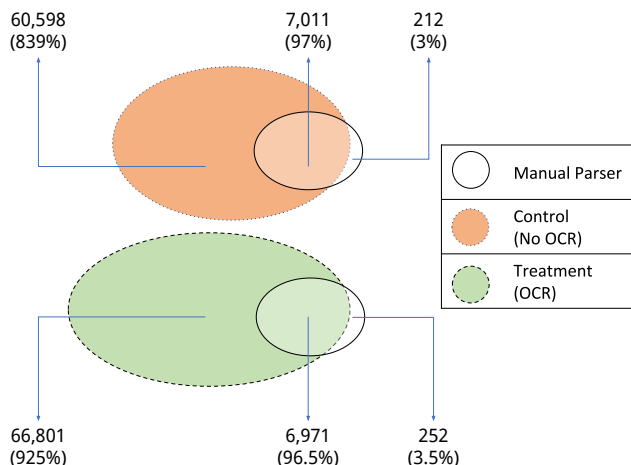
#### 4.5 Experimental Results

The number of score-0.9 templates and the respective template precision for both models are reported in Table 5. A two-tailed z-test shows that the two models have equal precision, while the treatment model discovers 9.12% more score-0.9 offer templates, which translates to about 5% additional email coverage. Note that control and treatment models are tested on the same set of templates, and the only difference is whether OCR text features are used in email-level model inference.

The immense number of emails and their associated privacy constraints make it extremely difficult to accurately measure the recall of the treatment and control models. However, we can obtain a biased proxy: the recall of our models on the high-volume sender domains covered by the hand-crafted parsers described in Section 2.2.1. A subset of our testing templates is covered by such parsers; we call these templates *Manual Parser Templates* (MPTs). An MPT is treated as an offer template if at least 90% of its emails are labeled as offers by a hand-crafted parser, and are herein referred to as score-0.9 MPTs.<sup>2</sup>

Figure 7 depicts the overlapping and non-overlapping templates classified as offers by the manual parsers, control model, and treatment model. Our first observation is that the use of machine learning greatly complements the manual parser based approach by extending coverage well beyond the capabilities of hand-crafted rules. Furthermore, template generation and classification is an automatic process, which can be executed periodically to capture changes in templates over time, while hand-crafted parsers are brittle to these changes and must be manually updated. This is

<sup>2</sup>While hand-crafted parsers are precise, they often have low recall. Thus, a template that corresponds to one of these hand-crafted parsers, but still has a low score from the offer classifier is usually a sign of a false negative from the classifier.



**Figure 7: Number of templates classified by the control and treatment models compared to the manual parsers. The manual parsers cover 7,223 offer templates. The control model labeled 67,609 templates as offers. The treatment model labeled 73,772 templates as offers.**

	OCR Feature	
	Present	Not Present
Control	18,424	49,185
Treatment	20,515	53,257
Increment	11.35%	8.25%

**Table 6: Number of score-0.9 templates discovered by each model split by whether they contain OCR text features.**

particularly important for the offer vertical, since sales and deals are often temporal in nature (e.g. holiday sales, flash sales, etc.).

Our second observation is that the inclusion of OCR text features in the treatment model further extends the coverage of the control model by 9% in terms of the number of templates. Note that this translates to a 5% increase in email coverage.

In Table 6 we explore how coverage is affected by the presence of the OCR feature in the template at inference time. As expected, templates that contain the OCR feature benefit the most. Templates that do not contain OCR features also benefit from the OCR feature, despite its absence. This improvement can be attributed to the merging of the OCR features with the existing text features, which strengthens the model at training time. At inference time, templates with only textual features are able to take advantage of the additional knowledge the treatment model has learned from the OCR text. Hence, the use of OCR features is beneficial for both merchants that present offer information in images as well as those that do not.

We further break down the discovered offer templates into quartiles by email count to investigate whether the newly introduced OCR features benefit larger or smaller merchants. Row 1 in Table 7 depicts the lower bound (in terms of number of emails) of each quartile.

The increase in coverage made by the control model over MPTs skews more towards smaller templates. This makes sense since



	$Q_1$	$Q_2$	$Q_3$	$Q_4$
Treatment quartile boundary (emails)	$k$	$4.8k$	$15k$	$65k$
MPTs	1,123	1,656	1,822	2,622
Control	14,889	17,336	17,136	18,248
Treatment	16,350	19,170	18,787	19,465
Control/MPTs	13.26×	10.47×	9.41×	6.96×
Treatment/MPTs	14.56×	11.58×	10.31×	7.42×
Treatment/Control	1.10×	1.11×	1.10×	1.07×

**Table 7: The score-0.9 templates discovered by the treatment model broken into quartiles according to the number of emails per template, represented by multiples of  $k$ , the anonymity threshold. Row 1 shows the lower bound of each quartile. We use these boundaries in subsequent rows for comparison. Rows 2-4 depict the per-quartile number of score-0.9 templates discovered by each model type with the corresponding number of sample emails within the interval defined by row 1. Rows 5-7 depict the template coverage increase factor between the various models.**

manual parsers are generally written for larger senders, so the distribution will skew towards larger templates, while templates induced through automatic clustering will discover templates of all sizes, provided they pass the  $k$ -anonymity thresholds.

Recall that the total increase in coverage of the treatment model over the control model was 9.12%. In Table 7 we observe that this improvement holds across the board, irrespective of whether the templates are small or large. Hence, the use of OCR information in offer classification is beneficial for both small and large merchants.

## 5 RELATED WORK

The ongoing explosive growth in email volumes is largely dominated by machine-generated emails such as confirmations, newsletters, notifications, and promotions. Such a dramatic shift in the character of their inbox contents is clearly a challenge for the typical recipient—one that our research community has sought to address in a number of innovative ways [23, 25]. For instance, the traditional problem of spam classification [7] has now been broadened to that of identifying *important* emails [2] and semantically categorizing emails [16]. Similarly, webmail providers have broadened email threading from a syntactic approach (based on subject and RFC headers) appropriate for personal conversations, to a semantic approach (based on identification of sequences of causally-related emails) suited to machine-generated emails [3]. This idea of “causal threading” has even been extended to predict future emails in a thread [13, 35].

Machine-generated emails are usually generated by filling out generic templates (describing the layout and boiler-plate) with variable details specific to the context of the email. Reversing this process, i.e., decomposing an email into its constituent template and specific details, opens a plethora of avenues towards better managing such emails. Emails instantiated from the same template are highly likely to be semantically similar, and can therefore be treated similarly for the purpose of various different tasks. Indeed, the recognition of this fact can improve both the scalability and

accuracy of machine learning techniques for tasks such as spam classification, email categorization and causal threading mentioned above. A lot of recent work has therefore been aimed at addressing this *template induction* problem [4, 27, 34]. Section 2.1 describes how we use some of these techniques in our pipeline.

Related to the problem of generalizing the *common* template from a set of emails is the dual problem of extracting the *specific* details used to instantiate variable fields in the template. These details include, for instance, the product name, delivery date and tracking ID for a shipping confirmation, as well as the coupon code and offer expiration date from a promotional email. Fields extracted from emails in this way can be used for targeted advertising [17], or to create reminders, calendar events or notifications for the user. While this problem has been well-studied under the moniker of *wrapper induction* [20] in information extraction literature (particularly for HTML documents on the web), there is relatively little published work on the subject in the context of emails. Zhang et al. [36] studied the problem of extracting product information from machine-generated B2C emails.

The research problems highlighted here are often rendered much more challenging by virtue of the fact that emails are private documents and users accordingly have stringent expectations of privacy. In this work, we used a  $k$ -anonymity approach similar to that described in [11] to meet these privacy expectations. However, a number of practical challenges remain unsolved in developing industrial-scale machine learning models over private data (see [30] for some examples in the context of non-email, private documents).

What we have highlighted above is only a sample of the rich body of research literature on various aspects of email management at web-scale as well as information extraction from web documents.

## 6 CONCLUSION

This paper tackles the problem of detecting commercial offers in emails (promoting discounts, free shipping, special sales, etc.) so we can enable new experiences such as alerting a user to a discount in her email when she visits a relevant store. We argue that image content is key to understanding such emails. We described a technique for accomplishing this *without* having to run an expensive ML model over rendered emails online. The key insight in this paper is to leverage the template structure of emails, and use the OCR’d text of images to augment the existing text features *offline*. The resulting model associates templates with offer emails, and can be used inexpensively online to label incoming emails. Experimental results show that we are able to discover 9.12% more templates than the text-only model at the same precision level (~90%).

To the best of our knowledge, ours is the first study of email classification for image-rich emails. In this work, we went beyond simply using the location or count of images in document structure to analyzing the textual content of the images and incorporating it into the classification pipeline. We plan to extend this work beyond classification to extraction of key fields like expiration dates, coupon codes, and offer text (e.g. “Free Shipping”) from the image content in the email when we are unable to obtain them from the text in the email.

## REFERENCES

- [1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A System for Large-scale Machine Learning. In *Proc. of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI)*. 265–283.
- [2] Douglas Aberdeen, Ondrej Pacovsky, and Andrew Slater. 2010. The Learning Behind Gmail Priority Inbox. In *LCCC: NIPS 2010 Workshop on Learning on Cores, Clusters and Clouds*.
- [3] Nir Ailon, Zohar S Karnin, Edo Liberty, and Yoelle Maarek. 2013. Threading Machine Generated Email. In *Proc. of the 6th ACM International Conference on Web Search and Data Mining (WSDM)*. 405–414.
- [4] Noa Avigdor-Elgrabli, Mark Cwalinski, Dotan Di Castro, Iftah Gamzu, Irena Grabovitch-Zuyev, Liane Lewin-Eytan, and Yoelle Maarek. 2016. Structural Clustering of Machine-Generated Mail. In *Proc. of the 25th ACM International Conference on Information and Knowledge Management (CIKM)*. 217–226.
- [5] Michael Bendersky, Xuanhui Wang, Donald Metzler, and Marc Najork. 2017. Learning from User Interactions in Personal Search via Attribute Parameterization. In *Proc. of the 10th ACM International Conference on Web Search and Data Mining (WSDM)*. 791–799.
- [6] Alessandro Bissacco, Mark Cummins, Yuval Netzer, and Hartmut Neven. 2013. PhotoOCR: Reading Text in Uncontrolled Conditions. In *Proc. of the 2013 IEEE International Conference on Computer Vision (ICCV)*. 785–792.
- [7] Enrico Blanzieri and Anton Bryl. 2008. A Survey of Learning-based Techniques of Email Spam Filtering. *Artificial Intelligence Review* 29, 1 (2008), 63–92.
- [8] Chia-Hui Chang, Mohammed Kayed, Moheb R Girgis, and Khaled F Shaalan. 2006. A Survey of Web Information Extraction Systems. *IEEE Transactions on Knowledge and Data Engineering* 18, 10 (2006), 1411–1428.
- [9] Michael Chui, James Manyika, Jacques Bughin, Richard Dobbs, Charles Roxburgh, Hugo Sarrazin, Geoffrey Sands, and Magdalena Westergren. 2012. *The Social Economy: Unlocking Value and Productivity through Social Technologies*. McKinsey Global Institute.
- [10] Dotan Di Castro, Zohar Karnin, Liane Lewin-Eytan, and Yoelle Maarek. 2016. You've Got Mail, and Here is What You Could Do With It!: Analyzing and Predicting Actions on Email Messages. In *Proc. of the 9th ACM International Conference on Web Search and Data Mining (WSDM)*. 307–316.
- [11] Dotan Di Castro, Liane Lewin-Eytan, Yoelle Maarek, Ran Wolff, and Eyal Zohar. 2016. Enforcing K-anonymity in Web Mail Auditing. In *Proc. of the 9th ACM International Conference on Web Search and Data Mining (WSDM)*. 327–336.
- [12] Y. Fujii, D. Genzel, A. C. Popat, and R. Teunen. 2015. Label Transition and Selection Pruning and Automatic Decoding Parameter Optimization for Time-synchronous Viterbi Decoding. In *Proc. of the 13th International Conference on Document Analysis and Recognition (ICDAR)*. 756–760.
- [13] Iftah Gamzu, Zohar Karnin, Yoelle Maarek, and David Wajc. 2015. You Will Get Mail! Predicting the Arrival of Future Email. In *Proc. of the 24th International Conference on World Wide Web (WWW)*. 1327–1332.
- [14] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D. Sculley. 2017. Google Vizier: A Service for Black-Box Optimization. In *Proc. of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 1487–1495.
- [15] Google. 2015. Cloud Vision API. <https://cloud.google.com/vision/>. (2015).
- [16] Mihajlo Grbovic, Guy Halawi, Zohar Karnin, and Yoelle Maarek. 2014. How Many Folders Do You Really Need?: Classifying Email into a Handful of Categories. In *Proc. of the 23rd ACM International Conference on Conference on Information and Knowledge Management (KDD)*. 869–878.
- [17] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikrit Savla, Varun Bhagwan, and Doug Sharp. 2015. E-commerce in Your Inbox: Product Recommendations at Scale. In *Proc. of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 1809–1818.
- [18] The Radicati Group. 2017. Email Statistics Report, 2017-2021. <http://www.radicati.com/?p=8801>. (2017).
- [19] Anjali Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, Laszlo Lukacs, Marina Ganea, Peter Young, and Vivek Ramavajjala. 2016. Smart Reply: Automated Response Suggestion for Email. In *Proc. of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 955–964.
- [20] Nicholas Kushmerick, Daniel S. Weld, and Robert Doorenbos. 1997. Wrapper Induction for Information Extraction. In *Proc. of the 15th International Joint Conference on Artificial Intelligence (IJCAI)*. 729–737.
- [21] Himabindu Lakkaraju, Jon Kleinberg, Jure Leskovec, Jens Ludwig, and Sendhil Mullainathan. 2017. The Selective Labels Problem: Evaluating Algorithmic Predictions in the Presence of Unobservables. In *Proc. of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 275–284.
- [22] Philippe Le Hégaré, Ray Whitmer, and Lauren Wood. 2005. Document Object Model (DOM). <http://www.w3.org/DOM>. (2005).
- [23] Yoelle Maarek. 2017. Web Mail is not Dead!: It's Just Not Human Anymore. In *Proc. of the 26th International Conference on World Wide Web (WWW)*. 5–5.
- [24] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proc. of the 1st International Conference on Learning Representations: Workshop Track (ICLR)*. arXiv:1301.3781
- [25] Marc Najork. 2016. Using Machine Learning to Improve the Email Experience. In *Proc. of the 25th ACM International Conference on Information and Knowledge Management (CIKM)*. 891–891.
- [26] Christopher Olston and Marc Najork. 2010. Web Crawling. *Foundations and Trends in Information Retrieval* 4, 3 (2010), 175–246.
- [27] Julia Proskurnia, Marc-Allen Cartright, Lluís Garcia-Pueyo, Ivo Krka, James B Wendt, Tobias Kaufmann, and Balint Miklos. 2017. Template Induction over Unstructured Email Corpora. In *Proc. of the 26th International Conference on World Wide Web (WWW)*. 1521–1530.
- [28] Sunita Sarawagi. 2008. Information Extraction. *Foundations and Trends in Databases* 1, 3 (2008), 261–377.
- [29] Ray Smith. 2007. An Overview of the Tesseract OCR Engine. In *Proc. of the 9th International Conference on Document Analysis and Recognition (ICDAR)*. 629–633.
- [30] Sandeep Tata, Alexandrin Popescu, Marc Najork, Mike Colagrosso, Julian Gibbons, Alan Green, Alexandre Mah, Michael Smith, Divanshu Garg, Cayden Meyer, et al. 2017. Quick Access: Building a Smart Experience for Google Drive. In *Proc. of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 1643–1651.
- [31] W3C. 1999. Objects, Images, and Applets. <https://www.w3.org/TR/html4/struct/objects.html>. (1999).
- [32] W3C. 2009. HTML Microdata. <https://dev.w3.org/html5/md-LC/>. (2009).
- [33] Kai Wang, Boris Babenko, and Serge Belongie. 2011. End-to-End Scene Text Recognition. In *Proc. of the 2011 International Conference on Computer Vision (ICCV)*. 1457–1464.
- [34] James B. Wendt, Michael Bendersky, Lluís Garcia-Pueyo, Vanja Josifovski, Balint Miklos, Ivo Krka, Amitabh Saikia, Jie Yang, Marc-Allen Cartright, and Sujith Ravi. 2016. Hierarchical Label Propagation and Discovery for Machine Generated Email. In *Proc. of the 9th ACM International Conference on Web Search and Data Mining (WSDM)*. 317–326.
- [35] Aston Zhang, Lluís Garcia-Pueyo, James B Wendt, Marc Najork, and Andrei Broder. 2017. Email Category Prediction. In *Proc. of the 26th International Conference on World Wide Web Companion (WWW)*. 495–503.
- [36] Weinan Zhang, Amr Ahmed, Jie Yang, Vanja Josifovski, and Alex J. Smola. 2015. Annotating Needles in the Haystack Without Looking: Product Information Extraction from Emails. In *Proc. of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 2257–2266.