RiSER: Learning Better Representations for Richly Structured Emails

Furkan Kocayusufoglu*
University of California, Santa Barbara
furkan@cs.ucsb.edu

Ying Sheng
Google
yingsheng@google.com

Nguyen Vo
Google
nguyenvo@google.com

James B. Wendt
Google
jwendt@google.com

Qi Zhao
Google
zhaqi@google.com

Sandeep Tata
Google
tata@google.com

Marc Najork
Google
tata@google.com

ABSTRACT

Recent studies show that an overwhelming majority of emails are machine-generated and sent by businesses to consumers. Many large email services are interested in extracting structured data from such emails to enable intelligent assistants. This allows experiences like being able to answer questions such as "What is the address of my hotel in New York?" or "When does my flight leave?". A high-quality email classifier is a critical piece in such a system. In this paper, we argue that the rich formatting used in business-to-consumer emails contains valuable information that can be used to learn better representations. Most existing methods focus only on textual content and ignore the rich HTML structure of emails. We introduce *RiSER* (Richly Structured Email Representation) – an approach for incorporating both the structure and content of emails. RiSER projects the email into a vector representation by jointly encoding the HTML structure and the words in the email. We then use this representation to train a classifier. To our knowledge, this is the first description of a neural technique for combining formatting information along with the content to learn improved representations for richly formatted emails. Experimenting with a large corpus of emails received by users of Gmail, we show that RiSER outperforms strong attention-based LSTM baselines. We expect that these benefits will extend to other corpora with richly formatted documents. We also demonstrate with examples where leveraging HTML structure leads to better predictions.

CCS CONCEPTS

• **Information systems → Email**; • **Computing methodologies → Artificial intelligence**;

KEYWORDS

Email Representation; HTML Structure Encoding; Email Classification

*Work done while interning at Google.

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313720

1 INTRODUCTION

Classifying documents into two or more target classes is a core problem in many web applications, and has received attention from several research communities. Identifying the sentiment in text documents [43], detecting spam in email [12] and the web [36], and fake news detection [42] are well-known applications. Even enterprise applications like legal discovery [40] make use of the core abstraction of document classification.

Most existing approaches focus on short plain-text documents like tweets, reviews, and posts in online discussion forums. However, several interesting applications are on corpora where the documents are longer, and often have rich HTML markup structure. Such corpora have not received as much research attention, but are important for many applications. For instance, web pages have rich HTML structure; classifying them [39] is useful for several applications like focused crawling, faceted search, and even as a signal for web page search ranking.

Email is another area that contains longer documents with rich HTML markup structure. Recent studies [31] have shown that most email is richly formatted and machine-generated rather than plain-text sent by humans. Several interesting applications beyond spam-detection rely on learning good classifiers over email. This includes foldering [19], automatic prioritization [15], and even information extraction [2, 14, 41]. In particular, these studies show that learning a high-precision classifier over richly formatted emails is a key ingredient to extracting structured data from email that is then used to power several intelligent experiences [14, 41].

In this paper, we tackle the problem of learning a good representation (embedding) for structured documents like richly formatted emails. We focus on evaluating these embeddings for the task of classifying an email into one of k target classes. Recent literature has shown that for relatively short documents, a recurrent neural network with an attention mechanism [48] is a very strong baseline for learning good representations. With richly formatted documents, most existing techniques ignore the formatting information except

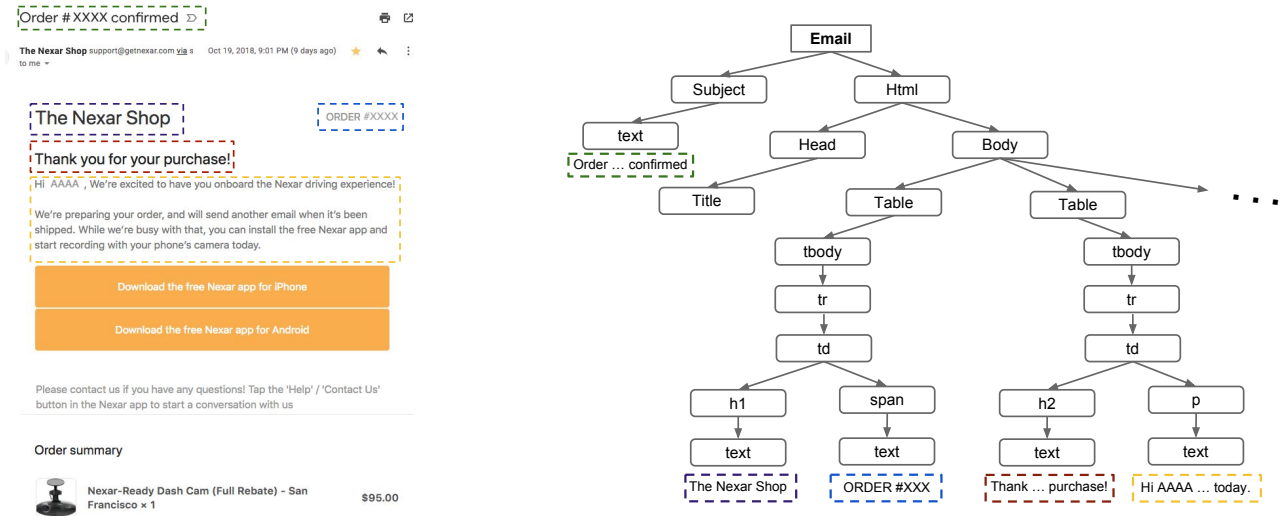

Figure 1: An example *purchase* email (left) and a simplified DOM tree (right) representing a portion of the email. For clarity, we trim long trivial HTML sequences such as nested *div* tags. Note that we extend the DOM tree beyond the email message body to include a branch representing the subject.

for assuming that documents are hierarchically organized [48] into sentences and paragraphs. We illustrate in Section 2 that the rich structure of HTML-formatted email can be a valuable signal for the relative importance of various pieces of information in the document. We introduce a neural architecture called RiSER (Richly Structured Email Representation) to take advantage of the structure of the email in addition to the content. RiSER projects the input email into a vector representation by jointly encoding the HTML structure of the email along with the terms in the email. It consists of a 2-level LSTM to first construct a structural encoding and combines this with an encoding of the terms for which there is a second LSTM layer. This provides a simple way to encode the formatting and layout information as opposed to manually engineering structural and visual features.

Experiments on real data from Gmail corpus show that the structure does indeed contain useful information and that RiSER is able to exploit this information to outperform a strong baseline that only considers textual information. While we focus on emails, the techniques introduced in RiSER are generic and can be used for any document corpus containing richly formatted information such as HTML web pages and PDF documents.

We make the following key contributions in this paper:

- We identify that structural information in richly formatted HTML emails is a valuable signal that is ignored by existing classification approaches focused on textual content.
- We propose a 2-level neural architecture called RiSER that jointly learns to encode both the structure and the content of the email by examining the sequence of HTML tags that each text term is associated with.
- We demonstrate through experiments on two classification tasks on data from Gmail that RiSER learns enhanced email representations.

The rest of this paper is organized as follows: Section 2 provides more detailed background on the information available in the structure and markup, explains how we represent the input document, and lays out the problem definition for applications like information extraction. Section 3 introduces our two-level neural architecture that jointly encodes the HTML markup and the textual content of documents to learn a good representation. Section 4 reports on the performance of this architecture on two real-world classification tasks using data from Gmail. We compare the performance of this architecture with several baselines including ones that try to leverage manual feature-engineering to represent aspects of this information. The experimental results show that the models with the proposed architecture significantly increase recall at high precision compared to previous baselines by up to 8.6%. Section 5 summarizes areas of related work from the data mining, NLP, and ML research communities. Finally, Section 6 concludes the paper with a summary and potential directions for additional research.

2 EMAIL STRUCTURE

We focus on business-to-consumer (B2C) emails, the vast majority of which are machine-generated instantiations of predefined templates, accounting for up to 90% of all email traffic on the internet [30].

Figure 1 (*left*) depicts an example email from the author's inbox that contains a purchase confirmation. Unlike personal emails sent by humans, B2C emails like these often have rich HTML structure. This structure introduces a complex hierarchy which can be represented by a Document Object Model (DOM) tree [27], such as the one illustrated in Figure 1 (*right*). Each node in this tree represents an HTML tag in the message body, and its children consist of the HTML tags nested within it. The text of the message resides at the leaf nodes. Note that we have extended this DOM tree beyond the

standard model to also include the subject of the email as a child node of the email root. A typical commercial email like the example in Figure 1 may contain around 2,000 leaf nodes. In fact, we have observed some messages, albeit few, that have as many as 10,000 leaf nodes and a depth of up to 200.

The use of rich HTML structure draws the attention of the human eye to different parts of the email. We find that the categorically similar emails often share similar structure. For example, bill reminder emails will format a "pay now" link close to the middle of the email, while hotel confirmation emails often contain tables of check-in and check-out information. This manifests as tables, headers, footers, highlighted and emboldened text in the markup. The key hypothesis we test in this paper is whether the markup in such richly formatted emails offers additional signal over just the textual content.

The remainder of this section describes how we convert this tree representation into a sequence of features that are suitable as input for the classification models described in the following section. The sequence consists of three types of features: *textual features*, *annotation features*, and *structural features*.

The textual features consist of a sequence of the first 200 terms (including punctuation) extracted from the DOM tree (including the subject branch) via in-order traversal. Limiting ourselves to 200 terms allows us to fully represent most emails without the risk of allowing very long emails to derail model training.

The annotation features indicate whether a particular text span contains an annotation. Several examples such as *Date*, *Price*, and *Time* are listed in Table 1. A term may correspond to one or more annotations. These annotations are extracted through a variety of methods including dictionary lookups and regular expression matching. Libraries of these annotators were developed over several years to support more traditional rule-based information extraction tasks. An entity detection library, for example, is also available as a Google Cloud API [18]. Readers can refer to Sheng et al. [41] for more detailed discussion on these annotators. In this work, we provide a way to incorporate the signals from these annotations.

Annotation Types	
Address	Location
Alphabet-Number	Price
Confirmation Number	Telephone Number
Date	Time
Establishment	Tracking Number

Table 1: Annotation features.

Finally, we represent the structure of the document using XPaths [7]. An XPath is the sequence of nodes (HTML tags) extending from the root of the DOM tree to a leaf node. For example, the XPath of the term "ORDER" in the example email in Figure 1 is */html/body/table/tbody/tr/td/span/text*. All the terms in a paragraph (in a *p* node), for instance, might share the same sequence of HTML tags for their XPath feature.

We considered several alternative approaches to represent document structure. For example, one can identify the XPath patterns for text in key HTML tags like headers, tables, lists and then use these to mark each term. This approach is labor-intensive and more importantly ad-hoc and may not take into account patterns than an engineer has not visually inspected and considered important. For our problem, we are interested in an approach that can be applied to all the emails from various domains with different structural patterns without having to manually engineer structural features. Another possible representation is to use a visual blocks structure using external parsers [11]. Visual blocks are tree structures based on the visual layout of the page. Such an approach would depend on an external parser to produce features that can then be represented as part of the input example. We chose to use the XPath since it is simple, and provides a detailed representation of the markup. This allows us to directly test the hypothesis of whether the information in the markup is valuable for learning a better representation. We expect that using visual features from a sub-system that understands document layout could be interesting future work.

A complete set of features representing a single email document thus consists of a stream of up to 200 terms, the set of annotations that each term is a part of, and the XPath that each term resides at. Figure 2 denotes the features corresponding to three terms – "Shop", "ORDER" and "#XXXX" from the email in Figure 1. The terms "Shop" and "ORDER" do not contain any annotation, but "#XXXX" has two annotations: *Confirmation Number* and *Alphabet-Number*. The bottom row contains the XPath in the DOM tree at which each of these terms appear.

We describe the architecture that combines these three features in Section 3 below, and explore the utility of different combinations of these features in Section 4.

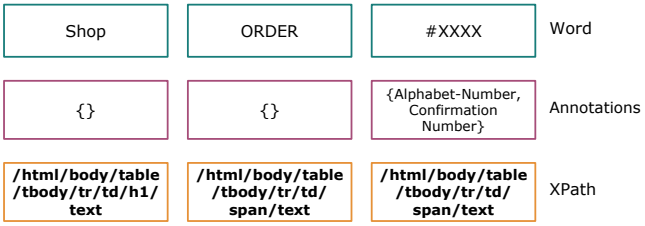

Figure 2: Feature sets of three terms "Shop", "ORDER" and "#XXXX" from the top of the email in Figure 1.

3 MODEL ARCHITECTURE

In this section, we present our email representation framework – RiSER. The proposed model consists of two main components: an XPath encoder and a word encoder. At a high level, the XPath encoder models an email's DOM structure by encoding the tags along the XPath to a leaf node using an LSTM layer [21] while the word encoder combines word embeddings with the corresponding XPath encodings and additional features to learn an enhanced representation of the email. The enhanced email representation is then used as a feature for email classification. Figure 3 shows an overview of the model architecture. We describe the details of the framework and its components in the following sections.

3.1 XPath Encoder

Recall from Section 2 that an XPath is a sequence of HTML tags from root to leaf of an email's DOM tree. The proposed XPath encoder

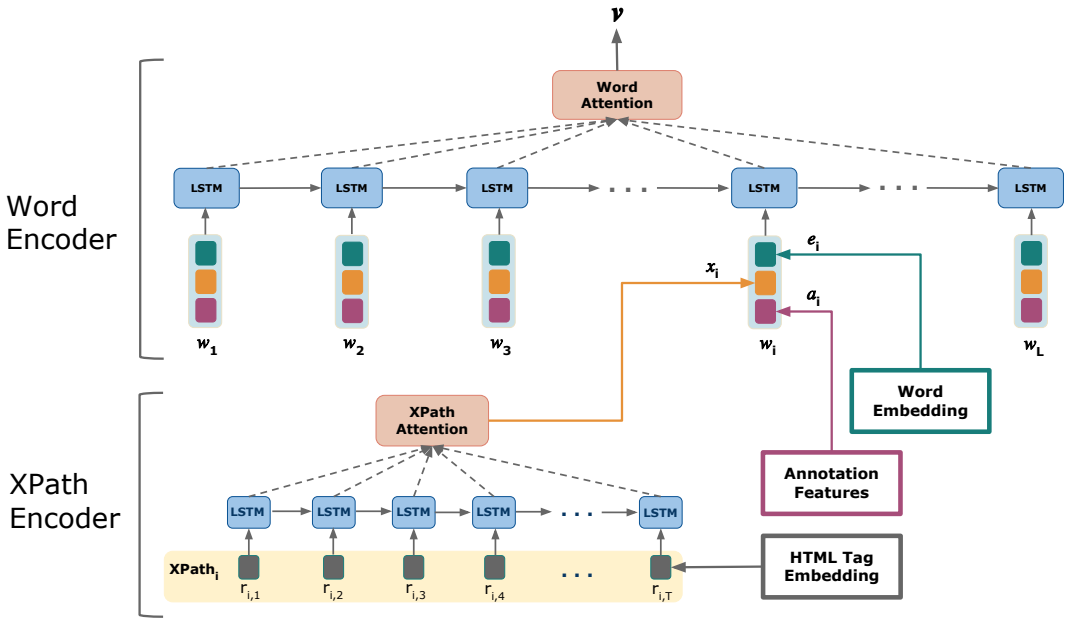

Figure 3: Overview of the RiSER architecture. The XPath Encoder iteratively encodes the tags in the XPath for each term in the email's DOM tree. Later, each XPath encoding (x_i) is combined with the corresponding word embedding (e_i) and annotation features (a_i) of the term to form a word representation (w_i). The Word Encoder then processes these word representations ($\{w_1, w_2, \cdots, w_L\}$) to learn an enhanced email representation (v). *(best viewed in color)*

in Figure 3 summarizes this sequence with a vector of fixed length. In this work, we use an LSTM encoder with an additional attention mechanism [48] to represent the sequential elements (HTML tags), however, one is free to choose any standard sequence encoder to model the sequential input. More precisely, let the $XPath_i$ be a sequence of HTML tags with embeddings $[r_{i1}, \cdots, r_{iT}]$. In practice, we randomly initialize these embeddings and update them during training. We pass these embeddings through an LSTM layer to produce output vectors:

$$h_{it} = \overrightarrow{LSTM}(r_{it}), t \in [1, T] \tag{1}$$

Typical practices for extracting a final output representation from an LSTM layer include selecting the final output or averaging across all outputs. However, we observe that XPath sequences can be very long and highly repetitive. Furthermore, not all HTML tags may contribute equally to the email structure. For instance, HTML tags such as `strong`, `em` or `big` can be more informative than other much more common tags, such as `div`. Hence, we apply an attention mechanism [48] over the outputs of the LSTM layer in order to (1) extract such informative tags from long sequences and reward their contributions to the XPath representation, and (2) simultaneously reduce the impact of highly repetitive tags.

The final XPath encoding vector is computed as follows. First, we feed the LSTM output h_{it} through a fully connected layer with trainable weight matrix W_r and bias vector b_r, as well as a *tanh* activation function to produce the hidden vector u_{it}:

$$u_{it} = \tanh(W_r h_{it} + b_r) \tag{2}$$

Next we calculate an importance weight α_{it} for each HTML tag r_{it} in the sequence as a normalized similarity score between the hidden vector u_{it} and a trainable structure vector u_r:

$$\alpha_{it} = \frac{\exp(u_{it}^T u_r)}{\sum_i \exp(u_{it}^T u_r)} \tag{3}$$

Finally, we compute the *XPath encoding* vector x_i as a weighted sum of all the LSTM outputs:

$$x_i = \sum_t \alpha_{it} h_{it} \tag{4}$$

The proposed XPath encoder has the following two desirable properties. First, as we show in Section 4.5, the XPath encoding vectors capture information that is highly valuable for an email classifier. Second, the XPath encoder captures the relative hierarchy of an email's structure, i.e. two XPath sequences will have similar embeddings in the vector space if they correspond to closer leaf nodes in the DOM tree since they share long common subsequences. In the next section, we explain how the XPath encoder interacts with the word encoder.

3.2 Word Encoder

The word encoder is responsible for learning a rich representation of the email which not only summarizes the email's content but also encodes its HTML structure. In order to accomplish this goal, we go

beyond the word-embeddings-based representations and capture additional features with structural information from the email. We create our rich word representation vector w_i by leveraging the following three components:

- *Word embeddings.* We construct our vocabulary with the most frequent 20,000 words in our data. Each word in the vocabulary is mapped to a word embedding vector through an embedding look-up matrix. Out-of-vocabulary (OOV) words are mapped to an *unknown* vector. In this work, we randomly initialize the word embeddings and jointly train them with the model. One can also use pre-trained word embeddings such as Word2Vec [35], Glove [38], or fastText [9].
- *XPath encodings.* The corresponding XPath of each word is passed through the XPath encoder to produce an XPath encoding. These XPath encodings implicitly capture the position of the word in the DOM tree. Since XPath sequences can be very long, we only keep the first T tags of the sequence and omit the rest. Note that T is a hyper-parameter in our framework.
- *Annotation vectors.* We represent the annotations a word corresponds to as a 10-D binary vector. Each dimension (0/1) indicates whether a word is annotated by a particular markup shown in Table 1. These features turn out to be very helpful, as we will show in Section 4.4.

To form improved word representations, the word embeddings, XPath encodings, and annotation vectors are concatenated for each term and fed into a fully connected layer with *tanh* non-linearity. Formally, given an email with L words, we represent each word in the sequence as:

$$w_i = \tanh(W_s[e_i, x_i, a_i]), i \in [1, L] \tag{5}$$

where $e_i \in \mathbb{R}^k$ is the word embedding vector, $x_i \in \mathbb{R}^l$ is the corresponding XPath encoding calculated in Equation (4), and a_i is the 10-dimensional binary vector representing the annotations mentioned in Section 2. $W_s \in \mathbb{R}^{m*m}$ is a trainable weight matrix, where $m = k + l + 10$.

With $[w_1, w_2, \cdots, w_L]$ we can now compute an email representation vector by using another LSTM layer to encode the sequence and apply an additional attention mechanism [48] over that layer:

$$h_i = \overrightarrow{LSTM}(w_i), i \in [1, L] \tag{6}$$

$$u_i = \tanh(W_w h_i + b_w) \tag{7}$$

$$\alpha_i = \frac{\exp(u_i^T u_w)}{\sum_i \exp(u_i^T u_w)} \tag{8}$$

$$v = \sum_i \alpha_i h_i \tag{9}$$

where v is the email representation vector that summarizes both the email structure and semantics.

3.3 Email Classification

We use the email representation computed above as a feature in email classification. More specifically, we use a linear layer to convert the email representation vector to a real-valued vector of size $|C|$, where C is the set of classes to predict. A final softmax layer is applied to obtain the normalized probabilities over the labels:

$$p = \text{softmax}(W_c v + b_c) \tag{10}$$

The training objective of our framework is cross-entropy loss:

$$loss = -\sum_{d \in D} \sum_{c \in C} p_c^g(d) \cdot log(p_c(d)) \tag{11}$$

where D is the set of training emails, C is the collection of email classes, $p_c(d)$ is the probability of predicting email d as class c, and $p_c^g(d)$ is a binary value indicating whether c is the correct label. The entire model is trained through back-propagation with respect to all parameters. Training details are further explained in Section 4.3.

4 EXPERIMENTS

This section presents the results from several experiments on data from a large Gmail corpus to illustrate the effectiveness of the RiSER architecture. We focus on two binary classification tasks for this purpose. The first task is one of detecting if an email contains a bill. This model is used by the email service to determine if it wants to proactively remind the user when a bill is due. The second task is one of detecting if an email contains a hotel reservation confirmation. This model is also used to support intelligent applications such as travel planning. We refer to these two as the *Bill* and *Hotel* tasks. We pick these two tasks for illustrative purposes. The overall system that these classifiers are a part of includes several other classifiers (and extractors) for flights, calendar appointments, shipping confirmations, etc.

The experiments are designed to compare the performance of different RiSER variants (introduced in Section 4.2) against a strong baseline and to understand the incremental advantage of incorporating the annotations and XPath embeddings. Before describing the experimental setting, we first describe how the datasets are constructed including the source for the labels and sampling techniques used to deal with biases.

4.1 Dataset

The training and testing datasets are constructed by sampling from the Gmail corpus. During the course of this work, user privacy was protected through strict data access controls and data preprocessing to avoid training on sensitive data. Nobody involved with this project had access to visually inspect any of the data. Only terms matching the dictionary of top 20,000 terms are used, while the rest are replaced by an *unknown* identifier. Any text spans that contain potentially private data (such as addresses, dates, and phone numbers) are denoted by the corresponding annotation feature, and the underlying terms are replaced by the *unknown* identifier.

4.1.1 Labels. Ground truth labels for this dataset are derived from three sources: Microdata [20], manually defined parsers, and rule-based extractors. Microdata is a standard that enables senders to explicitly label and mark up their outgoing emails with structured information. For example, when sending a confirmation email to a customer, hotels can include markup to indicate that their emails are reservation confirmations, and even specify details such as confirmation numbers, addresses, check-in and check-out out dates and times, etc. While Microdata markup is precise, it is not widely

adopted by all email senders, so the volume of annotations tends to be low.

Manually defined parsers are designed to extract category-specific fields from emails in lieu of Microdata. These are created on a per-sender basis and hand-crafted based on several instances of emails donated by users for this very purpose. For the purposes of email classification, we use the presence of a successful extraction from an email as a positive label for the category.

Note that we prioritize construction of parsers for high volume senders, since (a) parser construction is laborious, thus focusing on larger senders yields more labeled samples, and (b) the donated corpus is relatively small and thus contains a limited number of samples from low volume senders. Manually defined parsers tend to be highly precise but brittle, since small changes to the emails can often break the parsers, requiring human involvement and new donated emails to fix them.

Rule-based extractors consist of a manually engineered set of traditional information extraction techniques, such as dictionary lookups and regular expressions, that operate across senders. These are also constructed by leveraging the donated email corpus for both development and validation. These tend to have higher recall than the approaches above, but often at the cost of lower precision. Similar to the manually defined parsers, we use the presence of a successful extraction of an email as a positive label for the email.

4.1.2 Sampling. Despite drawing labeled examples from the three sources of ground truth mentioned above, the vast majority of emails remain unlabeled, and even among those that are labeled, the distribution is skewed towards higher volume senders. Thus, training classifiers using a uniformly random sample of data often leads to overfitting and poor generalization to smaller senders.

Domain-based sampling attenuates this issue by limiting the number of samples observed from high volume senders and boosting those from smaller senders, however this method can be too coarse for larger senders that have their own internal skew—e.g. the majority of amazon.com email might be classified as *purchase orders*, while a small fraction are *account memberships* or *bill reminders*.

We find a balance between these two methods by stratifying emails by the templates from which they are instantiated. This ensures that an even number of emails are represented from the *purchase order*, *account membership*, and *bill reminder* templates, along with emails from much smaller senders.

These templates are inferred through a one-time template generation process that clusters similarly structured emails into groups that are likely to have been instantiated from a single template. These techniques range from clustering on the sender and subject of an email [3] to clustering on the structure of the email body [2, 4]. In this work, we use structural clustering techniques based on a locality sensitive hash of the email body similar to the technique described in [41].

We split the data into train and test sets with a ratio of 9:1 while ensuring that samples belonging to the same sender domain appear in only one of these sets to prevent the effects of memorization. For the training data, we downsample the negative examples to yield a positive-to-negative ratio of about 1:100. We keep the original ratio in the test set. To put this into perspective, the resulting training sets have approximately *150M* total samples for the Bill task and *31M* samples for the Hotel task.

4.2 Experiment Configuration

We train and evaluate four variants of the RiSER architecture, each incorporating a different combination of the textual, annotation, and structural features described in Section 3.2.

- **RiSER-W** uses the Word Encoder only and represents words using the *word embeddings* component only. That is, in Equation 5, the word representation w_i is represented by the word embedding e_i only.
- **RiSER-WA** uses the Word Encoder and represents words with the *word embeddings* and *annotation features*.
- **RiSER-WX** uses both the Word Encoder and the Xpath Encoder and represents words with the *word embeddings* and *XPath encodings*.
- **RiSER-WXA** uses both the Word Encoder and the XPath Encoder and represents words with the *word embeddings*, *XPath encodings*, and *annotation features*. This variant is the one displayed in Figure 3.

The RiSER-W variant is simply an LSTM-based recurrent model with an attention mechanism, as it only includes word embeddings in its input layer. Recent studies in the literature have shown that attention-based LSTM models [29, 48] are excellent for modeling text documents and achieve state-of-the-art performance in document classification tasks by outperforming linear models, SVMs, and feed-forward neural networks. Therefore we consider RiSER-W as a baseline model to compare against the remaining RiSER variants.

Note that one can employ more complex neural architectures than just the RiSER-W architecture to improve email classification performance. However the focus of this work is to demonstrate the advantages of exploiting the rich formatting structure of emails through our proposed XPath Encoder and overall architecture. Moreover, we aim to demonstrate that textual content can be better utilized when combined with structural information for learning email representations.

We compare the performance of the RiSER-WA, RiSER-WX and RiSER-WXA variants against the baseline RiSER-W model to determine whether structural information and annotation features can improve the learned representations. We test this hypothesis by classifying emails into two semantic categories, *Bills* and *Hotels*. While we could have posed this as a multi-class classification problem, practical convenience of improving one model without affecting any other models motivated us to pose these as separate *binary classification* tasks. The general idea of leveraging structure to learn a better representation holds for multi-class, multi-label, or even regression tasks.

4.3 Hyper-parameters and Training

For all four variants, we perform grid search over the word embedding dimension ({50, 100, 200}), the word encoder LSTM output (hidden state) dimension ({64, 128, 256}), the initial learning rate ({0.01, 0.001, 0.0001}), the batch size ({50, 100, 200}), the word encoder dropout rate ({0, 0.25, 0.5}), and the optimizer type (AdaGrad [16] and Adam [24]). For the Adam optimizer, we use the default settings suggested by the authors ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$).

For the RiSER-WX and RiSER-WXA models we also search over the following additional hyper-parameters: the HTML tag embedding dimension ({25, 50, 100}), the XPath encoder LSTM output dimension ({16, 32, 64}), the XPath encoder dropout rate ({0, 0.25, 0.5}), and the maximum XPath length ({20, 30, 40}).

We use the norm clipping trick with a *threshold* of 5.0 in all four models to avoid the exploding gradient problem.

For each model, we use the Vizier [17] platform to select the best combination of hyper-parameters based on AUC-PR after a maximum of 1M training steps. The selected hyper-parameters for each model are shown in Table 2. We later continue training the selected models until they converge and report their highest numbers on the test data in the next section.

BILL				
Hyper-parameter	RiSER-W	RiSER-WA	RiSER-WX	RiSER-WXA
batch size	200	200	200	200
learning rate	0.001	0.0001	0.0001	0.001
optimizer type	Adam	Adam	Adam	Adam
word embedding dim	200	100	100	100
WE LSTM output dim	128	128	128	128
WE dropout rate	0	0.25	0.25	0
tag embedding dim	N/A	N/A	25	25
max XPath length	N/A	N/A	20	20
XE LSTM output dim	N/A	N/A	64	16
XE dropout rate	N/A	N/A	0.5	0.25

HOTEL				
Hyper-parameter	RiSER-W	RiSER-WA	RiSER-WX	RiSER-WXA
batch size	200	100	200	200
learning rate	0.001	0.0001	0.0001	0.001
optimizer type	Adam	Adam	Adam	Adam
word embedding dim	200	100	100	200
WE LSTM output dim	128	64	64	128
WE dropout rate	0.5	0.25	0.25	0.25
tag embedding dim	N/A	N/A	50	25
max XPath length	N/A	N/A	20	30
XE LSTM output dim	N/A	N/A	16	64
XE dropout rate	N/A	N/A	0.5	0.25

Table 2: Selected hyper-parameters for the RiSER variants for the Bill and Hotel classification tasks. WE stands for Word Encoder and XE stands for XPath Encoder.

4.4 Results and Discussion

Classifier performance is often measured using area under the receiver operating characteristic curve (AUC-ROC). However, when dealing in datasets with high class skew—which is often the case for email—the area under the precision-recall curve (AUC-PR) provides a much fairer representation of model performance. In practice, even the AUC-PR metric is inadequate given that applications for email generally require extremely high precision. For example, misclassifying an important email as spam is an incredibly bad user experience.

Thus, we evaluate our models based on their *recall at a fixed level of high precision*. By requiring that all models reach a prespecified precision threshold, we minimize bad user experiences. Fixing the precision then allows us to evaluate different models by the extent of their coverage at that level of precision. That being said, we also report the AUC-PR metric for clarity. The experimental results on both datasets are summarized in Table 3.

BILL				
Model	R@P=0.85	R@P=0.90	R@P=0.95	AUC-PR
RiSER-W	76.4	71.5	57.4	85.4
RiSER-WA	76.8	73.0	63.1	85.9
RiSER-WX	76.6	73.5	64.1	86.2
RiSER-WXA	78.0	73.7	66.0	85.6

HOTEL				
Model	R@P=0.85	R@P=0.90	R@P=0.95	AUC-PR
RiSER-W	95.9	95.0	92.4	97.1
RiSER-WA	97.3	95.7	93.2	98.0
RiSER-WX	96.3	95.5	93.5	97.2
RiSER-WXA	97.4	96.3	92.9	97.7

Table 3: Recall at a fixed level of precision and AUC-PR performance metrics for four RiSER variants trained for the Bill and Hotel classification tasks.

4.4.1 **Recall-at-fixed-precision (R@P).** Results show that our RiSER-WXA variant overall gives the best performance across both datasets. The improvement in performance depends on the data type. For more complex dataset, such as Bill, the RiSER-WXA variant outperforms the baseline model (RiSER-W) by 2.2% on R@P(0.9), and by 8.6% on R@P(0.95) metrics. Note that recall tends to decrease drastically for the baseline model as the precision threshold increases, while RiSER-WXA demonstrates more robust performance. This leads to a larger gap between both models' recall at higher precision thresholds. On the other hand, improvements in the Hotel task are smaller compared to the Bill task, leading to 1.3% on R@P(0.9), and by 0.5% on R@P(0.95) metrics. We also report the experimental results of RiSER-WA and RiSER-WX variants in order to better evaluate the effects of each component.

Effect of annotation features. Our experiments show that using crafted annotation features on top of word embeddings provides important performance gains over the baseline. With respect to the R@P(0.95) metric, the RiSER-WA variant outperforms the RiSER-W variant by 5.7% on the Bill task, and 0.8% on the Hotel task. Similarly, the gap between recall is smaller (1.5% and 0.7%) as we decrease the precision threshold.

We believe the reason behind the performance boost that comes with the annotation features is that they allow us to recover meanings in words that cannot be captured with the defined vocabulary. To be more specific, emails include dates, numbers, addresses such that most of these terms are anonymized due to privacy constraints and are embedded with the *unknown* vector. Thus, including the annotation features allows our framework to capture these additional signals, which intuitively improves the model's ability to correctly classify an email. As a toy example, an email without a date is most likely not a hotel reservation.

Effect of email structure. Combining XPath encodings with word embeddings leads to an improvement of 6.7% in the Bill task, and 1.1% in the Hotel task with respect to the R@P(0.95) metric. More importantly, by comparing the performance of RiSER-W and RiSER-WX, we verify that incorporating the *HTML structure* into the learning process leads to *improved email representations*. In

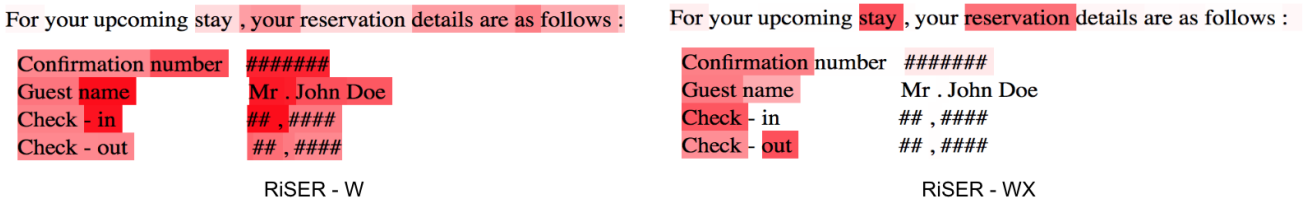

Figure 4: Word attention weights on an example donated Hotel email using the RiSER-W and RiSER-WX variants. Brighter red highlighting indicates a higher attention weight for the corresponding term relative to the other terms in the email.

Section 4.5, we illustrate this with some example emails in which the inclusion of HTML structure results in better classification.

While improvements of the order of 5% to 10% may seem small, they represent a significant reduction in "bad experiences" for users of Gmail. Consider, for example, a service that extracts structured data from these emails to remind people when their bills are due or answers questions about their hotel reservations. The improved models directly translate to improved recall for both these assistive experiences.

4.4.2 **AUC-PR.** The AUC-PR metric generally improves for RiSER-WA, RiSER-WX and RiSER-WXA variants compared to the baseline variant on both classification tasks. However, the improvements are not always in line with the improvements to the recall-at-fixed-precision metric. Surprisingly, unlike the R@P metric, the RiSER-WXA variant does not have the highest AUC-PR for either task. For the Bill task, the RiSER-WX variant performs the best (86.2%), while for the Hotel task, the RiSER-WA variant has the highest AUC-PR (98.0%).

4.5 Qualitative Analysis

In order to get an intuition into how RiSER utilizes the HTML structure of an email, we visualize and compare the attention weights of two architecture variants: one that excludes the XPath encodings and one that includes the XPath encodings. Here, we use the RiSER-W and RiSER-WX variants trained for the Hotel classification task. We illustrate the resulting attention weights of the two models applied to a single donated email example in Figure 4. Note that we obfuscate key information for privacy reasons.

In comparing the term attention weights of the RiSER-W and RiSER-WX models, we find that the latter is much better at attending to terms that are semantically relevant to the hotel reservation class. For example, terms such as `'stay'`, `'reservation'`, `'confirmation'`, and `'guest'` are attended to while the RiSER-W model attends to nearly all words. In fact, it attends more to those that are less semantically relevant in this example.

To get an intuition as to why structural information helps these models attend differently to these terms, we inspect the attention weights of the XPath tags in addition to the term attention weights. We visualize these weights for a donated email example in Figure 5. Red highlighting indicates term attention weights while yellow highlighting indicates the XPath's HTML tag attention weights.

Our first observation is that the use of structure helps the model understand the boundaries of different parts of the email. In this particular example the model attends to terms that belong to the subject more than the remainder of the text stream. Note that the word attention weights drop suddenly at the boundary of the subject and the beginning of the message body. We observed this pattern across many donated examples.

Our second observation is that the model also attends more to HTML tags that are often used to highlight important text. In this example, the model pays particular attention to meaningful tags such as `strong` (which indicates if a word is bold-ed), `h1`/`h2`/`h3` (which indicate headers), and `ul` and `li` (which indicate list items). These types of markup are generally used to draw the user's attention to specific text in a rendered email, often containing important information. Similarly, note that the `center` tag, which aligns all inner HTML to the center of the display, is also highly attended to.

In summary, from inspecting the attention weights of the RiSER architecture with and without XPath encodings, we observe that including structural information improves attention at the term

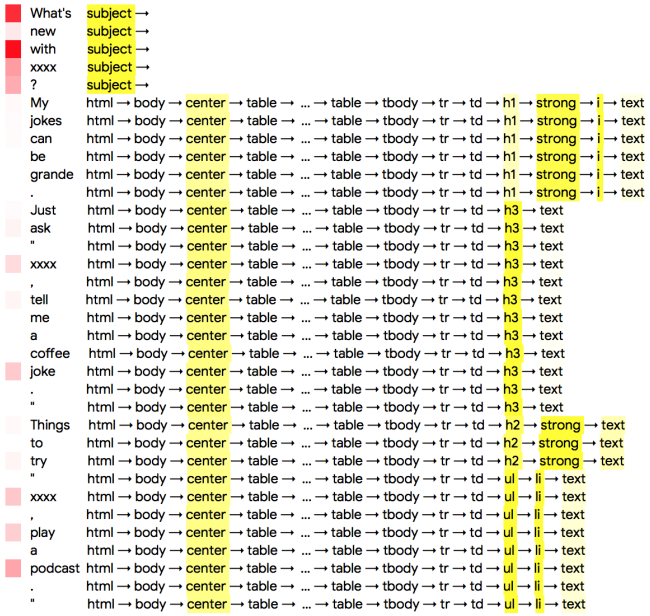

Figure 5: Term and XPath attention weights. Red highlighting indicates the term attention weights. Yellow highlighting indicates the XPath's HTML tag attention weights (row normalized). Brighter colors indicate higher attention weights.

level by helping the model learn about boundaries in documents (e.g. subject vs. message body), as well as learn to focus on parts that the original sender intended to draw the user's attention to (e.g. `strong`, `center`, etc.).

5 RELATED WORK

Our work builds on contributions from several research areas including natural language processing, data mining, web mining, and deep learning. Here we discuss related work in two main categories: text representation and email representation.

There is a rich history of work in text representation [33, 47] for various tasks like sentiment classification [37] (e.g. positive or negative reviews) and genre detection (e.g. sports, news, entertainment, etc.). More recently, the availability of large amounts of text data has led to unsupervised approaches for learning a good distributed representation for words. Techniques like Word2Vec [34] and Glove [38] have produced pre-trained embeddings for words that can be combined with more complex neural approaches.

The success of these approaches are later combined with the power of recurrent neural networks (RNNs) which led others to learn distributed representations of a larger piece of text like sentences [26], paragraphs [28], and documents [25]. Several variants of RNNs like convolutional-gated RNN [45], tree-LSTM [44] and Quasi-RNN [10] are proposed as alternative architectures. Apart from recurrent neural networks, methods employing convolutional networks [23, 49] have also produced compelling results for text classification tasks.

Recently, authors in [48] have argued that the hierarchical structure of documents helps to learn better representations. Acknowledging that the importance of words or sentences may vary depending on the context, they use an attention mechanism [5] while hierarchically encoding words into sentences and sentences into documents. In addition to the hierarchical structure, discourse structure [32] in the text, which represents the linguistic organization of a text as a tree, has also been studied in recent document representation work [8, 22, 29].

Each of these lines of research are based solely on the textual content assuming that the documents are plain-text. In this paper, we focus on machine-generated emails which have a rich HTML structure. In addition to leveraging the textual context, we encode the structural markup to learn a better representation. Note that in our work, the term "structure" corresponds to a formatting structure of an email, rather than the linguistic structure mentioned above. Combining our framework with the aforementioned architectures is possible and makes for promising future work.

Learning good email representations has been an area of interest for multiple research groups [2, 14, 41]. In addition to information extraction, email classifiers are useful for other applications like automatic foldering [6, 19], spam classification [13], and message priority ranking [1].

Lastly, systems like Fonduer [46] tackle richly formatted documents for the task of Knowledge Base Construction (KBC). Fonduer shows that augmenting textual features with structural and visual features produced by a library that parses the HTML markup and providing it to a strong baseline like a bi-directional LSTM with attention can indeed provide a significant improvement over just using the text features. Our results agree with the findings in Fonduer – formatting information does indeed help us learn a better representation for an email. In contrast to the approach of engineering a set of structural features (examples from Fonduer include HTML tag of the term, HTML tag of the parent, tabular features like row number, column number, n-grams from all the cells in the same row) our approach simply relies on learning an encoding using the sequence of HTML tags that appear in the path from the root to the node containing the term. Our results show that this is a powerful approach when combined with well-known building blocks like LSTMs and an attention mechanism. While we do not implement the nearly 40 structural, tabular, and visual features implemented in Fonduer, an interesting piece of future work is to study if XPath encodings as learned by our model can be as effective as the sophisticated features manually engineered in Fonduer.

6 CONCLUSION

In this paper, we study the problem of learning representations for richly structured emails. Considering the fact that most of today's emails are machine generated, we argue the rich formatting used in these emails contains highly valuable information that can be used to learn better representations. To address this gap in the literature, we proposed a novel framework called RiSER. Our framework combines three components from the email in order to learn an enhanced email representation: 1) textual content, 2) HTML structure, 3) manual annotation features. We show the effectiveness of our approach by evaluating the individual components as well as the full framework on email classification task across two datasets from a large email service. The experimental results and visualizations further indicate that our framework is learning an improved representation for emails and is capable of capturing valuable information buried in the email's HTML structure.

We are exploring multiple avenues of future research. We expect that leveraging structure will be valuable for other challenging tasks like information extraction from emails and web pages. We are considering ways to extend the RiSER framework to other types of documents with rich structure. For instance, PDFs and scanned images contain many informative layout and structural signals. However such documents do not contain XPath-style markup. We are exploring alternative ways to represent the layout and formatting information from such documents.

REFERENCES

[1] 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*.

[2] Manoj K Agarwal and Jitendra Singh. 2018. Template Trees: Extracting Actionable Information from Machine Generated Emails. In *Proceedings of the 30th International Conference on Database and Expert Systems Applications (DEXA)*. 3–18.

[3] Nir Ailon, Zohar S Karnin, Edo Liberty, and Yoelle Maarek. 2013. Threading machine generated email. In *Proceedings 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 *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM)*. 217–226.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[6] Ron Bekkerman, Andrew McCallum, and Gary Huang. 2005. *Automatic Categorization of Email into Folders: Benchmark Experiments on Enron and SRI Corpora.* Center for Intelligent Information Retrieval report IR-418. University of Massachusetts.

[7] Anders Berglund, Scott Boag, Don Chamberlin, Mary F Fernández, Michael Kay, Jonathan Robie, and Jérôme Siméon. 2007. XML Path Language (XPath). *World Wide Web Consortium (W3C)* (2007).

[8] Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. 2015. Better Document-level Sentiment Analysis from RST Discourse Parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2212–2218.

[9] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606* (2016).

[10] James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2016. Quasi-recurrent neural networks. *arXiv preprint arXiv:1611.01576* (2016).

[11] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. 2003. Extracting content structure for web pages based on visual representation. In *Asia-Pacific Web Conference*. Springer, 406–417.

[12] Godwin Caruana and Maozhen Li. 2012. A survey of emerging approaches to spam filtering. *ACM Computing Surveys (CSUR)* 44, 2 (2012), 9.

[13] Gordon V Cormack. 2008. Email spam filtering: A systematic review. *Foundations and Trends® in Information Retrieval* 1, 4 (2008), 335–455.

[14] Dotan Di Castro, Iftah Gamzu, Irena Grabovitch-Zuyev, Liane Lewin-Eytan, Abhinav Pundir, Nil Ratan Sahoo, and Michael Viderman. 2018. Automated Extractions for Machine Generated Mail. In *Companion Proceedings of the The Web Conference 2018 (WWW)*. 655–662.

[15] 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 *Proceedings of the 9th ACM International Conference on Web Search and Data Mining (WSDM)*. 307–316.

[16] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research (JMLR)* 12, Jul (2011), 2121–2159.

[17] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D Sculley. 2017. Google Vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 1487–1495.

[18] Google. 2019. Analyzing Entities. https://cloud.google.com/natural-language/docs/analyzing-entities.

[19] 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 *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM)*. 869–878.

[20] Ramanathan V Guha, Dan Brickley, and Steve Macbeth. 2016. Schema. org: evolution of structured data on the web. *Commun. ACM* 59, 2 (2016), 44–51.

[21] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[22] Yangfeng Ji and Noah Smith. 2017. Neural discourse structure for text categorization. *arXiv preprint arXiv:1702.01829* (2017).

[23] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).

[24] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[25] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent Convolutional Neural Networks for Text Classification. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*. 2267–2273.

[26] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*. 1188–1196.

[27] Philippe Le Hégaret, Ray Whitmer, and Lauren Wood. 2005. Document Object Model (DOM). http://www.w3.org/DOM.

[28] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057* (2015).

[29] Yang Liu and Mirella Lapata. 2018. Learning structured text representations. *Transactions of the Association of Computational Linguistics* 6 (2018), 63–75.

[30] Yoelle Maarek. 2016. Is Mail The Next Frontier In Search And Data Mining?. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining (WSDM)*. 203–203.

[31] Yoelle Maarek. 2017. Web Mail is not Dead!: It's Just Not Human Anymore. In *Proceedings of the 26th International Conference on World Wide Web (WWW)*. 5–5.

[32] William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse* 8, 3 (1988), 243–281.

[33] Andrew McCallum, Kamal Nigam, et al. 1998. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, Vol. 752. 41–48.

[34] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[35] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS)*. 3111–3119.

[36] Marc Najork. 2009. Web spam detection. In *Encyclopedia of Database Systems*. Springer, 3520–3523.

[37] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 79–86.

[38] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543.

[39] Xiaoguang Qi and Brian D Davison. 2009. Web page classification: Features and algorithms. *ACM Computing Surveys (CSUR)* 41, 2 (2009), 12.

[40] Herbert L Roitblat, Anne Kershaw, and Patrick Oot. 2010. Document categorization in legal electronic discovery: computer classification vs. manual review. *Journal of the American Society for Information Science and Technology (JASIST)* 61, 1 (2010), 70–80.

[41] Ying Sheng, Sandeep Tata, James B Wendt, Jing Xie, Qi Zhao, and Marc Najork. 2018. Anatomy of a Privacy-Safe Large-Scale Information Extraction System Over Email. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 734–743.

[42] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter* 19, 1 (2017), 22–36.

[43] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1631–1642.

[44] Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* (2015).

[45] Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*. 1422–1432.

[46] Sen Wu, Luke Hsiao, Xiao Cheng, Braden Hancock, Theodoros Rekatsinas, Philip Levis, and Christopher Ré. 2018. Fonduer: Knowledge base construction from richly formatted data. In *Proceedings of the 2018 International Conference on Management of Data (SIGMOD)*. 1301–1316.

[47] Yiming Yang and Xin Liu. 1999. A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 42–49.

[48] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. 1480–1489.

[49] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS)*. 649–657.