Automated Content Labeling using Context in Email

Aravindan Raghuveer

Yahoo!, Bangalore
India
aravindr@yahoo-inc.com

Abstract

Through a recent survey, we observe that a significant percentage of people still share photos through email. When an user composes an email with an attachment, he/she most likely talks about the attachment in the body of the email. In this paper, we develop a supervised machine learning framework to extract keywords relevant to the attachments from the body of the email. The extracted keywords can then be stored as an extended attribute to the file on the local filesystem. Both desktop indexing software and web-based image search portals can leverage the context-enriched keywords to enable a richer search experience. Our results on the public Enron email dataset shows that the proposed extraction framework provides both high precision and recall. As a proof of concept, we have also implemented a version of the proposed algorithms as a Mozilla Thunderbird Add-on.

1 Introduction

The life of a digital photograph starts with a innocuous click on a camera. Thereafter it is copied through several forms of digital media and sometimes enriched using image editing software. Finally the photo is shared with a larger community, either by means of emails or by means of an online photo-sharing service (e.g. Flickr) or by publishing it on a web-page (e.g. as part of a blog update or a news article). Once shared, the daunting task of building indexes to enable search on photos, both in the context of a personal computer and on the Internet, begins.

The well known bane of semantic gap [10] coupled with the rapid increase in number of digital photographs [24] has made the problem of search an extremely challenging one. To bridge the semantic gap, recent research tries to exploit information collected about a digital photograph during different stages of its life. Below is a short overview of such approaches, grouped by the life-stage to which they belong:

17^{th} *International Conference on Management of Data COMAD 2011, Bangalore, India, December 19–21, 2011*

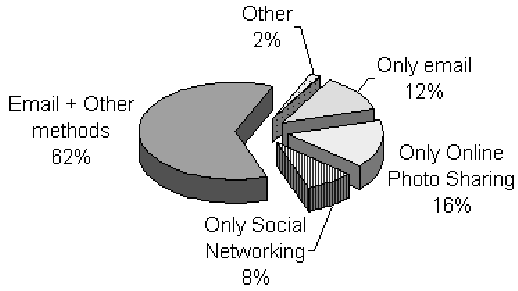

Figure 1: How do people share photographs? Answer: 74% of users use email in one way or other.

Inception: A digital camera embeds EXIF and GPS metadata on a photo, the instant the photo is shot. A variety of techniques have been proposed to exploit the EXIF metadata and the GPS data to label unseen images [6, 24, 15].

Enrichment: In the next stage of a photo's life, personal photo management and editing software like Picasa [1] help users create tags for image regions like faces. This human generated annotations are propagated to other photos that have similar visual regions.

Sharing: Search engines use the text in the vicinity of a photo on a HTML page to infer what the photo is about. A variety of approaches combine the Flickr tag information and the content to infer the semantics of a photograph [21, 23, 25, 16]. The ESP game [27] is a novel method to label publically available photographs through an engaging game.

In this paper, we propose a supervised learning framework to glean information about a photograph during one specific phase of its life cycle: *photo sharing through email*. To gauge the importance of email in photo sharing, we conducted a survey among a technology-savvy community. From the 50 volunteers we had, we found that 12% of them still use email as their only means of sharing their photos with friends and family. 62% of the partipants use email in addi-

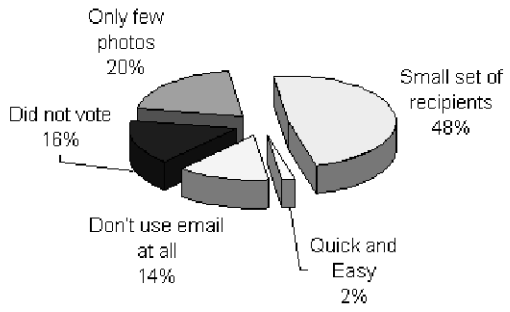

Figure 2: Why do people use email for sharing photos?

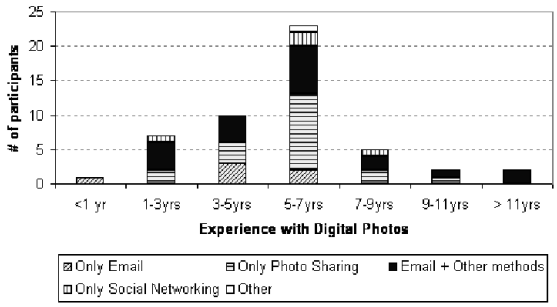

Figure 3: Frequency of various photo sharing methods as seen as function of participant's experience with digital photos.

tion to other methods like online photo sharing sites and social networking sites. Figure 1 shows the results of the survey. The results were surprising, yet true.

When users send photos through email, they tend describe about the photos in the email. In this paper we address the problem of automatically extracting tags for the photos, sent as attachments, from the body of the email. We face two key challenges in this problem:

1. **Not all sentences are relevant to the attachment:** Our analysis on a subset of the Enron Email Corpus[12, 18] shows that in only 67% of the emails, all sentences in the email are relevant to the attachment(s) in the email. Finding the sentences that are relevant to the attachment(s) is the first challenge.

2. **Hidden Context:** While composing emails, users tend to rely on context that is easily understood by humans. Some common examples are usage of pronouns, nick names and cross referencing information from a different conversation in the same email thread. This context is however not easily perceivable (in other words, hidden) by a machine. Designing an algorithm that is aware of hidden context proves to be the next challenge.

We make three key contributions in this paper:

- Interesting insights from a user study about how technically-advanced users share photos.

- Systematic design of natural language based features and problem-specific features that we use in conjunction with well-known supervised learning algorithms. By using the proposed features and tweaking the parameters of the learning algorithms, we are able to achieve a precision of 0.89 and a recall of 0.85 on the publically available Enron Email Dataset [18].

- A prototype implementation of a Mozilla Thunderbird extension to automatically extract tags from the body of the email based on the ideas proposed in this paper.

The remainder of the paper is organized as follows. In Section 2, we describe the results of the survey we conducted about photo sharing. In Section 3, we formulate the problem of extracting relevant keywords about an attachment from the body of an email. We also present an analysis of the Enron Email Corpus[12] that we use for measuring the performance of the proposed algorithms. We describe the design of the various features we use in Section 4. Section 5 covers the performance analysis of the proposed algorithms. In Section 7, we present our prototype *TagSeeker*, a Mozilla Thunderbird extension that can automatically generated tags from the body of the email. We discuss related work in Section 8. Section 9 concludes the paper.

2 Photo Sharing Survey

We conducted a survey to find out how people share photos with friends and family. We solicited participants from a community that we knew had medium to highly experienced users of the personal computer. The survey covers fifty participants and the demographics statistics also indicate that all of them are conversant users of the home computer and the Internet. In Figure 1, we see that 12% of the participants still use email as the only means of sharing photos. This outcome came as a surprise to us because the participant community is one that is technology-savvy. We can expect this percentage to go up with less technically advanced users. It is also seen that 62% of the participants use email in addition to other means of sharing photos. Figure 2 shows why people use email for sharing photos. Close to half of the participants use email when they need to share the photos with a small set of people.

Figures 3 and 4 throw light on the demographics of the participants while simultaneously drilling down on how each category of the participants share photos. In Figure 3, we see that irrespective how experienced users are with digital photographs, they still use email as one of the means to share photos. However, it also

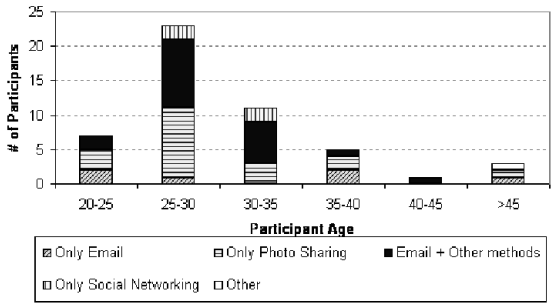

Figure 4: Frequency of various photo sharing methods seen as a function of participant age.

points out that long-time users of digital photographs (> 7 years of experience) are moving away from using email as the only mode to share photos.

Figure 4 shows that using emails as the only mode of photo sharing is prevalent across almost all age groups. Also surprisingly, we see that the younger group of participants do not use social networking sites as their only means of sharing photos when compared to their older counterparts.

We also were able to collect one more very interesting datapoint about how many attachments in an user's inbox and sent mails tend to be photos. We found that if a user receives an email with an attachment, there is a 38% chance that the attachment is a photograph. Similarly, when a user sends an email with an attachment, there is a 43% chance that the attachment is a photograph. This evidence further adds credibility to the hypothesis that emails are still frequently used to share photos.

To summarize the learnings from the user study above:

- A non-trivial percentage of the population still use email as their only mode to share photographs.
- Most of the other users use email in addition to other methods of sharing photos.

3 Preliminaries

In this section, we first formulate the problem we set to solve in this paper. Next, in Section 3.2, we present an analysis of the Enron email dataset and argue why it is a well suited dataset for measuring the performance of the proposed algorithms.

3.1 Problem Formulation

Problem Statement: Given an email E that has a set of attachments A, find the set of words K_{EA} that appear in E and are relevant to the attachments A.

We divide the above problem into the following two sub-problems.

3.1.1 Selecting Sentences

Let S_E be the set of sentences belonging to email E. The first sub-problem deals with identifying a subset of sentences $\hat{S}_{EA}$ from S_E which are relevant to the attachments A. Our primary contribution in this paper is to develop minimally supervised machine learning algorithms to solve this sub-problem. Sections 4 and 5 describe the proposed solution.

3.1.2 Selecting keywords from the sentences

The second sub-problem extracts useful keywords from the set $\hat{S}_{EA}$ generated by the previous step. We use a sequence of heuristics to convert the sentence set $\hat{S}_{EA}$ into the keyword set K_{EA}. We describe the heuristics in Section 6.

3.2 The Enron email dataset

For our experiments we use the Enron Email Corpus described in [12]. Dredze et al [12] curated the original Enron Email corpus[18] to make it suitable for research about email attachments. One of the key steps in the curation process was to remove emails that contained a forwarded message as an attachment. For more information about the curated corpus, please refer [12].

In this section, we describe those characteristics of the Enron email corpus that are important to our problem formulation in Section 3.1. We also argue why this dataset is well suited for measuring performance of the proposed algorithms.

The corpus consists of 157,510 emails belonging to 150 users. A total of 30,968 emails have at least one attachment. Further drilling down, 33,072 emails are present in the sent folders of users, of which 2,576 have at least one attachment. From the above statistics, we see that the dataset has substantial amount of real world data (30,968 emails) about emails that have attachments.

Figure 5 shows the number of emails with attachments sent/received by a particular user. A co-ordinate (x, a) on the *sent with attachment* curve represents that the user x sent a emails with at least one attachment. The corresponding co-ordinate (x, b) on the *received with attachment* curve represents the number of emails b the same user x received with at least one attachment. This graph shows that the dataset has emails with attachments sent by a wide variety of users. Therefore the dataset captures the different styles users adopt while describing an attachment in an email. Also it is seen that users always (but for 4 outliers) receive more emails with attachments than they send with attachments.

Figure 6 shows the histogram of attachment types in received and sent emails. We have not shown in Figure 6 the long tail of rare attachment types. Since the enron corpus consists primarily of official emails, the attachment types histogram is dominated by Word

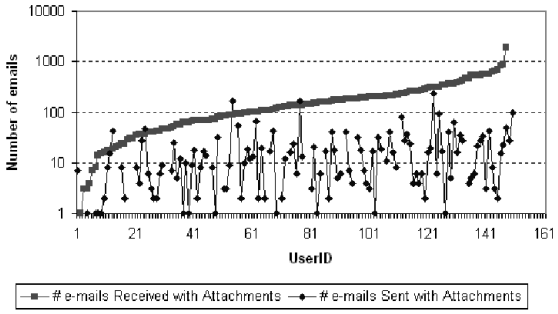

Figure 5: Number of received and sent emails with atleast one attachment. Note that the X-axis (userID) is sorted in increasing order by the number of received emails with attachments.

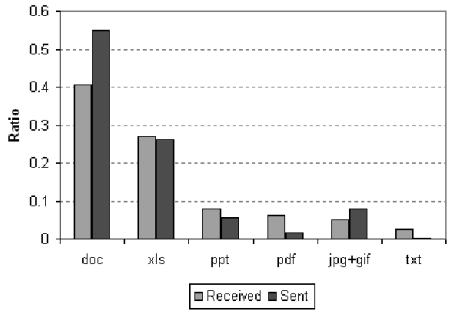

Figure 6: Histogram of attachment file extensions. Y-axis shows the ratio of number of attachments sent/received with a particular file type to the total number of attachments sent/received.

Documents and Spreadsheets. Consequently, we do not have a substantial sized subset of emails with image attachments alone. However, the techniques we propose in this paper are not restricted to emails with image attachments alone. Therefore this dataset is still very useful in measuring the performance of the proposed algorithms.

Finally, Figure 7 shows the Cumulative Distribution Function (CDF) of number of sentences contained in an email with attachment. According to this graph, 80% of the emails sent with attachment(s) have less than 8 sentences. Interestingly enough, in our analysis in a later section (Section 4.7), we find that even in emails that have less than 3 sentences, not every sentence is related to the attachment. So the problem of finding relevant sentences to an attachment remains real even in small emails.

4 Feature Selection

As described in Section 3.1, the task of assigning keywords to attachments can be split into two phases. In the first phase we identify the sentences in the email that are relevant to the attachments in the email. We

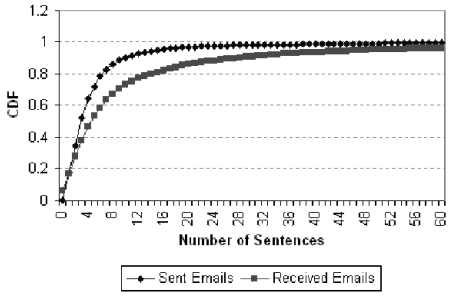

Figure 7: CDF of Number of Sentences. Y-Axis denotes the probability of an email having <= x sentences. #emails sent with attachment=2,576, #emails received with attachment=28,392

formulate the problem of identifying sentences relevant to attachments as a binary classification problem. We represent each sentence in an email as a feature vector and then use supervised learning algorithms to learn models that can predict the label of an unseen sentence. The label here refers to whether the sentence is relevant to the attachment or not.

In this section, we describe the features that were selected for this classification problem and the rationale behind selecting each feature. All the features proposed in this section are binary, sentence level features. Unlike previous approaches [12], we do not use any user specific features. We make this choice deliberately because getting user-specific training data is either not practical or very hard to get to.

The proposed features are broadly grouped into six categories as described by the subsections 4.1 through 4.6. In Section 4.7, we illustrate the prediction potential each feature by means of a correlation analysis.

4.1 Anchor Features

We define an *anchor sentence* as a sentence in the email that has a very high probability of directly referring to the attachment. We define the following four boolean, sentence-level features that can be used to detect anchor sentences.

4.1.1 Strong Phrase Anchor

The strong phrase anchor is marked true for a sentence if it has one of the following phrases in it: *attach*, *here is*, *enclosed*. We term these phrases as *strong anchor phrases*. We observe that when one of the strong anchor phrases appears, there is a high chance that the sentence it appears in is relevant to the attachment. However, the anchor sentence may not be the only sentence that is relevant to the attachment. We also observed that in the Enron dataset we used, out of the 30968 emails that have an attachment, 52% of them had a strong anchor phrase. This shows that the strong anchor phrase feature is a powerful signal to

Feature Set Name	Features
Phrase Anchor-Only	Section 4.1.1
Phrase and Extension Anchors only	Sections 4.1.1- 4.1.2
All Anchors	Section 4.1
No Lexicon Anaphora	Sections 4.1- 4.4
No Lexicon	Sections 4.1- 4.5
All	Sections 4.1- 4.6

Table 1: Feature Sets: Please note that the feature sets are cumulative as we move down the table.

predict sentences relevant to the attachment. On the other hand, it signals that we need more features to cover the remaining 48% of the emails.

4.1.2 Extension Anchor

By manually perusing through the emails in the corpus, we observe a pattern that users tend to refer to an attachment by its file type. For example, if the attachment is a .xls file, users tend to refer to the attachment as "'spreadsheet"', "report", "'excel file"'. If the attachment is a .jpg file, users refer to it as "'picture"', "'image"', "'snap"', "'photo"'. The following email snippet taken out of the enron email corpus gives an example:

```
''Please refer to the attached spreadsheet
for a list of Associates and Analysts who
will be ranked in these meetings and their
PRC Reps."
```

We use a dictionary to map a file extension to the keywords the file extension could be referred to as. Given an attachment extension, we fetch the keywords from the dictionary and mark the extension anchor as true for those sentences that have one of the keywords. Note that this feature is designed for any kind of file extension; not necessarily limited to image files.

With keyword based match, it possible to have false matches with the same word used in a different context. We use part of speech tagging to minimize this category of noise. We use the Stanford Part of Speech Tagger [2] to tag all the words with their part of speech elements. We add the constraint that the keyword found in the sentence needs to be a noun for the extension anchor feature to fire.

4.1.3 Attachment Name Anchor

Another useful behavioral pattern that we observe is that users tend to use tokens from the file name of the attachment to refer to the attachment. We tokenize attachment names on case transitions boundaries(lowercase to uppercase and vice versa) and punctuation symbols boundaries. Consider the following email snippet taken out of the enron email corpus:

```
‘‘ These are book requests for the Netco
books for all regions.  Please let me know
if you have any questions."
```

The email contained 5 attachments each of the form "Book Request Form *region_name*.xlsẅith region_name replaced with Financial, East, Central, Texas, West. With the attachment name anchor, we would recognize that the first sentence is relevant to the attachment.

Another useful property of the attachment name anchor is that when the attachment name anchor is true for a given sentence s_i, then s_i is most likely relevant to that attachment whose file name tokens were found in s_i. We exploit this property to assign attachment specific tags when an email contains more than one attachment.

4.1.4 Weak Phrase Anchor

The weak phrase anchor feature is the same as the strong phrase anchor by principle. It differs in the aspect that the weak anchor phrases provide weaker signals compared to the strong anchor phrases. The weak anchor phrases that we use are: *above*,*below*, *following*, *forward*, *file*, *draft*, *report*. It might seem that the weak anchor phrases can lead to loss in precision. However we observe in our experiments that when the weak anchor feature is used in conjunction with the word lexicon feature we describe later in (Section 4.6), the precision improves. Again to minimize noise, we place the restriction that when the words *file* or *report* or *draft* appear, they have to be used as nouns.

4.2 Noisy Sentences

We use two features to mark sentences that are most likely not interesting. These features are put in place to capture artifacts like signature sections of emails and header sections in email threads.

- **The Noisy Noun Feature:** This feature is marked true if more than 85% of the words in the sentence are nouns.
- **Noisy Verb Feature:** This feature is marked true if there are no verbs in the sentence.

4.3 Short Emails

The short email feature is marked true if the email to which the sentence belongs has less than or equal to 4 sentences. The hypothesis behind this feature is that all sentences in a short email tend to be relevant to the attachment.

4.4 Conversation Level in an EMail Thread

An email thread consists of multiple conversations. Each conversation is an email sent by a user either in response to a previous conversation or is the first email in the thread. The level of a conversation refers to how close a given conversation is to the last conversation. The last conversation has level equal to zero. So higher the level of the conversation, older it is in the email thread.

Consider a case where the strong phrase anchor feature is triggered for a sentence with level of five (i.e. it is part of the fifth conversation of an email thread). It is very unlikely that this sentence refers to the attachment in the current conversation in the zeroth level. We have observed that levels beyond the second conversation do not have content relevant to the attachment used in the first level conversation. We define two features here to capture this hypothesis:

- **Conversation Level <=2** This feature is marked true if the sentence appears in the first or second level conversation of the thread.
- **Conversation Level > 2** This feature is marked true if the sentence appears in a conversation of level strictly more than two.

We use a simple heuristic to calculate the level of a given sentence. According to [8], the > symbol is a very strong indicator of a reply thread. The authors report that if a sentence starts with the > symbol, there is a 95.1% chance that the sentence is part of a reply thread. We first validated this finding on the Enron email dataset and found it to be consistent. We then extended the heuristic to count the number of leading > symbols to indicate the level of the thread. Therefore if a sentence starts with >>, we would mark the level of the sentence to be two. Note that we assume that any number of whitespace characters can be present between the two > symbols.

4.5 Anaphora Detection

The Anchor features described in Section 4.1 help identify sentences in an email that directly refer to the attachment. The anaphora feature we describe in this section helps capture indirect references to attachments. By capturing indirect references, the anaphora feature aids the learning algorithm to track the hidden context in the email. Consider the following email snippet:

"`I made my staff burn the midnight oil last night putting together TW/NNG revenue summaries for review by the bankers that Rod/Kevin are working with. Thought you might me interested in the report. It gives a nice snapshot of our activity with our major counterparties.`"

In the above example, the first sentence has an attachment name anchor and the second sentence has a extension anchor. Hence both the sentences can be identified as relevant to the attachment. The third line uses a pronoun reference to continue the description about the attachment. However using the features we have seen so far, it is not possible to track this dependency between the second and third sentence. Therefore it is not possible to identify that third sentence as related to the attachment.

In the context of linguistics, the pronoun (*It*, in our example above) is called the anaphora and the noun (*the report* in our example above) that it refers to is called the antecedent. Note that the anaphora can also be another noun (e.g. abbreviation). The problem of resolving references that represent the same real world entity is called Anaphora Resolution and is a very active research topic in the Natural Language Processing community [26].

We propose the anaphora feature which is marked true for a sentence that has a back reference to an anchor sentence, thus help track indirect references to the attachment. The anaphora feature is computed as follows. We first use the Stanford POS Tagger to identify all the nouns in an anchor sentence. Then we use the BART framework [26] to identify pronouns and nouns in other sentences that refer to the nouns used in the anchor sentence. This operation establishes a link between an anchor sentence and other sentences that depend on the anchor sentence. We mark the anaphora feature to be true for the dependent sentences.

4.6 Lexicon Features

We build a dictionary from all the words that are used in the Enron email corpus. We remove stop words and use the Porter's stemming algorithm to compute the words in the dictionary. The size of the dictionary is 10100 words. Each word that appears in the dictionary is a lexicon feature. For a given sentence, we set a lexicon feature to true if the corresponding word appears in the sentence. In our experiments, we find that the lexicon feature help boost the precision of weak signal features like the weak phrase anchor.

4.7 Correlation Analysis

We build a ground truth data set with 6472 sentences. Each sentence is labeled whether it is relevant to an attachment or not. We discuss in detail about how the ground truth data set was built in Section 5.1. We calculated the Pearson Correlation coefficient between each feature and the label associated with the sentence. Note that all features we use are binary features. Therefore high positive correlation between a feature f and the label would signify that whenever feature f appears, the sentence it appears in has a high probability of being a relevant sentence. Similarly, a high negative correlation would mean that when the

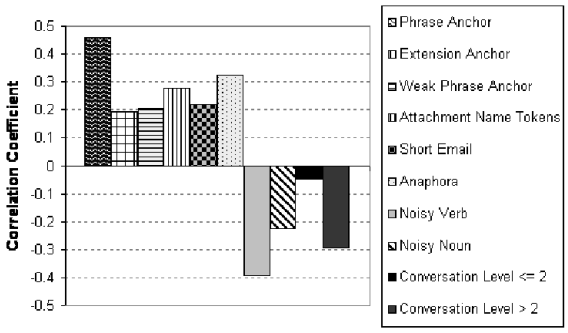

Figure 8: Feature - Class Label Correlation

feature f appears, the sentence is most likely not a relevant sentence. Figure 8 shows the correlation coefficient between each feature and the class label of the sentence. Note that the strong phrase anchor feature and the anaphora feature have the best positive correlation to the class label. Also, the short email feature surprisingly has a comparatively low correlation coefficient. This means that all sentences in a short email do not necessarily need to refer to the attachment in the email. Observe that the noisy verb feature has a good negative correlation with the class label. Another interesting aspect is that the conversation level <=2 feature has lower negative correlation when compared to the conversation level > 2 feature.

5 Experiments

5.1 Ground-Truth Data

The Enron Dataset has a total of 33072 *sent* emails by 150 users, out of which 2576 emails have at least one attachment. From the sent emails that had at least one attachment, we randomly sampled 1800 emails. The reason we chose *sent emails* as opposed to *received emails* was to eliminate the chance of redundant emails. Two editors, working independently, examined each sentence from the 1800 emails and decided whether that sentence referred to an attachment in the email. In addition to the sentences in the body of the email, the editors were given the names of the attachments as well. We then juxtaposed the annotations from the two editors and discarded those emails that had at least one sentence with conflicting labels. After this reconciliation step, we were left with 1150 emails whose sentences were marked either relevant to attachment or not. This data corresponds to 112 users from the Enron corpus and 6472 sentences.

5.2 Machine Learning Algorithms

With the ground truth data available, we then chose three machine learning algorithms for building models to label unseen data. Using the features proposed in Section 4 , we model the attachment sentence selection problem described in Section 3.1 as a binary classification problem. We choose two non-sequential machine learning techniques: Naive Bayes and Support Vector Machines and one sequential algorithm: Conditional Random Fields The light-weight aspect of Naive Bayes is useful in our case because we want to embed the sentence classification algorithm inside an email client. Next, we chose Support Vector Machines (SVM), a more sophisticated non-sequential model that has been proved to be very effective with text corpus. Finally, we also choose Conditional Random Fields (CRFs), a sequential learning algorithm.

Since our contribution in this paper is not a new machine learning technique, we do not cover all well known machine learning algorithms. We have selected 3 techniques to cover a few well known state of the art techniques available. In the following subsections, we briefly summarize each learning algorithm and also describe how we tweaked the parameters of each algorithm to suit our dataset better.

5.2.1 Naive Bayes

The Java-ML framework [11] provides a simple and consistent interface to implement various classifiers. We used the Weka Naive Bayes classifier [14] to implement the Java-ML classifier interface. We used the default parameters set by the Weka package.

5.2.2 Support Vector Machines

We used the libSVM [9] to implement the Java-ML classifier interface. We found that the Radial Basis Function (RBF) kernel to give better performance than the linear kernel [8]. We set the margin parameter C to 10 after finding that to yield best performance among 20 values we tried in the range 10^{-2} to 10^{2}.

5.2.3 Conditional Random Fields

Conditional Random Fields (CRF) [19] have been recently used widely in the context of labeling sequential data. The motivation behind employing CRFs in our problem is to exploit any relation that subsequent sentences in an email may have. We use the CRFSuite implementation [22], whose main feature is the fast running times at the expense of high memory usage. In terms of tweaking the parameters, we found the L2 regularization to give better results consistently than the L1 regularization. The regularization parameter sigma (variance of feature weights) [22] did not have a significant effect on the performance.

5.3 Feature Ranking

In the first experiment, we study the relative importance of the features proposed in Section 4 for the classification problem. We divide the features into six sets as shown in Table 1. Features were bucketed into the sets based on two criteria a) If a feature appears to be important through the initial study conducted shown

		Predicted Class	
		Yes	No
Actual Class	Yes	True Positive (TP)	False Negative (FN)
	No	False Positive (FP)	True Negative (TN)

$P = \frac{TP}{(TP+FP)}$; $R = \frac{TP}{(TP+FN)}$; $F1 = \frac{2 \cdot P \cdot R}{(P+R)}$
where: P is precision, R is recall, F1 is F1 measure

Table 2: Confusion Matrix: "Yes"' indicates that the sentence is a relevant sentence to attachment. Precision, Recall and F1 score computation.

Features	Naive Bayes			SVM			CRF		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Phrase Only	0.85	0.06	0.11	0.91	0.09	0.1638	0.69	0.48	0.56
Phrase + Extension	0.38	0.12	0.1824	0.56	0.17	0.26	0.7089	0.5359	0.61
All Anchor	0.61	0.32	0.41	0.83	0.31	0.45	0.84	0.64	0.72
No Lexicon Anaphora	0.76	0.39	0.51	0.84	0.31	0.45	0.89	0.6	0.71
No Lexicon	0.73	0.59	0.65	0.79	0.71	0.74	0.86	0.81	0.83
All	0.795	0.7	0.74	0.85	0.74	0.79	0.89	0.85	0.87

Table 3: Precision, Recall and F1 scores for combinations of learning algorithms and feature sets.

in Figure 8, it is observed in isolation. b) Related features are put into the same bucket. Please note that the latter feature sets consume the features that occur in previous feature sets.

We have two parameters in this experiment:

1. Which feature set F_i to use?: We have 6 choices shown in Table 1

2. Which learning algorithm A_j to use?: We have 3 choices described in Section 5.2.

The goal of this experiment is to evaluate the performance of each (F_i, A_j) pair where F_i is a feature set and A_j is a learning algorithm. For each (F_i, A_j) pair, we represent the 1150 emails from the ground truth dataset described in Section 5.1 as feature vectors in the space of F_i and then run a 5 fold cross validation of algorithm A_j on the feature vectors generated. The precision, recall and the F1 measure are calculated for each (F_i, A_j) pair as shown in Table 2. We use the (Phrase Anchor Only feature, Naive Bayes) set as our baseline.

Table 3 shows the result of this experiment. First of all it is evident that CRF has the best performance overall in terms of precision, recall and F1 measure. CRF has better performance across all feature sets as well. Similarly SVM outperforms Naive Bayes across all feature sets.

Next, we see that the *Phrase anchor* feature has the best prediction capability. This result concurs with the high co-relation coefficient between occurrence of Phrase Anchors and positive labels as seen in Figure 8. The anaphora feature provides an improvement of 0.12 to the F1 score of CRFs. The combination of attachment name anchor and the weak phrase anchors also provide a boost of 0.12 to the F1 score of CRFs. The other features contribute less significantly to the F1 scores.

We now highlight some interesting insights in Table 3 that show the behavior of a feature set or learning algorithm. When phrase anchors are used, SVM and Naive Bayes show high precision and low recall while CRF shows substantial high recall with low precision. The reason for this behavior was found to be because of a sequential learning of the CRF algorithm. The CRF algorithm learns that under the absence of any other feature, the first sentence of an email has a high probability of being a relevant sentence. Due to this learning, if a sentence had no phrase anchor, CRF would mark that sentence as relevant with high probability while SVM and Naive Bayes would label the sentence as not relevant.

We see that the anaphora feature provides the best increase in recall, next only to the phrase anchor feature. This is because the anchor features first identify the anchor sentences in the email. The anaphora feature kicks in and expands the scope of relevant sentences around the anchor sentences. Finally, we observe a minor dip in recall when the *No Lexicon Anaphora* Feature set is used. This is because the noisy verb feature caused few sentences to be labeled not relevant when they were actually relevant.

5.4 Training Set Sizes

In the next experiment, we study the sensitivity of a learning algorithm to the amount of training data available. This experiment is crucial in the context of our problem of sentence labeling because a learning algorithm that requires large amount of training data to produce consistently good results is not practical. We randomly sample 230 emails from the ground truth

	No Lexicon		All Features	
Algorithm	Train Time (msec)	Tag Time (msec)	Train Time (msec)	Tag Time (msec)
Naive Bayes	0.05	0.03	0.09	0.05
SVM	0.19	0.1	0.67	0.17
CRF	0.02	0.003	0.09	0.01

Table 4: Training and Tagging Time per sentence

data, a fifth of 1150 emails present in the ground truth data. This sampling rate is chosen in order to be comparable to the previous experiment which used five-fold cross-validation. From the remaining 920 emails, we randomly sample t% of the emails as training data. In this experiment we vary t from 0.1% to 100%. We train the learning algorithm with $(t/100) \cdot 920$ emails and test it with 230 emails. The results of this experiment is shown in Figure 9. We see that Naive Bayes reaches its best performance with just 15% of the full training set while SVM requires close to 90% of the training data to reach its steady-state, best performance. CRF, on the other hand, is positioned in middle ground requiring 50% of the training data to achieve stable performance.

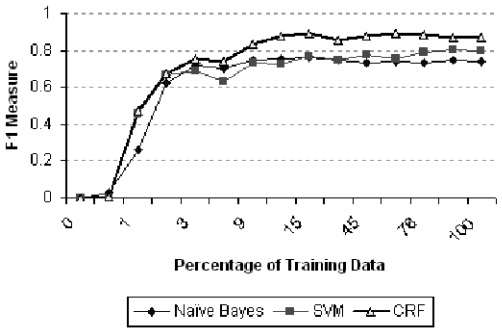

Figure 9: Training Set Size vs F1 score

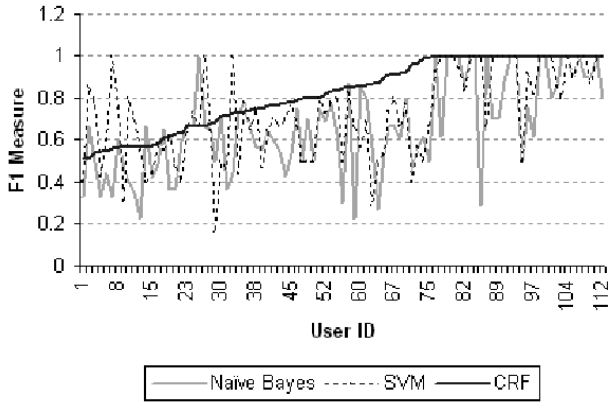

Figure 10: User Level Performance. The X-Axis is sorted by the F1 score of the CRF method.

5.5 User-specific Performance

The features we proposed in Section 4 do not depend on any user-specific information. An ideal (feature set, learning algorithm) combination should work equally well across different users. In the next experiment, we study the performance of the three algorithms with the full set of features at the granularity of a user. We repeat the five-fold cross validation experiment with 1150 emails and for each user, we calculate the F1 scores for the emails that he/she sent. Figure 10 shows the F1 scores for the three learning algorithms with all features across 112 users. Therefore every email would be in a test set atleast once. In a majority of the cases we see CRF outperforming the other two methods. Also, SVM consistently outperform Naive Bayes across all users. This experiment proves that with the same set of features, the CRF can learn a more generic model applicable to a variety of users.

5.6 System Level Performance

The previous experiments were designed to measure the precision and recall of the learning algorithms with the proposed features. In the next experiment, we measure the system level performance of the learning algorithms. Specifically, we measure the time taken by a learning algorithm for training a model and the time taken to label unseen sentences. These system level performance metrics are crucial to our application because the sentence labeling algorithm will eventually be embedded into an email client. An email client being a desktop application is expected to be light-weight and cannot consume CPU resources for an extended period of time.

We perform the five-fold cross validation experiment for two sets of features: No Lexicon and All Features. Table 5.3 summarizes the performance of the three learning algorithms. Note that the times mentioned in the table are at per-sentence granularity. It is not fair to compare the absolute times across algorithms because we use different implementations. However, the purpose of this experiment is to observe how the the train and tag time is affected by the two feature sets.

We observe that when lexicon features are used, tag times of CRFs increase by an order of magnitude. The performance of SVM also degrades but with a smaller fraction. Naive Bayes being a simple technique is not affected much by the lexicon features. By adding lexi-

con features, we introduce 10100 more dimensions into the feature space. This explosion in number of features affects the performance of the more complicated SVM and CRF algorithms.

6 Keyword Selection

In the previous two sections we described how to identify a set of sentences $\hat{S}_{EA}$ from an email E that are relevant to an attachment set A. In this section, we address the next sub-problem of identifying words K_{EA} from $\hat{S}_{EA}$ that are relevant to A.

Once the set $\hat{S}_{EA}$ is generated, we identify the keywords K_{EA} by applying the following heuristics in sequence on each sentence $s_i \subseteq \hat{S}_{EA}$:

1. Replace relative date references like "'tomorrow"', "'Monday"', "'next week"' with actual calendar dates like 8^{th},April 2010. We use the date in the email header to convert relative date references to actual calendar dates.

2. Replace pronouns with the nouns they refer to. We use BART to resolve pronoun-noun coreferences as explained in Section 4.5.

3. First person pronouns (e.g. I, mine, my) that refer to the sender cannot be resolved to the sender's name by the anaphora resolution module. So we replace first person pronouns with the sender's name. We also replace the second person pronouns (e.g. you, your)with the receiver's name. Both the sender's name and the receiver's name are got from the email header. Note that the sender's name and the receiver's name change based on the conversation level as defined in Section 4.4.

4. Use a regex to remove sensitive information like credit card numbers, objectionable words.

5. Remove stop words

6. Add the words that remain in sentence s_i to the keyword set K_{EA}.

7. Exception to Step- 6: If s_i contains tokens from the name of an attachment $A_j \subseteq A$, then the keywords generated from s_i are added as tags to attachment A_j alone. These keywords generated from s_i are not added to the set K_{EA} which contains keywords common to all attachments. Please refer to Section 4.1.3 for the reasoning behind this step.

7 Discussion

Keywords for attachments can be generated when sending them in an email or when saving an attachment from a received email to the filesystem. The keywords are especially useful for non-indexable content like images. The keywords could also prove to be useful for directly indexable content like spreadsheets and documents to provide additional context that the content does not capture.

Figure 11: TagSeeker: A ThunderBird Addon for Automatic Tag Detection for Attachments.

One of the use cases that we explore is to add the keyword as tags to the file on the local filesystem. All modern filesystems support the concept of extended attributes to tag the file with file-specific metadata. With the growing popularity of native Javascript extensions like BrowserPlus [3], embedding this usecase into popular email clients like Gmail and Yahoo! mail is now feasible. Once the files have been tagged with relevant keywords, a desktop indexing system like Google Desktop [4] can index the keywords which can be later used to search for the file.

As a proof of concept of this use case, we have implemented a Mozilla Thunderbird add-on called *TagSeeker*. *TagSeeker* automatically scrapes keywords for the attachments from the body of the email. Figure 11 shows a screen-shot of *TagSeeker* in action. We have been able to embed the following features into *TagSeeker*:

- Naive Bayes based sentence classification with no anaphora features used.
- User can register "blacklist" regexes to remove those keywords that match any "blacklist" regex
- BrowserPlus [3] integration under progress to add tags into the files themselves.

In the following two subsections we discuss some scope for improvement that is left for future work.

7.1 Privacy concerns

Since the keywords are generated automatically, they can possibly contain information that is private to

the user. When tagged, the keywords become part of the meta-data of the file. Due to this concern, users can register "blacklist" regexes with *TagSeeker*. Those keywords that match any "blacklist" regex will be discarded. This solution however does not suit a non-technical user. An ideal solution would be an online machine learning framework [5] where the user accepts or rejects each keyword that is suggested by *TagSeeker*. Based on this feedback, a model is learnt over time to predict which keywords the user is most likely to accept.

7.2 User-Specific Trained Models

Dredze et al[12] suggest that the lexicons of users vary widely. Hence by building a user-specific lexicon, the models can be trained better. We chose not to build user-specific prediction models because in practical applications, obtaining good quality, user-specific training data is extremely challenging. Investigating practical means of learning user-specific models with minimal training is a very promising direction.

8 Related Work

Research on email data has been greatly facilitated by the availability of the Enron Email Corpus [18, 12]. Klimt et. al [18] use the Support Vector Machine model to classify the folder to which an email should belong. Dredze et al [12] propose an unsupervised learning algorithm to predict if an email requires a file to be attached along. In [13], Dredze et. al. propose an unsupervised learning framework to select keywords from the email to concisely represent the gist of an email. The criteria used to judge the quality of the keyword is that it should be characteristic of the email it represent and at the same time it should relate to the latent topics already present in the user's inbox. Our work differs from [13] both with respect to the goal of keyword generation and with the metric used to measure quality of an extracted keyword. A keyword extracted using [13] summarizes an email but may not accurately capture the context for the attachment, which is our goal. Yoo et al [28], mine the social network graph from an email corpus and use this graph to prioritize the unread emails in an user's inbox. Vitor et al [8] address the problem of learning to extract signature and reply lines from an email. They evaluate the proposed features using both sequential and non-sequential learning algorithms.

A variety of novel methods have been proposed to enable fully automated and semi-automated tagging of photos. Boutell et al [6] use features from both content and EXIF metadata to predict semantic labels for unseen photographs. [6] classifies whether a photo was taken indoors or outdoors, whether a given scene is a sunset scene or not. This contribution represents moving from pixels of information to semantic labels understandable by a user. Sinha et al [24]also exploit EXIF metadata and content to transfer labels from a training sample to an unseen image. Naaman et al [15] showed how geographic information embedded in the photo can be used for image summarization and management.

A variety of supervised and semi-supervised learning approaches [7, 20] have been proposed to use visual features to generate semantic tags. Jia et. al [17] propose a web-mining based approach to label personal photos. They propose a semi-supervised learning algorithm to compensate the noisy annotations that can be introduced by web-based annotation.

Community Tagging popularized by Flickr has led to a wide variety of research to summarize, find trends and derive higher semantic knowledge out of the tags that users generate [21, 16, 25]. Rattenbury et. al [23] use the geographic and timestamp metadata available in Flickr photos to associate place and event semantics to tags.

9 Conclusion

In this paper, we present a complete picture of an automated content labeling scheme. We start with a survey that proves email is still a widely prevalent method of sharing images. Next, we develop a supervised learning framework that can predict the labels for attachments given the content of the email. Through detailed experiments we show that the precision and recall of the proposed approach is superior. Finally, we demonstrate the feasibility of our approach through a prototype embedded into Mozilla Thunderbird.

References

[1] `http://www.picasa.google.com`.

[2] `http://nlp.stanford.edu/software/tagger.shtml`.

[3] `http://browserplus.org`.

[4] `http://desktop.google.com`.

[5] A. Blum. On-line algorithms in machine learning. In *In Proceedings of the Workshop on On-Line Algorithms, Dagstuhl*, pages 306–325. Springer, 1996.

[6] M. Boutell and J. Luo. Bayesian fusion of camera metadata cues in semantic scene classification. *Computer Vision and Pattern Recognition*, 2:623–630, 2004.

[7] G. Carneiro and N. Vasconcelos. Formulating semantic image annotation as a supervised learning problem. In *CVPR '05*, pages 163–168, Washington, DC, USA, 2005. IEEE Computer Society.

[8] V. R. Carvalho and W. W. Cohen. Learning to extract signature and reply lines from email. In *in Proceedings of the Conference on Email and Anti-Spam*, 2004.

[9] C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines. `http://www.csie.ntu.edu.tw/~cjlin/libsvm`, 2001.

[10] R. Datta, D. Joshi, J. Li, and J. Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Comput. Surv.*, 40(2):1–60, 2008.

[11] A. T. de Peer and Y. V. S. Y. Java-ml: A machine learning library. volume 10, pages 931–934, 2009.

[12] M. Dredze, T. Brooks, J. Carroll, J. Magarick, J. Blitzer, and F. Pereira. Intelligent email: reply and attachment prediction. In *IUI '08*, pages 321–324, New York, NY, USA, 2008. ACM.

[13] M. Dredze, H. M. Wallach, D. Puller, and F. Pereira. Generating summary keywords for emails using topics. In *IUI '08: Proceedings of the 13th international conference on Intelligent user interfaces*, pages 199–206, New York, NY, USA, 2008. ACM.

[14] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. 2009.

[15] A. Jaffe, M. Naaman, T. Tassa, and M. Davis. Generating summaries and visualization for large collections of geo-referenced photographs. In *MIR '06*, pages 89–98, New York, NY, USA, 2006. ACM.

[16] R. Ji, X. Xie, H. Yao, and W.-Y. Ma. Mining city landmarks from blogs by graph modeling. In *MM '09: Proceedings of the seventeen ACM international conference on Multimedia*, pages 105–114, New York, NY, USA, 2009. ACM.

[17] J. Jia, N. Yu, and X.-S. Hua. Annotating personal albums via web mining. In *ACM Multimedia*, pages 459–468, 2008.

[18] B. Klimit and Y. Yang. The enron corpus: A new dataset for email classification research. In *In Proceedings of the European Conference on Machine Learning (ECML*, 2004.

[19] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML '01*, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[20] W. Li and M. Sun. Semi-supervised learning for image annotation based on conditional random fields. In *CIVR*, pages 463–472, 2006.

[21] M. Naaman, S. Ahern, R. Nair, and T. Rattenbury. How flickr helps us make sense of the world: context and content in community-contributed media collections. In *In Proceedings of the 15th International Conference on Multimedia (MM2007*, pages 631–640. ACM, 2007.

[22] N. Okazaki. Crfsuite: a fast implementation of conditional random fields (crfs), 2007.

[23] T. Rattenbury, N. Good, and M. Naaman. Towards automatic extraction of event and place semantics from flickr tags. In *SIGIR '07*, pages 103–110, New York, NY, USA, 2007. ACM.

[24] P. Sinha and R. Jain. Classification and annotation of digital photos using optical context data. In *CIVR '08*, pages 309–318, New York, NY, USA, 2008. ACM.

[25] J. Tang, S. Yan, R. Hong, G.-J. Qi, and T.-S. Chua. Inferring semantic concepts from community-contributed images and noisy tags. In *MM '09: Proceedings of the seventeen ACM international conference on Multimedia*, pages 223–232, New York, NY, USA, 2009. ACM.

[26] Y. Versley, S. Ponzetto, M. Poesio, V. Eidelman, A. Jern, J. Smith, X. Yang, and A. Moschitti. Bart: A modular toolkit for coreference resolution. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may 2008.

[27] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 319–326, New York, NY, USA, 2004. ACM.

[28] S. Yoo, Y. Yang, F. Lin, and I.-C. Moon. Mining social networks for personalized email prioritization. In *KDD '09*, pages 967–976, New York, NY, USA, 2009. ACM.