Detection and Extraction of Events from Emails

Shashank Senapaty
Department of Computer Science
Stanford University, Stanford CA
senapaty@cs.stanford.edu

December 12, 2008

Abstract

I build a system to detect emails that are informing the reader of an event and automatically extract structured information describing the event such as the title, date, time and venue of the event using various machine learning and natural language processing techniques. Such a system if run on an email client can be used to alert the user of events he/she may be interested in, and either automatically add the event to the user's calendar or facilitate one-click add since the system would have automatically extracted attributes describing the event.

1 Introduction

1.1 Motivation

Many of us get a large volume of email hitting our inbox especially from a variety of mailing lists that one may be subscribed to. We may be receiving emails on certain mailing lists about events that may interest us such as recruting events, talks, and other social events and it would be convenient if we could add the event with one click instead of opening up the calendar and manually entering the title, date, time and venue. Indeed, GMail already has such a feature where it suggests a one-click add to a user's calendar in certain cases (Figure 1). But it is observed that in many cases GMail does not provide such a suggestion even when the email contains event information and the user would like to add it to his/her calendar quickly. In the example shown in Figure 1, while GMail does make a suggestion to add to calendar it is unable to extract the date, time or venue in this particular case; as shown later the system presented here does accurately extract this information for this example. The goal of this project is to explore how well we can solve this problem and whether a robust reliable system can be built that in most cases will firstly detect that the email is about an event and secondly extract the correct information for the event.

1.2 Problem definition

This problem comprises of two parts. The first part is a classification problem where we decide whether an email contains information about an event or not. If the email is in fact an event email, the second part of the problems can be characterized as an information extraction task where we determine structured information of the event from unstructured text. In this system, I attempt to extract the following attributes: title, venue, date, start time, and end time if available.

There are some inherent amiguities in both parts. For the classification task, it may not necessarily be clear whether an email should get classified as an event email. For example, an email that informs of a deadline (such as for submitting an application) is not strictly an event that can be attended but still maybe something you want added to your calendar. I have considered such a case not to be an event and thus the system here attempts to classify this as a

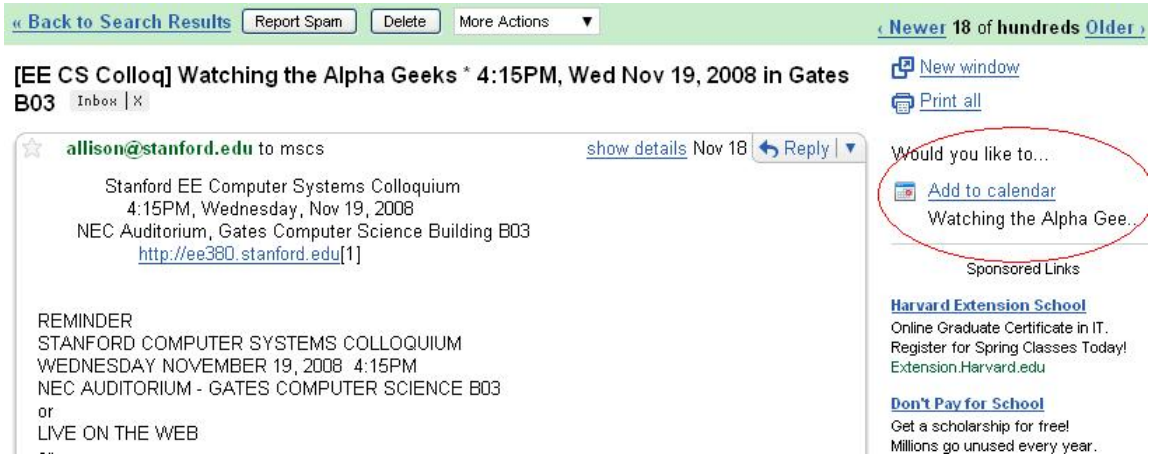

Figure 1: Gmail gives an option to add to calendar when it detects an event.

negative example (corresponding to non-events). In general, I have used my judgement in resolving such ambiguities by using the general narrow definition of an event as something that you would attend like a talk or a social gathering.

For the information extraction task, it is obviously upto a person's judgement what exactly qualifies as the title of an event. Therefore, in such cases I use an evaluation metric that reasonably captures whether the system is correct; for example, using exact string comparison would be completely unreasonable. I instead use a metric that captures the similarity between the strings but is more accomodating; details about the metrics used for evaluation are in section 3.4. For the time attribute of an event, however, the system must understand the semantics of the value and identify an exact time point so that the event can be added to the calendar appropriately; therefore, in this case the system is evaluated on the actual time value inferred. For example, "5pm", "5:00pm", "5:00 pm", "5:00" and "17:00 hrs" must all be inferred as the time 5:00 PM.

1.3 Challenges

As seen in the example in Figure 1, there are actually two dates in the email body; the system must, however, accurately predict which of the two dates is the actual date of the event. Similarly, an email can contain both start and end times and other irrrelevant times, and the system must accurately predict which is the start time and which is the end time of the event. For this it is necessary to rely on contextual clues and careful and thoughtful feature selection is necessary. Moreover, it can be quite hard to discern the venue of an event because in many cases there may not be contexual information aiding in detecting the venue; Figure 1 illustrates this also where the venue "NEC Auditorium, Gates Computer Science Building B03" is simply written alone and not in the context of a sentence. In general, this line may just say "Gates B03" and the system has the task of knowing that it is a venue.

1.4 Approach

Figure 2 shows the system design for the whole system. For the information extraction task, I used a Maximum Entropy Markov Model (MEMM), which combines a Maximum Entropy classifier ([3]) with a Viterbi decoder. This is a popular algorithm for performing information extraction in the field of natural language processing. However, since training data was not plentiful, this requires careful feature engi-

neering to avoid the sparseness problem. The algorithm is described in more detail in section 3.1.

For the event email classification task, I tried the Naive Bayes Multinomial Event model using a specialized tokenization of the input. This tokenization used special tokens like MONTH, TIME, DATE to replace months, times and dates respectively in addition to basic tokens like HTTPADDR, EMAILADDR, and NUMBER to replace urls, email addresses, and numbers. While this algorithm has certain shortcomings for the task at hand, particularly the fact that it doesn't capture the sequence structure of the text (such as if we used bigrams or trigrams), it still serves well as a baseline since it is an algorithm known to perform well for a text classification task. I later provided some suggestions on how performance on the event email classification task can be improved even though I did not get to implementing it myself.

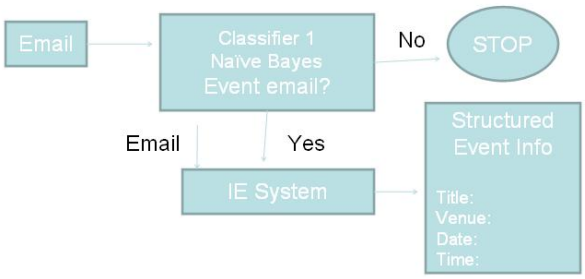

Figure 2: System design.

2 Dataset

The dataset was formed using emails sent to mscs@cs.stanford.edu mailing list. 180 emails were manually labelled as positive and negative examples corresponding to event emails and non-event emails. The dataset was divided into a training set of 140 examples and a test set of 40 examples.

Moreover, the MEMM approach used for information extraction (described in section 3) requires each token in the emails to be tagged with tags describing their role, i.e., the relevant parts of the email have to be tagged as corresponding to the TITLE, VENUE, DATE, START TIME, END TIME of the email. This was done for each positive example in the dataset using the Stanford JavaNLP[1] document annotation tool.

3 Information Extraction

Section 3.1 describes the MEMM model that is at the core of the information extraction system. Thereafter, section 3.2 discusses the overall system design for the information extraction task. Section 3.3 discusses the features used in the MEMM classifiers, section 3.4 discusses the evaluation metric for the information extraction task and finally section 3.5 discusses the implications of certain choices regarding system design and features.

3.1 The Model

The MEMM classifier assigns to each token in an email one of several labels/classes corresponding to the attributes of interest; the labels in this case are: TITLE, VENUE, DATE, START TIME, END TIME, OTHER. In this model, we therefore model the conditional probability of a token having one of these six labels given the features associated with the token. For each token in the email, features are computed based on the token itself, the neighboring tokens, and the label of the previous token. The conditional probability of a class for a particular token is modelled as a linear combination of the features combined via a softmax function. There is a weight associated with each feature-class pair and these are the parameters of the model. Formally, the conditional probability of a class for a token d and a set of features f_j is modelled as:

$$P(y = i|x; \theta) = \frac{exp({\theta_i}^T x)}{\sum_j exp({\theta_j}^T x)}$$

The objective function is the negative log likelihood of the data and the weights are learned to

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

minimize this objective function using gradient descent. Given a test email, instead of labelling each token to maximize conditional likelihood individually, a Viterbi decoder is used to label the tokens in the email so as to maximize the joint likelihood of the email.

3.2 System Design

Figure 3 shows the system design for the information extraction system. Two MEMM classifiers are used, one for the subject and one for the body of the email, because the subject and body have very different structure. Further motivation for this decision is in section 3.5. The two classifiers are trained on the subjects and bodies of the emails in the training set respectively. Given a new email, the two classifiers label the subject and body of the email. For extracting the value of an attribute, all candidates labelled with that particular label are considered, and one of the values is chosen based on the probabilites associated with each of them. However, if a candidate for an attribute is available from the subject this is always preferred over a candidate from the body since we expect the subject to have more of this important information.

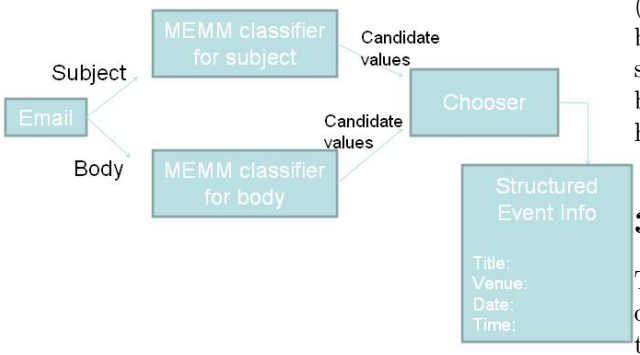

Figure 3: System design for the information extraction system.

3.3 Features

The current word, contexual features like the previous and next words, the label of the previous word, and orthographic features were used as features. A special feature that checked if the word represented a month was used. Specialized orthographic features were also used to dectect dates and times.

3.4 Evaluation Metric

For date, start time, and end time exact symantic value are inferred by the system and as such the extracted value is taken to be correct if it matches exactly the semantic value of the gold standard. For example, all of "5pm", "5:00pm", "5:00 pm" must be inferred as the time 17:00 hrs. For the title and venue, an exact string match is not reasonable because even humans cannot agree on what should be the exact title in many cases. Therefore, these values are marked correct based on a "recall-type measure" (R-measure) and "precision-type measure" (P-measure); the R-measure is defined as the fraction of words in the gold standard that are present in the extracted value and the P-measure is defined as the fraction of words in the extracted value that are present in the gold standard. If both R-measure and P-measure thresholds are met for a particular value, only then is it marked correct. The used R and P thresholds for title were (0.5, 0.5) and for venue it was (0.8, 0.8). It should be noted that the final algorithm used is not very sensitive to these thresholds (particularly for venue) because the precision of the extracted values are very high; this is discussed further in section 5.

3.5 Design and Feature Analysis

The decision to use two different MEMM classifier, one for subject and one for body, was made because the subject and body have very different structure and typically contain different attributes. The subject typically contains important information pertaining to the attributes we are interested in whereas the body contains a lot of irrelevant information and noise. The difference is clear when we compare the

features learnt for the subject and body classifier (tables 1 and 2).

Label	Feature	Description
TITLE	INITCAPS	Starts with capital letter
VENUE	WORD_Gates	Word is "Gates"
DATE	PREV_LABEL_DATE	Label of previous word is DATE

Table 1: Highly weighted features after training for Subject classifier.

Label	Feature	Description
TITLE	PREV_LABEL_TITLE	Label of previous word is TITLE.
TITLE	NEXTWORD_EE	Next word is "EE"
TITLE	INITCAPS	Starts with capital letter
VENUE	PREV_LABEL_VENUE	Label of previous word is VENUE.
VENUE	PREVWORD_in	Previous word is "in"
DATE	MONTH	Word represents a month. (e.g. 'Nov.')
DATE	PREV_WORD_,	Previous word is ","
START TIME	PREV_WORD_from	Previous word is "from".
END TIME	NEXTWORD_p.m	Next word is "p.m"

Table 2: Highly weighted features after training for Body classifier.

Note that contextual features turn out to be useful; for example, one of the most highly weighted features for VENUE in the body classifier is that the previous word is "in". The feature that detects months is a valuable feature for DATE. Also, using a Viterbi decoder instead of a greedy decoder to label allows us to take into account the previous label as a feature, which turns out to be useful in many cases as can be seen from tables 1 and 2.

4 Event Email Classification

The Naive Bayes Multinomial Event Model was implemented using the tokenization mentioned previously. This achieved a test set error of 7.5 percent. The error is as high as it is because this is a simple bag of words approach and does not capture the structure of the data like contextual information. However, for this application it is not necessary for this classification to be perfect. In particular, even if we are only mildly confident that the email contains an event, we can run the information extraction system on the email and show the one-click add option to the user. If the email does not contain an event the user simply ignores it but if it does indeed contain an event then we have served the purpose of the system since the user can enjoy one-click add.

Moreover, on exploring this issue more, it seems like it might be prudent to run the MEMM classifier on the email first. Using inputs from the output of the information extraction system, such as confidence in predictions etc., we can make a much better judgement of whether the email is about an event. This is likely to work much better since the MEMM classifier captures the true structure of the data. We can use the output of the MEMM classifiers as features for a logistic regression algorithm for example.

5 Results

The final results obtained are shown in table 3. It can be the seen that the precision is perfect for each attribute and this is desirable since we certainly do not want to extract wrong information for any attribute; rather the system should leave this attribute blank. Therefore, it is desirable that failures in the system hurt the recall numbers rather than the precision numbers and this is the case here. Two examples of the results obtained are shown in figures 4 and 5.

6 Conclusion

The system here is a fairly good system to extract various attributes of events from emails. With minor

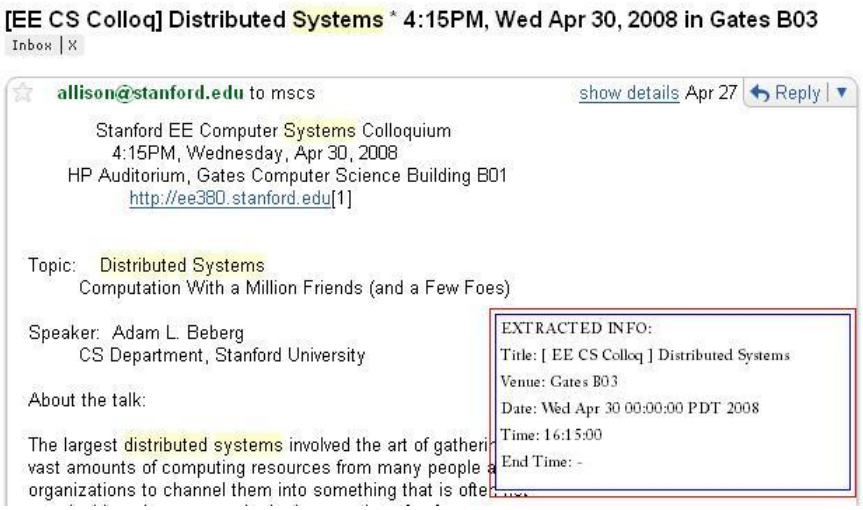

Figure 4: Example result.

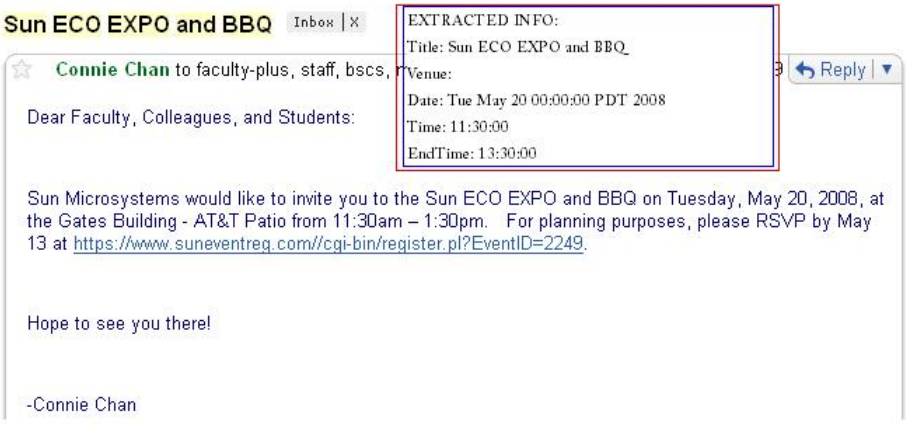

Figure 5: Example result.

Label	Recall	Precision	F1
TITLE	1.0	1.0	1.0
VENUE	0.41	1.0	0.58
DATE	0.67	1.0	0.8
START TIME	0.88	1.0	0.94
END TIME	0.62	1.0	0.77

Table 3: Recall, Precision and F1 using only body classifier

refinements, such as better tokenization, word stemming and such basic NLP techniques the performance on certain attributes can be increased. The recall for the venue is a bit low at 0.41. This is partly because in many cases there is not enough contextual information surrounding the venue as mentioned previously. In this case we must rely on having seen the word such as "Gates" or "Packard" before. This problem can be fixed by increasing the training size to a more representative set or by building a corpus of locations.

7 Acknowledgement

Some of the code for the implementation of the MEMM classifiers was taken from partial code provided for Stanford's CS 224n course. The JavaNLP annotation tool by the Stanford NLP group was useful in annotating the dataset.

8 References

1. Jurafsky, D. and Martin, J.H. 2008. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition. Second Edition. Prentice Hall.

2. Ion Muslea, 1999. Extraction Patterns for Information Extraction Tasks: A Survey

3. Adwait Ratnaparkhi. A Simple Introduction to Maximum Entropy Models for Natural Language Processing. Technical Report 97-08, Institute for Research in Cognitive Science, University of Pennsylvania.