Automated Extraction of Event Details from Text Snippets

Kavi Goel, Pei-Chin Wang

December 16, 2005

1 Introduction

We receive emails about events all the time. A message will typically include the title of the event, the date and time, the location, and sometimes a description. Unless the information is received as part of an integrated email/calendar system, this data must be entered into calendar software manually. Similarly, finding text on a web page about an event must be entered manually into a calendar. The extra time required to do this data entry discourages users from recording the event at all which reduces the usefulness of both the email and their calendar software.

For our CS 229 project, we would like to partially automate this data entry process. In our proposed tool, the user could select a block of text and choose "Add to Calendar" to automatically extract the event information from the text snippet. This information could then be piped into a calendar program. For example, given the block

COMPUTER SYSTEMS LABORATORY COLLOQUIUM
4:15PM, Wednesday, October 26, 2005
NEC Auditorium, Gates Computer Science Building B03
http://ee380.stanford.edu
Topic: The Reiser 4 Filesystem

The algorithm should be able to return the title, date, time, and location of the event. To address this issue, we design a modified HMM that incorporates context features.

2 Previous work

In our literature search, we found three previous works on extraction of meeting information from text documents. Dalli[1] applied a Named Entity Recognizer to first pre-process the data and then used an e-mail summarizer to extract attributes. However, we were not able to find the results of this strategy. Black and Ranjan[2] used RAPIER and a hand-coded pattern matcher. The most similar approach to ours was done by Almgren and Berglund[3]. In their approach, a hand-coded pattern and keyword matcher was used to identify obvious attributes. When no matching pattern was found, a Hidden Markov Model was used. The system produced an accuracy of 85% for date and time extraction, while title and location accuracy was approximately 50%.

3 Data Source

The scarcity of annotated data was found to be a difficulty in our project. We labeled personal emails to produce a training corpus of 64 snippets. The corpus included "formal" examples – semi-structured snippets in which formatting provided significant clues regarding the underlying states (like the example shown in the Introduction) – as well as "informal" natural language snippets in which there were generally fewer formatting clues about state transitions.

4 Algorithm Description

4.1 Hidden Markov Model

We use a modified Hidden Markov Model (see Figure 1) to identify the meeting attributes (title, date, time, and location) of the text snippet.

The hidden states of the HMM correspond to the attributes listed above with the addition of an "other" state for extraneous data. An observation for a given time slice consists of a set of features related to a specific word from the snippet. For each word, we consider the previous, current, and next word to get more contextual information. Features are assumed to be conditionally independent given an underlying state, so the overall emission probability is simply the product of emission probabilities for each feature. The full list of features is as follows:

- the current word token, as well as the previous and following words in the snippet
- the part-of-speech tags of each of these words

  We use the maximum entropy part-of-speech tagger by Adwait Ratnaparkhi.

- the named-entity tag of each of these words

  We use LingPipe, a suite of linguistic tools, as our named-entity tagger. The named-entity tagger tags words recognized as ORGANIZATION, PERSON, LOCATION, or PRONOUN.

- capitalization of the current word (capitalized or lowercase)

Typically, in snippets containing event information, a new line or a colon is indicative of a transition between states. In order to capture these state transition indicators, the probability of a transition is conditioned on both the previous state and on the observed formatting or punctuation symbol (blanks permitted).

4.2 Constraining the hidden state sequence

For the event extraction task, hidden states should not be repeated after they have already been identified once. For example, it does not make sense to have a state sequence: LOCATION DATE TIME LOCATION TITLE. A traditional HMM cannot capture this constraint since the next state depends only on the current state and not upon the entire sequence of previous states.

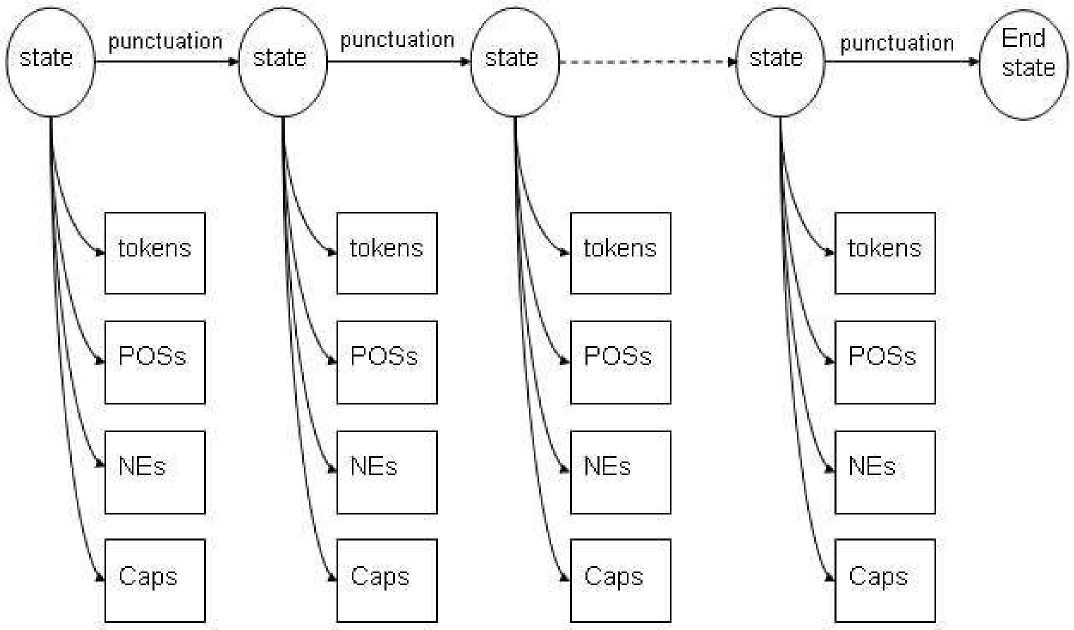

Figure 1: Modified HMM

In order to prevent the HMM from making predictions of this sort, we keep track of the states that have already been visited. When calculating the transition probabilities at a given time slice, any previously visited state is prohibited. The probabilities of the transitions to permitted states are not renormalized to sum to 1 because this would incorrectly inflate the probabilities of paths where a hidden state had been incorrectly chosen in the past.

In addition, we constrain the HMM to end in a special END state which allows the algorithm to incorporate knowledge that in a typical snippet some states are much more likely to be the last state seen than others.

4.3 Feature selection

In our HMM, We use 10 features to represent the observation in each time slice, in the hope that the POS tags and NE tags would be useful indicators to the HMM. To test this, we performed feature selection on the feature set by varying the add-n smoothing constants for each feature. Intuitively, note that increasing a smoothing constant causes all emission probabilities from that feature to become closer together. Given that the overall likelihood of an observation is the product of the output of each feature, a feature which outputs relatively uniform probabilities will have less impact on the overall emission probability than one that outputs widely varying probabilities. To do optimization, we used a greedy hill-climbing algorithm that iterates through all the smoothing constants and updates the smoothing constant if the performance based on our metric improved. Ultimately, no smoothing constants grew to a large value (optimal values ranged from 0.4 to 0.7), suggesting that all of the features contained useful information.

5 Results

5.1 Metrics used

We use three different metrics to measure the performance.

1. Number of correctly annotated words.
2. Number of perfect attributes, (i.e. a completely correct Title for a particular snippet)
3. Jaccard similarity. We use the formula

$$AvgJaccard = \frac{1}{n}\sum_{n}\frac{|A \cap B|}{|A \cup B|}$$

   to measure the overlapping range of A, the set of words that comprise an attribute, and B, the set of words that comprise the prediction for all examples n.

The number of correctly annotated words is an optimistic measurement of how well the HMM is doing. On the other hand, the number of perfect attributes gives no credit if the algorithm misses a Title by a single word. The Jaccard metric provides a balance between the two other metrics presented.

5.2 Performance of HMM

The HMM was trained and tested using 3-fold cross validation on the data set. Performance is as shown in Figures 2 and 3. Overall, none of the meeting attributes were extracted with sufficient reliability to be end-user ready. We noted that correctly extracting a title from natural language snippets is a particularly difficult task that might require more sophisticated semantic understanding techniques than are available with an HMM. However, there are some reasons to believe that our HMM strategy could still be effective for extraction of times, dates, and locations. In some examples, the algorithm correctly extracted essential information about the time and date even though it did not match the labeling of the "correct" answer. For example, "Saturday, November 25" (the output of the HMM) and "this Saturday, November 25" (test example labeling) are both correct answers from a practical standpoint even though a mismatch is marked as incorrect. A complete system would not only extract the time and date but convert them to canonical forms so that the resulting content could be checked for accuracy, eliminating the need to check the exact word selection.

Accuracy	Num of correct states	Num of perfect attributes	Jaccard
Title	0.527	0.296	0.401
Date	0.880	0.513	0.675
Time	0.883	0.598	0.750
Location	0.817	0.699	0.716

In some cases, there was evidence that insufficient training material had been provided. For example, the algorithm in one case marked [ONEDIGIT]pm (numbers are replaced with tokens during preprocessing) as a location even though, given a larger set of training data, this should have been easily identified as a time. More training data would also allow us to compare performance on formal (semi-structured) versus informal (natural language) test examples and to do more reliable feature selection.

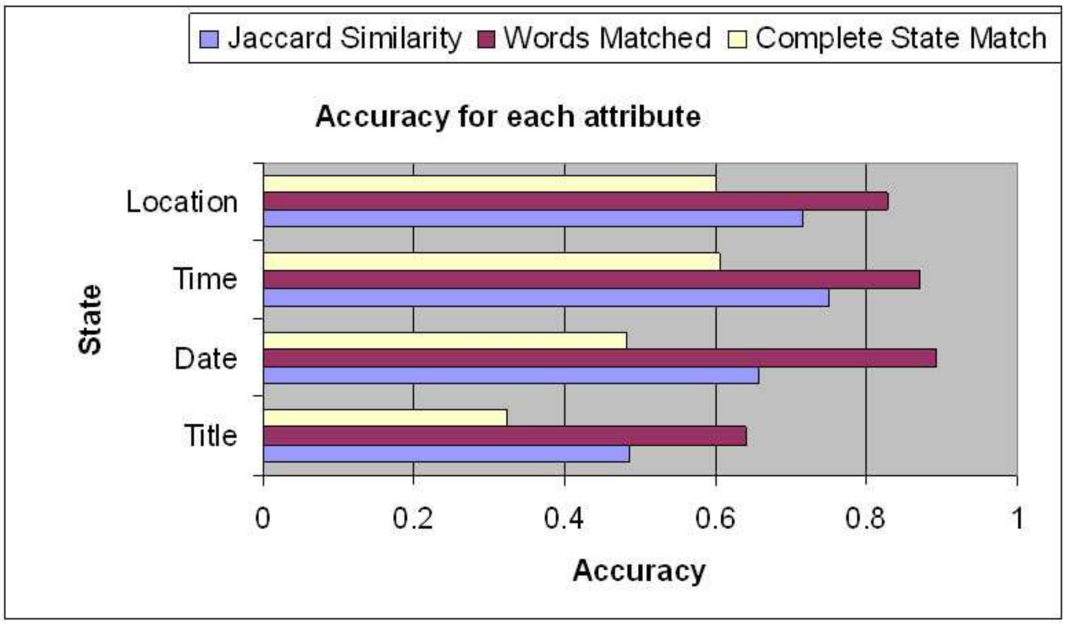

Figure 2: Performance of HMM

6 Conclusion

This paper has described a system that can perform event information extraction based on a flexible HMM-based learning algorithm. Performance was reasonable but not end-user ready. Although it is difficult to compare performance to that of other systems ([2] and [3]) since each algorithm was trained and tested on different data sets, it appears as if our algorithm extracts attributes with accuracy comparable to a hand-tuned pattern-matching system. However, an HMM-based solution provides a framework that may prove to be more robust with respect to unexpected test examples (i.e. misspellings or unusual conventions) than a pattern-matching-based system. Providing more training data and embedding the HMM in an end-to-end system would provide more realistic conditions in which to test the full potential of the algorithm.

7 Reference

1 Dalli, Angelo,“Automated Email Integration with Personal Information Management Applications” 2004.
2 Black , Julie and Ranjan, Nisheeth, “Automated Event Extraction from Email” 2004.
3 Almgren, Magnus and Berglund, Jenny, “Information Extraction of Seminar Information” 2000.