

Automated Event Extraction from Email

Julie A. Black
jblack@stanford.edu

Nisheeth Ranjan
nisheeth@stanford.edu

1 Introduction

On a daily basis, most email users will send and receive numerous emails of varying types thereby generating massive amounts of digital communication. At present, it is the user's responsibility to filter and archive all of her email data which, depending upon the average number of messages she receives, may be an extremely time consuming task. Often important information is not properly archived or noted and, as a result, lost in a sea of email communication.

Specifically, we believe that email often leaves event information (proposed and planned meetings, announced upcoming events, etc.) in the users inbox, far away from her daily calendar. We hypothesize that users often fail to add event information received in email format to their calendars because of the effort required to manually search the email for the specifics of the upcoming event – date, time, location, etc. – and the somewhat awkward mechanism by which data must be entered into calendar programs.

To mitigate the aforementioned problem, we propose an architecture for automated event extraction from incoming email messages. A successful system will properly classify messages that contain event information, attempt to perform information extraction to isolate the specifics of the event (date, time, location, title, and participants), present all of this information to the user for confirmation and correction, and, finally, automatically insert the extracted event into the users calendar.

In this paper, we will focus on the two middle components of this architecture. We present algorithms for isolating event emails from incoming messages and algorithms for extracting specific information about an event from an email. In the future, we hope to improve our algorithms as well as complete the pipeline described above, integrating this system into existing mail clients.

2 Related Work

The problem of event extraction from incoming email has been approached in the past by numerous research efforts. Here we will discuss other proposed solutions to the problem at hand.

At present, many email clients have been integrated with personal calendars. Perhaps the most commonly used hybrid application is Microsoft's email client duet Outlook and Entourage for the PC and Mac respectively. Although these email clients work to bridge the gap between an incoming email and the user's personal calendar, they do not provide any mechanism for automated event extraction. In other words, the user is still wholly responsible for manually determining which emails contain event information, manually extracting the specifics of the

event from these emails, and, using the data entry method supported by the calendar component, adding this event to their calendar. We view these systems as a step in the right direction as all of the personal data has become centralized, however, we believe that the system proposed here will provide the missing link in applications like Outlook and Entourage by automatically constructing the event entry for the user's calendar.

Outside of the commercial sector, numerous researchers have approached this problem with numerous algorithms. In 2000, Almgren and Berglund presented an approach for information extraction of seminar data [2]. Their architecture was similar to that proposed here, in that messages are first classified as seminar announcements before information extraction techniques are used to isolate specific information about the seminar such as date, location, speaker, etc. Our system broadens the scope of the event extraction problem and attempts to correctly extract meeting information not only from seminar messages, but also from less structured personal meeting proposals.¹

Additionally, at the University of Sheffield, Angelo Dalli has worked to integrate email with Personal Information Managers to provide a seamless experience for users [3]. In his work Dalli, relies upon four main components to extract all requisite calendar information from incoming email messages: (1) a named entity recognizer, (2) a threaded email filter, (3) an email signature filter, and (4) an email summarizer.

Although numerous additional attempts have been made to address the problem of event extraction from email, we believe that the three projects presented above highlight many of the commonly employed techniques.

3 Training and Testing Corpus

We created our own training and testing corpus by exporting messages from our own inboxes and manually tagging them into three categories: official event announcements, personal meetings, and other messages. Our final corpus was composed of 235 personal meeting emails, 652 official meeting emails, and 1772 other emails.

4 Email Types

For the purposes of this project, we have defined three email types as follows:

- ***Official meeting emails*** are all messages that contain event information clearly delimited from all other text in the email. Most frequently, seminar announcements are presented in this format and would fall into this category. A snippet of such an email is presented here:
-

¹ For the purposes of this paper, we use the terminology "Personal Meeting" to refer to any email message that contains a relatively unstructured meeting proposal. For example, "Let's do lunch at Bytes Cafe." would be considered such a personal meeting. Email types are described in detail in section 4.

“COMPUTER SYSTEMS LABORATORY COLLOQUIUM
4:15PM, Wednesday, June 02, 2004
NEC Auditorium, Gates Computer Science Building B03
[http://ee380.stanford.edu\[1\]](http://ee380.stanford.edu[1])
Topic: Automating CapCom....”

- **Personal meeting emails** are meeting proposals in which the specifics of the meeting are presented within the body of the email itself. We term such emails as personal meetings as this format is most commonly used for friends and acquaintances to propose meetings. An example of such an email is presented here:

“hey i also wanted to invite you to the final software faire for cs194. It's where we will be showing off our senior project. It's next wed from like 12 - 4 i think or something.”

- **Other emails** are all incoming messages that we deem to be non-event related.

5 System Framework

A diagram that describes the architecture of our information extraction system is shown in Figure 1 below. It can be viewed as a pipeline of processes that takes raw email as input, determines whether the email is event related, and, if it is, performs information extraction on it. Each step of the pipeline is discussed in more detail below.

5.1 Pre-processing

The raw email data is exported to a single text file where personal, official, and other emails are demarcated by XML tags. This single email file is then processed by a Perl script

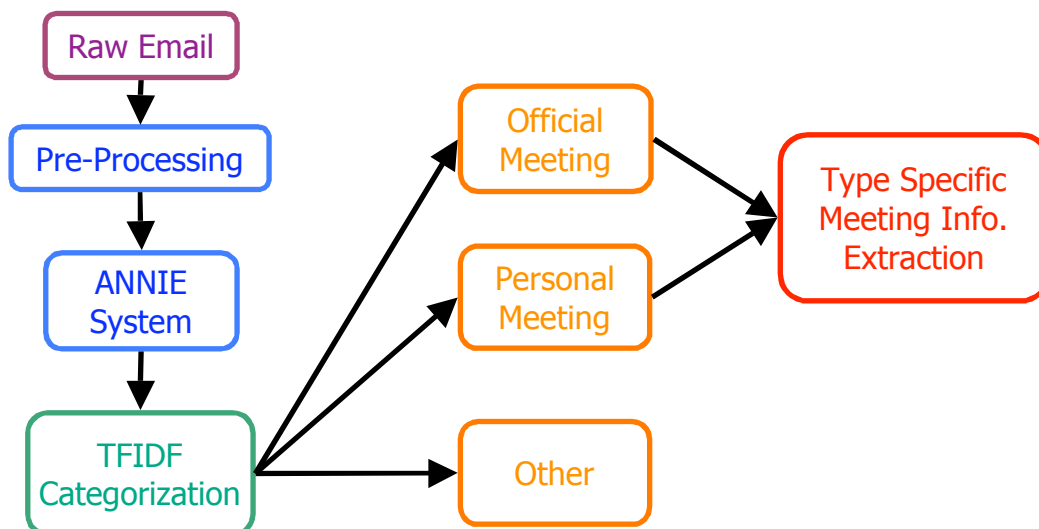


Figure 1. A schematic diagram of the architecture for automated event extraction from incoming email messages proposed in this paper.

(pre-process.pl) and separated out into three separate files containing personal meeting emails, official meeting emails, and other emails respectively. The email headers and body are demarcated with XML tags, <header> and <msgbody> and the MIME attachments present in the input email file are stripped.

All three output files are passed through another Perl script (strip-quoted.pl) that strips all quoted text from the emails. We do this because we only want to find event information in the real body of the email message, not in the entire thread of the email conversation contained within the email.

Next, all the output files are passed through a Perl script (add-meta-data.pl) that demarcates the From, Subject, To, and CC email headers by XML tags (<from>, <subject>, <to>, and <cc> respectively).

5.2 ANNIE

Developed at the University of Sheffield, ANNIE – a Nearly New Information Extraction System – exists inside the larger GATE infrastructure designed for “developing and deploying software components that process human language” [4]. For the present project, we have used ANNIE as a named entity recognizer after the pre-processing step. Although ANNIE can provide many different types of annotations useful for many different applications, we rely upon ANNIE to provide tags of people, locations, dates, parts of speech, and sentence boundaries. These tags appear in the outputted body of the raw email as XML tags wrapping the recognized text. Later, in our own modules we rely upon these tags for additional information about the email.

Although more detailed information can be found in the paper referenced above, ANNIE provides a tokenizer, a gazetteer, a sentence splitter, a part of speech tagger, a semantic tagger, an orthographic coreferencer, and a pronominal coreferencer. These components can be used independently of one another to provide different annotations required at different points in the system developed here. All of the ANNIE components are accessible through GATE’s graphical user interface in which raw documents can be imported into corpuses for processing by the ANNIE system. Although we have experimented with a stand-alone version of ANNIE that can be run from the command-line, we were unsuccessful at extracting all required information with this system. In the future, we hope to tweak this system, in order that it provides us with all requisite information.

The ANNIE pipeline that we used for our system was composed of the tokenizer, the gazetteer, the sentence splitter, the part of speech tagger, and the named entity transducer. We found that ANNIE was particularly good at tagging both explicitly formatted dates like “April 12th, 2004” as well as more implicit date words and phrases like “tomorrow”, “next week”, etc. ANNIE uses a modified version of the Brill POS tagger which also performed very well based on a random, manual evaluation of its output for a few emails.

5.3 TF-IDF-based Categorization

In order to categorize different emails into the three categories described in Section 4, we use a similarity measure based on TF-IDF (Term Frequency – Inverse Document Frequency). In

addition to this similarity measure, we define numerous domain specific heuristics with hand-tuned weights. This is described in more detail below.

5.3.1 Pre-processing ANNIE tags

Because, at this point in our system, we are working on email text that has been tagged by the ANNIE named entity recognizer, we must, before using the TF-IDF categorization, parse the XML into a format usable by the TF-IDF classification algorithm (using the `pre-process-test.pl` Perl script). Because the TF-IDF algorithm relies upon word frequencies in order to determine the similarity of a test email message to the training messages of a given folder, we must remove all unnecessary ANNIE tags so as not to skew the frequency counts. Similarly, we choose to compact all text tagged by ANNIE (using the `merge-annie-tags.pl` Perl script) as either people, locations, or dates containing numbers into specialized tags in order to denote that a person, location, or date was present within the given email without skewing the frequency counts by maintaining the specific person, location, or date. Once this pre-processing step has been completed we remove all remaining, un-merged ANNIE tags from the email text.

5.3.2 TF-IDF

In order to classify email messages into the three desired folders – official meetings, personal meetings, and all other messages – we rely upon a TF-IDF-based similarity measure as proposed by Almgren and Berglund. We selected this similarity measure after experimenting with an implementation similar to the TF-IDF classification approach defined in Manning and Schütze[5] on page 543. It turned out that our tweaked version of the Manning and Schütze algorithm did not perform as well as Almgren and Berglund’s version and we ended up using the latter implementation. We achieved reasonably good results for categorizing emails into the three desired folder classes, though not as good as Almgren and Berglund. We think that this was because we were trying to classify a broader range of messages and were, therefore, dealing with a more complex classification problem. The TF-IDF measure we used is defined in the paragraphs below (reproduced from Almgren and Berglund).

Let M be a the message we are trying to classify represented by a word-frequency vector $F(M)$ where each component $F(M, w)$ represents the frequency of word w in message M . From this definition, we compute the weight for a given folder, F , as follows.

Let $FF(F, w)$ be the number of times that word w occurs in any message belonging to folder F divided by the total number of words in folder F and let $FF(A, w)$ be the number of times that word w occurs in any training message divided by the total number of words across all training messages. Thus we can compute the *term frequency* as follows.

$$TF(F, w) = \frac{FF(F, w)}{FF(A, w)}$$

Using this term frequency, and the *document frequency* $DF(w)$ which we define to be the number of folders in which the word w occurs at least once divided by the total number of folders, we can compute the weight, $W(F, w)$ as follows.

$$W(F, w) = \frac{TF(F, w)}{DF(w)^2}$$

Using these weights, we can compute a similarity measure, $SIM(M, F)$ for the given message M to a given folder F as follows:

$$SIM(M, F) = \frac{\sum_{w \in M} F(M, w)W(F, w)}{\min(\sum_{w \in M} F(M, w), \sum_{w \in M} W(F, w))}$$

Thus, when attempting to categorize and email, we select the folder that produces the largest similarity score.

5.3.3 Domain Specific Features

To complement the general classification provided by the TF-IDF similarity measure, we used three main hand-weighted domain specific features. With additional time, we would have used a boosting algorithm to learn these feature weights. We discuss the features in greater detail below.

1. The presence of the string *list* in the message's to field. We hypothesize that official messages are more commonly sent to mailing-lists, many of which contain *list* in their email address (for example cs224n-spr0304-all@lists.Stanford.edu).
2. The presence of a date in the subject of the email message. In our own experience, it is common that most event announcements (either personal or official) contain a date in the subject field. We rely upon ANNIE's date tags to determine this feature.
3. The presence of both a date and a location in the subject of the email message. Once again, in our own experience, we have found that if there is a date and location present in the subject line, the email message is extremely likely to contain an event announcement. We rely upon ANNIE's location tags to determine this feature.

5.4 Information Extraction

In order to extract information from the categorized email messages, we first attempted to use the RAPIER algorithm proposed by Califf and Mooney [6]. RAPIER uses pairs of sample documents and filled templates to induce pattern match rules that directly extract fillers for the slots in the template. It employs a bottom up learning algorithm which incorporates techniques from several inductive logic programming systems to learn patterns that include constraints on the words and part of speech tags present in the filler and surrounding text.

In parallel with our tests of the RAPIER algorithm, we wrote our own Perl script (ie.pl) that uses hand coded rules and logic to extract event information from ANNIE tagged output.

The methods we used in these two approaches are described below.

5.4.1 RAPIER (Robust Automated Production of IE Rules)

We first ported the RAPIER system so that it compiled with GCC 3.3 running on Mac OSX 10.3.3. RAPIER's learning program learns rules that use constraints on words and on part of speech tags. It trains on a training set where each training example is a collection of three files: (a) email.orig, the original email, (b) email_out, the original email after passing it through a sentence splitter and part of speech tagger, (c) email.template, a filled event template containing the date, time, location, and title of the event contained in the email.

ANNIE's sentence splitter and POS tagger export their output as verbose XML files where the POS tags were contained in attributes on <token> tags around each text token created by the ANNIE tokenizer. RAPIER expected email_out to be in the format of "word/tag" where 'tag' is the POS tag of 'word'. Instead of writing a script to convert ANNIE's XML output into a format consumable by RAPIER, we decided to use the maximum entropy based sentence splitter (MXTERMINATOR, see [7]) and POS tagger (MXPOST, see [8]) written by Advait Ratnaparkhi. This tagger's output was of the form "word_tag" which could be converted into RAPIER consumable format via a single "tr" command.

We manually created 160 filled templates for the official meeting emails and used 130 of them in our training set and 30 of them in our test set. The results are described in Section 6.

5.4.2 Perl based Pattern Matching

In addition to the RAPIER system discussed briefly above, we constructed our own information extraction pattern matcher which relies upon the information returned by ANNIE to complement the raw email data when extracting the specifics of a meeting or event from an incoming email message. To simplify the problem, we focus on extracting only the date, location and title of the event using a few key features. We describe these patterns and features here.

5.4.2.1 "Gimmies"

When attempting to isolate the date and location of an event from an official announcement, we can get reasonably good performance on many messages by searching for data preceded by certain key phrases: "DATE:", "when:", "where:", etc. As a result, before attempting any more elaborate patterns, we first check the message body for a small set of common prefixes to the desired data fields. To simplify the code, and to provide us with more flexibility in determining what the list of prefixes should be, we have separated these prefix lists into individual files: location-preceders and date-preceders. If we successfully find a match for both prefix types – a match for a date prefix and a match for a location prefix – in the message body, we then assert that the message is a highly-structured official announcement and as a result return the Subject of the email as the title of the event. This decision is motivated mainly by our observation of numerous highly structured official event announcements.

5.4.2.2 Finding the Date

Because we are trying to produce complete meeting annotations, we prefer to return, if possible, complete dates in which both the month and day are specified. Thus we begin our search for the correct date of the email by considering all of the dates tagged by ANNIE that adhere to this desired structure. For each of these dates, we consider 5 words (this is a modifiable window size) on either side of the date in question in an attempt to extract the location and title. The decision to search for the location and title close to the date was motivated by our own observation of many personal messages in which we found that most meeting and event announcements and proposals adhere to this structure. The actual isolation of locations and titles will be discussed in more detail in the following two sections. In the event that we were unsuccessful at locating a date that adhered to our optimal date structure (in which both the month and day were specified), we expanded our search to consider all dates returned by ANNIE that were not solely years. We chose to omit dates containing only year information because most email messages are sent within a year of the actual event time, and thus isolating this information would not be useful for the user.

5.4.2.3 Finding the Location

As mentioned above, we attempt to isolate the location of a given event by considering the text within a small window (5 words) on either side of dates as annotated by ANNIE. In order to flag those words that most frequently are used to describe locations – i.e. building, room, park, etc. – we define a list of common location key-words (lockeywords). We then iterate over the tokens defined by our small window and append them to our running location string if ANNIE has annotated them to be nouns. In addition we allow determinants to be present in our location only if they occur between two nouns. Because we must select from the two locations found (one on either side of the date) we define a weighting scheme in which we provide a confidence measure asserting that the returned string is in fact a location. We boost the confidence measure significantly whenever we append a word to the location that belonged to our previously defined key-word list. Additionally, we significantly boost the confidence measure in the event that either the word *at* or the symbol @ occurs in the string in question.

5.4.2.4 Finding the Title

As in isolating the location, to find the title, we consider tokens within a small window on either side of each date tagged by ANNIE. In order to isolate the title, we consider the capitalization of the words, and select all nouns that either begin with a capital letter, contain all capital letters, or contain a mixture of capital and lowercase letters. As was the case, for the location strings, we must also define a confidence measure for the returned titles in order that our system properly select the correct title. Thus we weight words that have been tagged “Unknown” by ANNIE highly – assuming that these tokens are more likely to be names and thus are more likely to define the title of the event. Finally, we weight those words that we have defined to be location key words, exceptionally low, in order that we do not mistakenly return a location as a title.

6 Results

As has been described above, we have divided our system into two main components, (1) the TF-IDF-based categorization mechanism which attempts to classify incoming messages into one of three categories: official meetings, personal meetings, and all other messages, and (2) the information extraction mechanism that isolates the specifics of the event from messages categorized as meetings. We discuss the performance results of both these components below.

6.1 TF-IDF-based Categorization

Using the system described above, we have been able to achieve reasonably good performance from the TF-IDF based classification algorithm as shown in Table 1. We trained our algorithm on a subset of our original data: 259 official meeting emails, 113 personal meeting emails, and 764 other emails. Our test set consisted of 101 unseen and uncategorized email messages which were pre-processed in order to remove MIME attachments, quoted text, and all unnecessary headers (anything other than subject, to, from, and date). 9 of these 101 messages were manually classified to be official meetings, 30 were manually classified to be personal meetings, and the remaining 62 messages were manually classified as other messages.

When originally implemented, our TF-IDF algorithm performed poorly. In reviewing the results, we found that implementing a smoothing algorithm would help mitigate the problem of data sparsity often present in natural language processing problems. Thus, we implemented an absolutely discounted smoothing algorithm. In addition, it was clear that differences in capitalization were increasing the data sparsity, and thus we opted to convert all words in training and test set to lowercase before attempting to classify the incoming email message. Finally, we were able to improve our results slightly by removing all words from the training and test input that did not contain at least one letter character.

On the test set described above, we were able to obtain a precision of 52.0% and a recall of 66.67% for the correct classification of meeting emails. Although the TF-IDF classifier classified emails into three classes (personal meeting, official meeting, and other), the precision and recall numbers mentioned earlier measure the performance of classifying email into two classes (meetings and other). We chose to collapse the two meeting types into a single one because our information extraction approaches (RAPIER and Perl based pattern matching) that operate on the TF-IDF classified emails did not end up requiring the meeting type information. Although these results are reasonable, our classification algorithm was not as successful as we would have hoped. We think that our lower success rate (as compared to Almgren and Berglund) was because our training set, which included meeting messages of all kinds and other messages, was more diverse and harder to classify than the training set used by Almgren and Berglund which consisted only of seminar announcements and other messages.

We opted to rely upon the TF-IDF formula used by Almgren and Berglund as a result of its performance on our test data. We found that our implementation of the traditional TF-IDF

	Almgren-Berglund-based	Manning- Schütze -based
Precision	52.00%	39.80%
Recall	66.67%	100.00%

Table 1. Results from the two TF-IDF classifiers we tried.

algorithm as presented in Manning and Schütze, had precision and recall measures of 39.80% and 100% respectively. The 100% recall measure is misleading because the classifier simply returned the same class (official meeting) for all test messages. We chose to use the TF-IDF classifier proposed by Almgren and Berglund classifier because of the more reasonable precision and recall numbers achieved by it.

6.2 RAPIER Information Extraction

We trained RAPIER on a training set of 30 files first and then tested it on 10 files to see how it performed. Its performance was terrible, around 1-5% precision and recall, even though it did exceptionally well when tested on the training data (98% precision, 80% recall). We hypothesized that this was a result of over-fitting to the 30 file training set and decided to train it on a larger training set of a 100 messages. After training on the larger set, we tested on a set of 30 messages and achieved a precision of 68.5% and recall of 29.3%. We looked at the final patterns generated by RAPIER to fill the fields of our event template and found that they were very specific to the text in the training set. We hypothesize that with additional time, we could have created an even larger training set and further improved RAPIER's precision and recall measures.

We had thought that we would look at the patterns generated by RAPIER and base our Perl based information extractor on them. But, when we examined the filler patterns generated by RAPIER, we found them to be very specific to the training data and decided to forego using them. Instead, we used our own heuristics and domain specific knowledge generated after eyeballing a lot of the meeting emails to create our information extraction Perl script (ie.pl).

6.3 Perl-Based Pattern Matching

We ran our Perl-based Pattern Matcher described above on 50 test emails that we had previously created event templates for when we used them for training the RAPIER system. As anticipated, we found that we were more successful at isolating the date from event emails and less successful at correctly isolating location and title information. We present the results below in Table 2. We hypothesize that these results are due to the inherent ambiguity and variability in the structure of meeting emails. Since we did not focus only on seminar announcements, our set of meeting related emails contained a very wide variety of meeting proposal structures. Despite the low numbers we feel that the approach of using ANNIE's date tags to isolate dates in an email and then look for locations and titles around the date is a promising one because it fits the structure of many personal and official meetings we saw in our corpus. We had many ideas on how we could improve our information extraction system which we couldn't implement due to lack of time. Some of these ideas are mentioned below.

	Percent correctly extracted
Date	62.00 %
Location	24.00 %
Title	26.00 %

Table 2. Results from the Perl-Based Pattern Matcher for the three desired data fields: date, location, and title.

- Use semantic information from WORDNET to complement the POS and named entity tags from ANNIE to help determine location and title in the surrounding context around dates.
- Be more intelligent in determining the context window around the date. For example, rather than hardcoding a length of 5 words before and after the date, search until the beginning and end of the sentence that the date appears in; or, if the date is on its own line, search the previous and next sentences.
- Flag the information in the subject as important. Often subject contains part of the title and location of the meeting. If a word in the surrounding context of the date matches one in the subject, boost the confidence of picking up that word as a location or title.

7 Conclusion

Although we weren't able to achieve the performance necessary for deploying our information extraction system as a plugin to an email client, we think that the architecture we have developed is sound and given more time we can obtain high performance required for practical use. We are excited by the knowledge we have gained while building this system and the ideas that we have generated for future improvements. We now have a better appreciation of the depth and complexity of the information extraction problem on general, conversational text. We look forward to make more headway against this problem in the future.

Works Cited

1. Microsoft Outlook. 2003. <<http://office.microsoft.com/home/office.aspx?assetid=FX01085793>>
2. Almgren, Magnus and Jenny Berglund. "Information Extraction of Seminar Information." 2000. <<http://nlp.stanford.edu/courses/cs224n/2000/berglund/report.pdf>>
3. Dalli, Angelo. "Automated Email Integration with Personal Information Management Applications." 2004. <<http://www.cs.bham.ac.uk/~mgl/cluk/papers/dalli.pdf>>
4. Cunningham, Hamish, et al. "Developing Language Processing Components with GATE (a User Guide)." 2003. <<http://www.gate.ac.uk/sale/tao/>>
5. Manning, Christopher D. and Hinrich Schütze. Foundations of Statistical Natural Language Processing. Cambridge, MA: The MIT Press. 2003.
6. Califf, Mary Elaine and Raymond J. Mooney. "Bottom-Up Relational Learning of Pattern Matching Rules for Information Extraction." 2003.
7. Jeffrey C. Reynar and Adwait Ratnaparkhi. "A Maximum Entropy Approach to Identifying Sentence Boundaries". In Proceedings of the Fifth Conference on Applied Natural Language Processing, March 31-April 3, 1997. Washington, D.C.
8. Adwait Ratnaparkhi. "A Maximum Entropy Part of Speech Tagger". In Proceedings of the Empirical Methods in Natural Language Processing Conference, May 17-18, 1996. University of Pennsylvania