CS498AD Final Project: Extracting Publication Entries from Homepages

William Lee (wwlee1@uiuc.edu)

12/15/2004

Abstract

In a project such as DBLife, name entity extraction has played an major role in a people-centric data integration model. Before the full realization of semantic web, however, monitoring and following the trends in any scattered web-based communities have proved to be a difficult problem.

DBLife faces many challenges, one problem is the accurate extraction of publication entries. This paper focuses on extracting and matching publication entries listed on researcher homepages. In particular, we focuses on leveraging structural properties of the HTML document and an effective border alignment heuristics in a Hidden Markov model (HMM) in order to improve the exact match performance. A classification approach using SNoW (Sparse Network of Winnow) has also been tried for comparison. Overall, our system has the best exact match accuracy of 29.2% with 45.9% recall. Our contribution also includes an extensive corpus with manually tagged publication entries that can be used in future experiments in order to improve the results.

1 Introduction

There are many challenges to overcome before the full realization of semantic web. For the most basic level, semantic web requires a set of identifying and describing tags that give semantic meanings to text. The tagged entries then are served as the basis for web ontology construction and autonomous, intelligent agents [2, 1, 3].

There are many approaches that describe the authoring of such tagged information. Doan et. al [2] describes a way to alleviate this mundane process by a special tagging tool. The tool is designed to be as unobtrusive and backward-compatible as possible. Ideally, the bulk of the work done by machine learning and other techniques place a much lighter load on the user. The integration of such tool in the document authoring environment should help to kickstart the semantic web movement.

Furthermore, the DBLife project provides database researchers an integrated view about the ongoing activities of database researcher. The source for the information comes from monitoring researcher home pages daily and apply data integration tool on such untagged data.

As all researchers would probably know, making aware of the kind of publication out there provides great insight to the general research direction within the field. Services such as DBLP and Citeseer have shown to be useful when finding information about a particular researcher and his or her publications. However, in many cases, it can be difficult to associate a particular publication with a researcher.

This implies that it is essential to accurately extract publication entries from researcher homepages. One can think of this as a similar problem than the phrase extraction problem in NLP

(Natural Language Processing) [4]. However, there remains several challenges that are unique in the publication extraction problem.

One fundamental problem is the varied ways to represent a publication entry in a homepage. The publication entries are typically presented in the following forms:

- With <a> link:

```
<LI><B>A. Balmin, Y. Papakonstantinou, V. Vianu &#8220;
<a href="http://www.db.ucsd.edu/publications/IncrValTODS.pdf">
Incremental Validation of XML Documents</a>
```

- With <a> link at the end:

```
<li>Obradovic, Z, Peng, K, Vucetic, S., Radivojac, P., Brown,
C., and Dunker, A.K. and Dunker, A.K. (in press) Predicting
Intrinsic Disorder from Amino Acid Sequence, Proteins: Structure,
Function and Genetics. <a href="obradovic_casp5.ps">(PS)
```

- Without <a> link:

```
<li>Smith, D.K. Radivojac, P.Obradovic, Z., Dunker,A.K. and Zhu,
G.(2003) &quot;Improved Amino Acid Flexibility Parameters,&quot;
Protein Science, vol. 12, pp. 1060-1072.<br>
```

However, not all publication entries follow the general forms. Furthermore, there are many ambiguous entries that look like publication entries. For example, talks, tutorials, and other lecture titles can look very similar to publication entries:

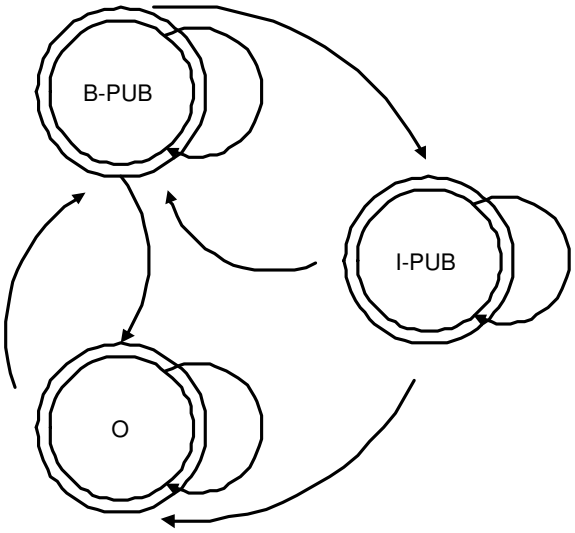
```
...
<P>
<SPAN> Recent Talks and Tutorials</SPAN>
</P>
<UL>
<LI>
<A href="http://www.db.ucsd.edu/people/yannis/XQueryTutorial.htm">
XQuery Tutorial</a>: Introduction to XQuery and its basic deployment
options in data integration and application integration scenarios.
<LI>
<A href="http://www.db.ucsd.edu/people/yannis/JeffSymposium.pdf">
XML View-Based Mediation: TSIMMIS and Beyond (pdf)</A>
```

One can note that the HTML tag structure is extremely similar to a publication entry. This make it hard for a learner to learn the border that can segment each publication entry. In this paper, talk and class titles are not tagged as publication entries.

Also, machine generated HTML files usually introduce a lot of noise. For example:

```
<p
style='margin-left:.5in;text-align:justify;text-justify:inter-ideograph;
text-indent:-.25in;mso-list:l4 level1 lfo3;tab-stops:list
.5in'><![if !supportLists]><span
style='font-size:10.0pt;mso-fareast-font-family:"Times New
Roman"'><span style='mso-list:Ignore'>15.<span
style='font:7.0pt "Times New
Roman"'>    
</span></span></span><![endif]><a
href="http://www.cs.uiuc.edu/~hanj/pdf/sigmod04_maids.pdf">
MAIDS: Mining Alarming Incidents from Data
Streams</a>&#8221;, (system demonstration), Proc.
2004 ACM-SIGMOD Int.  Conf. Management of Data (SIGMOD'04),
Paris, France, June 2004.<o:p></o:p></span></p>
```

Figure 1: Basic BIO model



The noise introduced in the document can have an effect on the extraction performance, since many feature extraction scheme depend on surrounding tokens.

The Winnow algorithm and its subsequent derivations, SNoW, or a Sparse Network of Winnow , has also shown to be quite effective in natural language processing [5, 6]. In this paper, we will evaluate the performance difference between the HMM approach and the Winnow approach. The main focus is to see how HTML structural information can affect the overall extraction performance.

2 Methods

2.1 Document Representation and Problem Definition

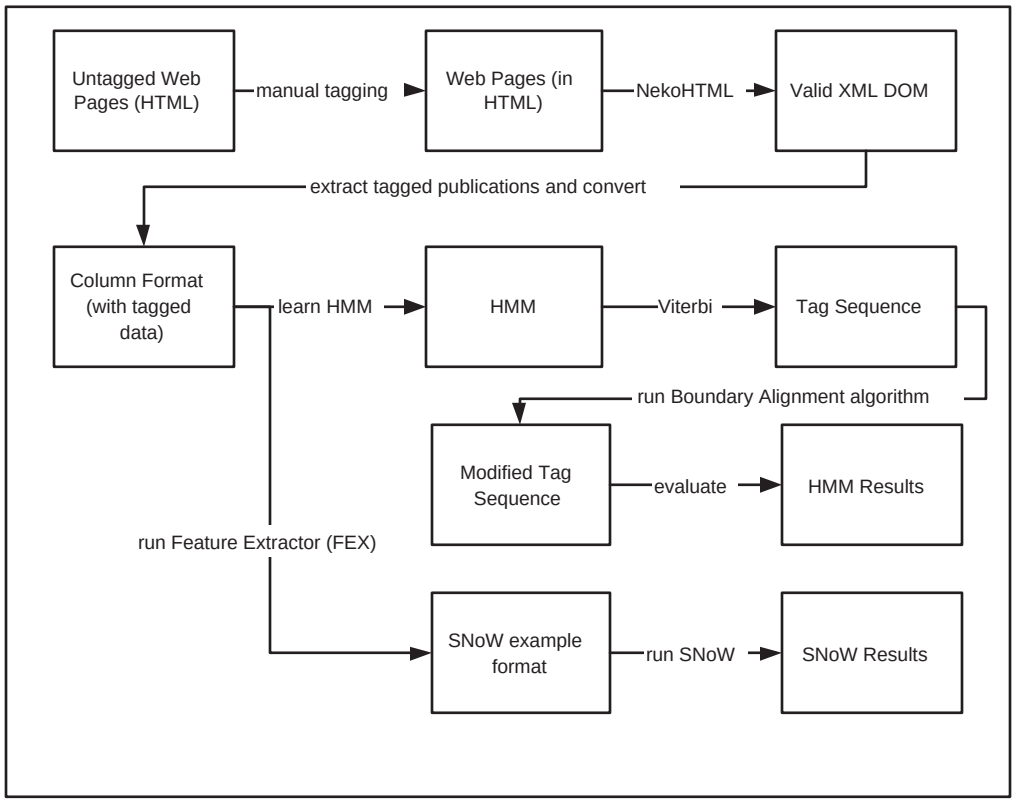
We first need to define how we model the documents in the experiment. We first represent each document d as a stream of tokens $\{w_1, ..., w_n\}$. Each token can be a word or a special tag that indicates the start and end of an HTML tag. Since it is well-known that the original design of HTML makes it harder for the parser to parse its content in a structured way, we first use the NekoHtml toolkit available at http://www.apache.org/~andyc/neko/doc/index.html to transform each HTML files into a valid XML DOM (Document Object Model). We then process the DOM and token each HTML tag and word into a column format that can be read easily. Note that the attributes for each tag is ignored in the corpus.

We define the phrase extraction as a labeling problem. The goal of our algorithm is to label each token with a tag. We use the simple BIO approach in [4], for each token. There are three states: Begin Publication (B-PUB), Inside Publication (I-PUB), and Other (O). Figure 1 shows the possible state transitions for each state.

Several methods have been tried in this project:

- Classification-based approach using FEX (Feature EXtraction) and SNoW (Sparse Network of Winnow)
- HMM on raw text data

Figure 2: System Flow



- HMM on HTML tag data and text data
- HMM on HTML tag data, text data, and with boundary alignment

Figure 2 demonstrates the basic system flow in the experiment setup.

2.2 Classification-based Approach

We can transform the publication extraction problem into a simple classification problem. The high level goal is to train a classifier that can assign a label (B-PUB, I-PUB, or O) to each token in the document d.

We use SNoW as our training algorithm. Essentially, SNoW[5] is a linear classifier that can decides whether an example represented by a boolean vector $X = [x_1, ..., x_n]$ has the label y given that the following condition is true:

$$\sum_{i \in A} w_{t,i} x_i \quad \geq \quad \theta_t$$

. The constant θ_t is the threshold for the target node t. In our case, each of the BIO state is a label (target) node t. A is the set of active features for the example X.

During classification, the goal is to find the optimal target t^* given an example X:

$$t^*(X) \quad = \quad argmax_{t \in T} \sigma(\theta_t, \Omega(X))$$

where $\Omega(X) = \sum_{i \in A} w_{t,i} x_i$ and $\sigma(\theta_t, \Omega(X))$ is a softmax function such that $\sigma(\theta_t, \Omega(X)) = \frac{1}{1-e^{\theta_t - \Omega(X)}}$.

Note that one can not represent the transition constraints in HMM, but we can use more features than what Markov assumption can take into account.

2.2.1 Extracting Features

FEX (Feature EXtractor) can automatically extract boolean features from the list of examples. An example in our case is centered on each token in the document. Here are the features for the example that we extract for each token:

- w[-2,2] - generate feature for any word that's within 2 words before and 2 words after the current word
- scoloc(w,w) [-2, 2] - generates the sparse colocation of any two words within the [-2, 2] window
- w|r [-2,2] - generates the feature that detects the presence of the word or the role (where the current word is within the [-2,2] window, the role in our case is a special tag that indicates whether a particular word is within a <a> tag with the value the href attribute ending in either "ps", "pdf", "gz", and "tgz".
- scoloc(w|r, w|r) [-2,2] - generates the sparse colocation version of w|r

One can see the exact definition in the FEX manual. We have experiment with 3 settings in our evaluation. The results are given in Section 4.1.

Setting	w[-2,-2]	scoloc(w,w) [-2,2]	w|r [-2,2]	scoloc(w|r, w|r) [-2, 2]	Total Features
base	X				14293
coloc	X	X			279219
full			X	X	310731

2.3 HMM (Hidden Markov Model)

As our goal is to evaluate how the structural information in the HTML document can help improve our overall result, one approach is to use HMM to find the "hidden" sequence of tags given a stream of "output" including the HTML tags. Ideally, HMM has the advantage of leveraging sequential information. Since we represent the HTML tags as tokens in the text stream as well, HMM should yield interesting results.

Given B-PUB, I-PUB, and O as states, the problem formulation become:

Find sequence of states $S = \{s_1, s_2, ..., s_n\}$ that maximizes the probability given the observation sequence $O = \{o_1, ..., o_n\}$:

$$S^* = argmax_{S \in all\ possible\ sequences} P(S, O)$$

Given that HMM follows the Markovian assumptions

$$\begin{aligned} P(s_t | s_{t-1}, s_{t-2}, ..., s_1, o_{t-1}, o_{t-2}, ..., o_1) &= P(s_t | s_{t-1}) \\ P(o_t | s_t, s_{t-1}, ..., s_1, o_{t-1}, o_{t-2}, ..., o_1) &= P(o_t | s_t) \end{aligned}$$

, we can write $P(S, O)$ as:

$$P(S, O) = P(s_1) \prod_{t=2}^{|S|} P(s_t | s_{t-1}) P(o_t | s_t)$$

2.3.1 Viterbi

We use Viterbi, a dynamic programming method, to find the most likely hidden state sequence given the output of the sequence.

In the simplest sense, for all possible states in the HMM S and at each output o_t at time t, the update rule for Viterbi is

$$\delta_t(s) = \max_{s' \in S} \delta_{t-1}(s') P(s|s') P_t(o_t|s)$$

, where $\delta_t(s)$ is the score for best sequence at time t that ends at state s. Given such update rule, all we need to calculate is $P(s|s')$ and $P_t(o_t|s)$. We can do so through counting from the training example.

It is also important to note that since the output o_t in our case only contains tokens that are used in the training corpus. Therefore, smoothing is also important since we want to ensure that we can generalize to the test data. In our case, for any output that we have not seen in the training corpus, we assume a very small weight for $P(o|s) = 0.00001$.

2.3.2 Boundary Alignment

One problem with HMM is that although it can find the tag correct for a subsequence, the border is often fuzzy and does not exactly match the entire publication entry. Here, we propose a simple algorithm to combat this problem.

- Given a tagged entry that contains the starting word of index i in the corpus. Let the length of tagged entry be k.
- Given a window size of n , if $\exists j$ within the interval $[i-n, ..., i-1]$ where $token(j) \in SB$ where SB is a set of starting tags, then we move the start of the phrase to the largest $j+1$
- Similarly, if $\exists m$ within the interval $[i+k-1, ..., i+k-1+n]$ such that $token(m) \in EB$, where EB is a set of ending tags, we extend the end of the phrase to the smallest $m-1$.
- In our case, we tested $SB = \{<$ `a` $>, <$ `i` $>, <$ `font` $>\}$, $EB = \{<$ /`a` $>, <$ /`i` $><$ /`font` $>\}$, and a window size of 5.

3 Constructing the Training and Testing Corpus

Since there is no existing corpus for tagged publications within a researcher's homepage, we have to construct a new corpus for evaluation. One of our contribution to this project is this training and testing corpus. We believe the corpus can be used in the future to improve the performance for similar information extraction problems.

3.1 Researcher Homepages

The main corpus used in this project is derived from the file set originally used in the DBLife project. There are 565 HTML files downloaded from 20 researchers' home pages. The distribution of the 565 files is:

```
Num Name
  2 Michael J. Franklin
```

```
  4 Gio Wiederhold
  4 Jayavel Shanmugasundaram
  5 Luis Gravano
  6 Jignesh M. Patel
  7 Johannes Gehrke
  9 Christos Faloutsos
  9 Inderjit S. Dhillon
 11 ChengXiang Zhai
 12 Rajeev Motwani
 15 Jennifer Widom
 15 Longin Jan Latecki
 17 Zoran Obradovic
 22 Yannis Papakonstantinou
 23 Jiawei Han
 24 Cindy Chen
 25 Kajal Claypool
 36 Dan Suciu
121 Alon Y. Halevy
198 Subbarao Kambhampati
```

where "Num" denotes the number of HTML pages from a particular researcher homepage.

With HTML tags, the corpus consists of 517673 tokens. Without the tags, there are 475459 tokens. There are 1824 publication entries tagged and 1342 unique entries.

The publication entries are sparse within the corpus. Most of the researchers put their publication entries in a long HTML page. Within that page, the publication entries may be very dense. Some publication entries are duplicated across two or more pages. This typically happens when a researcher put a entry once in the resume and once again in a web page with title "Publications".

Note that we do not remove the pages that do not contain publication entries. We believe that this can better simulate the real-world scenario, where one do not know which page may contain publication entries.

Each of the 565 files are inspected manually and the publication entries are tagged with a special tag. The special tags are then extracted and the entire corpus is later converted to a column format that can be used by FEX.

4 Results

For our experiment, we calculate the accuracy and recall as:

$$
\begin{aligned}
Accuracy_{exact} &= \frac{|correct\,exact\,prediction|}{|exact\,prediction|} \\
Recall_{exact} &= \frac{|entries\,that\,matches\,prediction\,exactly|}{|total\,publication\,entries|} \\
Accuracy_{token} &= \frac{|correct\,label\,predictions|}{|predictions|} \\
Recall_{token} &= \frac{|correct\,B-PUB\,and\,I-PUB\,predictions|}{|labels\,with\,B-PUB\,or\,I-PUB|}
\end{aligned}
$$

The $Accuracy_{exact}$ indicates the accuracy for an exact publication match. This implies that the system tags an publication entry exactly. The $Recall_{exact}$ is a metric to measure how many

Figure 3: SNoW Results

Bayesian	Exact			Token		
	Accuracy	Recall	F1	Accuracy	Recall	F1
base	0.010	0.006	0.007	0.965	0.368	0.532
coloc	0.022	0.035	0.027	0.961	0.490	0.650
full	0.037	0.081	0.050	0.959	0.568	0.728

Perceptron	Exact			Token		
	Accuracy	Recall	F1	Accuracy	Recall	F1
base	0	0	0	0.974	0.031	0.061
coloc	0	0	0	0.973	0.106	0.192
full	0.164	0.106	0.129	0.959	0.435	0.599

Winnow	Exact			Token		
	Accuracy	Recall	F1	Accuracy	Recall	F1
base	0	0	0	0.974	0.067	0.126
coloc	0.001	0.005	0.001	0.949	0.320	0.479
full	0.001	0.005	0.002	0.953	0.332	0.493

publication entries can be uncovered by our system. The $Accuracy_{token}$ shows how many B and I tags that our system has gotten correctly. It treats each token as an example and does not care whether it is a partial match or not. Lastly, the $Recall_{token}$ is recall for the B and I tags. We also calculate the F1 values by:

$$F_{1\,exact} = \frac{2Accuracy_{exact}Recall_{exact}}{Accuracy_{exact} + Recall_{exact}}$$
$$F_{1\,token} = \frac{2Accuracy_{token}Recall_{token}}{Accuracy_{token} + Recall_{token}}$$

4.1 SNoW Results

Figure 3 shows the results for SNoW. We run the three versions of feature extraction for each algorithm (Bayesian, Perceptron, and Winnow). We note that using just SNoW in general does not yield satisfactory result for the exact match. This is probably due to the fact that the classification-base approach does not enforce the state transition property as described in Figure 1.

There are many possible ways to improve the results, one way is to use inference with classifier in order to force HMM-like inference with any kind of classifier. Punyakanok[4] has shown this method is quite effective on the phrase extraction task in Natural Language Processing.

4.2 HMM Results

Figure 4 shows the results for the HMM experiments. The "notag" run uses just the text data with no tag information. As we can see, it has very poor accuracy and recall. Adding the HTML tag information improves the recall by 7 percent.

The "dlink" run, which is not shown in the graph, looks into the "href" attribute for the <a> tag and see if a the href ends with either "ps", "pdf","gz", or "tgz". If so, our preprocessing would create a special tag in the token stream to indicate so. Surprisingly, it does not seem to help performance. The drop in performance may be attribute to the noise that it introduces into the training set, since the HMM will have both the <a> and special <a> tag to be used for training. Also, ambiguity such as talks can also have an <a> tag with the href value that ends in one of those extensions.

The rest of the runs are using the border alignment algorithm with different sets of border tags. For example, the "tag + a + font" indicates the use of the border alignment algorithm that uses the <a> and <font> tag as the border sets. The "skipa" is an heuristics used if the start of a publication entry lies on the <a> tag. In such case, it will skip the <a> tag, assign it to be of label "O" (other), and assign the next token as "B-PUB".

Figure 5 shows the accuracy and recall for the token accuracy. The accuracy is high since the publication entries are sparse, and most of the tokens are assigned with the "O" tag. One can also see a significant improvement when the border alignment algorithm is used in the "tag + a" case. This indicates that although HMM can find publication entries approximately, the border is often not aligned correctly in order to match them exactly.

5 Conclusion and Future Work

The classification approach such as FEX and SNoW has been tried in our experiments with not much success. There are many possible reasons:

- Lack of enforcement of the inference rules such as HMM's state transitions
- Lack of meaningful features that can distinguish a publication entry
- Unlike NLP tasks like noun phrase extraction, publications are generally very sparse with few repeating words and surrounding words that can give meaningful features.
- Extracting publication is difficult due to
  - hard-to-find phrase boundary
  - noise in HTML and markup data
  - ambiguity with other types of entries (such as talk title)

We can see, however, in the HMM case, the structural information with the boundary alignment heuristic can indeed help performance. This indicates a strong need to combine the two approaches.

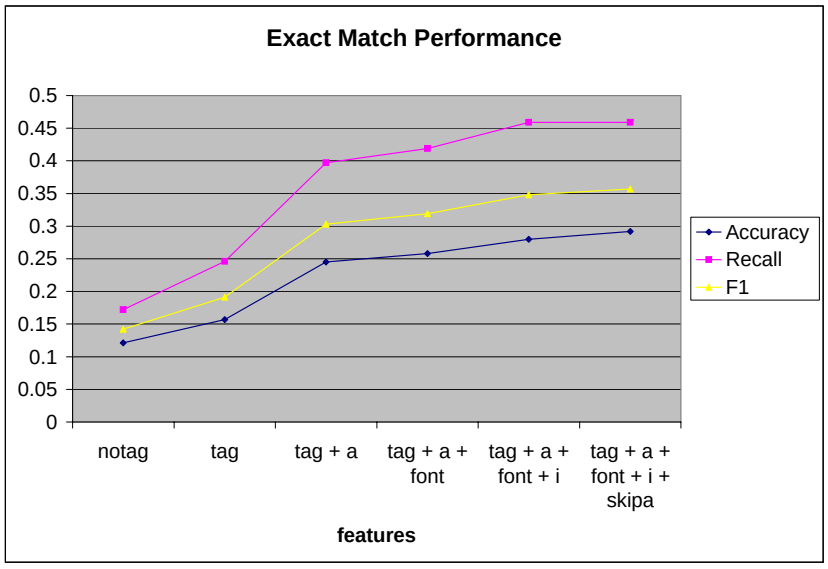
One possible way to improve on the performance is the use of inference with classifier. Simple HMM only utilize $P(s_t|s_{t-1})$ and the output probability $P(o_t|s_t)$. It is desirable to incorporate more features into the model by utilizing more features in order to calculate $P(o_t|s_t)$ in a more sophisticated way.

For example, one can change the Viterbi update rule according to [4]:

$$\begin{aligned}\delta_t(s) &= \max_{s' \in S} \delta_{t-1}(s') P(s|s') P_t(o_t|s) \\ &= \max_{s' \in S} \delta_{t-1}(s') P(s|s') \frac{P(s|o_t) P_t(o_t)}{P_t(s)}\end{aligned}$$

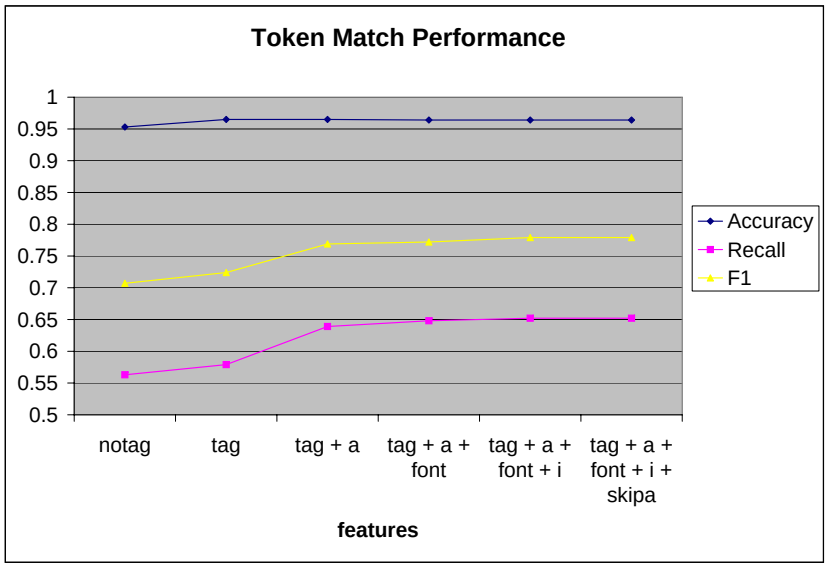
An assumption is made in [4] that $P_t(o_t)$ can be reduced to a constant based on time t and $P_t(s)$ can be ignore in the Viterbi algorithm. Furthermore, $P(s|o_t)$ can be approximated by the weights for each state given in a classifier such as SNoW.

Figure 4: HMM Exact Match Performance



Exact Match	Accuracy	Recall	F1
notag	0.121	0.172	0.142
tag	0.157	0.246	0.191
tag + dlink	0.152	0.237	0.185
tag + a	0.245	0.397	0.303
tag + a +font	0.258	0.419	0.319
tag + a + font + i	0.280	0.459	0.348
tag + a + font + i + skipa	0.292	0.459	0.357

Figure 5: Token Match Performance



Token Match	Accuracy	Recall	F1
notag	0.953	0.563	0.707
tag	0.965	0.579	0.724
tag + dlink	0.965	0.590	0.732
tag + a	0.964	0.639	0.769
tag + a +font	0.964	0.648	0.772
tag + a + font + i	0.964	0.652	0.779
tag + a + font + i + skipa	0.964	0.652	0.779

Inferencing with classification is a novel way to incorporate the results into a model like HMM or PMM (Projection-based Markov Model) [4]. It should be interesting to try out the state-of-the-art phrase extraction model in NLP and apply it to the publication extraction problem.

References

[1] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. 2001.

[2] AnHai Doan, Jayant Madhavan, Robin Dhamankar, Pedro Domingos, and Alon Halevy. Learning to match ontologies on the semantic web.

[3] Alon Halevy, Oren Etzioni, AnHai Doan, Zachary Ives, jayant Madhavan, Luke McDowell, and Igor Tatarinov. Crossing the structure chasm.

[4] Vasin Punyakanok and Dan Roth. Inference with classifiers: The phrase identification problem.

[5] Dan Roth. Learning to resolve natural language ambiguities: a unified approach. In *Proceedings of AAAI-98, 15th Conference of the American Association for Artificial Intelligence*, pages 806–813, Madison, US, 1998. AAAI Press, Menlo Park, US.

[6] Libin Shen and Aravind K. Joshi. A snow based supertagger with application to np chunking.