Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence

Joint Extraction and Labeling via Graph Propagation for Dictionary Construction

Doo Soon Kim, Kunal Verma, and Peter Z. Yeh
Accenture Technology Labs
50 West San Fernando Street
San Jose, CA, 95113

Abstract

In this paper, we present an approach that jointly infers the boundaries of tokens and their labels to construct dictionaries for Information Extraction. Our approach for joint-inference is based on graph propagation, and extends it in two novel ways. First, we extend the graph representation to capture ambiguities that occur during the token extraction phase. Second, we modify the labeling phase (i.e., label propagation) to utilize this new representation, allowing evidence from labeling to be used for token extraction. Our evaluation shows these extensions (and hence our approach) significantly improve the performance of the outcome dictionaries over pipeline-based approaches by preventing aggressive commitment. Our evaluation also shows that our extensions over a base graph-propagation framework improve the precision without hurting the recall.

Introduction

Text contains vast amounts of information, which if systematically and effectively extracted can provide numerous benefits. For example, extracting information such as locality, items to be procured, etc. from tenders can help companies identify new sales opportunities and increase revenue. Similarly, extracting information such as products and their attributes from online listing sites (e.g., EBay) can help enterprises discover and understand new market trends.

Many solutions – especially commercial ones such as IBM Content Analytics and SAS Text Miner – have recently become available to automate the extraction of information from text, and have seen increased adoption. These solutions all require dictionaries that define the surface forms of the information to be extracted (e.g., Company Names : {*Apple*, *Samsung*, ...}). However, the creation of these dictionaries is a manual process, which is costly and error-prone.

Numerous approaches have been proposed to automate the creation of dictionaries (Kim, Verma, and Yeh 2012) (Whitney and Sarkar 2012) (Riloff and Jones 1999) (Collins and Singer 1999). These approaches typically employ a pipeline architecture of first extracting candidate tokens from a given corpus, and then labeling these tokens with the appropriate type/category using semi-supervised methods such as co-training (Riloff and Jones 1999) (Collins and Singer 1999) or self-training (Yarowsky 1995) (Whitney and Sarkar 2012).

This pipeline architecture, however, has a significant disadvantage. In this architecture, each step must commit to a solution before proceeding to the next (even if there is insufficient information to do so). This aggressive commitment can significantly hurt performance. For example, the extraction step must determine the boundaries of tokens based primarily on syntactic information, which is often insufficient. Once the decision is made, it cannot be reverted. Hence, this architecture will perform poorly if the documents are difficult to parse syntactically. Moreover, the labeling step often provides useful information for token extraction. For example, consider extracting tokens denoting a product type from informal listings. An extraction pattern such as *"sell <x> for <number> dollars"* learned during the labeling step could benefit token extraction by indicating the exact boundaries of the extracted tokens.

In this paper, we present a semi-supervised approach to jointly perform extraction and labeling for dictionary construction, based on a graph propagation framework (Whitney and Sarkar 2012) (Subramanya, Petrov, and Pereira 2010) [1]. Our approach extends this framework in two novel ways. First, our approach extends the graph representation to include mutually exclusive edges that indicate candidate tokens which cannot be simultaneously valid. These edges capture ambiguities that occur during token extraction. We also provide an effective way of constructing this graph from a given corpus. Next, we modified the labeling step (i.e., label propagation) to utilize this new representation, allowing evidence from labeling to be used for token extraction and preventing aggressive commitment.

We evaluate our approach in two domains – medical tenders and informal advertisements. We chose these domains because traditional syntactic analyses (e.g., off-the-shelf parsers) often fail on these types of text. Our evalu-

[1] Informally, the nodes in a graph propagation framework are the tokens to be labeled and the edges are the similarities between the nodes. An initial subset of nodes are provided the correct labels (i.e., seed nodes), and the goal is to determine the label distribution for all nodes such that similar nodes have similar label distributions.

ation shows that our approach outperforms pipeline-based approaches, and significantly improves precision without hurting recall compared to the "vanilla" graph propagation framework. These results confirm the importance of jointly performing extraction and labeling and the benefits of our extensions (and hence our contribution).

Related Work

We present related works from three areas – dictionary construction, non-pipeline architecture, and graph propagation.

Dictionary Construction. Automated approaches for dictionary construction often use a pipeline architecture where the tokens are extracted based on syntactic analysis, without utilizing information from the labeling step. For example, to identify named entities, (Whitney and Sarkar 2012) (Collins and Singer 1999) extract consecutive proper nouns within a noun phrase where the last word is the head of the noun phrase. (Riloff and Jones 1999) extracts the noun phrases from the parsed documents. (Kim, Verma, and Yeh 2012) uses co-occurrence statistic to identify multiword tokens before passing them to the labeling phase. (Talukdar et al. 2006) extracts the longest span of multiwords that satisfy pre-defined regular expressions.

Approaches based on sequence-modelling (e.g., (Putthividhya and Hu 2011)) could naturally combine extraction and labeling, but cannot exploit mutually exclusive relationships among the conflicting tokens.

Non-Pipeline Architecture. Pipeline architectures can hurt the accuracy of text interpretation due to its aggressive commitment at each step. One approach to address this problem is to maintain multiple candidate interpretations at each step. This allows downstream steps to resolve the ambiguities as more evidence becomes available. For example, (Sutton and McCallum 2005) maintains multiple parse trees to jointly perform parsing and semantic role labeling. (Kim, Barker, and Porter 2010) maintains multiple candidate interpretations compactly using a packed representation to jointly resolve different ambiguities. Our method, which takes this approach, generates all candidate tokens during the token extraction step and uses label propagation to determine the correct extractions during labeling. Another approach to the aggressive commitment problem is to jointly model multiple tasks in a unified framework (e.g., (Poon and Domingos 2007)).

Graph Propagation. Graph propagation can exploit rich structures in the data (represented as graphs), and have been shown to be effective in various applications such as recommendation systems (Baluja et al. 2008), named entity recognization (Whitney and Sarkar 2012) and part-of-speech tagging (Subramanya, Petrov, and Pereira 2010). To our knowledge, we are the first to use graph propagation to jointly perform extraction and labeling for dictionary construction to prevent aggressive commitment. In particular, we extend the original framework in two novel ways: 1) we extend the graph representation to capture ambiguities that occur during token extraction, and 2) we modify the labeling step (i.e., label propagation) to utilize this new representation, allowing evidence from labeling to be used for token extraction.

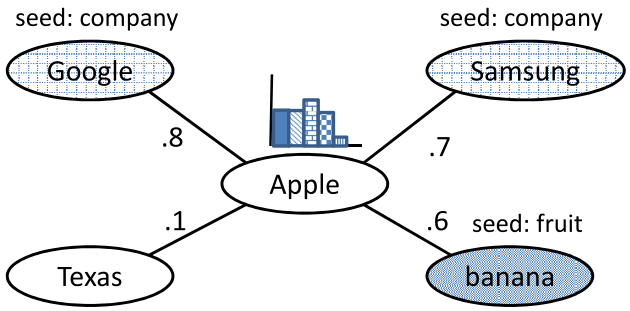

Figure 1: Example graph: the seed nodes are shaded.

Preliminary: Graph Propagation

We provide additional details on the graph propagation method presented in (Whitney and Sarkar 2012) [2], which extends (Subramanya, Petrov, and Pereira 2010), because it is the basis for our approach.

A graph, $G = (V, E)$, is defined in the following way. V is a set of nodes, v_i, that denotes a token to be labeled. E is a set of undirected edges, $e_{ij} = (v_i, v_j, w_{ij})$, that connects two nodes, v_i and v_j. w_{ij} is a positive real number, indicating the strength of the similarity between v_i and v_j. Given a set of labels, $q_u(y)$ denotes the score of assigning a label, y, to the node, u. q_u is normalized for each node (i.e., $\sum_y q_u(y) = 1$). Seed nodes are assigned a correct label a priori. For each seed node, a point distribution is defined: $r_u(y) = 1$ if the seed label is y and 0 otherwise. [3] Fig. 1 shows an example graph.

The goal of label propagation is to determine the label distribution across all nodes (i.e., $q_u(y)$) such that similar nodes (i.e., nodes connected by an edge with a high weight) have similar label distributions. To do this, (Subramanya, Petrov, and Pereira 2010) iteratively updates the label distribution using Eq. 1, which propagates the labels from the seed nodes to the non-seed ones.

$$\begin{aligned} q_u^{(m)}(y) &= \frac{\gamma_u(y)}{\kappa_u} \text{ where } \kappa_u \text{ is a normalizing constant} \\ \gamma_u(y) &= r_u(y)\delta(u \in V_l) \\ &+ \mu \sum_{v \in N(u)} w_{uv} q_v^{(m-1)}(y) + \nu U(y) \end{aligned} \quad (1)$$

where μ and ν are parameters and m is the iteration index. $q_u^{(m)}(y)$ is the normalized score of y on the node, u, at the m-th iteration. $\gamma_u(y)$ is the unnormalized score of y and consists of three terms. The first term considers the seed label. It returns 1 if u is a seed node for the label, y, and 0 otherwise. V_l is the set of the seed nodes and $\delta(x)$ is an indicator function that returns 1 if x is true and 0 otherwise.

[2] They present two types of label propagation – based on co-training (Collins and Singer 1999) and self-training (Yarowsky 1995). Our method is based on the self-training version but the general idea can be applied to the co- training version.

[3] We distinguish q_u from r_u. The label distribution, q_u, may be inconsistent with the initial seed labeling, r_u.

$r_u(y)$ is a point distribution that returns 1 if y is the seed label of u and 0 otherwise. The second term ensures that the label distribution becomes close to the distributions of similar neighboring nodes. If w_{uv} is high, γ_u becomes similar to $q_v^{(m-1)}$. $N(x)$ is the set of the neighboring nodes of x and w_{uv} is the weight of the edge connecting u and v. The final term is a regularizer based on the uniform distribution, U – that is, $U(y)$ is 1 over the number of labels.

The iteration stops when one of two conditions are met – either the change in $q_u^{(m)}$ is minimal or m exceeds a predefined number. Once the iteration is finished, nodes with the highest-scored labels (over a given threshold) are selected as new instances of those labels.

Our Approach

Our approach extends graph propagation to jointly infer the boundaries of tokens and their labels for dictionary construction. We first describe how the base graph representation (see previous section) is extended to capture ambiguities that occur during token extraction. We then present an effective way to construct this extended representation from a given corpus. Finally, we describe how the label propagation step is modified to utilize this new representation, allowing evidence from labeling to be used for token extraction.

Extended Graph Representation

The graph representation in existing graph propagation frameworks cannot capture ambiguities that occur during token extraction (i.e., tokens whose boundaries cannot be determined with high confidence). Hence, existing frameworks require an aggressive commitment to the boundaries of tokens (even if the syntactic evidence is insufficient) before label propagation can be performed. This aggressive commitment hurts the accuracy of the outcome dictionaries.

To capture this ambiguity, our approach introduces a new edge type, Mutually Exclusive(ME) edge, that captures mutually exclusive relationships among ambiguous candidate tokens. This ME edge allows alternative candidates to be generated when the boundaries of a token cannot be determined with high confidence (see example in Fig. 2). These edges (and hence the alternative candidates) are later resolved during label propagation, which jointly infers the correctly-segmented tokens and their labels.

Formally, a ME edge, $e'_{ij} = (v_i, v_j, w'_{ij})$, indicates that if either v_i or v_j is a correctly segmented token, then the other one cannot be [4]. w'_{ij}, a positive real number, indicates how strongly the mutually exclusive relationship should be enforced. [5] We also extend the label set to include a new label, ϕ, which indicates that a node should not be assigned any other labels because it is incorrectly segmented.

[4] Notice that the other direction does not hold. Even if v_i turns out to be incorrectly segmented, v_j is not necessarily correct, because both could be incorrectly segmented.

[5] The weight w'_{ij} allows a ME relationship to be a soft constraint rather than a hard constraint. If w'_{ij} for a ME edge is low, then our approach may not enfore the constraint, and hence allow competing candidate tokens to be extracted. For example, if a ME edge connects "blanket" and "teething blanket" (as in Fig 2), but the weight of the edge is low, then both "blanket" and "teething blanket" may be extracted.

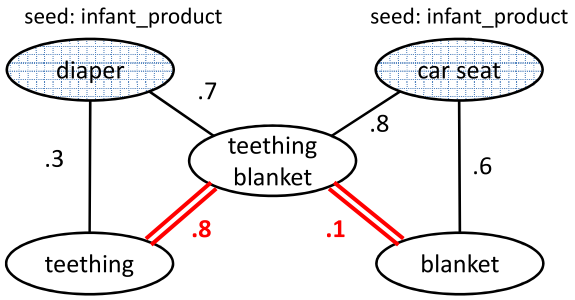

Figure 2: Extended graph: In this example, *teething blanket* (correct segmentation) is connected to *teething* and *blanket* (alternative segmentations) via ME edges (indicated by double edges) with weights 0.8 and 0.1 respectively.

Graph Construction

Our approach constructs the extended graph representation (described above) from a corpus via the following steps.

Step 1: Generate Candidate Tokens. Our approach generates candidate tokens (i.e., the nodes) from a given corpus using pre-defined rules like those shown in Table 1. Multiple rules can be applied to the same span of text to extract alternative candidate tokens. For example, the rule, <J|VBG>*<N|POS>*<N>, can be applied multiple times to the sentence, *"I am selling a teething blanket"*, to extract *teething*(<N>), *blanket*(<N>), and *teething blanket*(<N><N>). The resulting token, *teething blanket*, is mutually exclusive with *teething* and *blanket* because they overlap in the same span of text.

Our approach records $f_{me}(w_1, w_2)$, the number of instances (i.e., same span of text) within a given corpus where two candidate tokens w_1 and w_2 overlap. Our approach uses f_{me} later in *Step 4* to construct ME edges.

Step 2: Filter Candidates. The previous step can generate many wrong candidate tokens, making the graph unnecessarily large. To reduce the graph size, we filter these candidates using the rank-ratio measure presented in (Deane 2005), which measures the likelihood of a multiword token being correctly segmented. We chose this measure because it is widely applicable across multiple domains due to its non-parametric nature.

Specifically, the likelihood that a multiword token, $c_1 c_2 ... c_n$, is correctly segmented is defined as:

$$\sqrt[n]{RR(c_1, [c_2, ..., c_n]) * ... * RR(c_n, [c_1, ..., c_{n-1}])} \quad (2)$$

$$RR(word, context) = \frac{ER(word, context)}{AR(word, context)}.$$

$ER(word, context)$ is the expected rank – i.e., the rank of $context$ (based on the corpus frequency of $context$) among all contexts associated with $word$. $AR(word, context)$ is the actual rank – i.e., the rank of $context$ based on the frequency with which $word$ appears in the $context$.

Tenders	1. <**w**>**+** where <w> is a capitalized word or a preposition e.g., *Sodium Chloride, Department for Labor*
	2. <**w**>**+ ("certificate"|"license")** e.g., *drug manufacturing license*
	3. **("certificate"|"license") "of"** <**w**>**+** e.g., *Certificate of a Chamber of Commerce*
	4. <**w**>**+ ("vaccine"|"kit")** e.g., *pregnancy test kit, hepatitis B vaccine*
	5. **("office"|"department") "of"** <**w**>**+** e.g., *office of the postmaster general*
Ads	1. <**NP**><**NP**|**CC**|**POS**>***** (for named entities) e.g., *Fisher Price, Children's Place, Mellissa and Doug*
	2. <**J**|**VBG**>*****<**N**|**POS**>*****<**N**> (for noun phrases) e.g., *soccer cleats, boxing gloves, teething blanket*
	3. <**RB**><**VBN**> (for adverb phrases) e.g., *rarely used, hardly worn*
	4. **(**<**N**>|<**J**>**)**<**J**> (for adjective phrases) e.g., *brand new, perfect working, good used*

Table 1: The regular expressions used in our experiments to extract tokens, along with example extractions. <w> indicates a word. Except for rule 1 in Tenders, <w> is case-insensitive. <NP>, <N>, and <J> indicate proper noun, noun, and adjective respectively. Other POS notations follow Penn Treebank Tags. In our experiment, + and * are restricted to four repetitions. Hence, the maximum length of any token is four.

After all candidate tokens are scored, our approach takes the top p% of candidates.

Step 3: Construct Similarity Edges. Our approach measures the similarity between two candidate tokens (i.e., nodes) based on their distributional similarity. For each candidate token, our approach builds a feature vector where each feature captures the frequency of a contextual word of the token (within a window size of two in both directions). For example, given the sentence *"I am selling a teething blanket"*, the contextual words for *teething* are *l2-selling, l1-a, r1-blanket*. [6] Our approach then weighs each feature (i.e., the frequency of each contextual word) by the PMI (pointwise mutual information) between the contextual word and candidate token. To mitigate PMI's bias toward infrequent elements, our approach uses a modified formula suggested in (Turney and Pantel 2010). Finally, our approach calculates the edge weights for all node pairs by measuring the cosine similarity between their feature vectors. Our approach keeps only the top 5 edges for each node because most edges have a negligible weight.

Step 4: Construct Mutually Exclusive Edges. For each pair of candidate tokens (i.e., nodes), v_1 and v_2, our approach determines whether they should be connected by a ME edge, based on the conditional probability of v_1 and v_2 occurring overlapped when v_1 or v_2 occurs – i.e., $w_{me} = \frac{f_{me}(v_1,v_2)}{max(freq(v_1),freq(v_2))}$ (see *Step 1* for definition of f_{me}). If w_{me} exceeds a pre-defined threshold, then a ME edge is established between v_1 and v_2 with a weight of w_{me}.

We use a conditional probability instead of f_{me} directly to prevent connecting nodes that do not have a strong association (and hence are not alternative, competing candidate tokens). For example, if $freq(blanket)$ is much larger than $f_{me}(blanket, \textit{teething blanket})$, then they are not connected by a ME edge because both are valid candidate tokens.

[6] *l2, l1* and *r1* indicates the position. For example, $l2$ denotes the second to the left.

Modified Label Propagation

Our approach extends the label propagation step to utilize the new graph representation, allowing evidence from labeling to be used for token extraction. Our extension is based on the semantics of the ME edges. If a node, v_i, is assigned a label, l ($l \neq \phi$), with high confidence, then v_i is a correctly segmented token. Therefore, any other nodes, v_j, connected to v_i via a ME edge is incorrectly segmented (and hence should be assigned the label ϕ).

Our approach captures this notion formally by extending the original propagation equation, Eq. 1, as follows:

$$\begin{aligned} \gamma_u(y) &= r_u(y)\delta(u \in V_l) \\ &+ \mu \sum_{v \in N(u)} w_{uv} q_v^{(m-1)}(y) + \nu U(y) \\ &+ \lambda\delta(y = \phi) \sum_{v \in N_{me}(u)} w'_{uv} me\_score(v) \end{aligned} \quad (3)$$

Eq. 3 is the same as Eq. 1 except for the additional fourth term. This term increases the score of ϕ if neighboring nodes (connected by ME edges) are likely to be assigned a label, l ($l \neq \phi$). Specifically, $\delta(y = \phi)$ returns 1 only if y is ϕ. $N_{me}(u)$ is the set of nodes connected to u by ME edges. w'_{uv} is the weight of a ME edge. $me\_score(v)$ measures the likelihood of a node, v, being labeled. We observe that a node is likely to be labeled if the node has a label, l ($l \neq \phi$), whose score is significantly higher than all other labels. Based on this observation, our approach calculates $me\_score(v)$ by taking the difference between the top two labels, l_1 and l_2 s.t. $l_1 \neq \phi$, from the label distribution of v. If this difference exceeds a pre-defined threshold, $me\_score(v)$ is set to the difference. Otherwise, $me\_score(v)$ is 0. [7]

Following (Subramanya, Petrov, and Pereira 2010), we set μ and ν to 0.5 and 0.01 respectively. For λ, we use a decay function, $\lambda = \alpha e^{\beta * x}$, (where x is the difference between the average score of ϕ at the current iteration step and

[7] Entropy has been used to detect a spiky distribution. However, ϕ makes the application of entropy difficult because entropy would prefer any spiky distribution even with the peak at ϕ.

the initial step), instead of a constant because as more nodes are assigned labels, ϕ starts to proliferate (and eventually swamps the graph) due to ME edges. We set α and β to 0.3 and -100 respectively.[8] As ϕ increases overall, λ decreases, preventing over-proliferation of ϕ.

Evaluation

We compared our approach to multiple baseline solutions, including several pipeline-based approaches and a joint-inference approach without our extensions. Our evaluation was conducted over two domains, medical tenders and informal advertisements, with various settings. Our evaluation showed that our approach significantly outperformed pipeline-based solutions, confirming the advantage of joint inference. Our evaluation also showed that our approach improved precision without hurting recall when compared against a base graph propagation approach, confirming the benefits of our extensions (and hence our approach).

Baseline Approaches We constructed several baseline solutions for our evaluation. First, we constructed three pipeline solutions – i.e., PIPELINE1, PIPELINE2, and PIPELINE_KEYWORD. These approaches must commit to the boundaries of tokens (using a set of extraction rules) before labeling can be performed. All three pipeline solutions used the extraction rules in Table 1. The Stanford POS tagger (Toutanova et al. 2003) was used for the rules requiring part-of-speech tagging. PIPELINE1 prefers the shortest token (if multiple extraction rules apply to the same span of text) – e.g., *teething* is preferred over *teething blanket*. PIPELINE2 prefers the longest token, which is a commonly used scheme for tokenization in many information extraction systems such as (Whitney and Sarkar 2012) (Collins and Singer 1999) (Talukdar et al. 2006). PIPELINE_KEYWORD, which is based on a state-of-the-art tokenization method (Deane 2005), uses Eq. 2 to score candidate tokens and selects the one with the highest score. Once these pipeline solutions commit to the tokens, they then use the label propagation scheme with the update function, Eq. 1, to perform labeling.

We also constructed a baseline joint inference system – i.e., BASEJOINT_<P>. Like our approach – we'll call OURJOINT_<P> – BASEJOINT_<P> does not commit to the boundaries of token before labeling. Instead, it generates all possible candidate tokens (using a set of extraction rules), and uses additional information from label propagation to determine the best candidates. However, unlike our solution, BASEJOINT_<P> does not capture mutually exclusive relationships between candidates and hence uses the update equation, Eq. 1, instead of our modified one, Eq. 3.

<p> indicates the percentage of the extracted tokens retained by the filtering step (see p in subsection *Step 2: Filter Candidates*). In our evaluation, <p> varies in $\{25, 50, 75, 100\}$, and both BASEJOINT_<P> and OURJOINT_<P> use the same extraction rules in Table 1 to generate candidate tokens.

[8] We set α and β based on the best result from multiple test runs. We note, however, that our approach was not overly sensitive to the choice of these parameters.

Evaluation Domains We evaluated all solutions over two domains, medical tenders and infant product ads. We selected these domains because they are difficult to parse, and in the case of tenders, have not been widely studied in Information Extraction research. For example, tenders contain various forms of text such as tables and (semi or un)structured text, and ads have an informal and idiosyncratic style.

Medical tenders describe requests for bids on medical products to be procured. These bids include information such as drug type, agency, etc., and we assembled a corpus of 2,200 tenders from a public website [9]. Because most tenders are in Microsoft Word or PDF, we used off-the-shelf libraries [10] to convert them to plain text.

Product ads include information such as condition, pickup location, etc., and we assembled 10,500 ads from a public website [11]. The information types extracted for each domain are shown in Table 2.

Evaluation Setup First, we employed an external annotator to construct a gold standard in the following manner. This annotator began by identifying a set of information types (i.e., labels) shown in Table. 2. Our annotator then manually extracted relevant tokens for each information type from the two corpora. The result was a dictionary for each domain that contained the correct tokens and their labels.

For each domain, we applied all baseline approaches (see previous subsection) along with our approach (i.e., OURJOINT_<P>) to the corresponding corpus to construct a dictionary containing the tokens and their labels. We repeated this process ten times: each time using a different set of seeds (the same set of seeds was used by all solutions for the same run). We randomly selected k percentage of the gold standard to serve as the seed, and the remainder of the gold standard to serve as the test set. We varied k across {.05, .1, .2, .3, .4, .5, .6, .7}. After each solution finished labeling via propagation (10 iterations max), we included in the output dictionary those token-label pairs where the label with the highest score for that token (i.e., node) exceeded a threshold s. Unless otherwise specified, we fixed k and s to .4 and .5 respectively. We also measured the effect of varying s across {.1, .2, .3, .4, .5, .6, .7, .8} (see results below).

For each output dictionary, we compared the dictionary (and hence the solution's performance) against the corresponding test set. We measured the performance using the metrics of precision(P) and recall (R):

$$\text{P} = \frac{\text{\# correct extractions by our system}}{\text{Total \# extractions by our system}}$$

$$\text{R} = \frac{\text{\# correct extractions by our system}}{\text{Total \# extractions by gold standards}}$$

where a correct extraction is a token-label pair that match exactly those provided by the gold standard. We also used the F1-score, which is the harmonic mean of precision and recall.

[9] http://www.tendertiger.com

[10] We used PDFMINER for PDF and PYWIN32 for MS Word.

[11] http://sfbay.craigslist.org

Tenders	DrugName(*Upper GIT Drugs*), City(*New Delhi*), Ministry (*Ministry of Railways*), Department (*Medical Department*), State (*Andhra Pradesh*), Institution(*South Central Railway*) , Office (*District Development Office*) , Purchaser (*Chief Medical Director*), Requirement (*WHO Certification*), Disease (*Hepatitis B*)
Ads	Product Types (*child rocking chair*), Character (*Hello Kitty*) , Company (*Fisher Price*), Size (*18"x36"*) , PickupLocation (*El Camino Real*), Condition(*barely worn*), Color(*espresso brown*), SeriesName (*Discovery Loops*)

Table 2: Information types and their examples

	Tenders	**Ads**
# of total docs	2,262	10,539
# of docs used to generate nodes	80	100
# of words per doc	1,627.9	63.3
# of nodes	2,766	2,356

Table 3: Statistics of the datasets

Due to compute resource limitations [12] we could not construct the graphs (required by all solutions evaluated) from the entire dataset. These graphs could be extremely large. Hence, we randomly selected a subset of documents for each domain for graph construction and used the rest to construct the feature vectors. Table. 3 shows the detailed settings.

Before presenting the results, we note that two experimental factors adversely affect the recall of all solutions evaluated. First, our extraction rules are not complete. Second, the conversion of tenders from PDF and MSWord to plain text is error-prone. [13] These factors often prevented tokens identified by the gold standard from being extracted across all solutions evaluated. In one sample analysis, only 45% (475/1,048) and 72% (268/373) of the gold standard could be extracted for Tender and Ads respectively.

Eval1: Comparison of Systems

Fig. 3 shows the F1-score for all systems evaluated. In both domains, the joint learning systems – i.e., OUTJOINT_75 and BASEJOINT_75 – significantly outperform the pipeline-based systems. This result confirms the advantage of delaying the commitment in token extraction to utilize evidence from label propagation.

Furthermore, our evaluation shows that OUTJOINT_75 was able to significantly improve precision over BASEJOINT_75 without hurting recall (see Fig. 4). This same finding is confirmed across different threshold scores s (see Fig. 5). These results demonstrate the benefits of our extensions – i.e., 1) extending the base graph representation to include mutually exclusive edges that captures ambiguities during token extraction, and 2) a modified method to utilize this information during label propagation.

Eval2: Analysis of Filtering in Joint Learning

Our method extracts candidate tokens using the extraction rules in Table 1, and then selects the top p% of the tokens using Eq. 2 (See Step2. Filtering). We perform this filtering step to make the algorithm scalable by discarding the tokens that are likely to be wrong.

In this experiment, we investige the effect of this filtering step on performance by varying the number of nodes kept (i.e., filter threshold p). We vary p across {25,50,75,100} – i.e., we compare the F1-scores of OURJOINT_25, OURJOINT_50, OURJOINT_75 and OURJOINT_100.

Fig. 6 shows that filtering is helpful. A cut around 75% in Tender and 50% in Ads reduces a significant number of nodes without hurting the accuracy. However, the result also shows that the performance degrades below the cut – especially for Tenders. This result shows that if filtering is conservatively performed, it can improve the scalability. This result also reaffirms that syntactic analysis (such as Eq. 2) alone for token extraction is not enough.

Eval3:Performance for Individual Labels

Table. 4 shows the performance of the top five labels in terms of the size of the gold standard dictionary. In this table, two

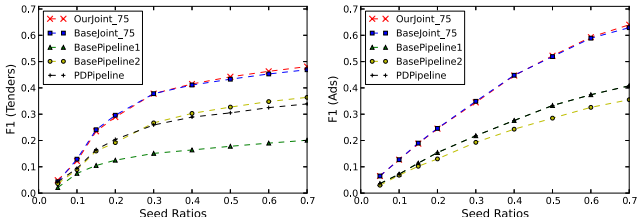

Figure 3: F1-scores of the systems

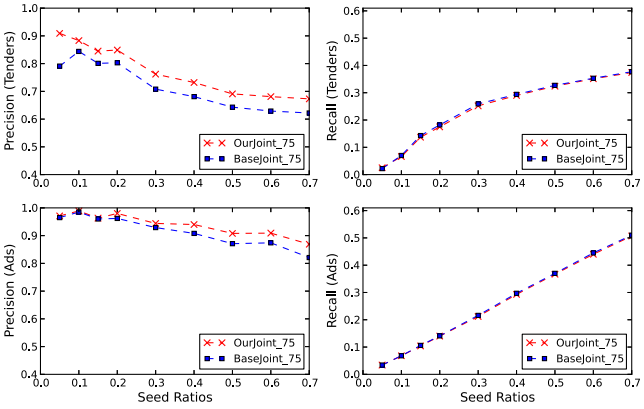

Figure 4: Precision and recall of OURJOINT_75 and BASEJOINT_75. The difference in precision was significant (paired t-test, $p < .01$) for seed ratio $>= .2$ for Tender and $>= .4$ for Ads.

[12] We only had access to 2.2 GHz dual CPU, 8G RAM node.

[13] All conversion tools we tried had poor performance. Given the explosively growing amount of PDFs and MSWords, we believe it is an important research to develop a reliable conversion method.

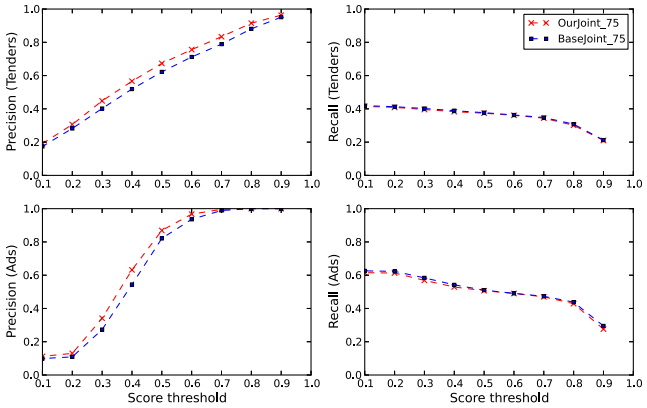

Figure 5: Precision and recall of OURJOINT_75 and BASEJOINT_75 over different scoring thresholds, s. The difference in precision was significant (paired t-test, $p < .01$) for s <= .8 for Tender and <= .6 for Ads.

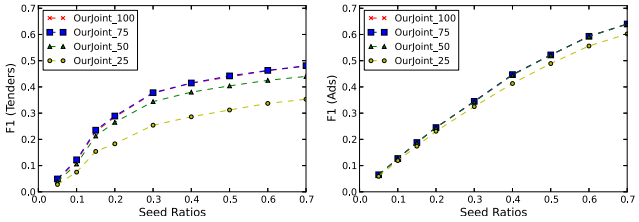

Figure 6: F1-Score for different filtering thresholds. Each curve represents a different filtering threshold p used by our system.

information types have low performance – CITY and CONDITION. For CITY, we found contextual words are uninformative because frequently co-occurring street names often differ for different cities. To address this problem, the latent meaning – the semantic labels of the surrounding words – should be captured.

CONDITION in the Ads also has a low F1-score. Our analysis shows CONDITION has a variety of surface forms (e.g., *used for 2 months, used only couple of times*) that could not be captured by our extraction rules. Furthermore, the context features are often uninformative. For example, the top three features of *barely used* are *l1-, l1-was* and *r2-in*, failing to include informative words.

Conclusion and Future Work

We proposed a dictionary construction method based on graph propagation. Our method allows a system to delay the decisions in token extraction to utilize the evidence from label propagation. Our evaluation shows the joint-inference approach improves the accuracy of the outcome dictionaries. Furthermore, the mutually exclusive edges representing conflicting relationship among the candidate tokens improves the precision without hurting the recall.

Based on the encouraging results, we plan to explore various features beyond distributional similarity, such as structural information (e.g., tables) which contains informative features. We plan to compare our approach against state-of-the-art sequence modeling approaches such as Conditional Random Fields. Finally, we plan to investigate the scalability of our approach by parallelizing it on distributed frameworks such as MapReduce.

		P	R	F1
Tender	DrugName(621)	0.63	0.38	0.475
	Requirement (101)	0.97	0.18	0.305
	City (95)	0.79	0.13	0.214
	Institution (48)	0.98	0.39	0.56
	Purchaser (37)	0.85	0.28	0.422
Ads	Product(181)	0.94	0.37	0.529
	Condition (51)	0.97	0.24	0.376
	Company (50)	0.95	0.42	0.582
	Character (29)	1	0.36	0.53
	PickupLoc (29)	0.69	0.52	0.579

Table 4: The performance of the top five labels. The parentheses indicate the number of tokens in the gold standard.

Acknowledgment

The authors would like to thank the anonymous reviewers for their helpful feedback and suggestions for improving the paper. The authors would also like to thank Anson Chu, Rey Vasquez, and Bryan Walker for their help with the experiments. Finally, the authors want to thank John Akred and Mary Ohara for their support on this project.

References

Baluja, S.; Seth, R.; Sivakumar, D.; Jing, Y.; Yagnik, J.; Kumar, S.; Ravichandran, D.; and Aly, M. 2008. Video suggestion and discovery for youtube: taking random walks through the view graph. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, 895–904. New York, NY, USA: ACM.

Collins, M., and Singer, Y. 1999. Unsupervised models for named entity classification. In *In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 100–110.

Deane, P. 2005. A nonparametric method for extraction of candidate phrasal terms. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, 605–613. Stroudsburg, PA, USA: Association for Computational Linguistics.

Kim, D. S.; Barker, K.; and Porter, B. 2010. Improving the quality of text understanding by delaying ambiguity resolution. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, 581–589. Stroudsburg, PA, USA: Association for Computational Linguistics.

Kim, D. S.; Verma, K.; and Yeh, P. Z. 2012. Building a lightweight semantic model for unsupervised information extraction on short listings. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language*

Learning, EMNLP-CoNLL '12, 1081–1092. Stroudsburg, PA, USA: Association for Computational Linguistics.

Poon, H., and Domingos, P. 2007. Joint inference in information extraction. In *Proceedings of the 22nd national conference on Artificial intelligence - Volume 1*, AAAI'07, 913–918. AAAI Press.

Putthividhya, D. P., and Hu, J. 2011. Bootstrapped named entity recognition for product attribute extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, 1557–1567. Stroudsburg, PA, USA: Association for Computational Linguistics.

Riloff, E., and Jones, R. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, AAAI '99/IAAI '99, 474–479. Menlo Park, CA, USA: American Association for Artificial Intelligence.

Subramanya, A.; Petrov, S.; and Pereira, F. 2010. Efficient graph-based semi-supervised learning of structured tagging models. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, 167–176. Stroudsburg, PA, USA: Association for Computational Linguistics.

Sutton, C., and McCallum, A. 2005. Joint parsing and semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, CONLL '05, 225–228. Stroudsburg, PA, USA: Association for Computational Linguistics.

Talukdar, P. P.; Brants, T.; Liberman, M.; and Pereira, F. 2006. A context pattern induction method for named entity extraction. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, 141–148. Stroudsburg, PA, USA: Association for Computational Linguistics.

Toutanova, K.; Klein, D.; Manning, C. D.; and Singer, Y. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, 173–180. Stroudsburg, PA, USA: Association for Computational Linguistics.

Turney, P. D., and Pantel, P. 2010. From frequency to meaning: vector space models of semantics. *J. Artif. Int. Res.* 37(1):141–188.

Whitney, M., and Sarkar, A. 2012. Bootstrapping via graph propagation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, ACL '12. Stroudsburg, PA, USA: Association for Computational Linguistics.

Yarowsky, D. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, ACL '95, 189–196. Stroudsburg, PA, USA: Association for Computational Linguistics.