Learning to Collectively Link Entities

Ashish Kulkarni  
IIT Bombay  
kulashish@gmail.com

Kanika Agarwal  
IIT Bombay  
kanika1712@gmail.com

Pararth Shah  
IIT Bombay  
pararthshah717@gmail.com

Sunny Raj Rathod  
IIT Bombay  
sunnyrajrathod@gmail.com

Ganesh Ramakrishnan  
IIT Bombay  
ganesh@cse.iitb.ac.in

ABSTRACT

Recently Kulkarni et al. [20] proposed an approach for collective disambiguation of entity mentions occurring in natural language text. Their model achieves disambiguation by efficiently computing exact MAP inference in a binary labeled Markov Random Field. Here, we build on their disambiguation model and propose an approach to jointly learn the node and edge parameters of such a model. We use a max margin framework, which is efficiently implemented using projected subgradient, for collective learning. We leverage this in an online and interactive annotation system which incrementally trains the model as data gets curated progressively. We demonstrate the usefulness of our system by manually completing annotations for a subset of the Wikipedia collection. We have made this data publicly available. Evaluation shows that learning helps and our system performs better than several other systems including that of Kulkarni et al.

Categories and Subject Descriptors
G.3 [PROBABILITY AND STATISTICS]: Markov processes

Keywords
entity disambiguation, associative markov network, learning

1. INTRODUCTION

Entity linking (EL) is the task of identifying and linking mentions embedded in unstructured text to their referent entity in a catalog like Wikipedia. This has been shown to benefit several systems, including search [4, 18], text classification [5], and other tasks. An entity linking system [21] typically consists of a spotter that first identifies short token segments (“spots”) as potential mentions of entities from its catalog. Many entities may qualify for a mention, e.g., “python” has over 15 senses in Wikipedia including Python (genus) and Python (programming language). In the second stage, a disambiguator assigns zero or more entities to selected mentions, based on mention-entity coherence, as well as entity-entity similarity. Collectively, these two stages comprise an annotator.

Some of the recent work [33, 24] shows that several mentions may have no associated sense in the catalog. This is referred to as the no-attachment (NA) problem (or NIL in the TAC-KBP challenge [25]). The other relatively lesser addressed challenge is that of multiple attachments [20], where, a mention might link to more than one entities from the catalog. This might often be a result of insufficient context and has been acknowledged by some of the recent entity disambiguation challenges.

As we describe in the next section, lot of earlier work on entity disambiguation focused on per-mention disambiguation. There has been several recent research on collective disambiguation of all mentions leveraging features derived from both mention-local context and global entity-entity relatedness. However, the complexity of these models often makes it computationally unfeasible to jointly learn their feature weights. We leverage the disambiguation model of Kulkarni et al. [20] and propose an efficient approach to jointly learn the local and global feature weights.

2. PRIOR WORK

Entity disambiguation: Earlier works [10, 3, 26] on entity annotation focused on per-mention disambiguation. This involves selecting the best entity to assign to a mention, independent of the assignments to other mentions in the document. Wikify! [26] for instance, uses context overlap for disambiguation and combines it with a classifier model that exploits local and topical features. Cucerzan [9] introduced the notion of agreement on categories of entities in addition to the local context overlap, in which the entity context comprised out-links from and in-links to their corresponding Wikipedia documents. Milne et al. [27] formulated a “relatedness” measure of similarity between two entities from Wikipedia, based on their in-link overlap. Relatedness, in conjunction with the prior probability of occurrence of an entity, was then used to train a classifier model. Han et al. [16] leveraged the semantic information in Wikipedia to build a large-scale semantic network and developed a similarity measure to be used for disambiguation. Kulkarni et al. [21] were the first to propose a general collective disambiguation approach, giving formulations for trade-off between mention-entity compatibility and coherence between entities. Several graph-based approaches [17, 11] followed

http://web-ngram.research.microsoft.com/ERD2014/
that cast the disambiguation problem as a problem of dense
subgraph selection from a graph of mentions and candidate
entities, making use of collective signals.

Most of these systems seem to prefer tagging conserva-
tively. Some of them [9, 17] restrict their tagging to named
etities, while others use a subset of entities from a back-
ground taxonomy such as TAP [10] or Wikipedia [27, 26].
Others [19, 1] have proposed LDA-based generative models
but focus only on person names. Some of the more recent
systems [21, 15] do perform aggressive spotting, aided by
the anchor dictionary of Wikipedia entities and study the
recall-precision tradeoff.

Kulkarni et al. [20] propose a joint disambiguation model
based on a Markov network of entities as nodes and edges for
their relatedness. Disambiguation is achieved by perform-
ing a MAP inference on this graph and it naturally handles
the NA and multiple attachment cases. Unlike other ap-
proaches [14, 28], their graph models candidate entities with
binary labels, instead of mentions with multiple labels. A
suitable assumption on cliques and their potentials makes
efficient computation of exact inference possible. However,
it is not clear as to how the node and edge feature weights
are set.

Joint learning: To the best of our knowledge, none of the
graph-based models above have attempted to jointly learn
the node and edge feature weights. While there is prior
work [31, 32] that applied graphical models to the problem
of information extraction and coreference resolution, exact
inference and estimation is intractable in these models. Sim-
ilar approaches have also been applied to the problem of
entity disambiguation [21, 14, 28], but hardly anyone has
attempted to jointly learn the feature weights. Taskar et
al. [29] proposed an approach to learn associative Markov
networks (AMNs). They provide an approximate quadratic
program for the problem of learning a margin maximizing
Markov network and show that it is guaranteed to return
optimal parameters for AMNs with binary-valued variables.
Our learning approach is based on this work but we are per-
haps the first ones to apply it in the text annotation domain.
Further, we also extend this learning approach and propose
an interactive active learning framework.

2.1 Our Contributions

We leverage the disambiguation model of Kulkarni et al. [20]
and propose an efficient approach to jointly learn the node
and edge feature weights. We also develop an interactive
active learning framework that progressively improves the
model as more training data becomes available. We imple-
memented our approach in an online annotation system and
used it to semi-automatically curate labeled data\(^2\). Our
trained model performs better than several other systems
including that of Kulkarni et al.

3. PRELIMINARIES

We borrowed the features and the disambiguation model
from the work described in Kulkarni et al. and present it in
brief here. We first start with the problem definition.

3.1 Problem Definition

The primary goal of document annotation is to link en-
tity mentions in an input document to entities in a catalog.

Mentions (or “spots”) are contiguous token sequences in a
text, e.g. Bush, that can potentially link to an entity, e.g.
George Bush in the catalog. Let \( \mathcal{M}_d \) be the set of all men-
tions in a document \( d \) and \( \varepsilon \) be the set of all entities in the
catalog. Then, the entity linking problem is to find for each
mention \( m \in \mathcal{M}_d \), the set of entities \( E_m \subset \varepsilon \cup \{ \text{NA} \} \) that
can link to.

As a first step, the input text \( d \) is processed by a “spotter”
to identify the set \( \mathcal{M}_d \) of mentions and the set of candidates
\( E_m \subset \varepsilon, \forall m \in \mathcal{M}_d \). \( E_m \) is called a candidate entity
for spot \( m \). The set \( E_d = \bigcup_{m \in \mathcal{M}_d} E_m \) forms the can-
didate entities set for document \( d \). This is then followed by a
“disambiguation” phase that obtains from \( E_m \), the set \( \hat{E}_m \) of
entities that the mention \( m \) can actually link to. When none
of the entities in \( E_m \) are valid, then \( \hat{E}_m = \{ \text{NA} \} \). Alterna-
tively, more than one entities from \( E_m \) might get promoted to
\( \hat{E}_m \). Assuming one sense per discourse [13], an entity in
the candidate set of more than one mentions, links (or does
not link) to all those mentions.

3.2 Entity Catalog

A catalog is a structured knowledge base comprising cat-
ergories with entities under them, along with their attributes
and relations. Wikipedia has seen an extensive organic growth
and covers entities spanning a vast set of domains. We re-
port experimental results using the Wikipedia dump from
May 2011, with approximately 4.4 million entities. For eval-
uation on ERD, we used as catalog, the snapshot of Freebase
as provided in the challenge.

3.3 Spotter

We processed the Wikipedia dump and indexed it in sep-
arate fields storing page title, synopsis, frequent words, out-
links, full text etc. Given a document \( d \), we first use the
Stanford POS tagger to obtain a set \( T_d \) of tokens (and their
spans) consisting of nouns, adjectives and verbs occurring in
the document text. Tokens appearing in close spans are con-
solidated into a phrase if the phrase is an anchor text for an
entity in Wikipedia. The set of tokens and phrases obtained
after consolidation, is the set \( \mathcal{M}_d \) of all potential mentions
in the document. For each mention \( m \in \mathcal{M}_d \), we then fire a
fielded query against the catalog store to obtain the set \( E_m \)
of candidate entities for the mention. In our experiments we
only retain the top \( k \) (empirically set to 8) entities from the
result set. In addition, we also used the Wikipedia Miner
toolkit\(^3\) to retrieve candidate senses for these tokens and
include them in the set \( E_m \).

3.4 Features

We used three types of features - (1) Popularity-based
features of an entity: Prior Sense Probability [26], In-Link
Count, Out-Link Count; (2) Mention-Entity compatibility
features [21]; (3) Entity-Entity relatedness features: Category-
based Similarity [9], In-link based Similarity [27], Out-link
based Similarity, Contextual Similarity.

3.5 The Disambiguation Model

Having identified the set of candidate entities for each
mention, a disambiguator attempts to link each mention to
zero or more entities. The label of a candidate is a collec-
tive result of the interplay of local mention-entity and global
entity-entity relatedness signals.

\(^2\)http://tinyurl.com/entitydisamb-data

\(^3\)http://sourceforge.net/projects/wikipedia-miner/
A Markov Random Field (MRF) is an undirected graphical model that captures local correlations between random variables [29].

![Candidate entities MRF model](image)

**Figure 1: Candidate entities MRF model**

A node is instantiated in the MRF graph for each possible entity mapping of each mention instance in a document. Edges capture entity-entity relatedness. Let $x_i$ be the node feature vector of candidate $i$ and $x_j$ be the edge feature vector of the edge joining candidates $i$ and $j$. Each candidate corresponds to a random variable that takes a binary label, $y_i \in \{0,1\}$, based on whether or not it correctly disambiguates the underlying mention. Let $C$ be the set of all cliques in the MRF and each clique $c \in C$ be associated with a clique potential $\phi_c(.)$. Cliques are restricted to nodes and edges and their potentials are parameterized by log-linear functions of feature vectors; i.e., $\log \phi_c(.) = w_c \cdot x_c$, where, $x_c$ is the feature vector of a clique and $w_c$, the corresponding weight vector. The potentials are assumed to be submodular [29], that is, they are associated with only those assignments, where, variables in a clique have the same label (associative). Moreover, these potentials are all greater than one and the rest are set to one. Thus, $\log \phi_c(.) = 0$ for non-associative cliques and therefore we define node and edge weights only for associative cliques. Let $w_0$ and $w_1$ be the node feature weights influencing node labels 0 and 1 respectively and $w_0$ and $w_1$ be the associative edge weights influencing the connected nodes to take the same label. The probability of a complete graph labeling $y$ is given by $P(y) = \frac{1}{Z} \prod_{c \in C} \phi_c(y_c)$, where $Z$ is the partition function. Disambiguation is achieved by performing MAP inference on this graph.

$$L(y) = \arg\max_{y \in \mathcal{Y}} \sum_{i \in \mathcal{N}} \log \phi_i(y_i) + \sum_{ij \in \mathcal{E}} \log \phi_{ij}(y_{ij})$$

$$= \arg\min_{w \geq 0} - (\sum_{i \in \mathcal{N}} w_0 \cdot x_i(1 - y_i) + w_1 \cdot x_i y_i$$

$$+ \sum_{ij \in \mathcal{E}} w_{00} \cdot x_{ij}(1 - y_{ij}) + w_{11} \cdot x_{ij} y_{ij})$$

(1)

$\mathcal{N}$ is the set of nodes and $\mathcal{E}$ is the set of edges in the MRF. $y_i \in \{0,1\}$ and $y_{ij} = y_i y_j$. For an MRF with binary labeled nodes and associative edge potentials, MAP inference can be computed exactly in polynomial time, by finding the min cut of an augmented flow graph [2].

**Construction of flow graph:** The MRF is augmented by adding two special terminal nodes $s$ and $t$ that correspond to the two labels 0/1. For each node $i$, we add terminal edges $s \rightarrow i$ with weight $(w_{00}, x_i)$ and $i \rightarrow t$ with weight $(w_{11}, x_i)$. For each neighborhood edge $i \rightarrow j$, we assign weight $(w_{00} + w_{11}, x_{ij})$. We also add weight $(w_{00}, x_{ij})$ to the edge $s \rightarrow i$ and $(w_{11}, x_{ij})$ to the edge $j \rightarrow t$.

MAP inference in original MRF corresponds to the s/t min cut on this augmented graph, with nodes on the $s$ side of the cut getting a label of 0, and the nodes on the $t$ side being assigned a label of 1.

### 4. LEARNING FEATURE WEIGHTS

**Algorithm 1: MRF Learning algorithm**

<table>
<thead>
<tr>
<th>Data:</th>
<th>Training set ${X, y}$, MRF graph $g$, Slack penalty $C$, Iterations $T$, Step size $\alpha_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result:</td>
<td>Weight vector $w$</td>
</tr>
<tr>
<td>$w \leftarrow 0$</td>
<td></td>
</tr>
<tr>
<td>$t \leftarrow 1$</td>
<td></td>
</tr>
<tr>
<td>$N_n \leftarrow$ number of nodes in $g$</td>
<td></td>
</tr>
<tr>
<td>$f_{opt} \leftarrow \infty$</td>
<td></td>
</tr>
<tr>
<td>$w_{opt} \leftarrow 0$</td>
<td></td>
</tr>
<tr>
<td><strong>while</strong> $t \leq T$ <strong>do</strong></td>
<td></td>
</tr>
<tr>
<td>$g \leftarrow$ construct flow network from $g$</td>
<td></td>
</tr>
<tr>
<td>$\tilde{q} \leftarrow s/t$ mincut of $g$</td>
<td></td>
</tr>
<tr>
<td>$\nabla w \xi(g) \leftarrow 2w + C(q - \tilde{q})^T X$</td>
<td></td>
</tr>
<tr>
<td>$w \leftarrow w - \alpha_t \nabla w \xi(g)$</td>
<td></td>
</tr>
<tr>
<td>Project $w$ onto the positive orthant</td>
<td></td>
</tr>
<tr>
<td>Compute function value $f$</td>
<td></td>
</tr>
<tr>
<td><strong>if</strong> $f &lt; f_{opt}$ <strong>then</strong></td>
<td></td>
</tr>
<tr>
<td>$f_{opt} \leftarrow f$</td>
<td></td>
</tr>
<tr>
<td>$w_{opt} \leftarrow w$</td>
<td></td>
</tr>
<tr>
<td><strong>end</strong></td>
<td></td>
</tr>
<tr>
<td>$t \leftarrow t + 1$</td>
<td></td>
</tr>
<tr>
<td><strong>end</strong></td>
<td></td>
</tr>
<tr>
<td><strong>return</strong> $w_{opt}$</td>
<td></td>
</tr>
</tbody>
</table>

The submodularity restriction and binary labels, make efficient implementation of learning possible. We jointly learn both node and edge feature weights following the general max-margin framework described in [29, 30]. Consider a graph with $\mathcal{N}$ nodes and $\mathcal{E}$ edges constructed as described above. Following Taskar et al., the learning problem can be formulated in terms of the cut vector, such that, we minimize the norm of the weight vector subject to the constraint that the desired labeling scores better than an arbitrary labeling by an amount that scales with the Hamming distance between the desired and incorrect labelings.

$$\min_{w \geq 0} \|w\|^2$$

subject to

$$\min_{q \in Q} \sum_{ij \in \mathcal{E}} w \cdot x_{ij}(q_{ij} - \hat{q}_{ij}) - (N_n - q_n^T \cdot q_o) \geq 0$$

Here, $Q$ is the set of all valid cuts and $q_{ij} \in \{0,1\}$ indicates if edge $i \rightarrow j$ is cut ($q_{ij} = 1$). $q_o$ is the cut vector for terminal edges with components $q_{st}$ and $q_{ts}$, where, $q_{st} = 1$ implies that $s$ is labeled 1. $\hat{q}$ is the cut vector corresponding to the desired labeling. The first component of the constraint captures the difference in cost of the min cut induced by the weights $w$ and that of the desired labeling. The other component corresponds to the number of labeling disagreements, $N_n$ being the number of nodes in the graph (excluding $s$ and $t$).

By rearranging terms, we obtain

$$\min_{w \geq 0} \|w\|^2$$

subject to

$$\sum_{ij \in \mathcal{E}} w \cdot x_{ij} - (N_n - q_n^T \cdot q_o) \geq 0$$

Here, $Q$ is the set of all valid cuts and $q_{ij} \in \{0,1\}$ indicates if edge $i \rightarrow j$ is cut ($q_{ij} = 1$). $q_o$ is the cut vector for terminal edges with components $q_{st}$ and $q_{ts}$, where, $q_{st} = 1$ implies that $s$ is labeled 1. $\hat{q}$ is the cut vector corresponding to the desired labeling. The first component of the constraint captures the difference in cost of the min cut induced by the weights $w$ and that of the desired labeling. The other component corresponds to the number of labeling disagreements, $N_n$ being the number of nodes in the graph (excluding $s$ and $t$).
subject to \( \min_{q \in Q} \sum_{i,j \in E} (w^T \cdot x_{ij} + \hat{q}_{ij} (\delta_{is} + \delta_{jt}))q_{ij} \) 
\[ \geq N_n + \sum_{i,j \in E} (w^T \cdot x_{ij})\hat{q}_{ij} \]  

Here, \( \delta_{ij} \) is the Kronecker delta. Now, consider the inequality constraint \( \min_{q \in Q} \sum_{i,j \in E} (w^T \cdot x_{ij} + \hat{q}_{ij} (\delta_{is} + \delta_{jt}))q_{ij} \geq N_n + \sum_{i,j \in E} (w^T \cdot x_{ij})\hat{q}_{ij} \). The left-hand-side of this inequality is equivalent to adding a capacity \( \hat{q}_{ij} (\delta_{is} + \delta_{jt}) \) to all cut terminal edges. Since each node participates in one terminal edge, this is equivalent to adding a total capacity of at-most \( N_n \) to the current flow graph. Therefore, \( \min_{q \in Q} \sum_{i,j \in E} (w^T \cdot x_{ij} + \hat{q}_{ij} (\delta_{is} + \delta_{jt}))q_{ij} \leq N_n + \sum_{i,j \in E} (w^T \cdot x_{ij})\hat{q}_{ij} \). It follows that the left-hand-side and right-hand-side of the inequality in the constraint must be equal [30]. Moving the constraint to the objective, we get, 
\[ \min_{w \geq 0} \|w\|^2 + C(N_n + \sum_{i,j \in E} (w^T \cdot x_{ij})\hat{q}_{ij}) \] 
\[ - \min_{q \in Q} \sum_{i,j \in E} (w^T \cdot x_{ij} + \hat{q}_{ij} (\delta_{is} + \delta_{jt}))q_{ij} \] 

Summing over all the documents in the training set, we get the final objective, 
\[ \min_{w \geq 0} \|w\|^2 + \sum_{d \in D} \left( C(N_d + \sum_{i,j \in E_d} (w^T \cdot x_{ij})\hat{q}_{ij}) \right) \] 
\[ - \sum_{q \in Q} \sum_{i,j \in E_d} (w^T \cdot x_{ij} + \hat{q}_{ij} (\delta_{is} + \delta_{jt}))q_{ij} \]  

Here, \( w = [w_0^T \ w_1^T \ w_0^T \ w_1^T]^T \), \( N_d \) is the number of nodes (excluding \( s \) and \( t \)) and \( E_d \) is the set of edges in the candidate entity MRF graph for a document \( d \in D \), the set of all training documents, \( s \) and \( t \) are special source and sink nodes, respectively. The term \( N_d - \hat{q}_{ij} (\delta_{is} + \delta_{jt}) \) gives the number of misclassified nodes and \( \sum_{i,j \in E_d} w^T \cdot x_{ij}\hat{q}_{ij} - w^T \cdot x_{ij}q_{ij} \) is the total capacity of incorrectly cut edges in the flow graph. \( C \) is the penalty associated with the incorrect labeling. We solved the formulation (4) using the subgradient descent method as described in Algorithm 1.

### 4.1 Handling Unbalanced Training Data

The training data has many more entities labeled 0 as compared to those labeled 1. In our datasets, we observed a skew of about 3:1. This results in a bias towards the over-represented class in the learning algorithm and the accuracy of the non-dominant class suffers. We addressed this problem by assigning separate misclassification penalties \( C_0 \) and \( C_1 \) for label 0 and 1 disagreements respectively in equation 4, where, disagreements are defined as below.

**Definition 1.** Let \( l_i \in \{0, 1\} \) and \( \hat{l}_i \in \{0, 1\} \) be the predicted and actual labels of node \( i \). We say that a node \( i \) has label 0 disagreement if \( l_i \neq \hat{l}_i \) and \( l_i = 0 \). Similarly it has label 1 disagreement if \( l_i = \hat{l}_i \) and \( \hat{l}_i = 1 \).

**Proposition 1.** For an edge \( i \rightarrow j \) with \( q_{ij} \neq \hat{q}_{ij} \), exactly one of the nodes agrees on the label i.e., \( l_i = \hat{l}_i \) (or \( l_j = \hat{l}_j \)) and the other node disagrees on the label i.e., \( l_i \neq \hat{l}_j \) (or \( l_j \neq \hat{l}_i \)).

**Proof.** Case 1: Let \( q_{ij} \neq \hat{q}_{ij} = 0 \). This implies that the edge is not cut in the actual labeling and therefore \( \hat{l}_i = \hat{l}_j \). However, \( q_{ij} = 1 \) implies that \( l_i \neq \hat{l}_j \). It follows that either \( l_i = \hat{l}_i \) or \( l_j = \hat{l}_j \).

**Case 2:** Let \( q_{ij} \neq \hat{q}_{ij} = 1 \). Following a similar argument as that for case 1 above, we have that \( \hat{l}_i \neq \hat{l}_j \) and \( l_i = l_j \). Again, it follows that either \( l_i = \hat{l}_i \) or \( l_j = \hat{l}_j \). \( \square \)

**Definition 2.** An edge \( i \rightarrow j \) with \( q_{ij} \neq \hat{q}_{ij} \), is said to have a label 0 disagreement if either \( l_i \neq \hat{l}_i = 0 \) or \( l_j \neq \hat{l}_j = 1 \). It is said to have a label 1 disagreement if either \( l_i \neq \hat{l}_i = 1 \) or \( l_j \neq \hat{l}_j = 1 \).

### 4.2 Active Learning

Our online annotation system presents an opportunity to continuously update the model as more labeled data becomes available. The commonly used passive learning approach involves manual annotation of randomly and independently sampled data. Due to the time and cost associated with this process, often there is not enough training data to meet certain level of performance. Active learning [22] aims to minimize the labeling effort, by requesting labels for the most informative samples, so as to achieve a desired level of accuracy. While there are several approaches to querying examples for labeling [23], we follow a pragmatic approach, that can be characterized as least certain querying method. The method samples examples with the smallest difference between two highest probability classes. Our binary labeled MRF model labels a node, based on the collective effect of the node potential and the edge potentials on the edges connecting the node to its neighbors. We define certainty \( C(i) \) at a node \( n_i \) as

\[ C(i) = \left( w_0 \cdot x_i + \sum_{(ij) \in E \setminus N(i)} w_{00} \cdot x_{ij} \right) \] 
\[ - \left( w_1 \cdot x_i + \sum_{(ij) \in E \setminus N(i)} w_{11} \cdot x_{ij} \right) \]  

where \( N(i) \) is the set of all neighboring nodes of the node \( i \). The certainty score \( C(d) \) for a document \( d \) is then computed as the average certainty score across all nodes in that document.

\[ C(d) = \frac{1}{|N|} \sum_{i \in N} C(i) \]  

The active learning algorithm then queries for a document with the lowest \( C(d) \) and presents the document for labeling. It might be possible to further reduce the labeling effort by requesting labels for only top \( k \) entities in the selected document, where the entities are ordered in increasing values of their \( C(i) \).

### 5. EXPERIMENTS AND RESULTS

### 5.1 Data Sets

Table 1 shows details of the datasets used. We use Wiki_cur (created by Kulkarni et al. [20]) for training our model and present cross-validation results. We also evaluate on several other datasets from the entity linking literature. Kulkarni et al. [21] had created a dataset (IITB_cur) based on aggressive spotting but assuming single attachment. We, therefore, used our annotation system to manually complete annotations (to create IITB_cur) for the documents in this dataset. Three volunteers put in a combined effort of close to 300 hours to curate 57 documents in the IITB_cur dataset. Each
document was reviewed by at least one other volunteer. For a given document, our system displays the spots along with their candidate entity sets. Volunteers were instructed to:

1. link all correct entities (multiple attachments) to a mention;
2. add an entity manually if it is not already in the candidate set;
3. link disambiguation pages sparingly and only if they begin with a definition that is relevant in the context of the mention;
4. leave the mention untagged (NA) only if none of the above qualify and
5. manually add mentions (and their entities) whenever they were missed by our spotter.

The number of mentions is indicative of our aggressive spotting (Refer 3.3). It is encouraging to note that close to 15% of these mentions have multiple attachments\(^3\) and around 30% have no attachment. In spite of our best efforts, data curation continues to be an extremely challenging task. We discuss some of these challenges in a later section (Refer to Section 6).

5.2 Evaluation Measures

We follow the fuzzy evaluation measure [7] that accounts for slight syntactic and semantic variations in the match of a predicted and true annotation, where an annotation \(a\) is defined as the mention-entity pair \((m, e)\). Using their notion of weak annotation match \(M_w(a_1, a_2)\), we use as performance metrics, Recall, Precision and F1 micro-averaged over all documents in a dataset. After factoring out spotter errors, we also separately report the accuracy of our disambiguation model alone (Referred to as “disambiguator only”). This accuracy can be easily computed by comparing the predicted and true labels. It was tuned on the training fold during two-fold cross-validation on Wiki\(\text{cur}\). Also, to account for the skew in label 0 and 1 instances in the dataset, we penalized label 0 and label 1 disagreements separately, using \(C_0\) and \(C_1\), respectively. A higher \(C_1\) for instance, improves the label 1 recall while adversely impacting the precision. It is this recall-precision tradeoff for varying values of \(C_0\), \(C_1\) that we capture in Figure 4. We chose the best \(C_0\) and \(C_1\) from these for all our experiments.

5.3 Experiments with only Node Features

5.3.1 Is there merit in data curation?

The data curation process presents an opportunity for continuous training where our inference model periodically evolves, as more and more data gets curated. Optionally, in the absence of any curated data to start with (at time \(t = t_0\) when our model is yet untrained), one could use a Logistic Regression model, trained on a large uncurated dataset, to warm-start the data curation process. As data gets curated and our model is trained, we switch to our trained model at time \(t = t_k\).

We trained binary label LR models using 10000 randomly sampled Wikipedia documents, replacing an original Wikipedia document with its curated version from Wiki\(\text{cur}\), one at a time. Figure 2 plots the training accuracies of these models for an increasing number of curated documents. The improvement in accuracy could be explained by the reduction in false negatives achieved by virtue of aggressive tagging and multiple attachments in the curated dataset. Based on this observation, we claim (and verify it in section 5.4.3) that our MRF model too would benefit from data curation. At the same time, the use of an LR model for warm-starting an online annotation system as ours is strongly recommended.

5.3.2 How does our model perform?

Thereafter, we trained our candidate entity MRF model on Wiki\(\text{cur}\) dataset using the node features alone. We report two-fold cross-validation results on Wiki\(\text{cur}\) and test results on IITB\(\text{part}\) (Refer to table 2). These serve as a baseline for our collective approach.

5.4 Collective Disambiguation

Next, we trained our model using node features and one or more edge features. Iterations \(T\) were fixed at 600, \(C\) was tuned as described below, and step size (at iteration \(t\)), \(\alpha_t = K/\sqrt{t}\), where \(K\) was empirically set to 0.01.

5.4.1 Effect of \(C\) on accuracy

The \(C\) parameter in equation 4 acts as a regularizer and is indicative of the tolerance of disagreement between predicted and true labels. It was tuned on the training fold during two-fold cross-validation on Wiki\(\text{cur}\). Also, to account for the skew in label 0 and 1 instances in the dataset, we penalized label 0 and label 1 disagreements separately, using \(C_0\) and \(C_1\), respectively. A higher \(C_1\) for instance, improves the label 1 recall while adversely impacting the precision. It is this recall-precision tradeoff for varying values of \(C_0\), \(C_1\) that we capture in Figure 4. We chose the best \(C_0\) and \(C_1\) from these for all our experiments.

5.4.2 Effect of Edge Features

Table 3 shows the effect of different edge features in a collective setting. The model seems to benefit the most from the inlink and outlink relatedness features, while context overlap-based features seem to be noisy. This is understandable as context overlap-based signals are useful only for topically coherent entities, which might not hold true for an aggressively tagged corpus like ours [21].

5.4.3 Does training help?

We sampled 50 documents from the Wiki\(\text{cur}\) dataset, 5 at a time and used them for training, applying both passive (PL) and active learning (AL). The \(F_1\) measure evaluated on an independent test set of 30 documents is shown in the plot (Refer to figure 3). The \(F_1\) on training set seems to fluctuate, more so for Train-AL, as has been observed by others [6]. The performance of an active learner depends not only on training on instances that the model is least certain about, but, also on the informative features contained in them. The \(F_1\) on test set does show a steady improvement and we expect this to improve further as more and more curated documents become available for training.

5.4.4 Comparison with collective approaches

We compared our system against several other collective annotation approaches: AIDA [17], Wikify! [26], TagMe [12], Wikipedia Miner [27] and Illionis Wikifier [28] on three datasets viz. IITB\(\text{part}\), AQUAINT (Wikipedia Miner) and MSNBC [9]. Our system consistently beats all these systems on all the three datasets (Table 4). Some of the other collective annotation systems like Cucerzan (\(F_1 : .45\)), CSAW [21] (\(F_1 : .45\))

\(^3\)includes synonyms

\(^4\)which is true iff mentions \(m_1\) and \(m_2\) overlap in the input text and entities \(e_1\) and \(e_2\) are synonyms
Table 1: Dataset statistics (Mean and Standard Deviation)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size</th>
<th>#Mentions M</th>
<th>SD</th>
<th>#Mentions as % of #words in a document M</th>
<th>SD</th>
<th>#NA M</th>
<th>SD</th>
<th>#Multiple attachments M</th>
<th>SD</th>
<th>#Overlapping mentions M</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Wiki</em>cur</td>
<td>106</td>
<td>61.16</td>
<td>20.22</td>
<td>22.62</td>
<td>3.9</td>
<td>12.9</td>
<td>7.32</td>
<td>10.82</td>
<td>8.54</td>
<td>3.32</td>
<td>3.18</td>
</tr>
<tr>
<td><em>IITBpart</em></td>
<td>103</td>
<td>191.58</td>
<td>100.26</td>
<td>31.12</td>
<td>12.15</td>
<td>74.28</td>
<td>41.41</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><em>IITBcur</em></td>
<td>57</td>
<td>196.02</td>
<td>82.68</td>
<td>31.5</td>
<td>5.9</td>
<td>63.89</td>
<td>37.54</td>
<td>25.5</td>
<td>18.43</td>
<td>3.07</td>
<td>2.85</td>
</tr>
</tbody>
</table>

Figure 2: Effect of data curation

Figure 3: Effect of training

Table 2: Non-collective results (only node features) on *Wiki*cur set and *IITBpart* datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Disambiguator only P</th>
<th>R</th>
<th>F</th>
<th>Weak annotation match P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Wiki</em>cur</td>
<td>.82</td>
<td>.67</td>
<td>.74</td>
<td>.82</td>
<td>.56</td>
<td>.67</td>
</tr>
<tr>
<td><em>IITBpart</em></td>
<td>.82</td>
<td>.66</td>
<td>.73</td>
<td>.82</td>
<td>.50</td>
<td>.62</td>
</tr>
</tbody>
</table>

Figure 4: Effect of varying $C$: $C_0$ and $C_1$

.69), [15] ($F_1 : .73$), and [14] ($F_1 : .8$) have used CSAW’s evaluation measure to evaluate on *IITBpart*. We achieved an $F_1$ of 0.6 using the same measure. The relatively lower $F_1$ on this dataset could be attributed to inconsistencies between the ground truth and our knowledge base. During our manual annotation of *IITBpart*, we came across over 8000 annotations that were either added or removed⁶ to create the *IITBcur* dataset.

⁶due to erroneous annotations or newer Wikipedia dump

Table 3: Effect of edge features: two-fold cross validation on *Wiki*cur. Edge features that showed improvement over node features are shown in bold.

<table>
<thead>
<tr>
<th>Edge feature</th>
<th>Disambiguator only P</th>
<th>R</th>
<th>F</th>
<th>Weak annotation match P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>.72</td>
<td>.74</td>
<td>.73</td>
<td>.72</td>
<td>.63</td>
<td>.67</td>
</tr>
<tr>
<td><em>Outlink</em> (O)</td>
<td>.84</td>
<td>.67</td>
<td>.74</td>
<td>.84</td>
<td>.57</td>
<td>.68</td>
</tr>
<tr>
<td><em>Inlink</em> (I)</td>
<td>.80</td>
<td>.73</td>
<td>.76</td>
<td>.80</td>
<td>.62</td>
<td>.70</td>
</tr>
<tr>
<td>Frequent (F)</td>
<td>.84</td>
<td>.64</td>
<td>.73</td>
<td>.84</td>
<td>.54</td>
<td>.66</td>
</tr>
<tr>
<td>Synopsis</td>
<td>.69</td>
<td>.61</td>
<td>.65</td>
<td>.69</td>
<td>.52</td>
<td>.59</td>
</tr>
<tr>
<td>Syn. V/Adj.</td>
<td>.69</td>
<td>.67</td>
<td>.68</td>
<td>.69</td>
<td>.57</td>
<td>.62</td>
</tr>
<tr>
<td>Full text</td>
<td>.85</td>
<td>.63</td>
<td>.73</td>
<td>.85</td>
<td>.54</td>
<td>.66</td>
</tr>
<tr>
<td>All features</td>
<td>.44</td>
<td>.50</td>
<td>.47</td>
<td>.44</td>
<td>.42</td>
<td>.43</td>
</tr>
<tr>
<td>I+O</td>
<td>.85</td>
<td>.67</td>
<td>.74</td>
<td>.85</td>
<td>.56</td>
<td>.68</td>
</tr>
<tr>
<td>I+O+F</td>
<td>.79</td>
<td>.74</td>
<td>.76</td>
<td>.79</td>
<td>.63</td>
<td>.70</td>
</tr>
</tbody>
</table>

We evaluated our system on the ERD dataset and achieved $R: .62$, $P: .66$, $F_1 : .64$. We believe that our system benefits from model training, thereby performing better than that of [20] ($F_1 : .61$). While some of the other systems [8] at ERD performed better, this could be attributed to their choice of features. Our system offers an end-end annotation framework that is interactive and jointly trains feature weights.

5.5 Results on *IITBcur*

Section 6 shows some examples of incomplete annotations in the *IITBpart* dataset. It is precisely such cases that we tried to correct during data preparation. Finally, we report the accuracy of our model on the *IITBcur* dataset - $P : 77.4\%$, $R : 54.3\%$, $F_1 : 63.8\%$. 
Table 4: Comparison with publicly available systems (as reported by Cornolti et al. [7]) on three datasets

<table>
<thead>
<tr>
<th>Annotator</th>
<th>ITBpart F</th>
<th>AQUAINT R</th>
<th>MSNBC P</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIDA</td>
<td>.07</td>
<td>.66</td>
<td>.04</td>
</tr>
<tr>
<td>Wikify!</td>
<td>.37</td>
<td>.55</td>
<td>.28</td>
</tr>
<tr>
<td>TagMe</td>
<td>.44</td>
<td>.45</td>
<td>.42</td>
</tr>
<tr>
<td>Wikipedia Miner</td>
<td>.52</td>
<td>.57</td>
<td>.48</td>
</tr>
<tr>
<td>Illionis Wikifier</td>
<td>.44</td>
<td>.58</td>
<td>.36</td>
</tr>
<tr>
<td>Our Model (Node+I)</td>
<td>.67</td>
<td>.76</td>
<td>.60</td>
</tr>
<tr>
<td>Our Model (Node+I+O+F)</td>
<td>.65</td>
<td>.69</td>
<td>.61</td>
</tr>
</tbody>
</table>

5.6 Performance Evaluation

While our model allows for efficient inference and learning, graph construction itself is an expensive operation. For a document with \(|E_d|\) candidate entities, the graph construction complexity is \(O\left(|E_d|^2\right)\). For documents in the Wiki\(_{cur}\) set with 190 candidate entities on an average, the average graph construction time was about 57 seconds. For the relatively larger documents in the ITB\(_{part}\) dataset, the average graph construction time was around 1.5 minutes. The performance could be improved by (a) pre-computing the entity-entity features for all entities in the knowledge base (b) dividing input document into chunks and performing graph construction and inference in parallel.

Figure 5: Running time for inference on Wiki\(_{cur}\)

The running time for inference (Figure 5) shows a slightly quadratic behavior in the number of candidate entities \(|E_d|\) of a document. Inference on most documents runs in under 0.5 seconds. On the relatively sparser Inlink+Outlink graphs (Refer to Figure 6), training is much faster than the more dense Category graphs. The faster training happens without trading off much on accuracy as can be seen in Table 3. For our experiments, the model was retrained at time \(t\) using all the available training data. While this might be acceptable for offline training, online systems might benefit from faster incremental training approaches.

6. CHALLENGES WITH DATA CURATION

Data curation is a tedious and challenging task. Its inherent ambiguity often introduces annotator bias leading to either incomplete or ambiguous annotations in the curated data.

1. There might be cases when two or more entities are correct as attachments for a mention. E.g., mention ‘Barack Obama’ can be tagged as Barack Obama or President of United States, and both might seem correct in the context that it appeared. A ‘one entity per mention’ assumption makes it impossible to honor such cases.

2. Human annotators often limit their attention to the candidate entities retrieved by the spotter and very rarely search the catalog for any missed candidates. This results in a lot of missed annotations and often many mentions getting no attachments (NA).

3. Annotators also seem biased towards entity names that match with the mention text. However, this is often not true. E.g. a mention of ‘cone snail’ disambiguates to Conidae and Conus.

4. Wikipedia contains many disambiguation pages that often show up in the candidate set for a mention. Tagging a mention with a disambiguation page seems to be the very purpose of a disambiguation system. Ideally, the mention should be annotated with one of the entities on the disambiguation page or NA if none of them is semantically right.

Table 5 shows some of these cases from the ITB\(_{part}\) dataset. It is cases like these that we attempted to correct in coming up with the curated ITB\(_{cur}\) dataset.

7. CONCLUSION

We presented an approach to jointly train the node and edge features of a collective disambiguation model for the purpose of entity linking. Our system leverages active learning to bring down labeling effort. Experiments show that the model benefits from training and improves with the availability of more labeled data. We consistently performed
better than many other systems on various datasets. It also scales reasonably well and with suggested tweaks can be used for large scale document annotation.

8. ACKNOWLEDGMENTS

This research was supported by the Intranet Search project from IRCC at IIT Bombay.

9. REFERENCES


