Efficient Collective Entity Linking with Stacking

Zhengyan He[†] **Shujie Liu**[‡] **Yang Song**[†] **Mu Li**[‡] **Ming Zhou**[‡] **Houfeng Wang**[†*]
[†] Key Laboratory of Computational Linguistics (Peking University) Ministry of Education,China
[‡] Microsoft Research Asia
hezhengyan.hit@gmail.com {shujliu,muli,mingzhou}@microsoft.com
songyangmagic@gmail.com wanghf@pku.edu.cn

Abstract

Entity disambiguation works by linking ambiguous mentions in text to their corresponding real-world entities in knowledge base. Recent collective disambiguation methods enforce coherence among contextual decisions at the cost of non-trivial inference processes. We propose a fast collective disambiguation approach based on stacking. First, we train a local predictor g_0 with learning to rank as base learner, to generate initial ranking list of candidates. Second, top k candidates of related instances are searched for constructing expressive global coherence features. A global predictor g_1 is trained in the augmented feature space and stacking is employed to tackle the train/test mismatch problem. The proposed method is fast and easy to implement. Experiments show its effectiveness over various algorithms on several public datasets. By learning a rich semantic relatedness measure between entity categories and context document, performance is further improved.

1 Introduction

When extracting knowledge from natural language text into a machine readable format, ambiguous names must be resolved in order to tell which real-world entity the name refers to. The task of linking names to knowledge base is known as entity linking or disambiguation (Ji et al., 2011). The resulting text is populated with semantic rich links to knowledge base like Wikipedia, and ready for various downstream NLP applications.

Previous researches have proposed several kinds of effective approaches for this problem. Learning to rank (L2R) approaches use hand-crafted features $f(d, e)$ to describe the similarity or dissimilarity between contextual document d and entity definition e. L2R approaches are very flexible and expressive. Features like name matching, context similarity (Li et al., 2009; Zheng et al., 2010; Lehmann et al., 2010) and category context correlation (Bunescu and Pasca, 2006) can be incorporated with ease. Nevertheless, decisions are made independently and inconsistent results are found from time to time.

Collective approaches utilize dependencies between different decisions and resolve all ambiguous mentions within the same context simultaneously (Han et al., 2011; Hoffart et al., 2011; Kulkarni et al., 2009; Ratinov et al., 2011). Collective approaches can improve performance when local evidence is not confident enough. They often utilize semantic relations across different mentions, and is why they are called *global* approaches, while L2R methods fall into *local* approaches (Ratinov et al., 2011). However, collective inference processes are often expensive and involve an exponential search space.

We propose a collective entity linking method based on stacking. Stacked generalization (Wolpert, 1992) is a powerful meta learning algorithm that uses two levels of learners. The predictions of the first learner are taken as augmented features for the second learner. The nice property of stacking is that it does not restrict the form of the base learner. In this paper, our base learner, an L2R ranker, is first employed to generate a ranking list of candidates.

[*]Corresponding author

At the next level, we search for semantic coherent entities from the top k candidates of neighboring mentions. The second learner is trained on the augmented feature space to enforce semantic coherence. Stacking is employed to handle train/test mismatch problem. Compared with existing collective methods, the inference process of our method is much faster because of the simple form of its base learner.

Wikipedians annotate each entity with categories which provide another source of valuable semantic information. (Bunescu and Pasca, 2006) propose to generalize beyond context-entity correlation $s(d, e)$ with word-category correlation $s(w, c)$. However, this method works at word level, and does not scale well to large number of categories. We explore a representation learning technique to learn the category-context association in latent semantic space, which scales much better to large knowledge base.

Our contributions are as follows: (1) We propose a fast and accurate stacking-based collective entity linking method, which combines the benefits of both coherence modeling of collective approaches and expressivity of L2R methods. We show an effective usage of ranking list as global features, which is a key improvement for the global predictor. (2) To overcome problems of scalability and shallow word-level comparison, we learn the category-context correlation with recent advances of representation learning, and show that this extra semantic information indeed helps improve entity linking performance.

2 Related Work

Most popular entity linking systems use the L2R framework (Bunescu and Pasca, 2006; Li et al., 2009; Zheng et al., 2010; Lehmann et al., 2010). Its discriminative nature gives the model enough flexibility and expressivity. It can include any features that describe the similarity or dissimilarity of context d and candidate entity e. They often perform well even on small training set, with carefully-designed features. This category falls into the *local* approach as the decision processes for each mention are made independently (Ratinov et al., 2011).

(Cucerzan, 2007) first suggests to optimize an objective function that is similar to the collective approach. However, the author adopts an approximation method because of the large search space (which is $O(n^m)$ for a document with m mentions, each with n candidates). Various other methods like integer linear programming (Kulkarni et al., 2009), personalized PageRank (Han et al., 2011) and greedy graph cutting (Hoffart et al., 2011) have been explored in literature. Our method without stacking resembles the method of (Ratinov et al., 2011) in that they use the predictions of a local ranker to generate features for global ranker. The differences are that we use stacking to train the local ranker to handle the train/test mismatch problem and top k candidates to generate features for the global ranker.

Stacked generalization (Wolpert, 1992) is a meta learning algorithm that uses multiple learners outputs to augment the feature space of subsequent learners. It utilizes a cross-validation strategy to address the train set / testset label mismatch problem. Various applications of stacking in NLP have been proposed, such as collective document classification (Kou and Cohen, 2007), stacked dependency parsing (Martins et al., 2008) and joint Chinese word segmentation and part-of-speech tagging (Sun, 2011). (Kou and Cohen, 2007) propose stacked graphical learning which captures dependencies between data with relational template. Our method is inspired by their approach. The difference is our base learner is an L2R model. We search related entity candidates in a large semantic relatedness graph, based on the assumption that true candidates are often semantically correlated while false ones scattered around.

Wikipedians annotate entries in Wikipedia with category network. This valuable information generalizes entity-context correlation to category-context correlation. (Bunescu and Pasca, 2006) utilize category-word as features in their ranking model. (Kataria et al., 2011) employ a hierarchical topic model where each inner node in the hierarchy is a category. Both approaches must rely on pruned categories because the large number of noisy categories. We try to address this problem with recent advances of representation learning (Bai et al., 2009), which learns the relatedness of category and context in latent continuous space. This method scales well to potentially large knowledge base.

3 Method

In this section, we first introduce our base learner and local features used; next, the stacking training strategy is given, followed by an explanation of our global coherence model with augmented feature space; finally we explain how to learn category-context correlation with representation learning technique.

3.1 Base learner and local predictor g_0

Entity linking is formalized as follows: given an ambiguous name mention m with its contextual document d, a list of candidate entities $e_1, e_2, \ldots, e_{n(m)} \in C(m)$ is generated for m, our predictor g will generate a ranking score $g(e_i)$ for each candidate e_i. The ranking score will be used to construct augmented features for the next level learner, or used by our end system to select the answer:

$$\hat{e} = \arg\max_{e \in C(m)} g(e) \quad (1)$$

In an L2R framework, the model is often defined as a linear combination of features. Here, our features $\vec{f}(d, e)$ are derived from document d and candidate e. The model is defined as $g(e) = \vec{w}\vec{f}(d, e)$. In our problem, we are given a list of training data $\mathcal{D} = \{(d_i, e_i)\}$. We want to optimize the parameter $\vec{w}$, such that the correct entity has a higher score over negative ones. This is done via a preference learning technique SVM^{rank}, first introduced by (Joachims, 2002). The following margin based loss is minimized w.r.t $\vec{w}$:

$$L = \frac{1}{2}\|\vec{w}\|^2 + C \sum \xi_{d,e'} \quad (2)$$

$$\text{s.t.} \quad \vec{w}(\vec{f}(d, e) - \vec{f}(d, e')) \geq 1 - \xi_{d,e'} \quad (3)$$

$$\xi_{d,e'} \geq 0 \quad (4)$$

where C is a trade-off between training error and margin size; ξ is slacking variable and loops over all query documents d and negative candidates $e' \in C(m) - \{e\}$.

This model is expressive enough to include any form of features describing the similarity and dissimilarity of d and e. We only include some typical features seen in literature. The inclusion of these features is not meant to be exhaustive. Our purpose is to build a moderate model in which some of the useful features like string matching and entity popularity cannot be easily expressed by collective approaches like (Hoffart et al., 2011; Han et al., 2011). The features for level 0 predictor g_0 are described in Table 1. The reader can consult (Li et al., 2009; Zheng et al., 2010; Lehmann et al., 2010) for further reference.

Surface matching:
1. mention string m exactly matches candidate e, i.e. $m = e$
2. neither m is a substring of e nor e is a substring of m
3. $m \neq e$ and m is a substring of e
4. $m \neq e$ and e is a substring of m
5. $m \neq e$ and m is a redirect pointing to e in Wikipedia
6. $m \neq e$ and e starts with m
7. $m \neq e$ and e ends with m
Context matching:
1. cosine similarity of TF-IDF score between context and entire Wikipedia page of candidate
2. cosine similarity of TF-IDF score between context and introduction of Wikipedia page
3. jaccard distance between context and entire Wikipedia page of candidate
4. jaccard distance between context and introduction of Wikipedia page
Popularity or prominence feature:
percentage of Wikipedia hyperlinks pointing to e given mention m, i.e. P(e|m)
Category-context coherence model:
cat_0 and cat_1 (details in Section 3.4)

Table 1: Features for local predictor g_0.

3.2 Stacking training for global predictor g_1

Stacked generalization (Wolpert, 1992) is a meta learning algorithm that stacks two "levels" of predictors. Level 0 includes one or more predictors $h_1^{(0)}, h_2^{(0)}, \ldots, h_K^{(0)} : \mathbb{R}^d \to \mathbb{R}$, each one is trained on the original d-dimensional feature space. The level 1 predictor $h^{(1)} : \mathbb{R}^{d+K} \to \mathbb{R}$ is trained in the augmented $(d+K)$-dimensional feature space, in which predictions at level 0 are taken as extra features in $h^{(1)}$.

(Kou and Cohen, 2007) proposed stacked graphical learning for learning and inference on relational data. In stacked graphical learning, dependencies among data are captured by *relational template*, with which one searches for *related instances* of the current instance. The augmented feature space does not necessarily to be $d + K$. Instead, one can construct any declarative feature with the original data and predictions of related instances. For instance, in collective document classification (Kou and Cohen, 2007) employ relational template to extract documents that link to this document, then apply a COUNT aggregator over each category on neighboring documents as level 1 features.

In our entity linking task, we use a single predictor g_0 trained with local features at level 0. Compared with (Kou and Cohen, 2007), both g_0 and g_1 are L2R models rather than classifier. At level 1, for each document-candidate entity pair, we use the *relational template* $\mathcal{N}(x)$ to find related entities for entity x, and construct global features with some function $\mathcal{G}(\{g_0(n)|n \in \mathcal{N}(x)\})$ (details in Sec. 3.3). The global predictor g_1 receives as input the original features plus $\mathcal{G}$.

One problem is that if we use g_0 trained on the entire training set to predict related instances in training set, the accuracy can be somehow different (typically lower) for future unseen data. g_1 with this prediction as input doesn't generalize well to test data. This is known as train/test mismatch problem. To mimic test time behavior, training is performed in a cross-validation-like way. Let $\mathcal{D}$ be the entire training set:

1. Split $\mathcal{D}$ into L partitions $\{\mathcal{D}_1, \dots, \mathcal{D}_L\}$

2. For each split $\mathcal{D}_i$:

   2.1 Train an instance of g_0 on $\mathcal{D} - \mathcal{D}_i$

   2.2 Predict all related instances in $\mathcal{D}_i$ with this predictor g_0

   2.3 Augment feature space for $x \in \mathcal{D}_i$, with $\mathcal{G}$ applied on predictions of $\mathcal{N}(x)$

3. Train level 0 predictor g_0 on entire $\mathcal{D}$, for expanding feature space for test data

4. Train level 1 predictor g_1 on entire $\mathcal{D}$, in the augmented feature space.

In the next subsection, we will describe how to construct global features from the predictions of g_0 on neighbors $\mathcal{N}(x)$ with $\mathcal{G}$.

3.3 Enforcing coherence with global features $\mathcal{G}$

If one wants to identify the correct entity for an ambiguous name, he would possibly look for related entities in its surrounding context. However, surrounding entities can also exhibit some degree of ambiguity. In ideal cases, most true candidates are inter-connected with semantic links while negative candidates are scattered around (Fig. 1). Thus, we ask the following question: Is there any highly relevant entity to this candidate in context? Or, is there any mention with highly relevant entity to this candidate in the top k ranking list of this mention? And how many those mentions are? The reason to look up top k candidates is to improve recall. g_0 may not perfectly rank related entity at the first place, e.g. "Mitt Romney" in Figure 1.

Assume the ambiguous mention set is M. For each mention $m_i \in M$, we rank each entity $e_{i,j} \in C(m_i)$ by its score $g_0(e_{i,j})$. Denote its rank as $Rank(e_{i,j})$. For each entity e in the candidate set $E = \{e_{i,j} | \forall e_{i,j} \in C(m_i), \forall m_i \in M\}$, we search related instances for e as follows:

1. search in E for entities with semantic relatedness above a threshold ({0.1,0.3,0.5,0.7,0.9});

2. select those entities in step (1) with $Rank(e)$ less than or equal to k ($k \in \{1, 3, 5\}$);

3. map entities in step (2) to unique set of mentions U, excluding current m, i.e. $e \in C(m)$.

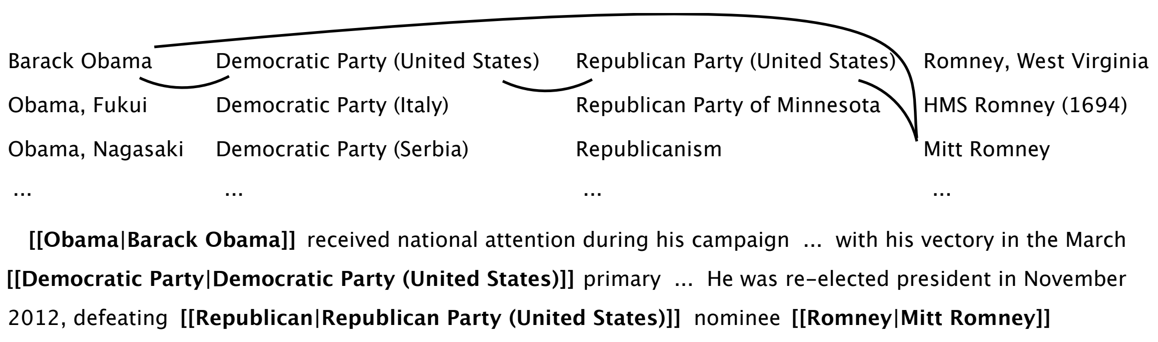
This process is relatively fast. It only involves a sparse matrix slicing operation on the large precomputed semantic relatedness matrix in step (1), and logical operation in step (2,3). The following features are fired concerning the unique set U:

- if U is empty;

- if U is not empty;

- if the percentage $|U|/|M|$ is above a threshold (e.g. 0.3).

The above process generates a total of 45 ($5 \times 3 \times 3$) global features.



Barack Obama
Obama, Fukui
Obama, Nagasaki
...
Democratic Party (United States)
Democratic Party (Italy)
Democratic Party (Serbia)
...
Republican Party (United States)
Republican Party of Minnesota
Republicanism
...
Romney, West Virginia
HMS Romney (1694)
Mitt Romney
...
[[Obama|Barack Obama]] received national attention during his campaign ... with his vectory in the March [[Democratic Party|Democratic Party (United States)]] primary ... He was re-elected president in November 2012, defeating [[Republican|Republican Party (United States)]] nominee [[Romney|Mitt Romney]]

Figure 1: Semantic links for collective entity linking. Annotation $[[mention|entity]]$ follows Wikipedia conventions.

Finally, the semantic relatedness measure of two entities e_i,e_j is defined as the common in-links of e_i and e_j in Wikipedia (Milne and Witten, 2008; Han et al., 2011):

$$SR(e_i, e_j) = 1 - \frac{\log(\max(|A|, |B|)) - \log(|A \cap B|)}{\log(|W|) - \log(\min(|A|, |B|))} \tag{5}$$

where A and B are the set of in-links for entity e_i and e_j respectively, and W is the set of all Wikipedia pages.

Our method is a trade-off between exact collective inference and approximating related instance with top ranked entities produced by g_0. Most collective approaches take all ambiguous mentions into consideration and disambiguate them simultaneously, resulting in difficulty when inference in large search space (Kulkarni et al., 2009; Hoffart et al., 2011). Others resolve to some kinds of approximation. (Cucerzan, 2007) construct features as the average of all candidates for one mention, introducing considerable noise. (Ratinov et al., 2011) also employ a two level architecture but only take top 1 prediction for features. This most resembles our approach, except we use stacking to tackle the train/test mismatch problem, and construct different set of features from top k candidates predicted by g_0. We will show in our experiments that this indeed helps boost performance.

3.4 Learning category-context coherence model cat

Entities in Wikipedia are annotated with rich semantic structures. Category network provides us with another valuable information for entity linking. Take the mention "Romney" as an example, one candidate "Mitt Romney" with category "Republican party presidential nominee" co-occurs frequently with context like "election" and "campaign", while another candidate "Milton Romney" with category "Utah Utes football players" is frequently observed with context like "quarterback" and "backfield". The category network forms a directed acyclic graph (DAG). Some entities can share category through the network, e.g. "Barack Obama" with category "Democratic Party presidential nominees" shares the category "United States presidential candidates by party" with "Mitt Romney" when travelling two levels up the network.

(Bunescu and Pasca, 2006) propose to learn the category-context correlation at word level through category-word pair features. This method creates sparsity problem and does not scale well because the number of features grows linearly with both the number of categories and the vocabulary size. Moreover, the category network is somewhat noisy, e.g. travelling up four levels of the hierarchy can result in over ten thousand categories, with many irrelevant ones.

Rather than learning the correlation at word level, we explore a representation learning method that learns category-context correlation in the latent semantic space. Supervised Semantic Indexing (SSI) (Bai et al., 2009) is trained on query-document pairs to predict their degree of matching. The comparison is performed in the latent semantic space, so that synonymy and polysemy are implicitly handled by its inner mechanism. The score function between query q and document d is defined as:

$$f(q, d) = q^T W d \tag{6}$$

where W is learned with supervision like click-through data.

Given training data $\{(q_i, d_i)\}$, training is done by randomly sampling a negative target d^-. The model optimizes W such that $f(q, d^+) > f(q, d^-)$. Thus, the training objective is to minimize the following margin-based loss function:

$$\sum_{q,d^+,d^-} \max(0, 1 - f(q, d^+) + f(q, d^-)) \tag{7}$$

which is also known as contrastive estimation (Smith and Eisner, 2005).

W can become very large and inefficient when we have a big vocabulary size. This is addressed by replacing W with its low rank approximation:

$$W = U^T V + I \tag{8}$$

here, the identity term I is a trade-off between the latent space model and a vector space model. The gradient step is performed with Stochastic Gradient Descent (SGD):

$$\begin{aligned} U \leftarrow & U + \lambda V(d^+ - d^-)q^T, \\ & \texttt{if } 1 - f(q, d^+) + f(q, d^-) > 0 \end{aligned} \tag{9}$$

$$\begin{aligned} V \leftarrow & V + \lambda U q(d^+ - d^-)^T, \\ & \texttt{if } 1 - f(q, d^+) + f(q, d^-) > 0. \end{aligned} \tag{10}$$

where λ is the learning rate.

The query and document are not necessary real query and document. In our case, we treat our problem as: given the occurring context of an entity, retrieving categories corresponding to this entity. Thus, we use context as query q and the categories of this candidate entity as d. We also treat the definition page of an entity as its context, and first train the model with definition pages, because definition pages exhibit more focused topic. This considerably accelerates the training process. To reduce noise, We input the categories directly connected with one entity as a word vector. The input can be a TF-IDF vector or binary vector. We denote model trained with normalized TF-IDF and with binary input as cat_0 and cat_1 respectively.

4 Experiments

4.1 Datasets

Previous researches have used diverse datasets for evaluation, which makes it hard for comparison with others' approaches. TAC-KBP has several years of data for evaluating entity linking system, but is not well suited for evaluating collective approaches. Recently, (Hoffart et al., 2011) annotated a clean and much larger dataset AIDA [1] for collective approaches evaluation based on CoNLL 2003 NER dataset. (Ratinov et al., 2011) also refined previous work and contribute four publicly available datasets [2]. Thanks to their great works, we have enough data to evaluate against. According to the setting of (Hoffart et al., 2011), we split the AIDA dataset for train/development/test with 946/216/231 documents. We train a separate model on the Wikipedia training set for evaluating ACE/QUAINT/WIKI dataset (Ratinov et al., 2011). Table 2 gives a brief overview of the datasets used.

For knowledge base, we use the Wikipedia XML dump [3] to extract over 3.3 million entities. We use annotation from Wikipedia to build a name dictionary from mention string m to entity e for candidate generation, including redirects, disambiguation pages and hyperlinks, follows the approach of (Cucerzan, 2007). For candidate generation, we keep the top 30 candidates by popularity (Tbl. 1). Note that our name dictionary is different from (Ratinov et al., 2011) and has a much higher recall. Since (Ratinov et al., 2011) evaluate on "solvable" mentions and we have no way to recover those mentions, we re-implement their global features and the final scores are not directly comparable to theirs.

4.2 Methods under comparison

We compare our algorithm with several state-of-the-art collective entity disambiguation systems. The AIDA system proposed by (Hoffart et al., 2011) use a greedy graph cutting algorithm that iteratively remove entities with low confidence scores. (Han et al., 2011) employ personalized PageRank to propagate evidence between different decisions. Both algorithms use simple local features without discriminative training. (Kulkarni et al., 2009) propose to use integer linear programming (ILP) for inference. Except our re-implementation of Han's

[1] available at http://www.mpi-inf.mpg.de/yago-naga/aida/

[2] http://cogcomp.cs.illinois.edu/Data, we don't find the MSNBC dataset in the zip file.

[3] available at http://dumps.wikimedia.org/enwiki/, we use the 20110405 xml dump.

Dataset	ndocs	non-NIL	identified	solvable
AIDA dev	216	4791	4791	4707
AIDA test	231	4485	4485	4411
ACE	36	257	238	209(185)
AQUAINT	50	727	697	668(588)
Wikipedia	40	928	918	854(843)

Table 2: Number of mentions in each dataset. "identified" means the mention exists in our name dictionary and "solvable" means the true entity are among the top 30 candidates by popularity. Number in parenthesis shows the results of (Ratinov et al., 2011).

method, both AIDA and ILP solution are quite slow at running time. The online demo of AIDA takes over 10 sec to process one document with moderate size, while the ILP solution takes around 2-3 sec/doc. In contrast, our method takes only 0.3 sec/doc, and is easy to implement.

(Ratinov et al., 2011) also utilize a two layer learner architecture. The difference is that their method use top 1 candidate generated by local learner for global feature generation , while we search the top k candidates. Moreover, stacking is used to tackle the train/test mismatch problem in our model. We re-implement the global features of (Ratinov et al., 2011) and use our local predictor g_0 for level 0 predictor. Note that we only implement their global features concerning common inlinks and inter-connection (totally 9 features) for fair comparison because all other models don't use common outgoing links for global coherence.

4.3 Settings

We implement SVM^{rank} with an adaptation of linear SVM in scikit-learn (which is a wrapper of Liblinear). The category-context coherence model is implemented with Numpy configured with OpenBlas library, and we train this model on the entire Wikipedia hyperlink annotation. It takes about 1.5d for one pass over the entire dataset. The learning rate λ is set to 1e-4 and training cost before update is below 0.02.

Parameter tuning: there aren't many parameters to tune for both g_0 and g_1. The context document window size is fixed as 100 for compatibility with (Ratinov et al., 2011; Hoffart et al., 2011). The number of candidates is fixed to top 30 ranked by entity's popularity. Increase this value will generally boost recall at the cost of lower precision.

We introduce the following default parameter for global features in g_1. The number of fold for stacking is set to $\{1,5,10\}$ (see Table 4, default is 10; 1 means no stacking, i.e. training g_0 with all training data and generating level 1 features for training data directly with this g_0). The number k for searching neighboring entities with relational template is set to $\{1,3,5,7\}$ (e.g. in step 2 of Section 3.3 $k = 5$; default is 5).

For category-context modeling, the vocabulary sizes of context and category are set to top 10k and 6k unigrams by frequency. The latent dimension of low rank approximation is set to 200.

Performance measures: For all non-NIL queries, we evaluate performance with micro precision averaged over queries and macro precision averaged over documents. Mean Reciprocal Rank (MRR) is an information retrieval measure and is defined as $\frac{1}{|Q|}\sum_i^{|Q|}\frac{1}{rank_i}$, where $rank_i$ is the rank of correct answer in response to query i. For ACE/AQUAINT/WIKI we also give the accuracy of "solvable" mentions, but this is not directly comparable to (Ratinov et al., 2011). Our name dictionary is different from theirs and ours has a higher recall rate (Tbl. 2). Hence, the "solvable" set is different.

k	recall	k	recall
1	78.56	6	96.31
2	89.59	7	97.04
3	93.01	8	97.37
4	94.97	9	97.62
5	95.78	10	97.81

Table 3: Top k recall for local predictor g_0.

4.4 Discussions

Table 4 shows the evaluation results on AIDA dataset and Table 5 shows results on datasets ACE/AQUAINT/WIKI.

Effect of cat: The first group in Table 4 shows some baseline features for comparison. We can see even if the categories only carry incomplete and noisy information about an entity, it performs much

Methods	Devset			Testset		
	micro p@1	macro p@1	MRR	micro p@1	macro p@1	MRR
cosine	33.25	28.61	46.03	33.33	28.63	46.54
jaccard	44.71	36.56	57.76	45.66	36.89	57.08
cat_0	54.75	47.14	67.70	61.52	54.72	72.55
cat_1	60.15	54.64	72.98	65.46	61.04	76.84
popularity	69.21	67.59	79.26	69.07	72.63	79.45
g_0	76.04	73.63	84.21	76.16	78.17	84.58
g_0+global(Ratinov)	81.30	78.03	88.14	81.45	81.89	88.70
g_1+1fold	**82.01**	78.52	88.90	**83.59**	**83.58**	**90.05**
g_1+5fold	81.99	78.42	88.87	83.52	83.37	89.99
g_1+10fold	**82.01**	**78.53**	**88.91**	**83.59**	83.55	90.03
g_1+top1	81.65	**78.76**	88.51	81.81	82.55	89.06
g_1+top3	**82.20**	78.64	**88.98**	83.52	83.34	89.94
g_1+top5	82.01	78.57	88.90	83.63	**83.76**	90.05
g_1+top7	82.05	78.40	88.90	**83.75**	83.58	**90.08**
g_0+cat	79.36	76.14	86.66	79.64	80.47	87.32
g_1+cat	82.24	78.49	89.02	84.88	84.49	90.65
g_1+cat+all context	**82.99**	**78.56**	**89.51**	**86.49**	**85.11**	**91.55**
(Hoffart et al., 2011)	-	-	-	**82.29**	82.02	-
(Shirakawa et al., 2011)	-	-	-	81.40	**83.57**	-
(Kulkarni et al., 2009)	-	-	-	72.87	76.74	-
(Han et al., 2011)	-	-	-	78.97	75.77	-

Table 4: Performance on AIDA dataset. Maximal value in each group are highlighted with bold font. top k means up to k candidates are used for searching related instances with relational template.

better than word level features. Group 5 in Table 4 shows cat information generally boosts performance for both predictor g_0 and g_1.

Effect of stacking: Group 3 in Table 4 shows the results with different fold in stacking training. 1 fold means training g_0 with all training data and directly augment training data with this g_0. Surprisingly, we do not observe any substantial difference with various fold size. We deduce it is possible the way we fire global features with top k candidates that alleviates the problem of train/test mismatch when extending feature space for g_1. Despite the ranking of true entity can be lower in testset than in training set, the semantic coherence information can still be captured with searching over top k candidates.

Effect of top k global features: Group 4 in Table 4 shows the effect of k on g_1 performance. Clearly, increasing k generally improves precision and one possible reason is the improvement in recall when searching for related instances. Table 3 shows the top k recall of local predictor g_0. Further increasing k does not show any improvement.

Our method benefits from such a searching strategy, and consistently outperforms the global features of (Ratinov et al., 2011). While their method is a trade-off between expensive exact search over all mentions and greedy assigning all mentions with local predictor, we show this idea can be further extended, somewhat like increasing the beam search size without additional computational overhead. The only exception is the ACE dataset, since this dataset is so small, the difference translates to only one mention. One may notice the improvement on ACE/AQUAINT datasets is a little inconsistent. These datasets are much smaller and the results only differ within 4 mentions. Because these models are

Method	micro p@1	macro p@1	MRR	correct / solvable
		ACE		
g_0	77.43	81.30	79.03	95.22
Ratinov	77.43	80.70	78.81	95.22
g_1+5fold	77.04	79.85	78.96	94.74
g_0+cat	**77.82**	**81.48**	**79.31**	**95.69**
g_1+cat	77.43	80.16	79.25	95.22
		AQUAINT		
g_0	84.46	84.69	87.49	91.92
Ratinov	85.14	85.29	87.90	92.66
g_1+5fold	**85.83**	**85.55**	**88.27**	**93.41**
g_0+cat	85.01	85.00	87.89	92.51
g_1+cat	85.28	85.14	88.23	92.81
		Wikipedia test		
g_0	83.19	84.30	86.63	90.40
Ratinov	84.48	85.96	87.62	91.80
g_1+5fold	84.81	86.29	88.13	92.15
g_0+cat	84.38	86.13	87.51	91.69
g_1+cat	**85.45**	**87.16**	**88.31**	**92.86**

Table 5: Evaluation on ACE/AQUAINT/WIKI datasets.

trained on Wikipedia, the annotation style can be quite different.

Finally, as we analyze the development set of AIDA, we discover that some location entities rely on more distant information across the context, as we increase the context to the entire contextual document, we can gain extra performance boost.

4.5 Error analysis

As we analyze the development set of AIDA, we find some general problems with location names. Location name generally is not part of the main topic of one document. Thus, comparing context with its definition is not realistic. Most of the time, we can find some related location names in context; but other times, it is not easily distinguished. For instance, in "France beats Turkey in men's football..." France refers to "France national football team" but our system links it to the country page "France" because it is more popular. This can be addressed by modeling finer context (Sen, 2012) or local syntactic pattern (Hoffart et al., 2011). In other cases, our system misclassifies "New York City" for "New York" and "Netherlands" for "Holland" and "People's Republic of China" for "China", because in all these cases, the latter ones are the most popular in Wikipedia. It is even hard for us humans to tell the difference based only on context or global coherence.

5 Conclusions

We propose a stacking based collective entity linking method, which stacks a global predictor on top of a local predictor to collect coherence information from neighboring decisions. It is fast and easy to implement. Our method trades off between inefficient exact search and greedily assigning mention with local predictor. It can be seen as searching related entities with relational template in stacked graphical learning, with beam size k. Furthermore, we adopt recent progress in representation learning to learn category-context coherence model. It scales better than existing approaches on large knowledge base and performs comparison in the latent semantic space. Combining these two techniques, our model consistently outperforms all existing more sophisticated collective approaches in our experiments.

Acknowledgments

This research was partly supported by Major National Social Science Fund of China(No. 12&ZD227),National High Technology Research and Development Program of China (863 Program) (No. 2012AA011101) and National Natural Science Foundation of China (No.91024009).

References

B. Bai, J. Weston, D. Grangier, R. Collobert, O. Chapelle, and K. Weinberger. 2009. Supervised semantic indexing. In *The 18th ACM Conference on Information and Knowledge Management (CIKM)*.

R. Bunescu and M. Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of EACL*, volume 6, pages 9–16.

S. Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of EMNLP-CoNLL*, volume 6, pages 708–716.

X. Han, L. Sun, and J. Zhao. 2011. Collective entity linking in web text: a graph-based method. In *Pro-*

ceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval, pages 765–774. ACM.

J. Hoffart, M.A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792. Association for Computational Linguistics.

Heng Ji, Ralph Grishman, Hoa Trang Dang, Kira Griffitt, and Joe Ellis. 2011. Overview of the tac 2011 knowledge base population track. In *Proceedings of the Fourth Text Analysis Conference*.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM.

S.S. Kataria, K.S. Kumar, R. Rastogi, P. Sen, and S.H. Sengamedu. 2011. Entity disambiguation with hierarchical topic models. In *Proceedings of KDD*.

Zhenzhen Kou and William W Cohen. 2007. Stacked graphical models for efficient inference in markov random fields. In *SDM*.

S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti. 2009. Collective annotation of wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 457–466. ACM.

J. Lehmann, S. Monahan, L. Nezda, A. Jung, and Y. Shi. 2010. Lcc approaches to knowledge base population at tac 2010. In *Proc. TAC 2010 Workshop*.

F. Li, Z. Zheng, F. Bu, Y. Tang, X. Zhu, and M. Huang. 2009. Thu quanta at tac 2009 kbp and rte track. In *Proceedings of Test Analysis Conference 2009 (TAC 09)*.

André FT Martins, Dipanjan Das, Noah A Smith, and Eric P Xing. 2008. Stacking dependency parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 157–166. Association for Computational Linguistics.

D. Milne and I.H. Witten. 2008. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 509–518. ACM.

L. Ratinov, D. Roth, D. Downey, and M. Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the Annual Meeting of the Association of Computational Linguistics (ACL)*.

P. Sen. 2012. Collective context-aware topic models for entity disambiguation. In *Proceedings of the 21st international conference on World Wide Web*, pages 729–738. ACM.

M. Shirakawa, H. Wang, Y. Song, Z. Wang, K. Nakayama, T. Hara, and S. Nishio. 2011. Entity disambiguation based on a probabilistic taxonomy. Technical report, Technical Report MSR-TR-2011-125, Microsoft Research.

N.A. Smith and J. Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 354–362. Association for Computational Linguistics.

Weiwei Sun. 2011. A stacked sub-word model for joint chinese word segmentation and part-of-speech tagging. In *ACL*, pages 1385–1394.

David H Wolpert. 1992. Stacked generalization. *Neural networks*, 5(2):241–259.

Zhicheng Zheng, Fangtao Li, Minlie Huang, and Xiaoyan Zhu. 2010. Learning to link entities with knowledge base. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 483–491, Los Angeles, California, June. Association for Computational Linguistics.