A Wikipedia-LDA Model for Entity Linking with Batch Size Changing Instance Selection

Wei Zhang[†] **Jian Su**[‡] **Chew Lim Tan**[†]

[†]School of Computing
National University of Singapore
{z-wei,tancl}@comp.nus.edu.sg

[‡] Institute for Infocomm Research
sujian@i2r.a-star.edu.sg

Abstract

Entity linking maps name mentions in context to entries in a knowledge base through resolving the name variations and ambiguities. In this paper, we propose two advancements for entity linking. First, a Wikipedia-LDA method is proposed to model the contexts as the probability distributions over Wikipedia categories, which allows the context similarity being measured in a semantic space instead of literal term space used by other studies for the disambiguation. Furthermore, to automate the training instance annotation without compromising the accuracy, an instance selection strategy is proposed to select an informative, representative and diverse subset from an auto-generated dataset. During the iterative selection process, the batch sizes at each iteration change according to the variance of classifier's confidence or accuracy between batches in sequence, which not only makes the selection insensitive to the initial batch size, but also leads to a better performance. The above two advancements give significant improvements to entity linking individually. Collectively they lead the highest performance on *KBP-10* task. Being a generic approach, the batch size changing method can also benefit active learning for other tasks.

1 Introduction

Knowledge base population (KBP)[1] involves gathering information scattered among the documents of a large collection to populate a knowledge base (KB) (e.g. Wikipedia). This requires either linking entity mentions in the documents with entries in the KB or highlighting these mentions as new entries to current KB.

Entity linking (McNamee and Dang, 2009) involves both finding name variants (e.g. both *"George H. W. Bush"* and *"George Bush Senior"* refer to the 41st U.S. president) and name disambiguation (e.g. given *"George Bush"* and its context, we should be able to disambiguate which president it is referring to).

Compared with Cross-Document Coreference (Bagga and Baldwin, 1998) which clusters the articles according to the entity mentioned, entity linking has a given entity list (i.e. the reference KB) to which we disambiguate the entity mentions. Moreover, in the articles, there are new entities not present in KB.

For name disambiguation in entity linking, there has been much previous work which demonstrates modeling context is an important part of measuring document similarity. However, the traditional approach for entity linking treats the context as a bag of words, n-grams, noun phrases or/and co-occurring named entities, and measures context similarity by the comparison of the weighted literal term vectors (Varma *et al*., 2009; Li *et al*., 2009; McNamee *et al*., 2009; Zhang *et al*., 2010; Zheng *et al*., 2010; Dredze *et al*., 2010). Such literal matching suffers from sparseness issue. For example, consider the following four observations of *Michael Jordan* without term match:

1) Michael Jordan is a leading researcher in machine learning and artificial intelligence.

2) Michael Jordan is currently a full professor at the University of California, Berkeley.

3) Michael Jordan (born February, 1963) is a former American professional basketball player.

4) Michael Jordan wins NBA MVP of 91-92 season.

To measure the similarity of these contexts, the semantic knowledge underlying the words is needed.

Furthermore, current state-of-the-art entity linking systems (Dredze *et al*., 2010; Zheng *et al*., 2010) are based on supervised learning approach requiring lots of annotated training instances to achieve good performance. However, entity linking annotation is highly dependent on the KB. When a new KB comes, the annotating process needs to be repeated. We have tried to automate this annotating process (Zhang *et al*. 2010). However, as discussed in that paper, the distribution of the auto-generated data is not con-

[1] http://nlp.cs.qc.cuny.edu/kbp/2010/

sistent with the real dataset, because only some types of instances can be generated.

In this paper, we propose two approaches: (1) a Wikipedia-LDA model to effectively mine the semantic knowledge from the contexts of the mentions. Such topic model allows us to measure the similarity between articles and KB entries in the semantic space of Wikipedia category. (2) An instance selection strategy to effectively utilize the auto-generated annotation through an iterative process of selecting a representative, informative and diverse batch of instances at each iteration. The batch sizes at each iteration change according to the variance of classifier's confidence or accuracy between batches in sequence, which makes selection insensitive to the initial batch size and performs better than fixed size.

We conduct evaluation on *KBP-10* data (Ji *et al.,* 2010). Experiments show that the Wikipedia-LDA model is able to effectively capture the underlying semantic information and produce statistically significant improvement over literal matching alone. Correspondingly, instance selection can make the dataset more balanced and it also produces a significant gain in entity linking performance. Collectively, the two advancements lead the highest performance on *KBP-10* task. Being a generic approach, the batch size changing method proposed in this paper can also benefit active learning for other tasks.

The remainder of this paper is organized as follows. Section 2 introduces the framework for entity linking. We present our Wikipedia-LDA model in Section 3, and the instance selection in Section 4. Section 5 shows the experiments and discussions. Section 6 concludes our work.

2 Entity Linking Framework

Entity linking is done through two steps: name variation resolution and name disambiguation. Name variation resolution finds variants for each entry in KB and then generates the possible KB candidates for the given name mention by string matching. Name disambiguation is to map a mention to the correct entry in the candidate set.

2.1 Name Variation Resolution

Wikipedia contains many name variants of entities like confusable names, spelling variations, nick names, etc. We extract the name variants of an entry in KB by leveraging the knowledge sources in Wikipedia: "titles of entity pages", "disambiguation pages"[2], "redirect pages"[3] and "anchor texts". With the acquired name variants for entries in KB, the possible KB candidates for a given name mention can be retrieved by string matching. If the given mention is an acronym, we will expand it from the given article, and then use entity linking process.

2.2 Name Disambiguation

First, using a learning to rank method, we rank all the retrieved KB candidates to identify the most likely candidate. In this learning to rank method, each name mention and the associated candidates are formed by a list of feature vectors. During linking, the score for each candidate entry is given by the ranker. The learning algorithm we used is ranking SVM (Herbrich *et al.*, 2000).

Next, the preferred KB candidate is presented to a binary classifier (Vapnik, 1995) to determine if it is believed as the target entry for a name mention. From here, we can decide whether the mention and top candidate are linked. If not, the mention has no corresponding entry in KB (*NIL*). The base features adopted for both learning to rank and classification include 15 feature groups divided to 3 categories. A summary of the features is listed in Table 1. Due to the space limit, we only show the feature name, leaving out the feature details which can be found in (Dredze *et al.*, 2010; Zheng *et al.*, 2010).

Categories	Feature Names
Surface	Exact Equal Surface, Start With String of Query, End With String of Query, Equal Word Num, Miss Word Num
Contextual	TF-IDF Similarity, Similarity Rank, All Words in Text, NE Number Match, Country in Text Match, Country in Text Miss, Country in Title Match, Country in Title Miss, City in Title Match
Others	NE Type

Table 1: Base Feature Set

3 Wikipedia-LDA Model

In the similar task cross-document coreference (Han and Zhao 2009) and other tasks (e.g. text classification) (Wang and Domeniconi, 2008), Wikipedia concepts are used to model the text. Wikipedia concept is a kind of entity-level topic. In our approach, we use the cross-entity topic Wikipedia Categories to represent the semantic knowledge.

[2] http://en.wikipedia.org/wiki/Wikipedia:Disambiguation
[3] http://en.wikipedia.org/wiki/Wikipedia:Redirect

Thus, we model the contexts as the distributions over Wikipedia categories. Then, the similarity between the contexts can be measured in a semantically meaningful space. Finally, such semantic similarity, together with other base features, is incorporated in the trainable models to learn the ranker and classifier.

3.1 Modeling the Contexts as Distributions over Wikipedia Categories

Wikipedia requires contributors to assign categories to each article, which are defined as "major topics that are likely to be useful to someone reading the article". Thus, Wikipedia can serve as a document collection with multiple topical labels, where we can learn the posterior distribution over words for each topical label (i.e. Wikipedia category). Then, from the observed words in the mention's context and KB entry, we can estimate the distribution of the contexts over the Wikipedia categories. To obtain this distribution, we use a supervised Latent Dirichlet Allocation (LDA) model – labeled LDA defined by Ramage *et al.* (2009), which represents state-of-the-art method for multi-labeled text classification. It performs better on collections with more semantically diverse labels, which we need in order to leverage on the large semantically diverse categories from Wikipedia as the topical labels.

Figure 1 shows us a graphical representation of the labeled LDA for the multi-labeled document collection. Labeled LDA is a three level hierarchical Bayesian model. β is the multinomial distribution over words for a Wikipedia category, which has a Dirichlet prior with hyperparameter η. Both the category set Λ as well as the topic prior α influence the topic mixture θ. These distributions can be used to generate documents in the form of a collection of words (w). D is the number of documents, N is the document length and K is the number of categories.

After the model is trained by Wikipedia data, the distributions of KB entry and the article over K categories are estimated by calculating the topic proportions θ. θ is given by an EM procedure that treats θ as a parameter with Z missing.

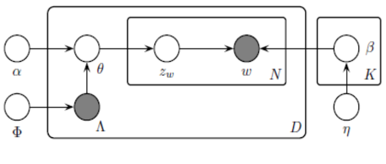

Figure 1: Graphical model of Labeled LDA

3.2 Context Similarity

We have mapped the contexts to a K-dimensional semantic space. Thus, we can calculate the context similarity by their distance in this space. To measure the context similarity in the K-dimensional topical space, we calculate the Cosine value as below:

$$Similarity_{d,e} = \frac{\sum_{k=1}^{K} \theta_{d,k} \times \theta_{e,k}}{\sqrt{\sum_{k=1}^{K} (\theta_{d,k})^2} \times \sqrt{\sum_{k=1}^{K} (\theta_{e,k})^2}} \quad (1)$$

Where d means the document with the name mention and e means the KB entry.

Such semantic similarity can be further combined with other term matching features for SVM ranker and classifier of entity linking.

3.3 Wikipedia Category Selection

Each article in Wikipedia is assigned several categories by the contributors as requested. However, from our observation some categories in Wikipedia may not be suitable to model the topics of a document. Thus, we shall consider selecting an appropriate subset from the Wikipedia categories to effectively model the contexts. We examined five possible category subsets: **All**, **All-admin**, **isa_all**, **isa_class**, and **isa_instance**.

Wikipedia contains 165,744 categories. This is the set **All.**

There are some meta-categories used for encyclopedia management in Wikipedia, e.g. "*Wikipedia editing guidelines*", which are unsuitable to describe the topics of a document. Thus, we remove the categories which contain any of the following strings: *wikipedia*, *wikiprojects*, *lists*, *mediawiki*, *template*, *user*, *portal*, *categories*, *articles* and *pages*. This leaves 127,325 categories (**All-admin**).

However, some categories such as "*people by status*" and "*Geography by place*" in the **All-admin** set cannot serve as the topics of a document properly. Thus, we need to remove them from the category set. From our observation, the topical categories are usually in *is-a* relation. For example, the relation between the two topical categories "*Olympic basketball players*" and "*Olympic competitors*" is an *is-a* relation, while the categories to be removed "*people by status*" and "*Geography by place*" are not in any *is-a* relation. We thus only select the categories connected by *is-a* relation to **isa_all** subset.

Since the categories are connected by unlabeled links in Wikipedia, we need to identify *is-a* relation links. We use the four methods as below proposed by Ponzetto and Strube (2007) to distinguish *is-a* and *not-is-a* relation links.

We first use a syntax-based method: assign *is-a* to the link between two categories if they share the same lexical head lemma (e.g. "*British Computer Scientists*" and "*Computer Scientists*").

Then, we use structural information from the category network: (1) for a category *c*, look for a Wikipedia article *P* with the same name. Take all *P*'s categories whose lexical heads are plural nouns $CP =\{cp_1, cp_2, \ldots, cp_n\}$. Take all supercategories of *c*, $SC=\{sc_1, sc_2, \ldots,sc_k \}$. If the head lemma of one of cp_i matches the head lemma of sc_j, label the relation between *c* and sc_j as *is-a*. (2) assign *is-a* label to the link between two categories if a Wikipedia article is redundantly categorized under both of them. For example, "*Internet*" is categorized under both "*Computer networks*" and "*Computing*" and there is a link between "*Computer networks*" and "*Computing*". Then this link is assigned *is-a*.

Next, we use lexical-syntactic patterns in a corpus. This method uses two sets of patterns. One set is used to identify *is-a* relations (Caraballo, 1999; Hearst, 1992), for example "*such* NP_1 *as* NP_2", NP_1 and NP_2 are the values of categories and their subcategories respectively. The second set is used to identify *not-is-a* relations. For example "NP_1 *has* NP_2", where the link between NP_1 and NP_2 will be assigned *not-is-a*. These patterns are used with a corpus built from Wikipedia articles, and separately with the Tipster corpus (Harman and Liberman, 1993). The label is assigned by majority voting between the frequency counts for the two types of patterns.

Finally, we assign *is-a* labels to links based on transitive closures **-** all categories along an *is-a* chain are connected to each other by *is-a* links.

Another fact is that the categories defined by Wikipedia are not all classes. For example, "*Microsoft*" is an instance of the class "*Computer and Video Game Companies*", and it appears both as an article page and as a category in Wikipedia. We would like to further examine the two different subsets: **isa_class**, and **isa_instance** in **isa_all** set for entity linking**.** To distinguish instance and class in **isa_all** set, we use a structure-based method (Zirn *et al*., 2008). The categories which have other subcategories or Wikipedia articles connected to them by *is-a* relation are assigned *class* label. In our problem, the remaining categories are approximately regarded as instances.

4 Instance Selection Strategy

In this section, we explore a method to effectively utilize a large-scale auto-generated data for entity linking.

In our pervious work (Zhang *et al*. 2010), we proposed automatically gathering large-scale training instances for entity linking. The basic idea is to take a document with an unambiguous mention referring to an entity *e1* in KB and replace it with its variation which may refer to *e1*, *e2* or others. For example, a mention "*Abbott Laboratories*" in a document only refers to one KB entry "*Abbott Laboratories*". "*Abbott Laboratories*" in the document is replaced with its ambiguous synonyms, including *"Abbott" "ABT"*, etc. Following this approach, from the 1.7 million documents in *KBP-10* text collection, we generate 45,000 instances.

However, the distribution of the auto-generated data is not consistent with the real dataset, since the data generation process can only create some types of training instances. In the case of *"Abbott Laboratories"*, more than ten "*Abbott*" mentions are linked to "*Abbott Laboratories*" entry in KB, but no "*Abbott*" example is linked to other entries like "*Bud Abbott*" "*Abbott Texas*", etc. Thus, we need an instance selection approach to reduce the effect of this distribution problem. However, the traditional instance selection approaches (Brighton and Mellish, 2002; Liu and Motoda, 2002) only can solve two problems: 1) a large dataset causes response-time to become slow 2) the noisy instances affect accuracy, which are different from our needs here. We thus propose an instance selection approach to select a more balanced subset from the auto-annotated instances. This instance selection strategy is similar to active learning (Shen *et al*., 2004; Brinker, 2003) for reducing the manual annotation effort on training instances through proposing only the useful candidates to annotators. As we already have a large set of auto-generated training instances, the selection here is a fully automatic process to get a useful and more balanced subset instead.

We use the SVM classifier mentioned in Section 2.2 to select the instances from the large dataset. The initial classifier can be trained on a set of initial training instances, which can be a small part of the whole auto-generated data, or the limited manual annotated training instances available, e.g. those provided by *KBP-10.*

Our instance selection method is an iterative process. We select an informative, representative and diverse batch of instances based on current hyperplane and add them to the current training instance set at each iteration to further adjust the hyperplane for more accurate classification.

We use the distance as the measure to select informative instances. The distance of an instance's feature vector to the hyperplane is computed as follows:

$$Dist(w) = \left|\sum_{i=1}^{N} \alpha_i y_i k(s_i, w) + b\right| \quad (2)$$

Where w is the feature vector of the instance, α_i, y_i and s_i correspond to the weight, class and feature vector of the i^{th} support vector respectively. $\boldsymbol{N}$ is the number of the support vectors.

Next, we quantify the representativeness of an instance by its density. Such density is defined as the average similarity between this instance and all other instances in the dataset. If an instance has the largest density among all the instances in the dataset, it can be regarded as the centroid of this set and also the most representative instance.

$$Density(w_i) = \frac{\sum_{j\neq i} Sim(w_i, w_j)}{N-1} \quad (3)$$

Where w is the instance in the dataset and N is the size of dataset. *Sim* is cosine similarity.

We combine the informativeness and representativeness by the function $\lambda(1 - Dist(w)) + (1-\lambda)Density(w)$, in which *Dist* and *Density* are normalized first. The individual importance of each part in this function is adjusted by a tradeoff parameter λ (set to 0.5 in our experiment). The instance with the maximum value of this function will be selected first to the batch. This instance will be compared individually with the selected instances in current batch to make sure their similarity is less than a threshold β. This is to diversify the training instance in the batch to maximize the contribution of each instance. We set β to the average similarity between the instances in the original dataset. When a batch of α instances is selected, we add them to the training instance set and retrain the classifier.

Such a batch learning process will stop at the peak confidence of the SVM classifier, since Vlachos (2008) shows that the confidence of the SVM classifier is consistent with its performance. The confidence can be estimated as the sum of the distances to hyperplane for the instances of an un-annotated development set. The development set guides the selection process to solve the distribution problem mentioned above. Alternatively, we can also leverage on some annotated development data and use accuracy instead to guide the selection process. We explore both approaches for different application scenarios in our experiments.

We now need to decide how to set the batch size α at each iteration. It is straightforward to set a fixed batch size α (**Fixed Number**), which never changes during the process. However, there are some limitations as demonstrated in our experiment in this paper. First, the performance is sensitive to the batch size. Second, if we set the batch size too big, it will impede further improvement allowed by small batch size. But if we set the batch size too small from the beginning, it will dramatically increase the number of iterations needed which will make the selection too slow. To resolve the above issues, we change the batch size according to the variance of classifier's confidence on an un-annotated set. Thus, we assign an integer to α_1 and α_2 in the first two iterations, and α_i $(i > 2)$ in the i^{th} iteration is computed as below (**Flexible Number)**:

$$\alpha_i = \frac{\alpha_{i-1} * (con_{i-1} - con_{i-2})}{con_{i-2} - con_{i-3}} \quad (4)$$

where con_i is the confidence of the classifier on the un-annotated dataset at i^{th} iteration.

Figure 2 summarizes the selection procedure.

```
Given: Initial Training Set $T=\{T_1,T_2...T_m\}$,
       Original Set to be selected $A=\{A_1,A_2...A_n\}$,
       Batch Set with the maximal size α.
Initialization: Batch Set = ∅
Loop until the confidence/accuracy of the classifier on
a development set does not increase
    Train a Classifier on $T$
    Batch Set=∅
    Update α according to Equation 4
    Loop until Batch Set is full
      • Select $A_i$ with the maximal value $P$ from $A$
        $P = \lambda(1 - Dist(w)) + (1-\lambda)Density(w)$
      • RepeatFlag=false;
      • Loop for each $A_k$ in Batch Set
            If $Sim(A_i,A_k) > \beta$ Then
                RepeatFlag=true
                Stop the Loop
      • If RepeatFlag==false Then
            Add $A_i$ to Batch Set
      • Remove $A_i$ from $A$
    $T = T \cup Batch\ Set$
```
Figure 2: Instance Selection Strategy

5 Experiments and Discussions

5.1 Experimental Setup

In our study, we use *KBP-10* knowledge base and document collection to evaluate our approach for entity linking. The KB is auto-generated from Wikipedia. The KB contains

Features	ALL	NIL	Non-NIL	ORG	GPE	PER
Base Features	83.2	88.2	77.2	82.1	75.1	92.5
Base + All	84.0	88.6	78.5	84.0	76.0	92.1
Base + All - admin	84.9	88.9	80.0	84.9	76.9	92.8
Base + isa_all	**85.9**	89.1	82.0	85.2	78.6	93.8
Base + isa_class	85.5	88.8	81.3	84.9	78.0	93.2
Base+isa_instance	83.9	88.9	77.8	82.9	76.6	92.1

Table 2: Results of Entity Linking for Semantic Features

818,741 different entries and the document collection contains 1.7 million documents. Each KB entry consists of the Wikipedia Infobox[4] and the corresponding Wikipedia page text. The test data has 2,250 mentions across three named entity types: Person (PER), Geo-Political Entity (GPE) and Organization (ORG). The documents containing these mentions are from newswire and blog text. The training set consists of 3,904 newswire mentions and 1,500 web mentions. In order to leverage name variant information mentioned in Section 2.1 and category network mentioned in Section 3.3, we further get Wikipedia data directly from Wikipedia website[5]. The version we use is released on Oct. 08, 2008.

For pre-processing, we perform sentence boundary detection derived from Stanford parser (Klein and Manning, 2003), named entity recognition using a SVM based system trained and tested on ACE 2005 with 92.5(P) 84.3(R) 88.2(F), and co-reference resolution using a SVM based resolver trained and tested on ACE 2005 with 79.5%(P), 66.7%(R) and 72.5%(F). In our implementation, we use the binary SVM^{Light} developed by Joachims (1999) and SVM^{Rank} developed by Joachims (2006). The classifier and ranker are trained with default parameters. The Stanford Topic Model Toolbox[6] is used for Labeled-LDA with default learning parameters.

We adopt micro-averaged accuracy used in *KBP-10* to evaluate our Entity Linker, i.e. the number of correct links divided by the total number of mentions.

5.2 System with Wikipedia-LDA

Table 2 lists the performance of entity linking with overall accuracy (*ALL*) as well as accuracy on subsets (*Nil, Non-Nil, ORG,GPE* and *PER*) of the data. In the first row, only base features described in Section 2.2 are used. This baseline system models the contexts with literal terms. The second to sixth rows report the results combining base features with semantic knowledge (i.e. the context similarity is computed by the five different subsets of Wikipedia categories mentioned in Section 3.3).

American novels	American film actors	Members of the National Academy of Sciences	American basketball players
novel	role	prize	**nba**
book	actor	**researcher**	**basketball**
story	films	**professor**	points
paperback	appeared	science	rebounds
plot	actress	nobel	games
print	television	institute	draft
edition	hollywood	theory	guard
isbn	california	physics	overall
hardback	roles	received	coach
characters	movie	sciences	**professional**
published	acting	medal	assists
man	married	chemistry	play
father	death	academy	**season**
love	character	award	forward
written	starred	ph. d	ncaa

Table 3: Sample Wikipedia Categories and Corresponding Top 15 Words

We see that all the five systems with semantic features perform better than the baseline system, which models the context similarity as literal term matching. Especially, the **isa_all** and **isa_class** can achieve significantly better result than the baseline ($\rho < 0.05$, χ^2 test). These results prove that the semantic knowledge underlying the contexts has good disambiguation power for entity linking. Table 3 tells the reason of the improvements. Table 3 shows us four sample Wikipedia categories and top 15 highly probable words identified by the topic model for these categories. The topic model successfully assigns a high probability to the words *"researcher"* and *"professor"* in the category *"Members of the National Academy of Sciences"*, and assign a high probability to the words "*nba*" "*basketball*" "*professional*" and "*season*" in the category *"American basketball players"*. Such semantic

[4] http://en.wikipedia.org/wiki/Template:Infobox
[5] http://download.wikipedia.org
[6] http://nlp.stanford.edu/software/tmt/tmt-0.3/

Methods	ALL	NIL	Non-NIL	ORG	GPE	PER
Auto_Gen	81.2	81.8	80.5	80.8	72.5	90.3
Auto_Gen+IS	**85.2**	87.5	82.5	84.4	78.5	92.8
KBP	83.2	88.2	77.2	82.1	75.1	92.5
KBP+Auto_Gen	82.2	83.8	80.4	81.7	75.6	89.5
KBP+Auto_Gen+IS	**85.5**	87.7	82.9	84.7	78.9	92.8

Table 4: Results of Entity Linking for Instance Selection

knowledge learned from Wikipedia data is helpful in the example of *"Michael Jordan"* mentioned in Section 1. This shows that entity linking can benefit from the semantic information underlying the words and overcome the shortcomings of literal matching.

We further compare the performances of the five different category subsets. From the last five rows of Table 2, we can see that **isa_all** subset performs best among the five subsets for disambiguation. This should be because **isa_all** includes more categories than **isa_class** and **isa_instance**, and thus can capture more semantic information. However, although **All** and **All-admin** include even more categories, they introduce many categories which are unsuitable to model the topics of a news article or blog text, such as the two categories mentioned in Section 3.3, "*people by status*" which is not in an *is-a relation* and "*Wikipedia editing guidelines*" which is used for encyclopedia management.

5.3 System with Instance Selection

Table 4 shows the results for evaluating our instance selection strategy. These experiments use the base features (Section 2.2).

5.3.1 With and Without Manual Annotated Data

We want to find out the effectiveness of our instance selection strategy if no manually annotated data is available. In the first block of Table 4, we compare the performances of the systems with and without instance selection. "*Auto_Gen*" uses the auto-generated dataset described at the beginning of Section 4 as the training set directly, and "*Auto_Gen+IS*" applies our instance selection to the auto-generated data for training. In the instance selection process, we use the KB entries with more than 15 linked documents in the auto-generated data as our Initial Training Set (1,800 instances) to train a classifier, and then use this classifier to select instance from the auto-generated dataset. The first block of Table 4 shows that our instance selection gives significant improvements ($\rho < 0.05$, χ^2 test). These improvements show our selection strategy makes the training set more balanced and it can effectively reduce the effect of distribution problem in the large scale dataset.

We further evaluate our instance selection strategy when a large manually annotated data is available in the second block of Table 4. *"KBP"* is trained on the manually annotated *KBP-10* training set. "*KBP+Auto_Gen*" is trained on *KBP-10* set and the auto-generated set. *"KBP+Auto_Gen+IS"* uses *KBP-10* training set as the Initial Training Set, and applies instance selection process to the auto-generated data. Comparing "*KBP+ Auto_Gen*" with "*KBP*", we can see that the unbalanced distribution caused serious problem which even pull down the performance achieved by the large manual annotation alone. The experiment results of "*KBP*" and "*KBP+Auto_Gen+IS*" show that our instance selection strategy appears very necessary to bring further improvements over the large manually annotated dataset (5,404 instances). These significant ($\rho < 0.05$, χ^2 test) improvements are achieved by incorporating more training instances in a reasonable way.

Comparing the performance of "*Auto_Gen+IS" with "KBP"* in Table 4, we can find that our method performs better without hard intensive work on annotating 5,404 articles. This proves that using our instance selection can save labor without compromise of entity linking accuracy. The pretty much same performance of "*Auto_Gen+IS" with "KBP +Auto_Gen+IS"* also confirms the above conclusion.

5.3.2 Fixed Size Vs. Changing Size

We are also interested in the effectiveness of the two schemes (i.e. Fixed Number and Flexible Number) of setting the batch size α mentioned in Section 4. In Figure 3, we set the batch size α in Fixed Number scheme and α_1 α_2 in Flexible Number scheme, to different numbers from 50 to 140 increasing 10 each time. We conduct instance selection to the auto-generated data. Figure 3 shows that flexible batch size outperforms the fixed size for entity linking. Especially, the improvement at α=50, 60 and 70 is significant ($\rho < 0.05$, χ^2 test). This proves that batch size should be in line with the variance of the classifier's confidence at each iteration of instance selection. Furthermore, in this Figure, the performance of flexible batch size is more stable than the Fixed Number scheme. This shows that Flexible Number scheme makes the entity linking system insensitive to the initial batch size during instance selection process. Thus the initial batch size of the experiments in Table 4 is set to 80, which we believe that very similar performance can be achieved even with a different initial size. Another fact is that the selection process is similar to active learning, which needs to manually annotate the selected instances in each batch. Thus, being a generic approach, the batch size changing method proposed in this paper can also benefit active learning for other tasks.

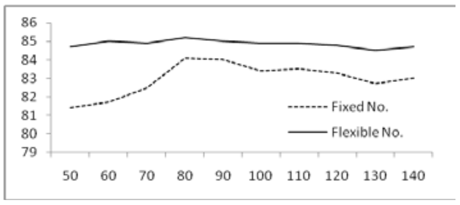

Figure 3: Performance Curves for Two Batch Size Schemes

5.3.3 (Un-)Annotated Development Set

In the above study, we directly use the test set without annotations as the development set for instance selection to optimize our solution to the application data. Such an approach will be useful when the application set is available in advance as in the case with *KBP* benchmarks.

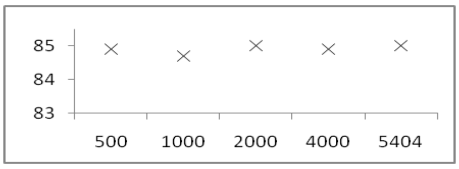

Figure 4: Annotated Development data

When the application set is unavailable beforehand, in other words, the articles to be linked only arrive one after the other in linking stage, we leverage on the accuracy on annotated development set for the instance selection. Figure 4 shows the performances on different sizes of annotated development set. The results show that the different sizes contribute more or less same performances. We only need to use a small amount of annotated development data, 500 articles in our study to guide the instance selection to achieve similar performance as with unannotated test set being development data.

5.4 Overall Result Combining Two Approaches

We also evaluate our model which combines the Wikipedia- LDA and Instance Selection together on *KBP-10* data, and compare our method with the top 7 systems in *KBP-10* shared task (Ji *et al*., 2010). As shown in Figure 5, the first column is the performance of our system for entity linking, which outperforms the best solution[7] in *KBP-10* shared task.

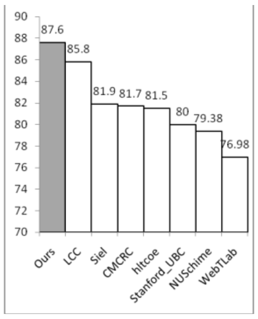

Figure 5: A Comparison with *KBP-10* Systems

6 Conclusion

In our paper, we explored using two innovative approaches for entity linking. We proposed a Wikipedia-LDA to entity linking, which can discover the semantic knowledge underlying the contexts. We also investigated the effectiveness of five subsets of Wikipedia categories to model the contexts. Furthermore, we proposed a batch size changing instance selection strategy to reduce the effect of distribution problem in the auto-generated data. It makes entity linking system achieve state-of-the-art performance without hard labor. Meanwhile, the flexible batch size not only makes the selection insensitive to the initial batch size, but also leads to a better performance than the fixed batch size. The above two advancements significantly improve entity linking system individually, and collectively they lead the highest performance on *KBP-10* task.

Acknowledgment

This work is partially supported by Microsoft Research Asia eHealth Theme Program.

[7] Another system submission shows 86.8%. However, it accesses web which is not allowed in *KBP* benchmark as the purpose to develop a standalone system, which is our focus here as well.

References

A. Bagga and B. Baldwin. 1998. Entity-Based Cross-Document Coreferencing Using the Vector Space Model. *36th Annual Meeting of the Association of Computational Linguistics*. 1998.

H. Brighton and C. Mellish. 2002. Advances in Instance Selection for Instance-Based Learning Algorithms. *Data Mining and Knowledge Discovery.*

K. Brinker. 2003. Incorporating Diversity in Active Learning with Support Vector Machines. In *Proceeding of ICML*. 2003.

S. A. Caraballo. 1999. Automatic construction of a hypernym-labeled noun hierarchy from text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, College Park, Md., 20-26 June. 1999.

M. Dredze, P. McNamee, D. Rao, A. Gerber and T. Finin. 2010. Entity Disambiguation for Knowledge Base Population. *23rd International Conference on Computational Linguistics (COLING 2010),* August 23-27, 2010, Bejing, China

X. Han and J. Zhao. 2009. Named Entity Disambiguation by Leveraging Wikipedia Semantic Knowledge. *Proceeding of the 18th ACM conference on Information and knowledge management* (2009).

D. Harman and M. Liberman. 1993. *TIPSTER Complete. LDC93T3A, Philadelphia, Penn. Linguistic Data Consortium* , 1993.

M. A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 15th International Conference on Computational Linguistics,* Nantes, France, 23-28 August 1992.

R. Herbrich, T. Graepel and K. Obermayer. 2000. Obermayer. Large Margin Rank Boundaries for Ordinal Regression. *Advances in Large Margin Classifiers (pp. 115-132).* 2000.

H. Ji, R. Grishman, H. Dang, K. Griffitt, and J. Ellis. 2010. Overview of the TAC 2010 Knowledge Base Population Track. In Proceedings of *Text Analysis Conference 2010.*

T. Joachims. 1999. Making large-Scale SVM Learning Practical. *Advances in Kernel Methods - Support Vector Learning,* MIT Press, 1999.

T. Joachims. 2006. Training Linear SVMs in Linear Time, *The ACM Conference on Knowledge Discovery and Data Mining (KDD),* 2006.

D. Klein and C. D. Manning. 2003. Fast Exact Inference with a Factored Model for Natural Language Parsing. In *Advances in Neural Information Processing Systems 15 (NIPS 2002),* Cambridge, MA: MIT Press, pp. 3-10.

F. Li, Z Zheng, F Bu, Y Tang, X Zhu, and M Huang. 2009. THU QUANTA at TAC 2009 KBP and RTE Track. *Text Analysis Conference 2009 (TAC 09).*

H. Liu and H. Motoda. 2002. On Issues of Instance Selection. 2002. *Data Mining and Knowledge Discovery*, 6, 115-130. 2002.

P. McNamee and H. T. Dang. 2009. Overview of the TAC 2009 Knowledge Base Population Track. In *Proceeding of Text Analysis Conference 2009*

P. McNamee *et al*. 2009. HLTCOE Approaches to Knowledge Base Population at TAC 2009. *In Proceedings of Text Analysis Conference 2009 (TAC 09).*2009.

S. P. Ponzetto and M. Strube. 2007. Deriving a Large Scale Taxonomy from Wikipedia. In *Proceedings of the 22nd National Conference on Artificial Intelligence*, Vancouver, B.C., 22-26 July, 2007, pp. 1440-1447.

D. Ramage, D. Hall, R. Nallapati and C. D. Manning. 2009. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In Proceedings *of the 2009 Conference on Empirical Methods in Natural Language Processing*, 2009.

D. Shen, J. Zhang, J. Su, G. D. Zhou and C. L. Tan. 2004. Multi-Criteria-based Active Learning for Named Entity Recognition. In *Proceedings of the ACL 2004.*

V. Vapnik. 1995. The Nature of Statistical Leaning Theory. *Springer-Verlag, New York. 1995*

V. Varma *et al*. 2009. IIIT Hyderabad at TAC 2009. In *Proceedings of Text Analysis Conference 2009 (TAC 09).*

A. Vlachos. 2008. A Stopping Criterion for Active Learning. *Computer Speech and Language*. 22(3):295-312. 2008.

P. Wang and C. Domeniconi. 2008. Building Semantic Kernels for Text Classification Using Wikipedia. 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2008

W. Zhang, J. Su, C. L. Tan and W. T. Wang. 2010. Entity Linking Leveraging Automatically Generated Annotation. *23rd International Conference on Computational Linguistics,* August 23-27, 2010.

Z Zheng, F Li, X Zhu and M Huang. 2010. Learning to Link Entities with Knowledge Base. In Proceedings of *the 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL).* 2010. Los Angeles, CA

C. Zirn, V. Nastase and M. Strube. 2008. Distinguishing Between Instances and Classes in the Wikipedia Taxonomy. In *Proceedings of the 5th European Semantic Web Conference*, Tenerife, Spain, 1-5 June 2008.