Learning to Link Entities with Knowledge Base

Zhicheng Zheng, Fangtao Li, Minlie Huang, Xiaoyan Zhu
State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
`zhengzc04,fangtao06 @gmail.com, aihuang,zxy-dcs @tsinghua.edu.cn`

Abstract

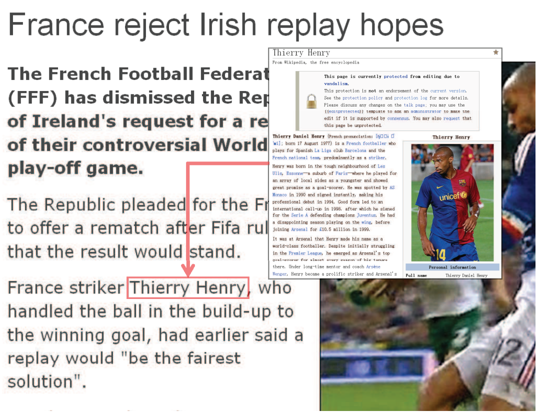
This paper address the problem of entity linking. Specifically, given an entity mentioned in unstructured texts, the task is to link this entity with an entry stored in the existing knowledge base. This is an important task for information extraction. It can serve as a convenient gateway to encyclopedic information, and can greatly improve the web users' experience. Previous learning based solutions mainly focus on classification framework. However, it's more suitable to consider it as a ranking problem. In this paper, we propose a learning to rank algorithm for entity linking. It effectively utilizes the relationship information among the candidates when ranking. The experiment results on the TAC 2009[1] dataset demonstrate the effectiveness of our proposed framework. The proposed method achieves 18.5% improvement in terms of accuracy over the classification models for those entities which have corresponding entries in the Knowledge Base. The overall performance of the system is also better than that of the state-of-the-art methods.

1 Introduction

The entity linking task is to map a named-entity mentioned in a text to a corresponding entry stored in the existing Knowledge Base. The Knowledge Base can be considered as an encyclopedia for entities. It contains definitional, descriptive or relevant information for each entity. We can acquire the knowledge of entities by looking up the Knowledge Base. Wikipedia is an online encyclopedia, and now it becomes one of the largest repositories of encyclopedic knowledge. In this paper, we use Wikipedia as our Knowledge Base.

[1]http://www.nist.gov/tac/

Entity linking can be used to automatically augment text with links, which serve as a convenient gateway to encyclopedic information, and can greatly improve user experience. For example, Figure 1 shows news from BBC.com. When a user is interested in ”Thierry Henry”, he can acquire more detailed information by linking *”Thierry Henry”* to the corresponding entry in the Knowledge Base.



France reject Irish replay hopes

The French Football Federat (FFF) has dismissed the Rep of Ireland's request for a re of their controversial World play-off game.

The Republic pleaded for the Fr to offer a rematch after Fifa rul that the result would stand.

France striker Thierry Henry, who handled the ball in the build-up to the winning goal, had earlier said a replay would "be the fairest solution".

Figure 1: Entity linking example

Entity linking is also useful for some information extraction (IE) applications. We can make use of information stored in the Knowledge Base to assist the IE problems. For example, to answer the question *”When was the famous basketball player Jordan born?”*, if the Knowledge Base contains the entity of basketball player Michael Jordan and his information (such as infobox[2] in Wikipedia), the correct answer *"February 17, 1963"* can be easily retrieved.

Entity linking encounters the problem of entity ambiguity. One entity may refer to several entries in the Knowledge Base. For example, the entity "Michael Jordan" can be linked to the basketball player or the professor in UC Berkeley.

Previous solutions find that classification based methods are effective for this task (Milne and Witten, 2008). These methods consider each candidate entity independently, and estimate a probability that the candidate entry corresponds to the target entity. The candidate with the highest probability was chosen as the target entity. In this way, it's more like a ranking problem rather than a classification problem. Learning to rank methods take into account the relations between candidates, which is better than considering them independently. Learning to rank methods are popular in document information retrieval, but there are few studies on information extraction. In this paper, we investigate the application of learning to rank methods to the entity linking task. And we compare several machine learning methods for this task. We investigate the pairwise learning to rank method, Ranking Perceptron (Shen and Joshi, 2005), and the listwise method, ListNet (Cao et al., 2007). Two classification methods, SVM and Perceptron, are developed as our baselines. In comparison, learning to rank methods show significant improvements over classification methods, and ListNet achieves the best result. The best overall performance is also achieved with our proposed framework.

This paper is organized as follows. In the next section we will briefly review the related work. We present our framework for entity linking in section 3. We then describe in section 4 learning to rank methods and features for entity linking. A top1 candidate validation module will be explained in section 5. Experiment results will be discussed in section 6. Finally, we conclude the paper and discusses the future work in section 7.

[2]Infoboxes are tables with semi-structured information in some pages of Wikipedia

2 Related Work

There are a number of studies on named entity disambiguation, which is quite relevant to entity linking. Bagga and Baldwin (1998) used a Bag of Words (BOW) model to resolve ambiguities among people. Mann and Yarowsky (2003) improved the performance of personal names disambiguation by adding biographic features. Fleischman (2004) trained a Maximum Entropy model with Web Features, Overlap Features, and some other features to judge whether two names refer to the same individual. Pedersen (2005) developed features to represent the context of an ambiguous name with the statistically significant bigrams.

These methods determined to which entity a specific name refer by measuring the similarity between the context of the specific name and the context of the entities. They measured similarity with a BOW model. Since the BOW model describes the context as a term vector, the similarity is based on co-occurrences. Although a term can be one word or one phrase, it can't capture various semantic relations. For example, "Michael Jordan now is the boss of Charlotte Bobcats" and "Michael Jordan retired from NBA". The BOW model can't describe the relationship between *Charlotte Bobcats* and *NBA*. Malin and Airoldi (2005) proposed an alternative similarity metric based on the probability of walking from one ambiguous name to another in a random walk within the social network constructed from all documents. Minkov (2006) considered extended similarity metrics for documents and other objects embedded in graphs, facilitated via a lazy graph walk, and used it to disambiguate names in email documents. Bekkerman and McCallum (2005) disambiguated web appearances of people based on the link structure of Web pages. These methods tried to add background knowledge via social networks. Social networks can capture the relatedness between terms, so the problem of a BOW model can be solved to some extent. Xianpei and Jun (2009) proposed to use Wikipedia as the background knowledge for disambiguation. By leveraging Wikipedia's semantic knowledge like social relatedness between named entities and associative relatedness between concepts, they can measure the similarity between entities more accurately. Cucerzan (2007) and

Bunescu (2006) used Wikipedia's category information in the disambiguation process. Using different background knowledge, researcher may find different efficient features for disambiguation.

Hence researchers have proposed so many efficient features for disambiguation. It is important to integrate these features to improve the system performance. Some researchers combine features by manual rules or weights. However, it is not convenient to directly use these rules or weights in another data set. Some researchers also try to use machine learning methods to combine the features. Milne and Witten (2008) used typical classifiers such as Naive Bayes, C4.5 and SVM to combine features. They trained a two-class classifier to judge whether a candidate is a correct target. And then when they try to do disambiguation for one query, each candidate will be classified into the two classes: correct target or incorrect target. Finally the candidate answer with the highest probability will be selected as the target if there are more than one candidates classified as answers. They achieve great performance in this way with three efficient features. The classifier based methods can be easily used even the feature set changed. However, as we proposed in Introduction, it's not the best way for such work. We'll detail the learning to rank methods in the next section.

3 Framework for Entity Linking

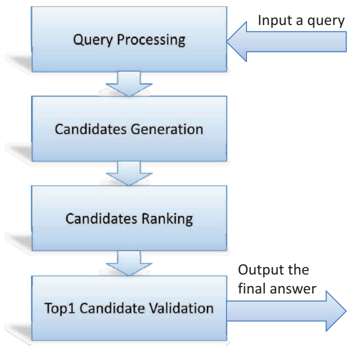


Figure 2: The framework for entity linking

Entity linking is to align a named-entity mentioned in a text to a corresponding entry stored in the existing Knowledge Base. We proposed a framework to solve the ”entity linking” task. As illustrated in Figure 2, when inputting a query which is an entity mentioned in a text, the system will return the target entry in Knowledge Base with four modules:

1. Query Processing. First, we try to correct the spelling errors in the queries by using query spelling correction supplied by Google. Second, we expand the query in three ways: expanding acronym queries from the text where the entity is located, expanding queries with the corresponding redirect pages of Wikipedia and expanding queries by using the anchor text in the pages from Wikipedia.

2. Candidates Generation. With the queries generated in the first step, the candidate generation module retrieves the candidates from the Knowledge Base. The candidate generation module also makes use of the disambiguation pages in Wikipedia. If there is a disambiguation page corresponding to the query, the linked entities listed in the disambiguation page are added to the candidate set.

3. Candidates Ranking. In the module, we rank all the candidates with learning to rank methods.

4. Top1 Candidate Validation. To deal with those queries without appropriate matching, we finally add a validation module to judge whether the top one candidate is the target entry.

The detail information of ranking module and validation module will be introduced in next two sections.

4 Learning to Rank Candidates

In this section we first introduce the learning to rank methods, and then describe the features for ranking methods.

4.1 Learning to rank methods

Learning to rank methods are popular in the area of document retrieval. There are mainly two types of learning to rank methods: pairwise and listwise. The pairwise approach takes as instances object pairs in a ranking list for a query in learning. In this way, it transforms the ranking problem to the classification problem. Each pair from ranking list is labeled based on the relative position or with the score of ranking objects. Then a classification model can be trained on the labeled data and then be used for ranking. The pairwise approach has advantages in that the existing methodologies on classification can be applied directly. The listwise approach takes candidate lists for a query as instances to train ranking models. Then it trains a ranking function by minimizing a listwise loss function defined on the predicted list and the ground truth list.

To describe the learning to rank methods, we first introduce some notations:

- Query set Q = .
- Each query is associated with a list of objects(in document retrieval, the objects should be documents), .
- Each object list has a labeled score list represents the relevance degree between the objects and the query.
- Features vectors from each query-object pair, .
- Ranking function f, for each it outputs a score . After the training phase, to rank the objects, just use the ranking function f to output the score list of the objects, and rank them with the score list.

In the paper we will compare two different learning to rank approaches: Ranking Perceptron for pairwise and ListNet for listwise. A detailed introduction on Ranking Perceptron (Shen and Joshi, 2005) and ListNet (Cao et al., 2007) is given.

4.1.1 Ranking Perceptron

Ranking Perceptron is a pairwise learning to rank method. The score function is defined as .

For each pair , is computed. With a given margin function and a positive rate , if , an update is performed:

After iterating enough times, we can use the function to rank candidates.

4.1.2 ListNet

ListNet takes lists of objects as instances in learning. It uses a probabilistic method to calculate the listwise loss function.

ListNet transforms into probability distributions both the scores of the objects assigned by the oracle ranking function and the real score of the objects given by human.

Let denote a permutation on the objects. In ListNet algorithm, the probability of with given scores is defined as:

Then the top k probability of $\mathscr{G}$ can be calculated as:

$\mathscr{G}$

The ListNet uses a listwise loss function with Cross Entropy as metric:

$\mathscr{G}$

Denote as the ranking function based on Neural Network model . The gradient of with respect to parameter can be calculated as:

$\mathscr{G}$

In each iteration, the is updated with in a gradient descent way. Here is the learning rate.

To train a learning to rank model, the manually evaluated score list for each query's candidate list is required. We assign 1 to the real target entity and 0 to the others.

4.2 Features for Ranking

In the section, we will introduce the features used in the ranking module. For convenience, we define some symbols first:

Q represents a query, which contains a named entity mentioned in a text. CSet represents the candidate entries in Knowledge Base for the query Q. C represents a candidate in CSet.

Q's nameString represents the name string of Q. Q's sourceText represents the source text of Q. Q's querySet represents the queries which are expansions of Q's nameString.

C's title represents the title of corresponding Wikipedia article of C. C's titleExpand represents the union set of the redirect set of C and the anchor text set of C. C's article represents the Wikipedia article of C.

C's nameEntitySet represents the set of all named entities in C's article labeled by Stanford NER (Finkel et al., 2005). Q's nameEntitySet represents the set of all named entities in Q's sourceText.

C's countrySet represents the set of all countries in C's article, and we detect the countries from text via a manual edited country list. Q's countrySet represents the set of all countries in Q's sourceText. C's countrySetInTitle represents the set of countries exist in one of the string s from C's titleExpand.

C's citySetInTitle represents the set of all cities exist in one of the string s from C's titleExpand, and we detect the cities from text via a manual edited list of famous cities. Q's citySet represents the set of all cities in Q's sourceText.

Q's type represents the type of query Q. It's labeled by Stanford NER. C's type is manually labeled already in the Knowledge Base.

The features that used in the ranking module can be divided into 3 groups: Surface, Context and Special. Each of these feature groups will be detailed next.

4.2.1 Surface Features

The features in Surface group are used to measure the similarity between the query string and candidate entity's name string.

StrSimSurface. The feature value is the maximum similarity between the Q's nameString and each string s in the set C's titleExpand. The string similarity is measured with edit distance.

ExactEqualSurface. The feature value is 1 if there is a string s in set C's titleExpand same as the Q's nameString, or the Candidate C is extracted from the disambiguation page. In other case, the feature value is set to 0.

StartWithQuery. The feature value is 1 if there is a string s in set C's titleExpand starting with the Q's nameString, and C's ExactEqualSurface is not 1. In other case, the feature value is set to 0.

EndWithQuery. The feature value is 1 if there is a string s in set C's titleExpand ending with the Q's nameString, and C's ExactEqualSurface is not 1. In other case, the feature value is set to 0.

StartInQuery. The feature value is 1 if there is a string s in set C's titleExpand that s is the prefix of the Q's nameString, and C's ExactEqualSurface is not 1. In other case, the feature value is set to 0.

EndInQuery. The feature value is 1 if there is a string s in set C's titleExpand that s is the postfix of the Q's nameString, and C's ExactEqualSurface is not 1. In other case, the feature value is set to 0.

EqualWordNumSurface. The feature value is the maximum number of same words between the Q's nameString and each string s in the set C's titleExpand.

MissWordNumSurface. The feature value is the minimum number of different words between the Q's nameString and each string s in the set C's titleExpand.

4.2.2 Context Features

The features in Context group are used to measure the context relevance between query and the candidate entity. We mainly consider the TF-IDF similarity and named entity co-occurrence.

TFSimContext. The feature value is the TF-IDF similarity between the C's article and Q's sourceText.

TFSimRankContext. The feature value is the inverted rank of C's TFSimContext in the CSet.

AllWordsInSource. The feature value is 1 if all words in C's title exist in Q's sourceText, and in other case, the feature value is set to 0.

NENumMatch. The feature value is the number of of same named entities between C's nameEntitySet and Q's nameEntitySet. Two named entities are judged to be the same if and only if the two named entities' strings are identical.

4.2.3 Special Features

Besides the features above, we also find that the following features are quite significant in the entity linking task: country names, city names and types of queries and candidates.

CountryInTextMatch. The feature value is the number of same countries between C's countrySet and Q's countrySet.

CountryInTextMiss. The feature value is the number of countries that exist in Q's countrySet but do not exist in C's countrySet.

CountryInTitleMatch. The feature value is the number of same countries between C's countrySetInTitle and Q's countrySet.

CountryInTitleMiss. The feature value is the number of countries that exist in C's countrySetInTitle but do not exist in Q's countrySet.

CityInTitleMatch. The feature value is the number of same cities between C's citySetInTitle and Q's citySet.

TypeMatch. The feature value is 0 if C's type is not consistent with Q's type, in other case, the feature value is set to 1.

When ranking the candidates in CSet, the features' value was normalized into [0, 1] to avoid noise caused by large Integer value or small double value.

5 Top 1 Candidate Validation

To deal with those queries without target entities in the Knowledge Base, we supply a Top 1 candidate validation module. In the module, a two-class classifier is used to judge whether the top one candidate is the true target entity. The top one candidate selected from the ranking module can be divided into two classes: target and non-target, depending on whether it's the correct target link of the query. According to the performance of classification, SVM is chosen as the classifier (In practice, the libsvm package is used) and the SVM classifier is trained on the entire training set.

Most of the features used in the validation module are the same as those in ranking module, such as StrSimSurface, EqualWordNumSurface, MissWordNumSurface, TFSimContext, AllWordsInSource, NENumMatch and TypeMatch. We also design some other features, as follows:

AllQueryWordsInWikiText. The feature value is one if Q's textRetrievalSet contains C, and in other case the feature value is set to zero. The case that Q's textRetrievalSet contains C means the candidate C's article contains the Q's nameString.

CountryInTextPer. The feature is the percentage of countries from Q's countrySet exist in C's countrySet too. The feature can be seen as a normalization of CountryInTextMatch/Miss features in ranking module.

ScoreOfRank. The feature value is the score of the candidate given by the ranking module. The ScoreOfRank takes many features in ranking module into consideration, so only fewer features of ranking module are used in the classifier.

6 Experiment and Analysis

6.1 Experiment Setting

Algorithm	Accuracy	Improvement over SVM
ListNet	**0.9045**	+18.5%
Ranking Perceptron	0.8842	+15.8%
SVM	0.7636	-
Perceptron	0.7546	-1.2%

Table 1: Evaluation of different ranking algorithm

Entity linking is initiated as a task in this year's TAC-KBP[3] track, so we use the data from this track. The entity linking task in the KBP track is to map an entity mentioned in a news text to the Knowledge Base, which consist of articles from Wikipedia. The KBP track gives a sample query set which consists of 416 queries for developing. The test set consists of 3904 queries. 2229 of these queries can't be mapped to Knowledge Base, for which the systems should return NIL links. The remaining 1675 queries all can be aligned to Knowledge Base. We will firstly analyze the ranking methods with those non-NIL queries, and then with an additional validation module, we train and test with all queries including NIL queries.

As in the entity linking task of KBP track, the accuracy is taken as

$$\frac{\texttt{correct answered queries}}{\texttt{total queries}}$$

6.2 Evaluation of Machine Learning Methods in ranking

As mentioned in the section of related work, learning to rank methods in entity linking performs better than the classification methods. To justify this, some experiments are designed to evaluate the performance of our ranking module when adopting different algorithms.

To evaluate the performance of the ranking module, we use all the queries which can be aligned to a target entry in the Knowledge Base. The training set contains 285 valid queries and the test set contains 1675.

[3]http://apl.jhu.edu/ paulmac/kbp.html

Set	Features in Set
Set1	Surface Features
Set2	Set1+TF-IDF Features
Set3	Set2+AllWordsInSource
Set4	Set3+NENumMatch
Set5	Set4+CountryInTitle Features
Set6	Set5+CountryInText Features
Set7	Set6+CityInTitleMatch
Set8	Set7+MatchType

Table 2: Feature Sets

Three algorithms are taken into comparison: ListNet, Ranking Perceptron, and classifier based methods. The classifier based methods are trained by dividing the candidates into two classes: target and non-target. Then, the candidates are ranked according to their probability of being classified as target. two different classifiers are tested here, SVM and Perceptron.

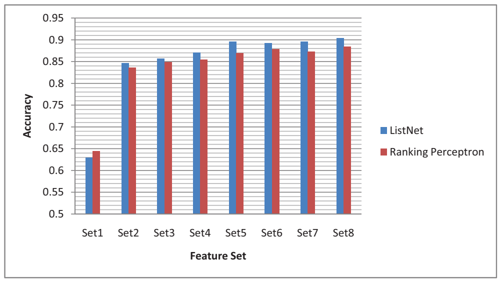


Figure 3: Comparison of ListNet and Ranking Perceptron

As shown in Table 1, the two learning to rank methods perform much better than the classification based methods. The experiment results prove our point that the learning to rank algorithms are more suitable in this work. And the ListNet shows slight improvement over Ranking Perceptron, but since the improvement is not so significant, maybe it depends on the feature set. To confirm this, we compare the two algorithms with different features, as showed in Table 2. In Figure 3, The ListNet outperforms Ranking Perceptron with all feature sets except Set1, which indicates that the listwise approach is more suitable than the pairwise approach. The pairwise approach suffers from two problems: first, the objective of learning is to minimize classification er-

Systems	Accuracy of all queries	Accuracy of non-NIL queries	Accuracy of NIL queries
System1	0.8217	0.7654	0.8641
System2	0.8033	0.7725	0.8241
System3	0.7984	0.7063	0.8677
ListNet+SVM	**0.8494**	**0.79**	**0.8941**

Table 3: Evaluation of the overall performance, compared with KBP results (System 1-3 demonstrate the top three ranked systems)

rors but not to minimize the errors in ranking; second, the number of pairs generated from list varies largely from list to list, which will result in a model biased toward lists with more objects. The issues are also discussed in (Y.B. Cao et al., 2006; Cao et al., 2007). And the listwise approach can fix the problems well.

As the feature sets are added incrementally, it can be used for analyzing the importance of the features to the ranking task. Although Surface Group only takes into consideration the candidate's title and the query's name string, its accuracy is still higher than 60%. This is because many queries have quite small number of candidates, the target entry can be picked out with the surface features only. The result shows that after adding the TF-IDF similarity related features, the accuracy increases significantly to 84.5%. Although TF-IDF similarity is a simple way to measure the contextual similarity, it performs well in practice. Another improvement is achieved when adding the CountryInTitleMatch and CountryInTitleMiss features. Since a number of queries in test set need to disambiguate candidates with different countries in their titles, the features about country in the candidates' title are quite useful to deal with these queries. But it doesn't mean that the features mentioned above are the most important. Because many features correlated with each other quite closely, adding these features doesn't lead to remarkable improvement. The conclusion from the results is that the Context Features significantly improve the ranking performance and the Special Features are also useful in the entity linking task.

6.3 Overall Performance Evaluation

We are also interested in overall performance with the additional validation module. We use all the 3904 queries as the test set, including both NIL and non-NIL queries. The top three results from the KBP track (McNamee and Dang, 2009) are selected as comparison. The evaluation result in Table 3 shows that our proposed framework outperforms the best result in the KBP track, which demonstrates the effectiveness of our methods.

7 Conclusions and Future Work

This paper demonstrates a framework of learning to rank for linking entities with the Knowledge Base. Experimenting with different ranking algorithms, it shows that the learning to rank methods perform much better than the classification methods in this problem. ListNet achieves 18.5% improvement over SVM, and Ranking Perceptron gets 15.8% improvement over SVM. We also observe that the listwise learning to rank methods are more suitable for this problem than pairwise methods. We also add a validation module to deal with those entities which have no corresponding entry in the Knowledge Base. We also evaluate the proposed method on the whole data set given by the KBP track, for which we add a binary SVM classification module to validate the top one candidate. The result of experiment shows the proposed strategy performs better than all the systems participated in the entity linking task.

In the future, we will try to develop more sophisticated features in entity linking and design a typical learning to rank method for the entity linking task.

Acknowledgments

This work was partly supported by the Chinese Natural Science Foundation under grant No.60973104 and No. 60803075, partly carried out with the aid of a grant from the International Development Research Center, Ottawa, Canada IRCI project from the International Development.

References

Bagga and Baldwin. 1998. *Entity-Based Cross-Document Coreferencing Using the Vector Spcae Model*. *in Proceedings of HLT/ACL*.

Gideon S. Mann and David Yarowsky. 2003. *Unsupervised Personal Name Disambiguation*. *in Proceedings of CONIL*.

Michael Ben Fleishman. 2004. *Multi-Document Person Name Resolution*. *in Proceedings of ACL*.

Ted Pedersen, Amruta Purandare and Anagha Kulkarni. 2005. *Name Discrimination by Clustering Similar Contexts*. *in Proceedings of CICLing*.

B.Malin and E. Airoldi. 2005. *A Network Analysis Model for Disambiguation of Names in Lists*. *in Proceedings of CMOT*.

Einat Minkov, William W. Cohen and Andrew Y. Ng. 2006. *Contextual Search and Name Disambiguation in Email Using Graph*. *in Proceedings of SIGIR*.

Ron Bekkerman and Andrew McCallum. 2005. *Disambiguating Web Appearances of People in a Social Network*. *in Proceedings of WWW*.

Xianpei Han and Jun Zhao. 2009. *Named Entity Disambiguation by Leveraging Wikipedia Semantic Knowledge*. *in Proceedings of CIKM*.

David Milne and Ian H. Witten. 2008. *Learning to Link with Wikipedia*. *in Proceedings of CIKM*.

Herbrich, R., Graepel, T. and Obermayer K. 1999. *Support vector learning for ordinal regression*. *in Proceedings of ICANN*.

Freund, Y., Iyer, R., Schapire, R. E. and Singer, Y. 1998. *An efficient boosting algorithm for combining preferences*. *in Proceedings of ICML*.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N. and Hullender, G. 2005. *Learning to rank using gradient descent*. *in Proceedings of ICML*.

Cao, Y. B., Xu, J., Liu, T. Y., Li, H., Huang, Y. L. and Hon, H. W. 2006. *Adapting ranking SVM to document retrieval*. *in Proceedings of SIGIR*.

Cao, Z., Qin, T., Liu, T. Y., Tsai, M. F. and Li, H. 2007. *Learning to rank: From pairwise approach to listwise approach*. *in Proceedings of ICML*.

Qin, T., Zhang, X.-D., Tsai, M.-F., Wang, D.-S., Liu, T.Y., and Li, H. 2007. *Query-level loss functions for information retrieval*. *in Proceedings of Information processing and management*.

L. Shen and A. Joshi. 2005. *Ranking and Reranking with Perceptron*. *Machine Learning*,60(1-3),pp. 73-96.

Silviu Cucerzan. 2007. *Large-Scale Named Entity Disambiguation Based on Wikipedia Data*. *in Proceedings of EMNLP-CoNLL*.

Razvan Bunescu and Marius Pasca. 2006. *Using Encyclopedic Knowledge for Named Entity Disambiguation*. *in Proceedings of EACL*.

Paul McNamee and Hoa Dang. 2009. *Overview of the TAC 2009 Knowledge Base Population Track (DRAFT)*. *in Proceedings of TAC*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. *Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling*. *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pp. 363-370.