Collaborative Ranking: A Case Study on Entity Linking

Zheng Chen
Computer Science Department
Graduate Center
City University of New York
zchen1@gc.cuny.edu

Heng Ji
Computer Science Department
Queens College and Graduate Center
City University of New York
hengji@cs.qc.cuny.edu

Abstract

In this paper, we present a new ranking scheme, collaborative ranking (*CR*). In contrast to traditional non-collaborative ranking scheme which solely relies on the strengths of isolated queries and one stand-alone ranking algorithm, the new scheme integrates the strengths from multiple collaborators of a query and the strengths from multiple ranking algorithms. We elaborate three specific forms of collaborative ranking, namely, micro collaborative ranking (MiCR), macro collaborative ranking (MaCR) and micro-macro collaborative ranking (MiMaCR). Experiments on entity linking task show that our proposed scheme is indeed effective and promising.

1 Introduction

Many natural language processing tasks can be formalized as a ranking problem, namely to rank a collection of candidate "objects" with respect to a "query". For example, intensive studies were devoted to parsing in which multiple possible parsing trees or forests are ranked with respect to a sentence (Collins, 2000; Charniak and Johnson, 2005; Huang, 2008), machine translation in which multiple translation hypotheses are ranked with respect to a source sentence (Och, 2002; Shen et al., 2005), anaphora resolution in which multiple antecedents are ranked with respect to an anaphora (Yang et al., 2008), and question answering in which multiple possible answers are ranked with respect to a question (Ravichandran et al., 2003). Previous studies mainly focused on improving the ranking performance using one stand-alone learning algorithm on isolated queries.

Although a wide range of learning algorithms (unsupervised, supervised or semi-supervised) is available, each with its strengths and weaknesses, there is not a learning algorithm that can work best on all types of data. In such a situation, it would be desirable to build a "collaborative" model by integrating multiple models. Such an idea forms the basis of ensemble methodology and it is well-known that ensemble methods (e.g., bagging, boosting) can improve the performance of many problems, in which classification is the most intensively studied (Rokach, 2009). The other situation is related with isolated queries handled by learning algorithms. The single query may not be formulated with the best terms or the query itself may not contain comprehensive information required for a high-performance ranking algorithm. Therefore, techniques of query expansion or query reformulation can be introduced and previous research has shown the effectiveness of those techniques in such applications as information retrieval and question answering (Manning et al., 2008; Riezler et al., 2007). Nevertheless, previous research normally considers query reformulation as a new query for the ranking system, it would be more desirable to form a larger-scale "collaborative" group for the query and make a unified decision based on the group.

Inspired from human collaborative learning in which two or more people form a group and accomplish work together, we propose a new ranking scheme, *collaborative ranking*, which aims to imitate human collaborative learning and enhance system ranking performance. The main idea is to seek collaborations for each query from two levels:

(1) query-level: search a group of query collaborators, and make the joint decision from the group together with the query using a stand-alone ranking algorithm.

(2) ranker-level: design a group of multiple rankers, and make the joint decision from the entire group on a single query.

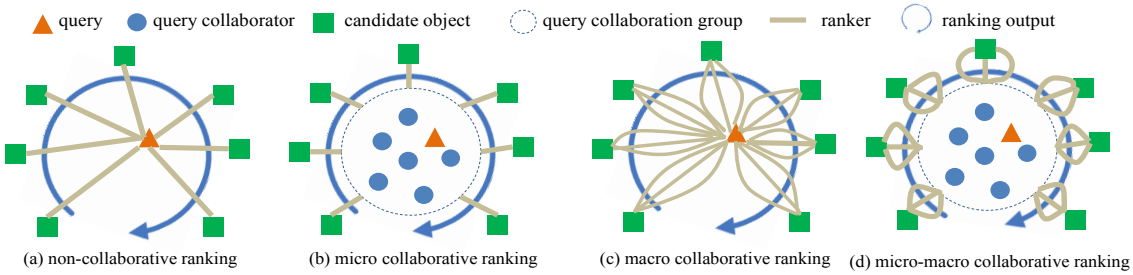

Figure 1: Non-collaborative ranking and three collaborative ranking approaches.

Figure 1 presents an intuitive illustration of four ranking approaches, including the traditional non-collaborative ranking and three collaborative ranking forms: micro collaborative ranking (MiCR), macro collaborative ranking (MaCR), and micro-macro collaborative ranking (MiMaCR).

Compared with the traditional non-collaborative ranking that only leverages the information contained in a single query and only applies one ranking function (Figure 1 (a)), the three collaborative ranking approaches have the following advantages:

(1)MiCR (corresponding to query-level collaboration[1]) leverages the information contained in the collaborators of a query. Figure 1 (b) demonstrates that 6 query collaborators together with the query form a query collaboration group.

(2)MaCR (corresponding to ranker-level collaboration[2]) integrates the strengths from two or more rankers. Figure 1 (c) demonstrates an example of 3 rankers.

(3)MiMaCR combines the advantages from MiCR and MaCR as shown in Figure 1 (d).

In this paper, we will show the efficacy of collaborative ranking on the entity linking task defined in the Knowledge Base Population (KBP) track (Ji et al., 2010) at Text Analysis Conference (TAC). Each query in the task is associated with a name string and its context document. Traditional approaches for entity linking only made use of the lexical or document level information contained in the query, however, it may not be sufficient for the task. The intuition why query-level collaboration may work is that it leverages more comprehensive information about the entity mention from multiple "collaborators" (related documents containing the name string). Furthermore, previous work on this task mainly focused on comparing one ranking algorithm with the others, however, each ranking algorithm has its own strengths, and therefore, ranker-level collaboration can potentially improve the performance. Last, the combination of query-level and ranker-level collaboration can lead to further performance gains.

[1]Query is normally expressed by small-scale data structure, so called micro.

[2]Ranker is normally implemented by large-scale algorithm, so called macro.

2 Non-collaborative Ranking

Let q denote a query. Let $o^{(q)} = \left\{o_1^{(q)}, \ldots, o_{n^{(q)}}^{(q)}\right\}$ denote the object set associated with q, where $n^{(q)}$ denotes the size of the $o^{(q)}$. The goal of non-collaborative ranking is to seek a ranking function f such that it computes ranking scores for the candidates in the object set, i.e., $y^{(q)} = f(o^{(q)}) = \left\{y_1^{(q)}, \ldots, y_{n^{(q)}}^{(q)}\right\}$.

Earlier studies on non-collaborative ranking mainly explored unsupervised approaches, e.g., vector space model, link based algorithm such as PageRank (Page et al., 1998). Unsupervised approaches are based on well-established statistical and probability theory, nevertheless, they suffer from some drawbacks, for example, it is hard to tune parameters. Recently, supervised approaches (named "learning to rank") that automatically learn ranking functions from training data become the focus of ranking research. In the literature, supervised approaches are categorized into three classes, namely, pointwise, pairwise, and listwise. We summarize a comparison of the three approaches in Table 1. We use the following notations in the table.

Let $\mathcal{Q} = \{q_1, \ldots, q_N\}$ denote the set of N queries in the training data, each query q_i is associated with a set of objects $o^{(q_i)} = \left\{o_1^{(q_i)}, \ldots, o_{n^{(q_i)}}^{(q_i)}\right\}$ and a set of ground-truth ranking scores $y^{(q_i)} =$

	pointwise	pairwise	listwise
approach overview	common: 1) use training samples; 2) learn the best ranking function by minimizing a given loss function; 3) apply the ranking function at ranking step		
	transform ranking to regression or classification on single objects	transform ranking to classification on object pairs	ranking by learning from lists of objects
training set	$\{\left(x_j^{(q_i)}, y_j^{(q_i)}\right)\}_{j=1,\ldots,n^{(q_i)};\ i=1,\ldots,N}$	$\{\left(x_j^{(q_i)}, x_k^{(q_i)}, y\right)\}_{j=1,\ldots,n^{(q_i)};k=1,\ldots,n^{(q_i)};\ k\neq j;\ i=1,\ldots,N}$ $y=\begin{cases}+1\ if\ x_j^{(q_i)} > x_k^{(q_i)}\\ -1\ if\ x_j^{(q_i)} \leq x_k^{(q_i)}\end{cases}$	$\{(x^{(q_i)}, y^{(q_i)})\}_{i=1,\ldots,N}$
loss function	pointwise loss, e.g., square loss(Chen et al., 2009)	pairwise loss, e.g., hinge loss(Zhang, 2004), exponential loss(Bartlett et al., 2003), logistic loss(Lin, 2002)	listwise loss, e.g., cross entropy loss(Cao et al., 2007),cosine loss(Qin et al., 2007)
pros and cons	pros: classification is well studied cons:1) only consider one object at a time ignoring relationship among objects	pros: classification is well studied cons:1) only consider pairwise orders; 2) biased towards lists with more objects	pros: fully consider relationship among objects cons: 1) less well studied in theory
selected algorithms	Discriminative model for IR (Nallapati, 2004); McRank (Li et al., 2007)	SVM Ranking(Joachims, 2002); RankBoost(Freund et al., 2003); RankNet(Burges et al., 2005)	ListNet (Cao et al., 2007); RankCosine(Qin et al., 2007); ListMLE (Xia et al., 2008)

Table 1: Comparison of pointwise, pairwise and listwise ranking approaches.

$\left\{y_1^{(q_i)}, \ldots, y_{n^{(q_i)}}^{(q_i)}\right\}$. Let $x_j^{(q_i)} = \varphi(q_i, o_j^{(q_i)})$ denote a feature vector associated with each query-object pair $(q_i, o_j^{(q_i)})$.

3 Collaborative Ranking

3.1 Micro Collaborative Ranking(MiCR)

Micro collaborative ranking is characterized by integrating joint strengths from multiple query collaborators and the query itself. It is based on the following assumptions:

• Expandability: Query is expandable, that is, it is able to find potential collaborators.

• Redundancy: Collaborators and query may share redundant information.

• Diversity: Collaborators exhibit multifaceted information that may complement the information contained in the query.

• Robustness: Noisy collaborators are allowable, and they could be put under control.

Let $cq^{(q)} = \{cq_1, \ldots, cq_k\}$ be the k collaborators of a query q. For each object $o_j^{(q)}$ associated with q, we form $k+1$ feature vectors $x_j^{(q)} = \varphi(q, o_j^{(q)}), x_j^{(cq_1)} = \varphi(cq_1, o_j^{(cq_1)}), \ldots, x_j^{(cq_k)} = \varphi(cq_k, o_j^{(cq_k)})$. Let f be a ranking function which is obtained by either an unsupervised or supervised approach. There are two important steps that distinguish MiCR from traditional non-collaborative ranking approaches:

• Step (1): searching the best k collaborators of q.

• Step (2): simulating the interaction of k collaborators at the ranking step.

Solutions for step (1) can vary from case to case. In our case study presented later, we transform the collaborator searching problem into a clustering problem. Collaborators of a query are then formed by members (excluding the query) in a cluster which contains the query and k is the size of the cluster minus one.

We transform the problem of step (2) into solving a function g_1 such that a ranking score $y_j^{(q)}$ can be computed for each object $o_j^{(q)}$. One approach to computing g_1 is to firstly compute the ranking scores of collaborators and query using the ranking function f and then combine those ranking scores in some way (Formula 1). The other approach is to learn a supervised ranking function f' which takes collaborators and query as input (Formula 2).

$$y_j^{(q)} = g_1(f\left(x_j^{(q)}\right), f\left(x_j^{(cq_1)}\right), \ldots, f\left(x_j^{(cq_k)}\right)) \quad (1)$$

$$y_j^{(q)} = g_1(\bullet) = f'\left(x_j^{(q)}, x_j^{(cq_1)}, \ldots, x_j^{(cq_k)}\right) \quad (2)$$

We present three specific forms of g_1 in Formula 1, namely, max, min, and weighted. We can also define a special case of weighted, called "average" in which $w_0 = w_1 \ldots = w_k = 1/(k+1)$.

- max: $y_j^{(q)} = \max(f\left(x_j^{(q)}\right), \ldots, f\left(x_j^{(cq_k)}\right))$
- min: $y_j^{(q)} = \min(f\left(x_j^{(q)}\right), \ldots, f\left(x_j^{(cq_k)}\right))$
- weighted: $y_j^{(q)} = w_0 f\left(x_j^{(q)}\right) + \sum_{i=1}^{k} w_i f\left(x_j^{(cq_i)}\right)$

We will discuss three supervised versions of g_1 (Formula 2) in section 4.4. A general algorithm for MiCR is presented in Algorithm 1.

Algorithm 1 MiCR Algorithm.

Input:
a query q; a set of objects $o^{(q)}$; a function g_1
Output:
a set of ranking scores $y^{(q)}$

Search k collaborators of q: $cq^{(q)} = \{cq_1, \ldots, cq_k\}$.
for $j = 1; j <= n^{(q)}; j++$ **do**
  Form $k+1$ feature vectors: $x_j^{(q)}, x_j^{(cq_1)}, \ldots, x_j^{(cq_k)}$.
  Compute function $y_j^{(q)} = g_1(\bullet)$.
end for
return $y^{(q)}$

3.2 Macro Collaborative Ranking(MaCR)

Macro collaborative ranking is characterized by integrating joint strengths from multiple rankers. It is based on the following assumptions:

- Independence: Each ranker can make its own ranking decisions.
- Diversity: Each ranker has its own strengths in making ranking decisions.
- Collaboration: Rankers in the group could collaborate to make a consensus decision under some mechanism.

Let $x_j^{(q)} = \varphi(q, o_j^{(q)})$ be the feature vector formed from the pair consisting of query q and an associated object $o_j^{(q)}$. Let $\mathcal{F}^* = \{f_1, \ldots, f_m\}$ be m existing ranking functions. We transform the computation of collaboration among rankers into solving the following composite function g_2:

$$y_j^{(q)} = g_2(f_1\left(x_j^{(q)}\right), \ldots, f_m\left(x_j^{(q)}\right)) \quad (3)$$

Similar with MiCR, g_2 can be expressed by max, min, weighted (average) respectively:

- max: $y_j^{(q)} = \max\{f_i\left(x_j^{(q)}\right)\}_{i=1}^{m}$
- min: $y_j^{(q)} = \min\{f_i\left(x_j^{(q)}\right)\}_{i=1}^{m}$
- weighted: $y_j^{(q)} = \sum_{i=1}^{m} w_i f_i\left(x_j^{(q)}\right)$

It is worth noting that max and min can be useful only if the ranking scores produced by various rankers can be compared to each other directly, however, in practice, this can hardly be true.

A special form of ranking problem is that only the best object is required as output. In this case, we have another version of g_2 which is called voting:

- voting: $y_j^{(q)} = \sum_{i=1}^{m} \text{sign}(f_i\left(x_j^{(q)}\right))$

in which $sign(\bullet)$ is an indicator function

$$sign(\bullet) = \begin{cases} 1 & \text{if } f_i \text{ outputs } o_j^{(q)} \text{ as the best object} \\ 0 & \text{otherwise} \end{cases}$$

A general algorithm for MaCR is presented in Algorithm 2.

Algorithm 2 MaCR Algorithm.

Input:
a query q; a set of objects $o^{(q)}$; a set of m ranking functions $\mathcal{F}^*$; a composite function g_2
Output:
a set of ranking scores $y^{(q)}$

for $j = 1; j <= n^{(q)}; j++$ **do**
  Form a feature vector $x_j^{(q)}$.
  Compute ranking scores: $f_1(x_j^{(q)}), \ldots, f_m(x_j^{(q)})$.
  Compute composite function: $y_j^{(q)} = g_2(\bullet)$.
end for
return $y^{(q)}$

3.3 Micro-Macro Collaborative Ranking (MiMaCR)

The above two ranking approaches can be further integrated into a joint model which is named Micro-Macro Collaborative Ranking (MiMaCR). In order to compute query-level and ranker-level collaboration jointly, we solve the following complex composite function g_3:

$$y_j^{(q)} = g_2(g_1(\bullet)) \quad (4)$$

in which, for each object $o_j^{(q)}$, firstly we compute m micro-ranking scores using m ranking functions on query-level collaborators:

$$m \begin{cases} g_1(f_1\left(x_j^{(q)}\right), f_1\left(x_j^{(cq_1)}\right), \ldots, f_1\left(x_j^{(cq_k)}\right)) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ g_1(f_m\left(x_j^{(q)}\right), f_m\left(x_j^{(cq_1)}\right), \ldots, f_m\left(x_j^{(cq_k)}\right)) \end{cases}$$

and secondly, we compute a macro-ranking score using g_2.

We can similarly define g_1 and g_2 as those in MiCR and MaCR. A general algorithm for MiMaCR is presented in Algorithm 3.

Algorithm 3 MiMaCR Algorithm.

Input:
a query q; a set of objects $o^{(q)}$; a set of ranking functions $\mathcal{F}^*$; functions g_1, g_2

Output:
a set of ranking scores $y^{(q)}$

1: Search k collaborators of q: $cq^{(q)} = \{cq_1, \ldots, cq_k\}$.
2: **for** $j = 1; j <= n^{(q)}; j++$ **do**
3: Form $k+1$ feature vectors: $x_j^{(q)}, x_j^{(cq_1)}, \ldots, x_j^{(cq_k)}$.
4: Compute m micro-ranking scores using $\mathcal{F}^*$ and g_1.
5: Compute the macro-ranking score using g_2.
6: **end for**
7: **return** $y^{(q)}$

4 A Case Study on Entity Linking

4.1 Task Definition

The entity linking task aims to align a textual mention of a named entity (person,organization or geo-political) to an appropriate entry in a knowledge base (KB), which may or may not contain the entity. More formally, given a large corpus $\mathcal{C}$, let $q = (q.id, q.string, q.text)$ denote a query in the task which is a triple consisting of query id ($q.id$), name string ($q.string$) and context document ($q.text \in \mathcal{C}$). Let $o^{(q)} = \left\{o_1^{(q)}, \ldots, o_{n^{(q)}}^{(q)}\right\}$ denote the candidate KB entries associated with the query. Each KB entry is a tuple consisting of KB id, KB title, KB infobox (a set of attribute-value pairs that summarize or highlight the key features of the concept or subject of this entry) and KB text. The goal is to rank the KB entries and determine whether the top entry id should be considered as the answer, otherwise NIL should be returned.

A specific example of the task is as follows, given a name string "Michael Jordan" and its context document "...England Youth International goalkeeper *Michael Jordan*...". From the name string, we retrieve a set of candidate KB entries including "Michael Jordan (mycologist)", "Michael Jordan (footballer)", etc. The entity linking system should return the id of "Michael Jordan (footballer)" as the answer, rather than the id of "Michael Jordan" who is most well known as a basketball player.

4.2 General Framework

A general framework of entity linking consists of two crucial components, one for candidate generation, the other for candidate ranking. In this paper, we developed the first component by following the procedures described in (Chen et al., 2010) which extensively leveraged resources mined from Wikipedia. The performance of the first component is 96.8% measured by recall (the percentage of queries in which the candidates cover the true answer). We then focus on the second component.

4.3 Baseline Rankers

We developed 8 baseline rankers, including 4 unsupervised rankers (f_1, f_2, f_3, f_4) and 4 supervised rankers(f_5, f_6, f_7, f_8). We use $f_1, \ldots, f_8$ to denote the 8 ranking functions.

•Naive (f_1): since the answer for each query can either be a KB id or NIL, the naive ranker simply outputs NIL for all queries.

•Entity (f_2): f_2 is defined as weighted combination of entity similarities in three types (person, organization and geo-political). Name entities are extracted from $q.text$ and KB text respectively using Stanford NER toolkit[3]. The formulas to compute entity similarities are defined in (Yoshida et al., 2010).

•Tfidf (f_3): f_3 is defined as cosine similarity between $q.text$ and KB text using tfidf weights.

•Profile (f_4): f_4 is defined as profile similarity between $q.text$ and KB text (Chen et al., 2010). We used a slot filling toolkit (Chen et al., 2011) to generate the profile (attribute-value pairs) for each query.

•Maxent (f_5): a pointwise ranker implemented using OpenNLP Maxent toolkit[4] which is based on maximum entropy model.

•SVM (f_6): a pointwise ranker implemented using SVM^{light} (Joachims, 1999).

[3]http://nlp.stanford.edu/software/CRF-NER.shtml
[4]http://maxent.sourceforge.net/about.html

•SVM ranking (f_7): a pairwise ranker implemented using SVM^{rank} (Joachims, 2006).

•ListNet (f_8): a listwise ranker presented in (Cao et al., 2007).

The four supervised rankers apply exactly the same set of features except that SVM ranking (f_7) needs to double expand the feature vector. The features are categorized into three levels, surface features (Dredze et al., 2010; Zheng et al., 2010), document features (Dredze et al., 2010; Zheng et al., 2010), and profiling features (entity slots that are extracted by the slot filling toolkit (Chen et al., 2011)).

4.4 MiCR for Entity Linking

We convert the collaborator searching problem into a clustering problem, i.e., for a given query q in the task, we retrieve at most $K = 300$ documents from the large corpus $\mathcal{C}$, each of which contains $q.string$; we then apply a clustering algorithm to generate clusters over the documents, and form query collaborators (excluding $q.text$) from the cluster that contains $q.text$.

We experimented the following two clustering approaches:

(1)*agglomerative clustering*: it iteratively merges clusters from singleton documents until a stop threshold is reached. Document similarity is defined as cosine similarity using tfidf weights. We applied group-average linking strategy to merge clusters (Manning et al., 2008).

(2)*graph-based clustering*: it iteratively partitions clusters from one single cluster until a stop threshold is reached. Document similarity is similarly defined as agglomerative clustering. We selected normalized spectral clustering as our clustering algorithm (Shi and Malik, 2000).

We first selected f_3 as our basic ranking function, and investigated whether the ranker can benefit from query collaborators formed by either agglomerative clustering or graph clustering. We implemented three versions of composite function g_1 (*max*, *min* and *average*), and experimented their performance using three unsupervised rankers f_2, f_3, f_4 respectively.

Last, we implemented three supervised versions of g_1 (Maxent, SVM and ListNet respectively) by adding cluster-level features and retraining the models in three supervised rankers f_5, f_6, f_8 respectively. Cluster-level features include maximum, minimum, average tfidf/entity similarities between the candidate and the query collaboration group.

4.5 MaCR for Entity Linking

We implemented two versions of composite function g_2, *average* and *voting*. Furthermore, we investigated how the performance can be affected by incrementally adding more rankers into the ranker set $\mathcal{F}^*$. To do so, we first sorted the 8 rankers according to their performance on the *development* set from the highest to the lowest, and starting with the highest performance ranker, we added one ranker at a time, until we have all the 8 rankers. It is worth noting that, when there are even number of rankers in the set $\mathcal{F}^*$, "ties" could take place using *voting* function. In order to break the ties, we rank the candidate higher if it is output as the answer from a higher performance ranker.

4.6 MiMaCR for Entity Linking

We investigated how the final performance can be boosted by jointly computing micro-ranking scores and macro-ranking score.

5 Experiments

5.1 Data and Evaluation Metric

We used TAC-KBP2009 evaluation data as our training (75%) and development set (25%), and used TAC-KBP2010 evaluation data as our blind testing set (shown in Table 2).

Corpus	Queries			
	PER	ORG	GPE	Total
Training&Dev	627	2710	567	3904
Testing	750	750	750	2250

Table 2: Training, development and testing corpus.

The reference KB consists of 818,741 entries which are extracted from an October 2008 dump of English Wikipedia. The source text corpus (denoted as $\mathcal{C}$ in section 4.1) consists of 1,777,888 documents in 5 genres (mostly Newswire and Web Text).

We used the official evaluation metric for TAC-KBP2010 entity linking task, that is, micro-averaged accuracy. It is computed by

$$\textit{micro-averaged accuracy} = \frac{\#\textit{correct answers}}{\#\textit{queries}}$$

An answer is considered as correct if the system output (either a KB entry id or NIL) exactly matches the key.

5.2 Performance of 8 Baseline Rankers

Table 3 shows the performance of the 8 baseline rankers in 4 columns: *Overall* for all queries, *PER* for person queries, *ORG* for organization queries, and *GPE* for geo-political queries. Each column is further split into *All*, *KB* (for Non-NIL queries) and *NIL* (for NIL queries). It shows that all the four supervised rankers perform better than the four unsupervised rankers. Naive ranker obtains the lowest overall micro-average accuracy (54.5%) but the highest NIL accuracy (100%). Among the four unsupervised rankers, *profile* ranker performs the best, which clearly shows that the extracted attributes of entities are effective for disambiguating confusable names. Among the four supervised rankers, ListNet outperforms SVM ranking and then SVM ranking outperforms the two pointwise rankers. It may confirm previous research findings that listwise ranking is superior to pairwise ranking and pairwise ranking is superior to pointwise ranking (Cao et al., 2007; Zheng et al., 2010).

5.3 Impact of MiCR

To study the impact of MiCR, we first select f_3 (*tfidf* ranker) as our ranking function. Figure 2 shows the performance of applying different query collaborator searching strategies (graph or agglomerative clustering) and different versions of g_1 (average, max and min respectively). We intentionally adjust the meaning of threshold (x-axis) for both graph clustering and agglomerative clustering, such that at threshold 0, both clustering algorithms generate the largest number of clusters (i.e., each document is a cluster), and at threshold 1, they generate only one cluster. We now take the *average* function (Figure 2 (a)) into considerations, as graph clustering algorithm gradually partitions from one cluster (corresponding to threshold 1) to more clusters, the number of query collaborators gradually reduces, meanwhile, the accuracy gradually increases and reaches the highest (73.6%) at threshold of 0.45, which clearly shows that removing noisy collaborators in the query collaboration group can improve the performance. As the threshold continues dropping below 0.45, the number of query collaborators reduces and the performance significantly drops until it reaches the baseline performance of *tfidf* ranker (68.3%). It clearly shows that maintaining a controllable number of query collaborators can improve the performance. For the agglomerative clustering, it is the other story. As it continues merging from singleton clusters (corresponding to threshold 0) to one single cluster, the performance continues increasing until in the end it reaches the highest accuracy of 72.6%. However, unlike graph clustering, a peak never appears in the middle which implies that agglomerative clustering is inferior to graph clustering.

The *max* function (Figure 2 (b)) leverages the strengths from the strongest collaborator in the group, which can potentially improve KB accuracy, but meanwhile hurt NIL accuracy. As shown in the figure, as more collaborators join in the group, the performance increases first for both graph and agglomerative clustering, however, it starts to deteriorate when arriving at a threshold, and in the end, the performance drops even lower than the baseline of tfidf ranker.

The *min* function (Figure 2 (c)) leverages the strengths from the weakest collaborator in the group, which can potentially improve NIL accuracy, but meanwhile hurt KB accuracy. Our data analysis shows that the gain in NIL accuracy can not afford the larger loss in non-NIL accuracy, therefore, the performance continues dropping as the threshold increases. Min function is a counter example showing that searching query collaborators can not always lead to benefits.

To summarize so far, the best strategy for *tfidf* ranker in MiCR approach is *graph-ave* (applying graph clustering and using average function) which obtains overall accuracy gain of 5.3% over the baseline (68.3%). We further validate the performance of *graph-ave* using f_2, f_4 ranking functions, for *entity* ranker, we obtain accuracy gain of 6.3%, and for *profile* ranker, we obtain accuracy gain of 3.0%.

We then experiment the three supervised g_1 functions (ListNet, Maxent, and SVM respectively) using graph clustering as the query collaborator searching strategy. Figure 5 shows that ListNet, Maxent, SVM rankers obtain accuracy gain of 1.4%, 4.6%, 4.2% respectively over the baselines (corresponding to those points at threshold 0).

	Overall (%)			PER (%)			ORG (%)			GPE (%)		
	All	KB	NIL	All	KB	NIL	All	KB	NIL	All	KB	NIL
Naive	54.5	0.0	**100**	70.8	0.0	**100**	59.7	0	**100**	33.0	0	**100**
Entity	65.6	48.6	79.7	82.1	52.1	94.5	68.4	46.2	83.3	46.1	48.5	41.3
Tfidf	68.3	45.0	87.7	83.6	54.3	95.7	66.2	45.9	80.0	54.9	40.3	84.6
Profile	75.0	58.7	88.6	90.8	82.2	94.4	73.3	62.7	80.4	61.0	46.1	91.1
Maxent	77.4	72.3	81.6	86.5	**82.6**	94.4	73.3	62.7	80.4	61.0	**71.5**	72.1
SVM	78.1	**73.0**	82.3	91.1	81.7	94.9	78.7	**70.0**	84.6	64.4	71.1	51.0
SVM Rank	80.3	66.7	91.7	**91.3**	76.3	97.6	77.3	59.7	89.1	72.3	66.7	83.8
ListNet	**81.1**	69.7	90.6	90.8	77.6	96.2	**79.0**	64.0	89.1	**73.5**	69.7	81.4

Table 3: Comparison of 8 baseline rankers.

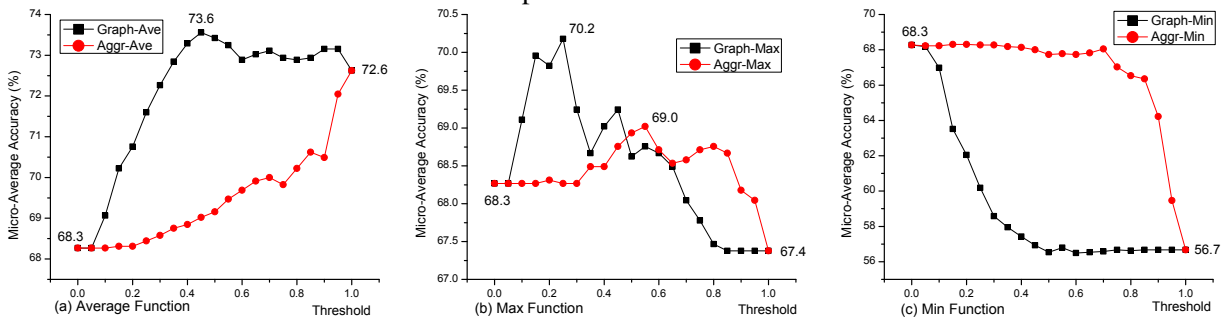

Figure 2: MiCR: comparison of average, max, and min functions with Graph and Agglomerative (Aggr)-based query collaborator searching strategies (*tfidf* ranker).

5.4 Impact of MaCR

Figure 3 shows that the MaCR approach obtains absolute accuracy gain of 1.3% (*voting* function) and 0.5% (*average* function) over the best baseline ranker (81.1%) when we add the 7th ranker (*entity* ranker). The improvement of *voting* function is statistically significant at a 99.6% confidence level by conducting Wilcoxon Matched-Pairs Signed-Ranks Test on the 10 folds of the testing set. However, the improvement of average function is not significant at the 0.05 level which implies that *average* is inferior to *voting*. We observe that the performance drops when there are even number of rankers in the ranker set using voting function, which implies that our tie breaking strategy is not very effective.

We also experimented the *voting* function on the top 10 KBP2009 entity linking systems (each system performance is shown in the table embedded in Figure 4, and experiment is similarly done as described in section 4.5). Figure 4 shows that it can obtain absolute accuracy gain of 4.7% over the top entity linking system (82.2%). The reasons why we achieve relative smaller gains using our own ranker set are as follows: (1) we use the same candidate object set for all rankers, while different KBP2009 systems may use their own set of objects. (2) our top 4 supervised rankers apply almost the same set of features, while different KBP2009 systems may apply more diversified features. Therefore, diversity is a highly important factor that makes MaCR approach effective.

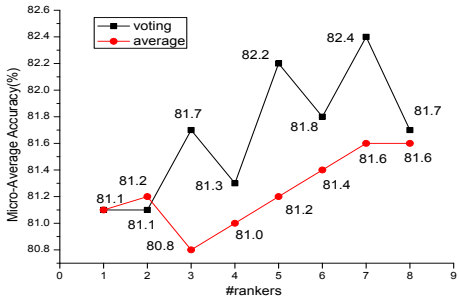

Figure 3: MaCR: comparison of voting and average.

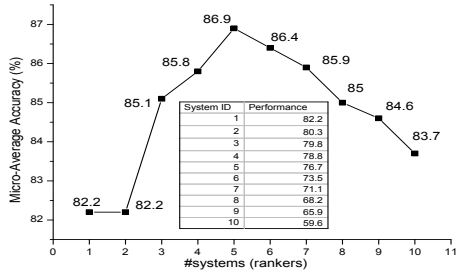

Figure 4: MaCR: voting function applied to the top 10 KBP2009 entity linking systems.

5.5 Impact of MiMaCR

We applied the following settings in our MiMaCR approach: selecting graph clustering as the query collaborator searching strategy, including five rankers (tfidf, entity, Maxent, SVM and ListNet) in the ranker set, using *average* function to compute micro-ranking scores for the tfidf and entity ranker, using the three corresponding supervised versions of g_1 to compute micro-ranking scores for Maxent, SVM and ListNet respectively, and finally applying *voting* function to compute the macro-ranking score. In Figure 5, the curve of "MiMaCR" shows how the performance of MiMaCR is affected by the threshold in graph clustering. We obtain the best micro-average accuracy of 83.7% at threshold 0.3, which is 2.6 % higher than the best baseline ranker (81.1%). The improvement is statistically significant at a 98.6% confidence level by conducting Wilcoxon Matched-Pairs Signed-Ranks Test on the 10 folds of the testing set. The score reported here is on par with the second best in the KBP2010 evaluation.

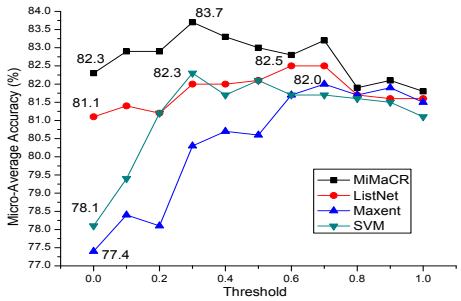

Figure 5: MiMaCR: Comparison of MiMaCR and three supervised versions of g_1 (ListNet, Maxent, and SVM respectively).

6 Related Work

In the literature of information retrieval, query expansion is a useful technique that involves the process of reformulating a query, and as a consequence, is capable to extend the ability of a query and improve the retrieval performance. Various approaches for query expansion have been proposed, as summarized in (Manning et al., 2008). The MiCR presented in this paper is superior to query expansion in two aspects, firstly, we leverage more information contained in multiple query collaborators; secondly, we place great emphasis on interactions among members in the query collaboration group.

In the literature of machine learning, there has been a considerable amount of research on ensemble-based classification, which is to build a predictive classification model by integrating multiple classifiers. A comprehensive survey is presented in (Rokach, 2009). In contrast, ensemble-based ranking has only recently attracted research interests (Hoi and Jin, 2008; Wei et al., 2010). Although the MaCR presented here is in essence ensemble-based ranking, we extend it to MiMaCR which integrates the strengths from both MiCR and MaCR.

It is worth noting that "collaborative ranking" presented here should be distinguished from "collaborative filtering" in that "collaborative filtering" uses the known preferences of a group of users to generate personalized recommendations while "collaborative ranking" leverages query collaborators and ranker collaborators to enhance the overall ranking performance.

There has been an increasing amount of research on entity linking, especially through KBP2009 and KBP2010. Various unsupervised or supervised approaches have been proposed, as summarized in (McNamee and Dang, 2009; Ji et al., 2010). However, most of the previous research mainly focused on one or two ranking algorithms on isolated queries. In this paper, we have extended the work by systematically studying the possibility of performance enhancement through query-level collaboration and ranker-level collaboration.

7 Conclusions

We presented a new ranking scheme called *collaborative ranking* with three specific forms, MiCR, MaCR and MiMaCR and demonstrated its effectiveness on entity linking task. However, our scheme is not restricted to this specific task and it is generally applicable to many other other applications such as question answering. In MiCR, effective searching of query collaborators and active interplay among members in the query collaboration group are two key factors that make MiCR successful. In MaCR, diversity is a highly important factor to make it successful. Overall, MiMaCR can bootstrap the performance to its maximum if integrating MiCR and MaCR properly. However, the better performance is at the expense of much more computations.

Acknowledgments

This work was supported by the U.S. Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053, the U.S. NSF CAREER Award under Grant IIS-0953149 and PSC-CUNY Research Program. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

P. L. Bartlett, M. I. Jordan and J. D. McAuliffe. 2003. Convexity, classification, and risk bounds. *Technical Report 638*, Statistics Department, University of California, Berkeley.

C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. 2005. Learning to Rank Using Gradient Descent. In *Proceedings of the 22th International Conference on Machine Learning (ICML 2005)*.

Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai and H. Li. 2007. Learning to rank: from pairwise approach to listwise approach In *Proceedings of the 24th International Conference on Machine Learning (ICML 2007)*, pages 129-136.

E. Charniak and M. Johnson. 2005. Coarseto-fine-grained n-best parsing and discriminative reranking. In *ACL-05*, pages 173-180.

W. Chen, T.-Y. Liu, Y. Lan, Z. Ma, and H. Li. 2009. Ranking measures and loss functions in learning to rank. In *Advances in Neural Information Processing Systems 22 (NIPS 2009)*, pages 315-323.

Z. Chen, S. Tamang, A. Lee, X. Li, W.-P. Lin, M. Snover, J. Artiles, M. Passantino and H. Ji. 2010. CUNYBLENDER TAC-KBP2010 Entity Linking and Slot Filling System Description. In *Proceedings of Text Analytics Conference (TAC2010)*.

Z. Chen, S. Tamang, A. Lee and H. Ji. 2011. A Toolkit for Knowledge Base Population. In *SIGIR*.

M. Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pages 175-182.

M. Dredze, P. McNamee, D. Rao, A. Gerber and T. Finin. 2010. Entity Disambiguation for Knowledge Base Population. *Proc. COLING 2010*.

Y. Freund, R. Iyer, R. Schapire, and Y. Singer. 2003. An efficient boosting algorithm for combining preferences. In *Journal of Machine Learning Research*, 4:933-969.

S. Hoi and R. Jin. 2008. Semi-supervised ensemble ranking. In *Proc. of the 23rd AAAI Conf. on Artificial Intelligence*.

L. Huang. 2008. Forest Reranking: Discriminative Parsing with Non-Local Features. In *ACL-HLT-08*, pages 586-594.

H. Ji, R. Grishman, H. T. Dang and K. Griffit. 2010. An Overview of the TAC2010 Knowledge Base Population Track. In *Proceedings of Text Analytics Conference (TAC2010)*.

T. Joachims. 1999. Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.

T. Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD 2002)*.

T. Joachims. 2006. Training Linear SVMs in Linear Time. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*.

Y. Lan, T.-Y. Liu, T. Qin, Z. Ma, and H. Li. 2008. Query-level stability and generalization in learning to rank. In *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)*, pages 512-519.

P. Li, C. Burges, and Q. Wu. 2007. Mcrank: Learning to rank using multiple classification and gradient boosting In *Advances in Neural Information Processing Systems 20 (NIPS2007)*.

Y. Lin. 2002. Support vector machines and the bayes rule in classification. In *Data Mining and Knowledge Discovery*, pages 259-275.

C. D. Manning, P. Raghavan and H. Schütze. 2008 . Introduction to Information Retrieval. Cambridge University Press.

P. McNamee and H. Dang. 2009. Overview of the TAC 2009 Knowledge Base Population Track. In *Proceedings of TAC*.

R. Nallapati. 2004. Discriminative models for information retrieval. In *SIGIR*.

F. J. Och. 2002. Statistical Machine Translation: From Single-Word Models to Alignment Templates. Ph.D. thesis, Computer Science Department, RWTH Aachen, Germany, October.

L. Page, S. Brin, R. Motwani, and T. Winograd. 1998. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project.

T. Qin, X.-D. Zhang, M.-F. Tsai, D.-S. Wang, T.-Y. Liu and H. Li. 2007. Query-level loss functions for information retrieval. In *Information Processing and Management*.

T. Qin, T.-Y. Liu, X.-D. Zhang,D.-S. Wang, and H. Li. 2008. Global Ranking Using Continuous Conditional Random Fields. In *Advances in Neural Information Processing Systems 21 (NIPS 2008)*.

D. Ravichandran, E. Hovy and F. J. Och. 2003. Statistical QA - Classifier vs. Re-ranker: What's the difference? In *Proceedings of the ACL Workshop on Multilingual Summarization and Question Answering*.

S. Riezler, A. Vasserman, I. Tsochantaridis, V. Mittal and Y. Liu. 2007. Statistical Machine Translation for Query Expansion in Answer Retrieval. In *Proceedings of ACL*.

L. Rokach. 2009. Ensemble-based classifiers. Artif Intell Rev DOI 10.1007/s10462-009-9124-7.

L. Shen, A. Sarkar, and F. J. Och. 2005. Discriminative reranking for machine translation. In *Proceedings of HLT-NAACL*.

J. Shi and J. Malik. 2000. Normalized Cuts and Image Segmentation. In *Machine Intelligence*, vol. 22, no. 8, pages 888-905.

F. Wei, W. Li and S. Liu. 2010. iRANK: A rank-learn-combine framework for unsupervised ensemble ranking. In *Journal of the American Society for Information Science and Technology*,61: 1232C1243. doi: 10.1002/asi.21296.

X. Yang and J. Su and C.L. Tan 2008. A Twin-Candidate Model for Learning-based Anaphora Resolution. In *Computational Linguistics*, vol. 34, no. 3, pages 327-356.

M. Yoshida, M. Ikeda, S. Ono, I. Sato, and H. Nakagawa. 2010. Person name disambiguation by boostrapping. In *SIGIR*.

T. Zhang. 2004. Statistical analysis of some multicategory large margin classification methods. In *Journal of Machine Learning Research*, 5, 1225-1251.

W. Zhang, J. Su, C. L. Tan and W.T. Wang. 2010. Entity Linking Leveraging Automatically Generated Annotation. *Proc. COLING 2010*.

Z. Zheng, F. Li, M. Huang, X. Zhu. 2010. Learning to Link Entities with Knowledge Base. *Proc. Proc. HLT-NAACL2010*.