CUNY-BLENDER TAC-KBP2012 Entity Linking System and Slot Filling Validation System

Suzanne Tamang, Zheng Chen and Heng Ji
Computer Science Department and Linguistics Department
Queens College and Graduate Center
City University of New York
hengji@cs.qc.cuny.edu

Abstract

This year the CUNY-BLENDER team participated in the English Entity Linking and Slot Filling Validation tracks. for entity linking, we apply two new techniques, collaborative clustering and query reformulation. For answer validation, we use a logistic regression model trained on within-system and cross-system features to re-rank the merged answer sets generated by individual systems. In this paper, we describe our approaches to each task in more detail, assess the impact of out techniques, and presents new directions for future work.

1 Introduction

This is the third year we participated in KBP2012 evaluation. Since our systems are based on extending our previous approaches, in the following we will focus on describing the new techniques that we have developed this year, for English Entity Linking (Section 2 and Section 3) and Slot Filling Validation (Section 5 and Section 6) respectively.

2 English Entity Linking Approach

2.1 System Overview

We enhanced our English entity linking system which has been developed in the past two years. There are two major improvements we made in the new system:

(1) Enhance the query preprocessing by reformulating ambiguous query names with less ambiguous query names.

(2) Apply a new clustering scheme which is called collaborative clustering (CC) to NIL clustering.

Figure 1 shows how we have enhanced our entity linking system through the year of 2010, 2011 and 2012. In 2010, we applied Wikipedia resources for query expansion (e.g., extra names from Wikipedia redirect and disambiguation pages), and used Lucene search engine to retrieve candidate KB entries. For candidate ranking, we only applied two simple unsupervised ranking methods (Chen et al., 2010).

In 2011, we proposed a new ranking scheme called "collaborative ranking" (CR) (Chen and Ji, 2011). 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 specified three forms of CR, namely, micro collaborative ranking (MiCR), macro collaborative ranking (MacR) and micro-macro collaborative ranking (MiMaCR). We applied this new scheme to enhance the candidate ranking component in our entity linking system. Furthermore, we extended ranker-level macro collaborative ranking to system-level macro collaborative ranking by combining CUNY entity linking system with UIUC entity linking system (Cassidy et al., 2010).

This year, we proposed a new clustering scheme called "collaborative clustering" (CC). In contrast with traditional non-collaborative clustering scheme which only relies on the information in clustering instances and one single clustering algorithm, the proposed scheme can leverage more information from collaborative instances which help to better reshape the clustering structure in the original data set, and furthermore, it can leverage the strengths from multiple clustering solutions. We specified three forms of CC, namely, micro collaborative clustering (MiCC), macro collaborative clustering (MaCC) and micro-macro collaborative clustering (MiMaCC). We have applied this new clustering scheme to entity clustering problem, especially NIL clustering problem. However, after analyzing the queries in this year's evaluation, we conclude that the NIL clustering problem is still not challenging enough such that our new clustering scheme cannot be taken full advantage of.

In the following two sections, we focus on the two major improvements we made in the new system.

2.2 Query Reformulation

Ambiguity is an important index to measure the difficulty of entity linking task. A name is *ambiguous* if it can refer to more than 2 entities. After analyzing the queries, we found that the queries in this year's evaluation are much more ambiguous than the previous two years, as there are many more single word queries which can indicate very diverse entities. More specifically, we found that

(1) if the query name is a person name, it is often expressed only using last name rather than the full name which can be found in the context document.

(2) if the query name is an organization name, it is often expressed using its acronym, rather than the expanded name which may or may not be found in the context document. Some queries are about the abbreviations of sport teams and their full names can be found in the context document.

(3) if the query name is a GPE name, it is often expressed using the city name, but the city name often appears in the structure "[City name], [State name]" in the context document.

We reformulated to significantly reduce the ambiguity in the queries. For example, we reformulated a person's last name by his/her full name, an acronym of an organization name by its expanded form, and a city name by a more precise form of [City name], [State name].

The ambiguity can be measured as the number of *ambiguous* names divided by the total number of names in the evaluation corpus. Using this metric, we obtained the ambiguity values of 19.6% (101/514)[1], 12.9% (97/752), 13.1% (173/1325) and 46.3% (376/812) for 2009, 2010, 2011 and 2012 evaluation respectively. We can clearly see that the names in this year's evaluation are far more ambiguous than previous years. However, by applying query reformulation, the ambiguity in 2012 evaluation drops significantly from 46.3% to 11.2% (200/1780). That clearly shows that query reformulation can significantly reduce the difficulty of entity linking task.

The main techniques for our query reformulation are as follows:

(1) if the query name is a person name, and it only has one token, we applied within document entity coreference resolution (part of our CUNY Information Extraction toolkit) to reformulate the query using the proper name with the maximum number of tokens. For example, if the query name is "Obama" and the following proper names are mentioned in the context document "Barack Obama", "Barack Hussein Obama", which are found to corefer to each other by within document coreference resolver, we then pick "Barack Hussein Obama" as the new query.

(2) if the organization name is an acronym, we applied pattern matching "[Expanded name] ([Query name])" or "[Query name] ([Expanded name])" to get the expanded name for the query. We picked the expanded name as the new query.

(3) if the GPE name starts with only [City name], we than applied pattern matching "[City name], [State name]" to get the expanded name for the query. We also picked the expanded name as the new query.

Although the above query reformulation techniques can significantly reduce the ambiguity of queries, however, due to the imperfect performance of within-document coreference resolution and pattern matching, erroneous reformulated queries may be introduced.

[1] in the parenthesis, the left side shows the number of ambiguous names, and the right side shows the number of total names

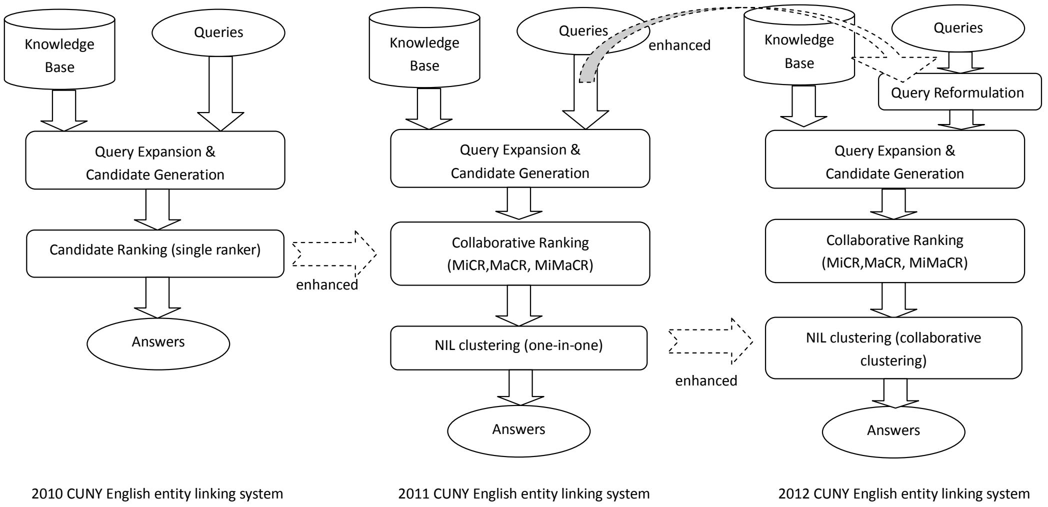
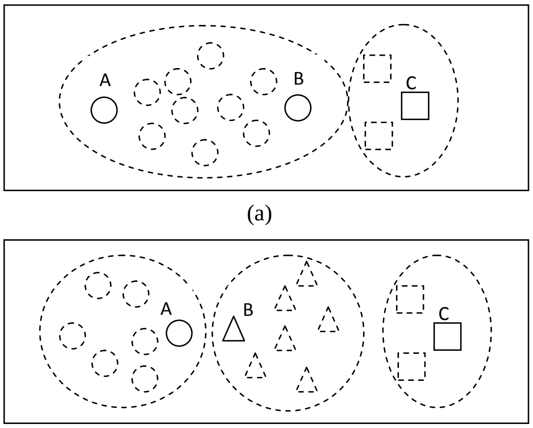

Knowledge Base
Queries
Query Expansion & Candidate Generation
Candidate Ranking (single ranker)
Answers
enhanced
Knowledge Base
Queries
enhanced
Query Expansion & Candidate Generation
Collaborative Ranking (MiCR,MaCR, MiMaCR)
NIL clustering (one-in-one)
Answers
enhanced
Knowledge Base
Queries
Query Reformulation
Query Expansion & Candidate Generation
Collaborative Ranking (MiCR,MaCR, MiMaCR)
NIL clustering (collaborative clustering)
Answers
2010 CUNY English entity linking system
2011 CUNY English entity linking system
2012 CUNY English entity linking system

Figure 1: Evolution of CUNY English Entity Linking System through 2010,2011 and 2012

2.3 Collaborative Clustering

2.3.1 Motivation

Clustering is generally formulated as grouping a set of instances into clusters such that members in each cluster are more similar than those in other clusters. Clustering is known to suffer from little prior knowledge of underlying data population. As a result, instances from the original data set may not always convey a good clustering structure. A promising solution is to add a predominant number of representative instances such that they can possibly reshape or enhance the clustering structure. For example, in Figure 2 (a), the original data set contains three instances A, B and C, since B and C are physically closer than A and B, it is likely that a clustering algorithm clusters B and C together. However, the actual case is that A and B belong to the same cluster and they are located at the border of the cluster. After we add instances that fill the core of the cluster which contains A and B, the clustering structure becomes visible. In the other example shown in Figure 2 (b), the original data set also contains three instances A,B and C. Since A and B are physically closer than B and C, it is likely that a clustering algorithm clusters A and B into one cluster. However, the actual case is that A and B belong to two clusters and each is located at the border of its own cluster. Only after adding representative instances as shown in Figure 2 (b), the clustering structure becomes visible. We call the extra added instances as "collaborative instances" and this style of clustering approach as "instance-level collaborative clustering".



(b)

Figure 2: Motivating examples for instance-level collaborative clustering.

It is also known that there does not exist a single perfect clustering algorithm that can always discover a good clustering structure in every data set. For example, K-means algorithm is good at clustering spherical shapes of clusters, but can be very sensitive to noise and can not handle well with non-convex shapes of clusters. A clustering algorithm normally optimizes some objective function which measures the clustering quality based on the information embedded in the data, however, it turns out that in most cases solving the optimization is a NP-Hard problem, and as a result, the clustering is always sub-optimal due to some approximative algorithm. A promising solution is to apply a set of diverse clustering approaches and then obtain a consensus clustering by combining multiple clustering results. We call this style of clustering approach as "clusterer[2]-level collaborative clustering" which has a well-known name of "clustering ensemble" in the literature.

Based on the above two motivations, we present a collaborative clustering scheme which includes the following three specific forms.

(1) MiCC (corresponding to instance-level[3] collaboration) leverages the information contained in the collaborative instances.

(2) MaCC (corresponding to clusterer-level[4] collaboration) integrates the strengths from multiple clustering algorithms.

(3) MiMaCC combines the advantages of MiCC and MaCC.

2.3.2 Micro Collaborative Clustering (MiCC)

Micro collaborative clustering (i.e.,instance-level collaborative clustering) takes the advantage of looking into a bigger and better vision of the clustering structure by adding collaborative instances. The success of MiCC involves the following key issues:

(1) *mechanism of populating collaborative instances* which deals with a method to produce potential collaborative instances;

(2) *internal measure* which computes the quality of clustering structure purely based on the data rather than the gold clustering;

(3) *algorithm of MiCC* which deals with how to gradually add collaborative instances and determines when to stop adding collaborative instances;

Mechanism of Populating Collaborative Instances

Generally, since we do not know the function of data distribution for a clustering problem, it is hard to populate collaborative instances out of the void. However, in many real clustering applications, it is possible to find such collaborative instances. For example, in document clustering, we can use other documents collected from the same source as potential collaborative documents, for example, if the testing data set is mostly from New York Times, we can collect more documents published from the same data source.

Internal Measure

The basic hypothesis of adding collaborative instances is that they can help uncover a good clustering structure. Therefore, an important question is how to measure the quality of clustering structure only using information in the data rather than external information (thus called internal measure). Most of the previous proposed internal measures are based on two criteria including *cohesion* which measures how cohesive the instances in a cluster are and *separation* which measures how separated a cluster is from other clusters (Tan et al., 2005).

We studied 12 internal measures that can represent a good coverage of the previous proposed measures, including 6 measures (I_1,I_2,ε_1,H_1,H_2 and G_1) discussed in (Zhao and Karypis, 2004) and another 6 measures (*CH*, *DI*, *SC*,*DB*, *SD*,*S_Dbw*) discussed in (Liu et al., 2010). Most of the previous work on internal measures has focused on whether they can correctly identify the true number of clusters. In this paper, we will examine the 12 internal measures from another aspect: can they help recover the good clustering structure? An important issue related with internal measure is whether the "optimal" clustering computed by an internal measure really turns out to be a good clustering by validating with an external gold clustering using some external measure. A possible solution is to compute the Pearson correlation between the score computed by an internal measure and the score computed by an external measure. The higher correlated they are, the better the internal measure is.

Algorithm of MiCC

The basic idea of MiCC is that we continually check the clustering quality of the original set of instances by incrementally add instances from the pool for multiple rounds ($n_{iterations}$). For each round, we identify a certain number of "best" collaborative instances (n_{step}) which help to achieve the best score

[2] A clusterer is a full-functional clustering approach that produces a clustering

[3] Instance is normally represented by a small-scale data structure, so we call it micro.

[4] Clusterer is normally implemented by a large-scale algorithm, so we call it macro.

computed by an internal measure. In order to identify the "best" n_{step} collaborative instances for each round, we repeat n_{trials} trials by randomly selecting n_{step} collaborative instances from the selection pool. At some point, the clustering quality reaches the optimal and adding more instances can no longer improve or even hurt the score computed by an internal measure. Once the optimal point is reached, we then obtain the best set of collaborative instances, and the clustering result on the expanded data set. The clustering on the original data set can be retrieved by only looking at the cluster ids of those instances in the original data set. The final evaluation using some external measure (i.e., validating with a gold clustering) is still conducted on the original data set. The detailed algorithm is presented in Algorithm 1.

Algorithm 1 MiCC algorithm.

```
Input:
    $X = \{x_1, x_2, ..., x_n\}$: instances in the data set;
    $Y = \{y_1, y_2, ..., y_m\}$: a pool of candidate collaborative instances;
    $\mathcal{F}$: a clustering algorithm;
    $\mathcal{M}$: an internal measure; //Assume optimal is achieved when this measure is maximized
    $n_{step}$: maximum number of collaborative instances picked in each iteration;
    $n_{trials}$: number of times
    $n_{iterations}$: number of iterations
Output:
    a set of best collaborative instances: $C_{opt}$;
    the optimal value computed by the internal measure $O_{opt}$;
Apply $\mathcal{F}$ on $X$ and output a clustering, compute the clustering quality by the internal measure $\mathcal{M}$.
initialize a list of best found candidate collaborative clustering instances $C_1$, ...,$C_{n_{iterations}}$
initialize a list of best found values of clustering quality $O_1$, ...,$O_{n_{iterations}}$
for $i = 1 \to n_{iterations}$ do
    for $j = 1 \to n_{trials}$ do
        Randomly pick $n_{step}$ collaborative clustering instances (naming the set as $Y_j$) from $Y$ and produce a new set of instances by $X \cup Y_j$.
        Apply $\mathcal{F}$ on $X \cup Y_j$, compute clustering quality $O_{ij}$ using $\mathcal{M}$.
    end for
    Find $O_{ik}$ such that $O_{ik} \geq O_{ij}$ for $j \in \{1, ..., n_{trials}\}$, expand $X$ such that $X = X \cup Y_k$, remove $Y_k$ from $Y$ such that $Y = Y - Y_k$.
    $C_i = Y_k$, $O_i = O_{ik}$
end for
Find $O_u$ such that $O_u \geq O_i$ for $i \in \{1, ..., n_{iterations}\}$
$O_{opt} = O_u$
$C_{opt} = C_1 \cup ... \cup C_u$
return $C_{opt}$ and $O_{opt}$
```

2.3.3 Macro Collaborative Clustering (MaCC)

Macro collaborative clustering (i.e., clusterer-level collaborative clustering) takes the advantage of leveraging the integration of diverse clustering results. There are two key issues involved in MaCC:

1. Ensemble generation: A clustering ensemble is a set of clusterings, each of which is generated by a clusterer. Diversified clusterers can be implemented through: (1) different clustering algorithms; (2) different distance functions to compute the distance between two instances; (3) different parameter settings for a specific algorithm; (4) different dimension reduction methods which project from high dimensional space to lower dimensional space.

We denote a clustering ensemble as $\Pi = \{\pi^1, ..., \pi^r\}$ in which r is the number of clusterers and each clustering π^i consists of k_i number of clusters, i.e., $\pi^i = \{C_1^i, ..., C_{k_i}^i\}$ where $C_1^i \cup ... \cup C_{k_i}^i = X$.

2. Consensus function: Given a clustering ensemble $\Pi = \{\pi^1, ..., \pi^r\}$, a consensus function Γ maps the ensemble to an integrated clustering, i.e., $\Gamma : \Pi \to \mathcal{C}$.

Ensemble Generation

We applied the following set of clustering algorithms:

(1) 3 linkage based agglomerative clustering algorithms including complete linkage, single linkage and average linkage (Manning et al., 2008). We use symbols *slink*,*clink*,*alink* to represent the three linkage based algorithms respectively.

(2) 6 agglomerative clustering algorithms that optimize the internal measure I_1,I_2,ε_1,H_1,H_2 and G_1 respectively (Zhao and Karypis, 2002). We use symbols I_1, I_2, ε_1, H_1, H_2 to represent the six algorithms.

(3) 6 repeated bisectional partitioning clustering algorithms that optimize the internal measure I_1,I_2,ε_1,H_1,H_2 and G_1 respectively (Zhao and Karypis, 2002). We use symbols rI_1, rI_2, $r\varepsilon_1$, rH_1, rH_2 to represent the six algorithms.

(4) 6 direct k-way partitioning clustering algorithms that optimize the internal measure I_1,I_2,ε_1,H_1,H_2 and G_1 respectively (Zhao and Karypis, 2002). We use symbols dI_1, dI_2, $d\varepsilon_1$, dH_1, dH_2 to represent the six algorithms.

Various clustering results can be obtained by combining the above clustering algorithms together with similarity functions. In our name entity clustering (NIL clustering), we applied four similarity functions (Chen, 2012), including cosine similarity of two documents, correlation similarity of two documents, maximum entropy based model to compute the similarity of two feature vectors representing the two query context, and SVM model to compute the similarity. We use symbols *cos*, *cor*, *maxen*, *svm* to represent the four similarity functions.

Consensus Functions

A consensus function maps a set of clusterings into a final clustering. Various approaches have been proposed, including mutual information based (Topchy et al., 2003; Luo et al., 2006), voting based (Fischer and Buhmann, 2003; Fred, 2001), co-association matrix based (Fred and Jain, 2002), mixture model based (Topchy et al., 2004), and three graph based approaches(Strehl and Ghosh, 2002; Fern and Brodley, 2004). In this paper, we focus on co-association matrix based and three graph based approaches, namely instance-based graph formulation (IBGF), cluster-based graph formulation(CBGF) and hybrid bipartite graph formulation (HBGF).

2.3.4 Micro Macro Collaborative Clustering (MiMaCC)

It is quite natural to combine instance-level collaborative clustering and clusterer-level collaborative clustering so that we can first recover good clustering structure by using collaborative instances and then apply multiple clustering algorithms to produce a final clustering. The hypothesis is that clustering on data in which good structure is embedded can produce better results than clustering on data in which ill structure is embedded.

A basic algorithm to implement MiMaCC is as follows:

(1) expand the data set by introducing collaborative instances;

(2) apply clusterer-level collaborative clustering to produce a final clustering on the expanded data set;

(3) down-scale the clustering by removing those collaborative instances from clusters.

2.3.5 Case Study: Name Entity Clustering

In this paper, we define name entity clustering as follows and NIL clustering is a specific form of name entity clustering in which each instance is NIL query.

Let $\mathcal{Q} = \{q_1, \ldots, q_n\}$ denote the set of n instances. Each instance $q = (q.id, q.string, q.text)$ is a triple consisting of instance id ($q.id$), name string ($q.string$) and context document ($q.text \in \mathcal{C}$) in which $\mathcal{C}$ is source document corpus.

The goal of name entity clustering is to generate a hard (non-overlapping) clustering for $\mathcal{Q}$, i.e., $\mathcal{Q} = (Q_1, ..., Q_K)$ such that

(1) $Q_i \neq \emptyset$ for $i \in \{1, ..., k\}$.

(2) $Q_i \cap Q_j = \emptyset$ for $i, j \in \{1, ..., k\}$ and $i \neq j$.

(3) $Q_1 \cup ... \cup Q_k = \mathcal{Q}$

in which each cluster Q_i refers to an entity.

3 Entity Linking Experiments

3.1 Data and Evaluation Metric

For the English entity linking experiments, we used KBP 2012 English entity linking evaluation corpus for testing, and used KBP 2009 English entity linking evaluation corpus for training as shown in Table 1. The training corpus was basically used to train supervised ranking models. The English reference Knowledge Base is the same as the one used in previous evaluations which consists of 818,741 nodes derived from an October 2008 dump of English Wikipedia. The English source collection is also the same as the one used in 2011 evaluation which includes 1,286,609 newswire documents, 490,596 web documents, and 683 other documents.

We applied the official evaluation metric which has been proposed during KBP 2011 entity linking evaluation. The metric is called modified B-Cubed (B-Cubed+) F-measure which is the harmonic mean

Table 1: Data sets

	Corpus	#Queries			
		PER	ORG	GPE	Total
Testing	KBP2012	918	706	602	2226
Training	KBP2009	627	2710	567	3904

of B-Cubed+ precision and B-Cubed+ recall. It is worth noting that this measure has shown to be highly correlated with the old micro-averaged accuracy metric in KBP2009 and KBP2010 (Ji et al., 2011).

As we will discuss in the later section, the NIL clustering is still not challenging enough in KBP 2012 evaluation, so simple clustering approach can perform reasonably well. To validate the effectiveness of collaborative clustering we proposed, we created a data set by following the procedure as follows:

(1) we first collected all the queries from KBP 2009, 2010 and 2011 entity linking evaluation corpus. We obtained 6652 queries distributed in 1379 unique spelling names.

(2) we wrote a program to automatically pick out names that satisfy the following conditions: (a) the name should be ambiguous, which means the queries that contain the name should be able to be clustered into 2 or more clusters according to the answer keys; (b) the number of queries that contain the name should be larger than 4; (c) the entity types in those queries that contain the name should be consistent and those name with mixed entity types are filtered; (d) besides the context documents in those queries which share a name, we can find at least 5 relevant documents that also contain the name.

As a result, we obtained a data set consisting of 1686 queries distributed in **106 names**, including **21 person names**, **67 organization names**, and **18 GPE names**.

Instead of B-Cubed+ F-measure, we applied the V-measure to evaluate name entity clustering, because in our previous study (Chen, 2012), we studied more than 20 evaluation metrics by taking into account whether they can capture the few constraints discussed in (Amigo et al., 1987) and whether they can overcome the “uniform effect” discussed in (Wu et al., 2009). We conclude that V-measure which was proposed by (Rosenberg and Hirschberg, 2007) has superior advantages and then in the later experiments, we use it as our evaluation metric.

3.2 Experiment Results

3.2.1 Scores of Our Submitted Runs

We submitted 6 runs, including 5 runs using Wikipedia texts and 1 run without using Wikipedia texts. The 6 runs are common in using the same query reformulation, candidate generation and NIL clustering techniques, however, they differ from each other in using different ranking models (Chen and Ji, 2011)

(1) query level collaborative ranking using maximum entropy based ranking model (Micro-Maxen)

(2) query level collaborative ranking using SVM ranking model (Micro-SVM)

(3) query level collaborative ranking using Listwise ranking model (Micro-Listwise)

(4) query level collaborative ranking using TF-IDF ranking model (Micro-TFIDF)

(5) query level and ranker level collaborative ranking by combining the above four ranking results together with the non-wiki result (Micro-Macro)

The only one run without using Wikipedia texts is based on ranking KB entries by popularity scores (Popularity), i.e., we always pick the most popular KB entry as output, and the popularity score is equivalent to the default relevant score retrieved from KB candidates using Lucene API.

The scores evaluated by B-Cubed+ F-measure are shown in Table 2, in which each row represents a submitted result and the column with scores represents a category of queries, i.e., ALL (for all queries), KB (for queries with KB entries), NIL (for queries which cannot be linked to KB), NW (for queries with associated documents from Newswire sources), WB (for queries with associated documents from Web blogs), PER (for queries with person type), ORG (for queries with organization type), and GPE (for queries with GPE type). The highest score for each category is highlighted in bold.

Not surprisingly, the runs using Wikipedia texts outperform those without using Wikipedia texts. Queries from web data proved more challenging than those from news wire documents. Among the three entity types, our systems performed the best on person queries.

Table 2: Scores of our submitted 6 results

	Submitted result	B-Cubed+ F-measure							
		ALL	KB	NIL	NW	WB	PER	ORG	GPE
wiki	Micro-Maxen	0.678	0.545	0.826	0.728	0.580	0.832	**0.671**	0.449
	Micro-SVM	**0.688**	**0.595**	0.789	**0.741**	**0.583**	**0.833**	0.666	**0.488**
	Micro-Listwise	0.660	0.530	0.804	0.717	0.547	0.802	0.655	0.446
	Micro-TFIDF	0.533	0.474	0.597	0.599	0.405	0.717	0.502	0.284
	Micro-Macro	0.680	0.561	0.812	0.733	0.576	0.831	0.666	0.461
non-wiki	Popularity	0.604	0.386	**0.847**	0.650	0.513	0.796	0.632	0.275

Among the 6 runs, the best overall score is Micro-SVM, Micro-Macro the second, Micro-Maxen the third, and Micro-Listwise the fourth. These contradict with the results we obtained in KBP2011 where we showed that Micro-Macro can perform the best, while Micro-Listwise can perform better than the other two pointwise based ranking approaches, i.e., Micro-Maxen or Micro-SVM. The reasons might be that (a) the training models created from KBP2009 cannot handle some new testing instances in KBP2012 because of unexpected data peculiarities or changes in data distribution; (b) Micro-Macro only integrates 5 runs which may not provide enough varieties and some results may actually hurt the overall performance after combination (e.g., the worst result from Micro-TFIDF). As discussed in our previous paper (Chen and Ji, 2011), the success of Micro-Macro depends on several factors, including the diversity of ranking results, choices of selecting ranking results (extremely bad results can hurt), and combination scheme (voting, average).

3.2.2 Impact of Query Reformulation

As we also discussed in section 2.2, the ambiguity of queries mainly come from three sources. We report the detailed impact of query reformulation as follows.

Source 1. Person Name: using last name as query string

For example, there are 48 queries with the query name of "Clark". However, in many cases, we can find the full name of "Clark" from the context document.

> ***Example 1:*** *District Attorney Mitch Morrissey announced at a news conference at the Denver Police Administration building that* ***Willie Clark*** *faces 39 counts ...*

> ***Example 2:*** *Instead,"figure out what kicks off asthma symptoms," says* ***Noreen Clark****, director of the Center for Managing Chronic Disease ...*

> ***Example 3:*** *Early on New Year's Day,* ***Bill Clark****, a long-distance trucker who was picking up cargo from Maine, ...*

Source 2. Organization Name: using acronym as query string For example, there are 4 queries with the query name of "MMA", and we can find their expanded names from the context documents.

> ***Example 4:*** *Trouble flared as police tried to arrest leaders of the six-partyIslamic alliance* ***Muttahida Majlis-e-Amal (MMA)*** *for staging arally in...*

> ***Example 5:*** *Meanwhile, the* ***Myanmar Medical Association (MMA)*** *has appealed tolocal private doctors to voluntarily provide ...*

Source 3. GPE Name: using city name as query string For example, there are 4 queries with the query name of "BRECKENRIDGE", and we can find the unambiguous form of "BRECKENRIDGE" followed by a state name from the context document.

> ***Example 6: BRECKENRIDGE, Minn.***
> *Minerva Hinojosa and her family migrated north again last month...*

> ***Example 7: BRECKENRIDGE, Texas***
> *2008-04-10 05:43:14 UTC*
> *Powerful storms including apparent tornadoes moved...*

> ***Example 8: BRECKENRIDGE, Colorado*** *2008-01-03 18:41:29 UTC*
> *An 11-year-old British boy died Thursday...*

To reduce the ambiguity from the three sources, we applied within-document coreference resolution, acronym expansion, city name expansion respectively.

We used the following formula to compute ambiguity (a name is ambiguous if it can refer to at least 2 entities):

$$ambiguity = \frac{\#ambiguous\_names}{\#names}$$

Table 3 shows that within-document coreference resolution contributes the most in reducing the ambiguity, from 46.3% to 14.6%, while acronym expansion and city name expansion further reduce the ambiguity to 11.2%. Without using any query reformulation, the popularity based non-wiki approach only obtains the B-Cubed+ F-measure of 0.471. However, after applying within-document coreference resolution and replacing the person names with the full names, the performance significantly increases to 0.576. Acronym expansion and city name expansion further help to improve the performance to 0.604.

Table 3: Impact of query reformulation

	ambiguity	B-Cubed+ F-measure (Popularity based approach)
Original query file	376/812=46.3%	0.471
+within document coreference resolution	242/1660=14.6%	0.576
+acronym expansion	229/1698=13.5%	0.577
+city name expansion	200/1780=11.2%	0.604

3.2.3 Impact of Collaborative Clustering

Long tail effect in name entity clustering

By analyzing the data set prepared for name entity clustering, we observed two types of long tail effect:

Type I: As shown in Figure 3, most ambiguous names only refer to a few entities (i.e., clusters). For example, 39 names (36.8% of the total) only have two clusters, and 91 names (85.8% of the total) have fewer than and equal to 6 clusters. There are only a few names that have extremely large number of clusters.

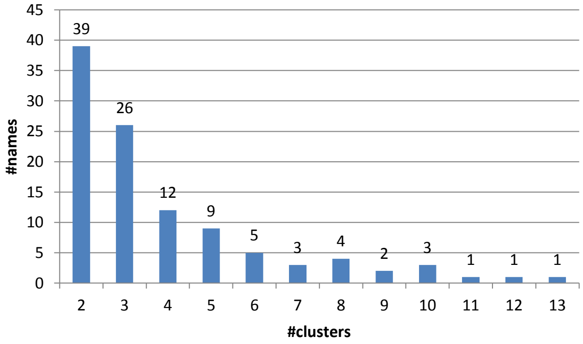


Figure 3: Type I long tail effect

Type II: As shown in Figure 4, most names have very unbalanced class distribution, in other words, most queries (instances) are clustered in a few large clusters, while the other queries are clustered into small clusters including many singleton clusters. The number on each bar is computed as follows: we rank the clusters for each name from high to low by computing the number of queries contained in each cluster, so the first cluster (as shown in the most left bar in Figure 4) always contains the largest number of queries; we then compute the average number of queries for the i_{th} cluster among all the names. The figure shows that the top cluster contains 9.4 queries in average, and the second largest cluster only contains 3.3 queries, and more clusters contains even fewer queries.

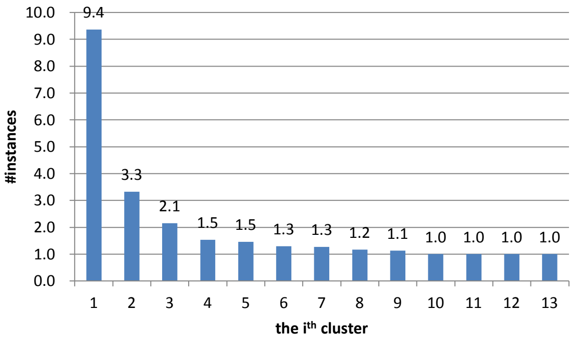


Figure 4: Type II long tail effect

Impact of Clustering Algorithms

As discussed in 2.3.3, we applied 21 clustering algorithms and 4 similarity functions to get 81 (21*4) clustering solutions. We also experimented two settings of each clustering algorithm, one is to use fixed K (assuming that we know the prior of the number of clusters), the other is to compute Silhouette Coefficient (Rousseeuw, 1987) to automatically determine the number of clusters. The results are shown in Figure 5 and 6 respectively.

We have the following observations based on the two figures:

(1) In general, the three linkage based agglomerative clustering algorithms can perform well using any of the four similarity functions. For example, in Figure 5 using cosine similarity, the highest score (0.658) is achieved by $clink$, and using maximum entropy based similarity function, the highest score (0.660) is achieved by $alink$.

(2) Normally, the scores of unknown prior K are lower than those of known prior K because determining the number of clusters is an extra effort which can introduce errors.

Impact of MiCC

similarity function	Agglomerative Clustering									Partitional Clustering											
	linkage			optimizing internal measure						repeated bisection						direct k-way					
	slink	clink	alink	I_1	I_2	$\mathcal{E}_1$	G_1	H_1	H_2	rI_1	rI_2	$r\mathcal{E}_1$	rG_1	rH_1	rH_2	dI_1	dI_2	$d\mathcal{E}_1$	dG_1	dH_1	dH_2
cos	0.587	**0.658**	0.645	0.545	0.554	0.513	**0.612**	0.529	0.535	0.544	0.572	0.521	**0.627**	0.541	0.544	0.542	0.573	0.530	**0.613**	0.546	0.547
cor	0.511	0.528	**0.538**	0.521	0.534	0.533	0.418	0.527	**0.540**	0.516	0.526	**0.545**	0.453	0.522	0.536	0.513	0.528	**0.546**	0.472	0.525	0.540
maxen	0.602	0.557	**0.660**	**0.626**	0.615	0.616	0.615	0.570	0.568	0.587	0.591	0.561	**0.609**	0.566	0.566	0.580	0.586	0.561	**0.596**	0.570	0.569
svm	0.603	0.567	**0.647**	**0.644**	0.643	0.614	0.561	0.567	0.561	0.585	**0.596**	0.575	0.586	0.576	0.575	0.575	0.584	0.578	**0.591**	0.570	0.565

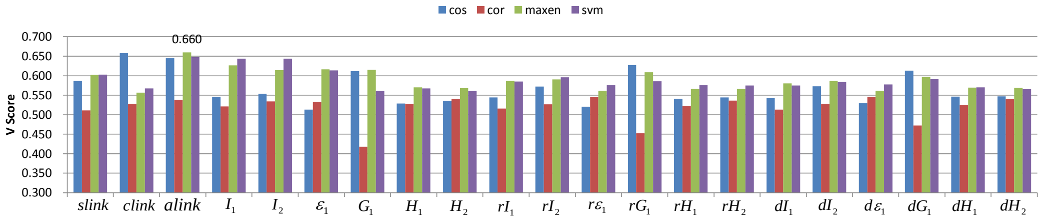


Figure 5: Performance of 21 clustering algorithms for all names (known prior K)

similarity function	Agglomerative Clustering									Partitional Clustering											
	linkage			optimizing internal measure						repeated bisection						direct k-way					
	slink	clink	alink	I_1	I_2	$\mathcal{E}_1$	G_1	H_1	H_2	rI_1	rI_2	$r\mathcal{E}_1$	rG_1	rH_1	rH_2	dI_1	dI_2	$d\mathcal{E}_1$	dG_1	dH_1	dH_2
cos	0.520	**0.632**	0.551	0.555	0.557	0.509	**0.561**	0.546	0.538	0.549	0.563	0.507	**0.615**	0.537	0.520	0.549	0.565	0.513	**0.605**	0.534	0.529
cor	0.474	**0.557**	0.515	0.551	0.558	0.556	0.417	0.563	**0.563**	0.556	0.560	**0.565**	0.480	0.554	0.557	0.552	**0.557**	0.555	0.484	0.556	0.555
maxen	0.525	0.493	**0.545**	0.532	0.537	0.537	0.515	0.537	**0.540**	0.536	0.528	0.525	0.498	0.536	0.536	0.531	0.524	0.520	0.510	0.531	0.531
svm	0.511	0.508	**0.552**	0.549	**0.553**	0.528	0.524	0.533	0.534	**0.536**	0.533	0.510	0.525	0.530	0.523	0.530	0.533	0.518	0.532	**0.534**	0.530

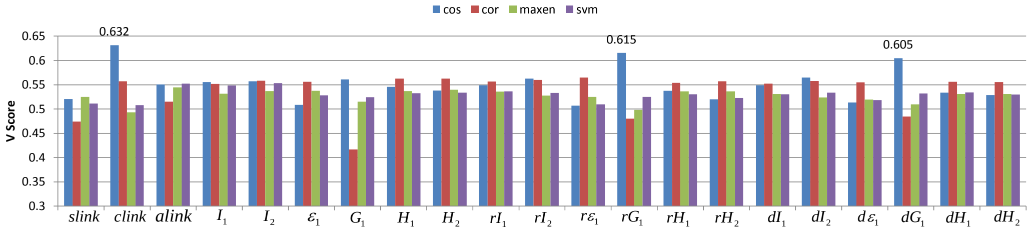


Figure 6: Performance of 21 clustering algorithms for all names (unknown prior K)

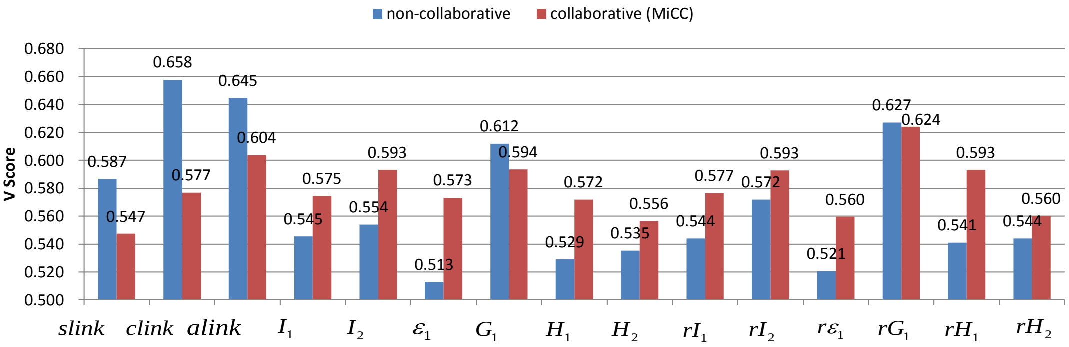


Figure 7: Impact of MiCC (known K)

For each name, we first applied Lucene[5] to search at most 100 documents that mention the name and use them as the pool of collaborative instances. It is worth noting that the selected instances are actually ranked by the default ranking model in the Lucene. Similar with document clustering, we selected Silhouette Coefficient (*SC*) as our internal measure. For each iteration, we selected at most $n_{step} = 5$ collaborative instances. To pick the best 5 collaborative documents in each iteration, we tried $n_{trials} = 10$ times by randomly selecting 5 documents from the pool. For each trial, we added the 5 documents into the expanded set of documents, and applied a clustering algorithm on the newly ex-

[5] http://lucene.apache.org/

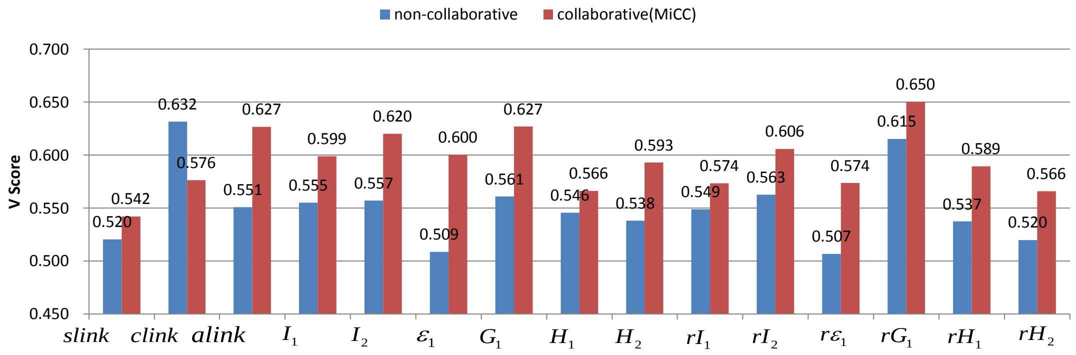


Figure 8: Impact of MiCC (unknown K)

panded set of document and then computed the clustering quality of the original set of instances using the internal measure *SC*. Among the 10 trials, we picked the best 5 collaborative documents which led to the best clustering quality. We set $n_{iterations} = 6$, in other words, we at most selected 30 collaborative documents in the end.

We tested on the 15 baseline clustering algorithms ($alink$,$clink$,$alink$,I_1, I_2, ε_1, G_1, H_1, H_2, rI_1, rI_2, $r\varepsilon_1$, rG_1, rH_1, rH_2) , and validated whether the MiCC algorithm can help each baseline algorithm to achieve better performance evaluated by V-measure. Figure 7, 8 show the results for known K and unknown K respectively.

We observed that MiCC achieves limited success for some clustering algorithms while fails for some other clustering algorithm if the number of clusters (K) is given. There are two reasons that lead to the possible failure: (1) in name entity clustering, the cluster distribution in instance collaborators are still quite close to the original set of instances. Therefore the added collaborators can only enhance some already well-formed clusters in the original data set and still lack the ability to help uncover bad clusters; (2) the added collaborators may belong to a new cluster that is not in the original data set. Thus if K is set to be fixed, those new added collaborators which belong to a new cluster will be distributed to clusters in the original data set, thus they are introduced as noises which make the results worse.

However, if K is not given, MiCC works for most of the clustering algorithms. That clearly shows the approach that automatically determines K helps to reduce the side-effect of introducing new clusters as K can be dynamically changed.

Impact of MaCC

We collected 84 clustering results, which is a combination of 21 clustering algorithms and 4 similarity functions.

We experimented four combination schemes and four consensus functions, specifically,

Scheme 1. The 84 clustering results are categorized into 4 serial groups, the first group uses cosine similarity, the second group uses correlation similarity, the third group uses maxen similarity and the final group uses svm similarity. Each group contains 21 clustering results. We use notations "cos", "cor", "maxen" and "svm" to represent the four groups.

Scheme 2. The 84 clustering results are categorized into 3 serial groups, the first group includes 24 repeated bisectional clustering results (6 repeated bisectional clustering algorithms times 4 similarity functions), the second group includes 24 direct k-way clustering results (6 direct k-way clustering algorithms times 4 similarity functions) and the third group includes 36 agglomerative clustering results(9 agglomerative clustering algorithms times 4 similarity functions), . We use notations "rbr", "direct" and "agglo" to represent the three groups respectively.

Scheme 3. The 84 clustering results are ranked from the highest to the lowest by the scores computed by the internal measure "silhouette coefficient", and then split into 4 groups, each group containing 21 ranked clustering results.

Scheme 4. The 84 clustering results are ranked from the highest to the lowest by the scores computed by the external measure "V-measure", and then split into 4 groups, each group containing 21 ranked clustering results.

The four consensus functions include co-association matrix based, IBGF, CBGF, and HBGF.

Figure 9 and Figure 10 show the performance of the four combination schemes and four consensus functions for known K and unknown K respectively. We can observe that:

(1) In Figure 9 (a), after cor related clustering results are added, the performance significantly drops which clearly show that they hurt the performance. However, after adding maxen and svm related clustering results, the performance increases again. The final best performance is obtained after adding all the 84 clustering results, however, the score 0.646 cannot beat the best one (0.660) among the 84 clustering results although it beats 95.2% of 84. This observation does not quite hold for unknown K in which case, cor related clustering results are significantly worse than the others, and the best combination performance (0.621) is obtained only by adding cosine related clustering results and it also cannot beat the best score among the 84 clustering results (unknown K) which is 0.632.

(2) In Figure 9 (b), we do not observe significant improvement after adding direct related clustering results which is consistent with the fact that direct related results are not significantly better than rbr, however, after adding agglomerative related results, the performance can increase significantly, because we already know that three linkage based agglomerative clustering algorithms perform well. This observation also holds for unknown K.

(3) Better performance can be obtained by Scheme 3 and Scheme 4 either for known K or unknown K. Most importantly, the best combined score can all outperform the best one in the 84 clustering results. For known K, in Figure 9 (c), we obtained the best score of 0.699, in Figure 9 (d), we obtained the best score of 0.750. They outperform the best individual score of 0.660 by 3.9% and 9.0% respectively. For unknown K, in Figure 10 (c), we obtained the best score of 0.646, in Figure 10 (d), we obtained the best score of 0.751. They outperform the best individual score of 0.632 by 1.4% and 11.9% respectively. All those performance gains are statistically significant.

We claim that

(1) the success of MaCC relies on clustering diversity. For example, direct related clustering results perform comparably with rbr related clustering results in Figure 9 (b), therefore, direct related results do not have enough diversity to further improve the combined score.

(2) the success of MaCC relies on selection of clusterers. Good clusterers tend to produce even better final result, while bad clusterers tend to produce negative result. This is obvious from Figure 9 (c) and (d).

(3) the success of MaCC also relies on consensus function. Co-association matrix based is a good choice according to our experiments.

3.2.4 Why NIL Clustering is So Simple in KBP2012

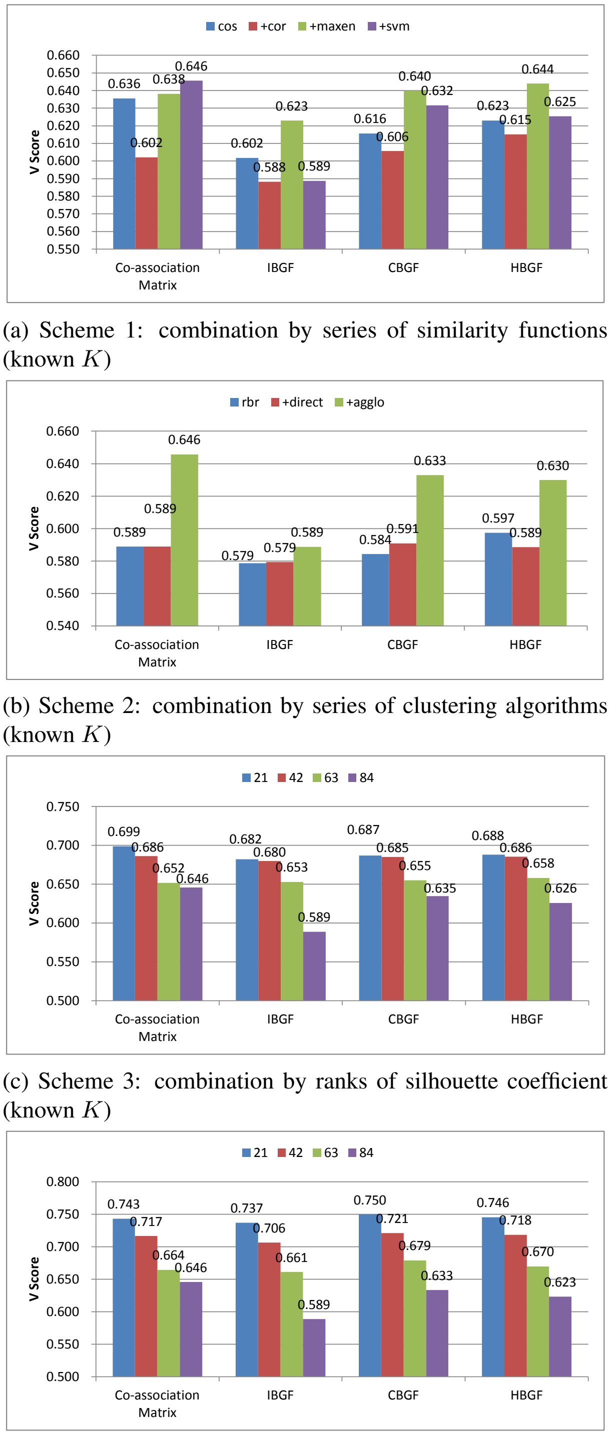
Among the NIL queries, we found that 34.9% (178/510) of names refer to more than 2 clusters, however, after applying our query reformulation techniques, only 5.8% (55/947) of names are ambiguous. That clearly shows that a simple strategy based on all-in-one (clustering all queries with the same name into one cluster) may beat any sophisticated clustering algorithms. What is more, we found that there are only 1049 NIL queries dispersed in 510 names before query reformulation, which means that for each name, there are only 2 queries in average. This also makes traditional clustering algorithms on such small scale of data ineffective. Although we have developed very sophisticated clustering approaches as presented in this paper for NIL clustering, unfortunately, they cannot be applied effectively because a simple all-in-one can do sufficiently well for the queries with the current selection criteria.

We propose the following criteria to select good queries to evaluate NIL Entity Clustering in the coming years:

(1) The query name after query reformulation should also be ambiguous, which means the full name refer to at least two entities.

(2) The number of queries (clustering instances) which share the same query name after query reformulation should be at least four.

(3) (optional to favor MiCC in collaborative clustering) at least 5 relevant documents can be retrieved from the source collection that contain the query name.



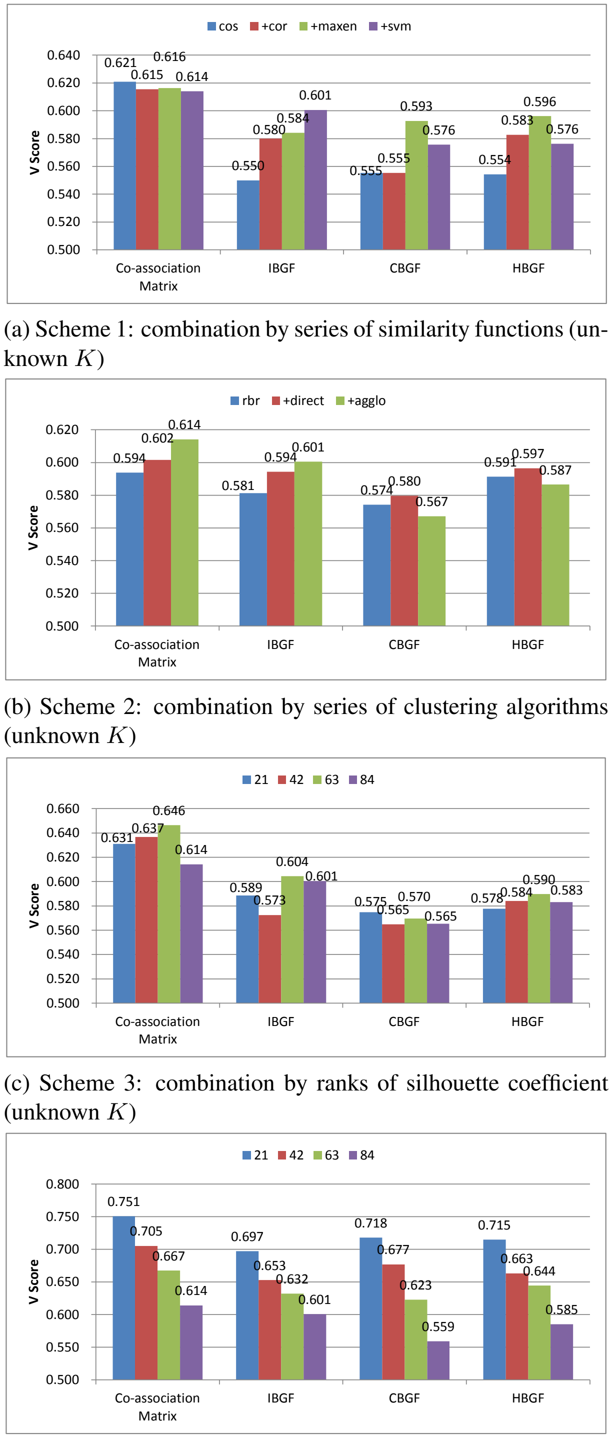
(a) Scheme 1: combination by series of similarity functions (known K)

(b) Scheme 2: combination by series of clustering algorithms (known K)

(c) Scheme 3: combination by ranks of silhouette coefficient (known K)

(d) Scheme 4: combination by ranks of V measure (known K)

Figure 9: Four combination schemes of MaCC (known K)



(a) Scheme 1: combination by series of similarity functions (unknown K)

(b) Scheme 2: combination by series of clustering algorithms (unknown K)

(c) Scheme 3: combination by ranks of silhouette coefficient (unknown K)

(d) Scheme 4: combination by ranks of V measure (unknown K)

Figure 10: Four combination schemes of MaCC (unknown K)

4 Entity Linking Related Work

Clustering is an extensively researched topic and there is a considerable amount of literature which covers every aspect of clustering problem including clustering algorithm, clustering validation, distance function computation etc. We refer readers to two most recent thorough surveys (Jain et al., 1999; Xu and II., 2005). A considerable amount of work has also published on clustering ensemble which is called as "macro collaborative clustering" in our term. As far as we know, this is the beginning work that takes both instance-level and clusterer-level collaboration into consideration.

The name entity clustering task defined here is closely related with other research topics such as person name disambiguation(Chen and Martin, 2007), web people search (Artiles et al., , 2009), cross-document entity coreference resolution. The common thing in these research topics is that they all can be formulated as a clustering problem. However, person name disambiguation and web people search focus on clustering person name entity mentions into unambiguous entities. Cross-document entity coreference resolution studies not only name entities "Mike Jordan", but also nominal entities such as "professor" and pronominal entities such as "he" and "she". The clustered results in cross-document entity coreference resolution are often mentioned as coreference chains, for example, a chain can contain the following mentions, "Mike Jordan", "professor" and "he". In this paper, we focus on name entities, and do not explore nominal entities and pronominal entities. We used name entity clustering as an application of "collaborative clustering", and validated its effectiveness.

System Validation: extensive work has been performed on reranking techniques to enhance the performance of NLP systems for a variety of tasks including but not limited to name tagging (Ji et al.,), parsing /citeCharniak2005, and machine translation (Huang et al.,). Although successful applications, these approaches generally focus on improving a stand-alone, or a limited number of systems to produce the n-best hypotheses. To this end, our previous work (Tamang and Ji, 2011) applies re-ranking to combine the aggregated output of many systems developed by different researchers, and demonstrates that overall gains can be accomplished even in the 'black-box' setting where intermediate system output is unavailable. Also, this work suggests the benefit of system combination is maximized when there are many systems available for combination, the component systems are developed using diverse resources, and systems demonstrate comparable performance.

5 Slot Filling Validation

This section describes our work on the new Slot Filling Validation task. After the evaluation runs are submitted by the participating teams, the answer set are merged, functioning as the input to a validation system. For each candidate answer in the merged answer set, a label the original answer to indicate a valid or invalid answer.

5.1 Validation System Overview

Our automatic slot filling validator was trained from the system output from previous years based on a logistic regression model for all 42 slot types, using several types of answer characteristics. We can broadly described our feature types as *shallow*, *contextual* or *emergent*. Shallow features exploit the basic knowledge about a query, slot fill and deeper, contextual features require context that is provided by a supporting document. Both types are examined on individual basis and the predictive information is independent of all other answers. Emergent features arise when the answers from multiple runs are aggregated, and draw from the notion of voting or consensus. For the purpose of answer validation, we can view this as a systems approach to tapping the"wisdom of crowds", which like human consensus, has it's strengths and weaknesses.

Our previous work (Tamang and Ji, 2011) has shown that there are benefits to system combination with as few as two systems. Based on out participation in the task and a detailed analysis of system results, we had the following observations using the merged results of twenty-six from X systems that we feel can help to improve framing the task and task evaluation. Currently, generating an answer key is a laborious task that involved multiple human assessors. In addition, we will identify the common error types across slot filling systems. In summary, our key insights are:

(1) *There are several conditions that should be met for successful combining of KBP slot-filling systems beyond that of requiring a uniform representation*. This includes the use of systems that can produce reasonably good results. When the performance of individual systems is variable, combining based on voting, or emergent features, becomes more challenging, especially when low performance systems produce a large number of answers relative to the number of answers provided by good system. In the extreme case, system or run consensus based features can amplify the contributions of poor performers and punish smarter systems.

(2) With the exception of syntactic features that embody known structural patterns a regular expressions, *features extracted from the answer justification, or contextual features, are more robust than other types of features*. That is, contextual features are not as sensitive to noisy component systems. However, it relies on the presence and accuracy of justification offsets and strategies for their use in validation should be shaped accordingly.

(3) *Shallow features alone are limited in their ability to discern answer correctness but helpful to training a classifier.* This is likely due to the fact that it is measuring consistency of answer type (i.e. title, country, lastname, number of tokens) and does not consider the context in which the answer was retrieved. Since systems with lower F1 measures typically report many more answers that top systems, contextual features have the most potential.

(4) The detailed methods of confidence estimation and the ranks of systems are unknown during evaluation. We will show that *information about the performance of component systems is important to our validator*. We demonstrate a simple method of automatically assessing quality to obtain a rough estimation of system quality that can be used to adjust the re-ranking based confidence. Also, we discuss the benefits and challenges of using system generated confidence values, and voting features when the quality of component systems is uncertain.

5.2 Feature Description

We hypothesize that implicit constraints can inform the likelihood that a slot fill is correct. Features that are extracted based on characteristics of a slot fill use gazetteers for labeling, structural features, or surface level answer features. For example, all residence slots will be filled with a country, state or city. Similarly, slots for a query's spouse or other family member will correspond with a name, and that name isn't likely to be 4 tokens in total length. The *shallow features* we extracted and are based on characteristics of the slot fill, and not the answer context from which the answer was derived fare in Table 4.

The next broad class of validation techniques shown in Table 4 we will describe as *contextual*. Based on the offsets provided in the standard answer format to indicate the justification for the slot-fill, it involves retrieving the supporting document from the test collection and analyzing the context from which a slot-fill was generated.

Lastly, the third category of slot-filling validation techniques, *emergent*, were weighted voting features, which make use of collective, group decision power. Our results in KBP2012 evaluation have shown that our validator can produce a combined system output with good performance, and promote good answers and thus can significantly reduce the cost of human assessment. We will report the impact of our validator on each individual system, the combined system, and speed up human assessment.

5.3 Learning Methods

The *first step* in answer validation is classification. To develop the slot-based classifiers for the evaluation, our automatic slot filling validator was trained with previous KBP output and uses a logistic regression model to obtain an answer likelihood on a scale from 0-1 for all 42 slot types. We extracted the appropriate features for each slot that are broadly described as *shallow*, *contextual* or *emergent* and collectively appear in Table 4.

For each slot's training data, and based on the results of 10-fold cross-validation, a stepwise procedure was used to test the contribution of each feature in the classification model to develop the slot-based classifiers for the KBP validation task. The model parameters that generated the lowest AIC were selected for the evaluation runs with one exception. In some cases where AIC-based selection resulted in a single voting feature. For these cases, if a model with more features had approximately the same AIC, it was used for the evaluation.

System consensus and answer consensus were significant predictors of answer validity for all but two slots. However, these two slots had a relatively few training examples. Surprisingly, document type was not a significant predictor in any of the classification models.

The *second step* of validation consists of confidence adjustment and rule based filtering. During training, we observed that on some slots, such as an organization's top member's and employee's, emergent features are sensitive to relatively bad systems with many answers. The impact of poor performers can be reduced by using a feature that captures system or slot level performance as shown in our previous work. Since the performance of component systems would be unknown at testing time, we used a heuristic for automatically assessing performance of a system. Specifically, we assessed the number of type inconsistent fills (e.g., returning a state name for a country slot) to get a rough estimate of high and low performers and used the training data to estimate an appropriate level of answer confidence adjustment.

Table 4: Validation Features

Feature	Description	Value	Type
document type	provided by document collection as news wire, broadcast news, web log	category	shallow
number of tokens	count of white spaces (+1) between contiguous character string	integer	shallow
acronym	identify and concatenate first letter of each token	binary	shallow
url	structural rules to determine if a valid url	binary	shallow
named entity type	label with gazetteer	category	shallow
city, state, country, title, ethnicity, religion	appears in specific slot-related gazetteer	binary	shallow
alphanumeric	indicate if numbers and letters appear	binary	shallow
date	structural rules to determine if an acceptable date format	binary	shallow
capitalized	first character of token(s) caps	binary	shallow
same	if query and fill strings match	binary	shallow
keywords	used primarily for spouse and residence slots	binary	context
dependency parse	length from query to answer	integer	context
system votes	proportion of systems with answer agreement	0-1	emergent
answer votes	proportion of answers with answer agreement	0-1	emergent

Based on the rough assessment of high and low performing systems, we adjusted confidence scores in the evaluation. Be did not weight or eliminate answers, rather if an answer was only returned by the set of lowest performers the validation confidence was reduced. Alternatively, is a answer did not have many votes, but a high performer was one of the voters, the confidence of the answer was increased.

For the contextual features, the training data provided the document ID not the provenance information that is new to KBP 2012. Using the document ID we retrieved the first sentence that the q,q pair co-occurrs. This heuristic introduced noise, but provided a useful approximation. The contextual features were not selected for the classification model based on AIC. However, we found the contextual characteristic derived from what we retrieved helped remove invalid answers when used in a filtering step. This involved adjusting the confidence predicted by the validator when a slot relevant keyword appeared, and setting a threshold to remove very long dependency parses. The best parameters for rule-based filtering were then used for evaluation.

Also, for two slots that were not well represented in the training data we could not train a classifier and only rule-based filtering could be applied. n the case of org:website, strong shallow feature predictors based on regular expressions can be used to eliminate invalid answers. On all other slots, syntactic feature were not as useful.

6 Slot Filling Validation Experiments

6.1 Experiment Results

Table 5 shows the recall, precision, and F1 measure for the LDC annotation, the aggregated output from the KBP 2012 slot-filling task that served as input to the validator, the results form phase one of the validation method, and phase two.

Table 5: Performance

System	Recall	Precision	F1
LDC	0.73	0.77	0.75
Validation input	0.70	0.03	0.06
Validator P1	0.12	0.07	0.09
Validator P2	0.35	0.08	0.13

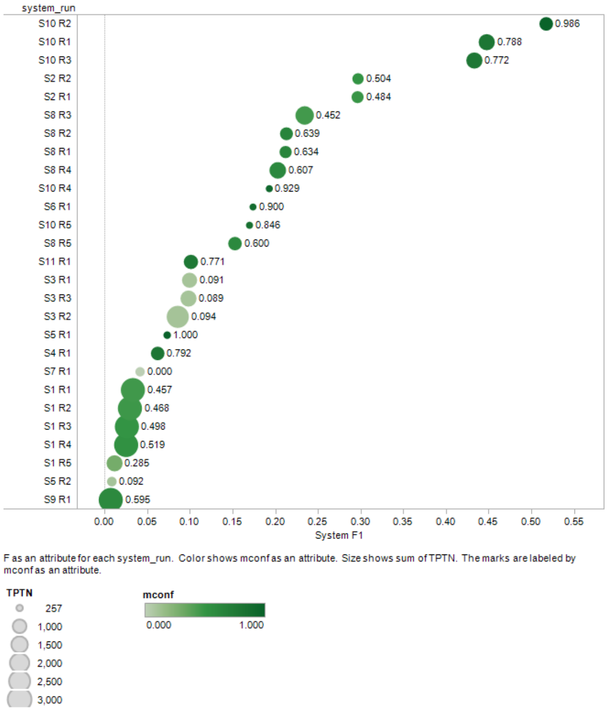
6.2 Analysis

6.2.1 Slot-filling Validation Features

Figure 11 shows the KBP 2012 F1, mean system confidence and total number of answers returned by each system. The size of each point is proportional to the size of the data set, and the color, the F1 measure. It suggests that better systems use higher mean confidence values. Unlike emergent features that are limited when favorable conditions for system combination cannot be met, confidence values appear to be more useful for validating and combining multiple systems even in the presence of poor performers. However, this is not a robust feature, and will not work in a setting with many 'ignorant systems' that are very confident about their invalid answers.

In the evaluation task, which reported 27 total runs for 10 different system submissions, the 10 worst performers accounted for more than half of the total answers. Not only has the range of system performance increased in 2012, but the relative number of invalid answers in the merges system output. This suggests a setting that can be unfavorable for voting based features. Also, the importance of detailed system characteristics such as methods used and quality.

Although the overall F1 measure was better than the median system performance rank, our post-evaluation analysis indicates that the validator was unable to improve on the results of top systems. Since emergent features were consistently the most significant predictors of valid answers on the training set, and the number of answers returned by systems with low F1 measures were two and even three times the size of top systems, it is likely that the validator was limited by its dependance on voting features, and surface features are too weak. Automatically assessing system quality by the proportion of consistent answers, and extracting contextual characteristics provided by answer provenance information proved beneficial as indicated in Table 5, corresponding with the gains from validation phases P1, and P2.



F as an attribute for each system_run. Color shows mconf as an attribute. Size shows sum of TPTN. The marks are labeled by mconf as an attribute.

Figure 11: System F1 measures by mean confidence and total answers returned

6.2.2 Answer Key Generation

To assess the impact of validation methods to expedite the generation of answer keys, we further explored the potential role of system confidence values, our current validator, and automatic methods to assess system quality when performance is unknown. Figure 12 shows several answer filtering strategies and the number of true-positives versus total candidate answers returned including: adding systems by known F1 measure (F1 rank), 'blind' assessment of system quality using answer consistency (consist), system reported answer confidence (sys_conf), our KBP evaluation validator (e_validator), random inclusion (random) and a combination of the e_validator at the threshold submitted for KBP2012 (0.5) with the incremental addition of system answers with a confidence at least .90 in the sequence of system consistency rank (valid_conf). For the methods *consist*, *F1 rank*, *random* and *valid_conf*, only the first ten out of the total 27 systems in each category were used. For *e_validator* and *sys_conf* and we show results starting with all answers at the 0.90 level cumulatively added at 0.10 increments.

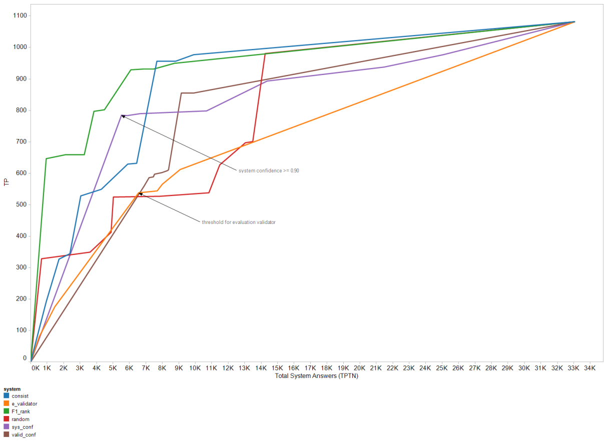


Figure 12: Alternative answer reranking strategies

Figure 12 suggests that knowing the system performance beforehand provides the most immediate rewards in terms of efficiency gains and saved human labor hours (F1 rank). However, when the quality of each system is unknown, it can be roughly assessed by simple inconsistency checks using gazetteers and heuristics. In practice, we examined the percent of answers that appeared in a country, state/province or title gazetteer to approximate system ranking, and show the cumulative results based on adding the top ten systems based on this approach (consist).

Also, this chart demonstrates that useful information can be obtained by system reported confidence levels. Adding the most confident answers first by 0.10 increments (sys_conf) indicates that almost 75 percent of the correct answers have been assigned a system confidence of 0.90 or greater. Also, starting with our validator at the evaluation threshold, adding answers with a confidence of 0.90 or greater in the order of system consistency also shows a notable improvement on the base validator.

7 Slot Filling Validation Related Work

Extensive work has been performed on reranking techniques to enhance the performance of NLP systems for a variety of tasks including but not limited to name tagging (Ji et al. 2006) and machine translation (Huang and Papineni, 2007). Although successful applications, these approaches generally focus on improving a stand-alone, or a limited number of systems to produce the n-best hypotheses. To this end, our previous work (Tamang and Ji, 2011) applies re-ranking to combine the aggregated output of many systems developed by different researchers, and demonstrates that overall gains can be accomplished even in the 'black-box' setting where intermediate system output is unavailable. Also, this work suggests the benefit of system combination is maximized when there are many systems available for combination, the component systems are developed using diverse resources, and systems demonstrate comparable performance.

8 Slot Filing Validation Discussions

This work describes a validator developed with previous KBP slot-filling output. Using surface features, consensus, and some deeper contextual analysis, tens of thousands of invalid answers can be filtered, notably improving (0.06 to 0.13 F1) on the combined output from all evaluation systems. Also, the number of invalid answers is not evenly distributed among systems, suggesting that for many low scoring teams, post-processing based on this framework will provide immediate benefits.

It is important to note that we view the goal of validation as part of an iterative loop involving that pairs KBP systems with human assessors (Tamang and Ji, 2011) to expedite the generation of answer keys. It is related too, but distinct from system combination, which seeks improve the overall F1 score. The motivation for validation is to cull less valid answers so that assessment hours be used more effectively. Currently, the task of composing answer keys to assess slot-filling performance is a bottleneck in that developers cannot fully understand the strengths and limitations of automated systems without comprehensive assessment keys.

Based on our experience validating the 2012 English KBP data, we were able to glean some insights into the main challenges of the task that should be considered for future validation work. The first is due to sampling variability between training and testing. That is, the collective systems represented are notably different in performance and quality. Our most predictive features for the training data are answer consensus across systems and runs for almost all of the slots. However, our evaluation systems analysis suggests that the disproportionate amount of answers and runs generated by poorer relative to better systems was an issue for reranking performance. Although we feel that consensus based features are useful, we also recommend the use of more robust features that are less sensitive to noise.

We found that surface features based on annotation of the answer with location, title, named entity and other gazetteers, or syntax checking for data formats or numerical values are useful but limited in their ability to discern answer correctness, and were not included in the classifier's feature set for many slots. Their advantage is that these features are easy to generate, and can be used to identify inconsistent answers types. Also, we found them helpful for automatically assessing very poor systems, which tend to produce many inconsistent answers relative to smarter systems.

Emergent, or voting based features to indicate agreement of an answer among runs and systems. Despite an inability to use the context provided by answer provenance as effectively as we'd like, simple checks for keywords within the answer context are useful based on training results, and filtering to eliminate long dependency parses is useful when a justification can be extracted.

Two new fields were added to KBP answer format for 2013 that can be used for validation: system confidence (0-1), and answer provenance. The first, system confidence is useful for validation the KBP 2013 data and a simple rule (i.e. returning only answers ¿.90). In our post-evaluation system analysis, we show that confidence values are informative as system features for KBP 2012 validation. However, their merit could easily be thrown off by a 'ignorant system' that is very confident but largely incorrect. To avoid worst-case scenarios, some knowledge about the performance of the system that generated the answer and there confidence scheme would be useful. One approach could be to combine our method for automatically ranking systems using a simple heuristic, with rescaling of confidence by system.

The second new answer attribute in the KBP 2012 system output is answer provenance, which provides the supporting text for a candidate answer. We feel this is key information for a good validator that in addition to shallow and consensus based features, makes use of deeper semantic features that can be good predictors and less sensitive to the limitations of other systems. However, justification offsets were not available in training data, and erroneous or missing in some evaluation answer sets.

For training out validator, we had only the document ID not the offset justifications for using provenance. For many training examples, we were able to identify an appropriate sentence based on the cooccurance of a q,a pair. Although no guarantee this was the original systems justification, we found this a reasonable approximation. These contextual features were not represented in the models selected using AIC based criteria. However, recognizing their value, they were used in a filtering step that involved adjusting the confidence predicted by the validator when a slot relevant keyword appeared, and to filter some bad answers associated with a very long dependency parse.

For our evaluation test set, we naively assumed offsets values would follow the guidelines. We were not prepared with a strategy for partially correct offsets, or missing offsets, which made it difficult to assign dependency parse, or keyword based features. Also, co-reference resolution was not used in either testing or training and it is possible some justifications had the q,a pair present, but was not detected.

The amount of data for developing classifiers that make use of deeper, semantic features for the KBP can be an issue, especially for slots that are less represented. We found rule-based filtering to adjust the confidence of answers is helpful in this case. Also, we feel that validation strategies that can approximate missing offsets, or partially correct provenance, and co-reference can more easily extract contextual features for classification. One trick that can be used for co-reference detection that does not entail the annotation of the entire corpus is to use the alternate names slot value for the entity already provided in the answer set.

In summary, this work describes a variety of features that were engineered for validation and our collective insights on their strengths and limitations. There are many ways to approach the validation task, and this framework is designed to benefit the generation of more comprehensive answer keys in a shorter amount of time. Our approach seeks to leverage syntactic and semantic information in a variety of features to provide predictive information about the validity of each slot-fill in the merged output of 27 KBP 2012 Slot Filling systems. Despite the challenges, we are excited by the new resources for validating KBP answers, and optimistic about their use for improving validation and ultimately, expediting answer-key generation.

Acknowledgments

This work was supported by the U.S. Army Research Laboratory under Cooperative Agreement No. W911NF-09-2-0053, the U.S. NSF Grants IIS-0953149 and IIS-1144111 and the U.S. DARPA BOLT 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 U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

S. Tamang and H. Ji. 2011. Adding smarter systems instead of human annotators: re-ranking for system combination. *Proc. CIKM2011 workshop on Search and mining entity-relationship data*.

H. Ji, C. Rudin and R. Grishman. 2006. Re-ranking algorithms for name tagging. *Proc. NAACL2006 workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*.

F. Huang and K. Papineni. 2007. Hierarchical System Combination for Machine Translation. *Proc. the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007).*

Y. Chen and J. Martin. 2007. Towards Robust Unsupervised Personal Name Disambiguation. *Proc. EMNLP2007.*

J. Artiles, J. Gonzalo and S. Sekine 2009. Weps 2 Evaluation Campaign: Overview of the Web People Search Clustering Task. *Proc. WePS 2009.*

Z. Chen 2012. Collaborative ranking and collaborative clustering. *Ph.d. thesis*, Graduate Center, City University of New York.

P. Rousseeuw 2012. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math., 20, 1987, pp. 53-65*

E. Amigo and J. Gonzalo and J. Artiles and F. Verdejo 2008. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*

H. Ji, R. Grishman and H. T. Dang. 2011. An Overview of the TAC2011 Knowledge Base Population Track. *Proc. Text Analysis Conference (TAC2011).*

X. Z. Fern and C. E. Brodley. 2004. Solving cluster ensemble problems by bipartite graph partitioning. *Proceedings of the Twenty First International Conference on Machine Learning*.

B. Fischer and J. M. Buhmann. 2003. Bagging for path-based clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 25(11):1411-1415.*

A. Fred 2001 Finding consistent clusters in data partitions. *In Multiple Classifier Systems, volume LNCS 2096, pages 309-318. Springer*.

A. Fred and A. K. Jain. 2002. Data clustering using evidence accumulation. *In Proc. of Sixteenth International Conference on Pattern Recognition, pages IV:276-280.*

A. K. Jain, M. N. Murty and P. Flynn. 1999. Data clustering: a review. *ACM Computing Surveys,31(3):264-323.*

G. Karypis 2007. Cluto - software for clustering high-dimensional datasets. *version 2.1.1.*

Y. Liu, Z. Li, H. Xiong, X. Gao, and J. Wu. 2010. Understanding of internal clustering validation measures. *Proceedings of the 2010 IEEE International Conference on Data Mining, p.911-916.*

H. Luo, F. Jing, and X. Xie. 2006. Combining multiple clusterings using information theory based genetic algorithm. *IEEE International Conference on Computational Intelligence and Security, vol. 1, pp. 84-89.*

J. B. MacQueen. 1967. Some methods for classification and analysis of multivariate observations. *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, University of California Press. pp. 281-297.*

C. D. Manning, P. Raghavan, and H. Schütze 2008. Introduction to information retrieval. *Cambridge University Press.*

A. Rosenberg and J. Hirschberg, J. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. *In Joint Conference on Empirical Methods in Natural Language Processing and Computational Language Learning*.

G. Salton, A. Wong, and C. S. Yang 1975. A vector space model for automatic indexing. *Communications of the ACM, vol. 18, nr. 11, pp. 613-620*.

M. Steinbach, G. Karypis, and V. Kumar 2000. A comparison of document clustering techniques. *KDD*.

A. Strehl and J. Ghosh. 2002. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research, 3: 583-617*.

P. N. Tan, M. Steinbach, and V. Kumar 2005. Introduction to data mining. *Addison Wesley*.

A. Topchy, A. Jain, and W. Punch. 2004. A mixture model for clustering ensembles. *In Proc. SIAM Data Mining, pages 379-390*.

A. Topchy, A. K. Jain,and W. Punch. 2003. Combining multiple weak clusterings. *Proceeding of the Third IEEE International Conference on Data Mining*.

J. Wu, H. Xiong, and J. Chen 2009. Adapting the right measures for k-means clustering. *SIGKDD09,pages 877-886*.

R. Xu and D. W. II. 2005. Survey of clustering algorithms. *IEEE Trans. Neural Networks,16, pp. 645-678*.

Y. Zhao and G. Karypis. 2002. Comparison of agglomerative and partitional document clustering algorithms. *Technical report , University of Minnesota*.

Y. Zhao and G. Karypis. 2004. Empirical and theoretical comparisons of selected criterion functions for document clustering. *Machine Learning, 55(3):311-331*.

Y. Zhao and G. Karypis. 2005. Hierarchical clustering algorithms for document datasets. *Data Mining and Knowledge Discovery, Vol. 10, No. 2, pp. 141 - 168*.

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

T. Cassidy, Z. Chen, J. Artiles, H. Ji, H. Deng, L.-A. Ratinov, J. Zheng, J. Han and D Roth. 2011. CUNY-UIUC-SRI TAC-KBP2011 Entity Linking System Description. *Proc. Text Analytics Conference (TAC2011)*.

Z. Chen and H. Ji. 2011. Collaborative Ranking: A Case Study on Entity Linking. *Proc. EMNLP2011*.

S. Tamang and H. Ji. 2011. Adding smarter systems instead of human annotators: re-ranking for system combination. *Proc. of the Workshop on Search and Mining Entity-relationship data, CIKM 2011*.

H. Ji., C. Rudin and R. Grishman. 2006. Re-ranking algorithms for name tagging. *Proc. of the Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing, CHSLP '06*.

E. Charniak, M. Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*.

F. Huang, K. Papineni. 2007 Hierarchical System Combination for Machine Translation. *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*.