Person Name Disambiguation by Bootstrapping

Minoru Yoshida
University of Tokyo
7-3-1, Hongo, Bunkyo-ku
Tokyo, 113-0033
mino@r.dl.itc.u-tokyo.ac.jp

Masaki Ikeda
University of Tokyo
7-3-1, Hongo, Bunkyo-ku
Tokyo, 113-0033
ikeda@r.dl.itc.u-tokyo.ac.jp

Shingo Ono
University of Tokyo
7-3-1, Hongo, Bunkyo-ku
Tokyo, 113-0033
ono@r.dl.itc.u-tokyo.ac.jp

Issei Sato
University of Tokyo
7-3-1, Hongo, Bunkyo-ku
Tokyo, 113-0033
sato@r.dl.itc.u-tokyo.ac.jp

Hiroshi Nakagawa
University of Tokyo
7-3-1, Hongo, Bunkyo-ku
Tokyo, 113-0033
nakagawa@dl.itc.u-tokyo.ac.jp

ABSTRACT

In this paper, we report our system that disambiguates person names in Web search results. The system uses named entities, compound key words, and URLs as features for document similarity calculation, which typically show high precision but low recall clustering results. We propose to use a *two-stage clustering algorithm* by *bootstrapping* to improve the low recall values, in which clustering results of the first stage are used to extract features used in the second stage clustering. Experimental results revealed that our algorithm yields better score than the best systems at the latest WePS workshop.

Categories and Subject Descriptors

H.3.3 [**Information Storage Retrieval**]: Information Search and Retrieval—*Clustering*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Text analysis*

General Terms

Languages

Keywords

person name disambiguation, Web people search, clustering

1. INTRODUCTION

World Wide Web (WWW) search engines are commonly used for learning about real-world entities, such as people. In such cases, users key the name of the target entity in search engines to obtain a set of Web pages that contain that name. However, ambiguity in names (*i.e.,* many entities having the same name) typically causes the search results to contain Web pages about several different entities.

For example, if we want to know about a "George Bush" other than the former U.S. president, many pages about the former president are returned in the search results, which may be problematic. Depending on the circumstances, we may have to search once more to find Web pages about the target person buried in the numerous unrelated ones.

Hereon, we will use the term "person name" to mean a string indicating the name of a person. Many studies have recently been carried out on disambiguating people's names, as was reported at the recent WePS (Web People Search) workshops [2, 3]. In this disambiguation task, the typical approach is to define similarities between documents based on features extracted from the documents, and cluster Web pages returned by search engines for person name queries by using the similarity. In terms of performance, named entities (NEs) have been reported as one of the most effective features for this task [9]. NEs are good features for distinguishing people because they concisely represent real-world concepts related to the people. For example, the person's name *Paul Allen* or the organization's name *Microsoft* indicate real-world entities that are related to the person *Bill Gates*. In addition, we also focus on Compound key words (CKWs) and URLs as additional useful features. These features show similar properties to NEs. For example, the compound noun *chief software architect* indicates a concept strongly related to *Bill Gates*. (These examples can be found in the Wikipedia article on *Bill Gates*.) Links to relevant pages such as the Microsoft homepage are also good indicators for distinction.

The problem we observed with such features is that they show high precision values but low recall values (*i.e.,* they are not observed frequently but work as strong evidence for entity identification). One typical approach to improve the recall value is to reduce the threshold value for document similarities, which involves the merging of non-similar documents and typically worsens the precision.

Another typical approach is to use *weak features* for calculation of document similarities. Features to represent documents for clustering are mainly categorized into two types:

SIGIR'10, July 19–23, 2010, Geneva, Switzerland.

strong features and *weak features*. Strong features have the ability to clearly distinguish between clusters and weak features do not. We categorized named entities (NEs), compound key words (CKWs), and URLs, as mentioned above, as strong features, and single words as weak features. Although we can improve the recall value by using weak features, it typically worsens the precision in the same way as reducing the threshold value discussed above.

We solve this problem by distinguishing *reliable* weak features from others by using *two-stage clustering algorithm*. The two-stage clustering algorithm clusters documents by using only strong features in the first stage, and revises them by using weak features in the second stage. The algorithm gives weights to weak features by using *bootstrapping* techniques, which is popular in the natural language processing community. For example, if a computer scientist and a baseball player share the same name, words like *memory*, *algorithm* are reliable weak features for the former, and words like *ball* and *batting* are reliable weak features for the latter. We report that we can use word features effectively by our feature weighting algorithm. We call the clusters produced in the first stage *first-stage clusters* and the clusters produced in the second stage *second-stage clusters*.

Bootstrapping is a category of algorithms for *instance extraction*, which start with some *seed instances* and iterate some process to repeatedly improve the *instances* (i.e., extracted collections such as *a set of movie names*) and *patterns* (i.e., linguistic rules that co-occur with movie names, in this case). Typical bootstrapping algorithm selects instances and patterns according to some *reliability scores*. We apply *Espresso* [20], one of such bootstrapping algorithms, to the person name disambiguation problem. In our case, patterns correspond to weak features and instances correspond to documents newly added to clusters, while seed instances correspond to documents in each first-stage cluster. Reliability scores give high weights to useful weak features and low weights to useless weak features. The experimental evaluation showed that using this algorithm dramatically improved the performance.

We compared the bootstrapping approach with two baseline methods for the second-stage clustering: compound key words and latent topics. The former uses strong features only, and the latter uses weak features. We observed that the bootstrapping algorithm showed the best performance, which suggests that bootstrapping approaches can get the most out of the ability of weak features.

Two-stage clustering has the same aim as pseudo-relevance-feedback for document retrieval in that both extract new features from a set of documents in the first stage. The main difference is that our purpose is not to produce words used as queries, but to refine clustering results. We therefore can choose other types of features to be extracted from documents than the features typically used for pseudo-relevance-feedback. Another difference is that we applied feature extraction to *all* the resulting clusters while pseudo-relevance-feedback does not focus on the ambiguity of a query itself. Because our purpose is to distinguish documents related to the same (person name) query, the differences between documents to be separated are often small. We therefore need more careful treatment of features to make distinctions among different clusters.

The word sense disambiguation (WSD) problem, extensively studied by the natural-language-processing community, is a task usually compared to the person name disambiguation problem [3]. Although some of the results can be used to solve our name disambiguation problem, there are also several important differences between these two disambiguation tasks. For example, the number of true entities is not known in advance in name-disambiguation problems while in WSD the task is to categorize each word into one of several predefined senses. Another difference is the knowledge source. Dictionaries or thesauruses that describe each sense are available in most word disambiguation cases, and are used in the algorithm as sources to discriminate the sense.

The remainder of this paper is organized as follows. Sections 2 and 3 explain our task and describe related work, respectively. Sections 4 and 5 explain our framework. Section 6 evaluates our framework with an actual Web document dataset. Section 7 summarizes our work.

2. TASK DEFINITION

Our task, the disambiguation of person names appearing on Web pages, is formalized as follows. The query (target person name) is referred to as q. The set of Web pages obtained by inputting query q to a search engine is denoted by $\mathcal{P} = \{d_1, d_2, \cdots, d_k\}$. Each Web-page d_i has at least one string q. We assume that q on the same page refers to the same entity. Therefore, person name disambiguation is achieved by document clustering where each cluster refers to a single entity. The input of the algorithm is query q. The output of the algorithm is a set of page clusters.

In this paper, we use the term *features* to indicate strings extracted from documents. The features include NEs, CKWs, URLs, and words.

3. RELATED WORK

Several important studies have tried to solve the task described in the previous section. Bagga and Baldwin [4] applied the vector space model to calculating similarities between names only using co-occurring words. Based on this, Niu et al. [17] presented an algorithm that uses information-extraction results in addition to co-occurring words. However, these methods had only been tested on small artificial test data, raising doubts as to their suitability in practical use. Mann and Yarowsky [15] employed a clustering algorithm to generate person clusters based on extracted biographic data. However, this method was also only tested on artificial test data. Wan et al. [24] proposed a system that rebuilt search results for person names. Their system, called WebHawk, was aimed at practical use like our systems, but their task was somewhat different. Their system was designed for actual queries that occurred frequently. The algorithm in their system was specialized for English person-name queries that consisted of three words: family name, first name, and middle name. They mainly assumed queries such as "<*first name*>" or "<*first name*> <*family name*>", and took middle names into consideration, which may have improved accuracy. So the problem setting of them is different from ours.

In another approach to this task, Bekkerman and McCallum [6] proposed two methods of finding Web pages that refer to a particular person. Their work consisted of two distinct mechanisms. The first was based on a link structure and the second used agglomerative/conglomerative double

clustering. However, they focused on disambiguating an existing social network of people, which is not the case when searching for people in real situations. In addition, as our experience is that the number of direct links between pages that contain the same name are fewer than expected, information on link structures would be difficult to use to resolve our task. Although there may be indirect links (*i.e.*, one page can be found from another page via other pages), it is far too time consuming to find these.

The method proposed by Bollegala et al.[7] used extracted keywords to calculate similarities between documents. They further extracted keywords from resulting clusters. However, their research was aimed at keyword extraction itself.

Bunescu et al.[8] reported using Wikipedia knowledge to disambiguate named entities. They used Wikipedia to extract features for supervised learning. Using knowledge sources like Wikipedia is an interesting direction for named entity disambiguation in general, but in our case it is difficult to use them because our targets contain many minor person names that are not defined in Wikipedia.

3.1 Web People Search Task (WePS)

A large workshop for disambiguating person names, called *WePS*, was held in 2007[2]. The workshop provided a common dataset for research on person-name disambiguation. Sixteen systems were introduced for participation at the workshop. Most of their methods can be categorized into one of the approaches described in the previous section. Their methods typically involved some preprocessing such as POS tagging and NE extraction, calculating similarities between documents, and creating clusters according to the calculated similarities. In 2009, the second WePS workshop[3] was also held.

Named entities were one of the most effective features for the task[2][9]. Preprocessing such as filtering out feature values from some "valueless" pages was also important. The best system used both "cleaning" of documents and selection of useful features such as named entities.

Several studies that have used the WePS corpus have been reported[12][5].

[12] extensively used named entities. They extracted person names and organization names from the documents, and calculated the similarities between documents using web counts of their co-occurrence. Although their method performed extremely well, counting from the Web required massive amounts of time (because they needed to query the Web search engines about 40,000 times for each cluster), making it difficult to use in real-time systems (they stated their system was better suited for servers.)

[5] proposed a simple algorithm that used Single-Pass Clustering with a bag-of-words model, which attained performance that was comparable to the best system at WePS. Their idea of making use of positions (ranks) in search results and HTML trees to extract relevant blocks for the query was complementary to ours, which can improve the performance of our algorithm. We plan to incorporate their method in our framework in the future.

3.2 Two-Stage Clustering

Previous research has tackled clustering problems with two-stage clustering approaches. Tishby et al. [23] proposed the information-bottleneck method, which finds the optimal clusters according to an information-theoretic measure for cluster goodness. Slonim et al. [22] applied this method to document clustering by a double-clustering approach, which extracts word clusters that preserve information about the document clusters, and used the extracted word clusters in turn to cluster documents. These previous methods were for document clustering in general, which are difficult to be directly applied to person name disambiguation problem because, as discussed in the introduction, general term (word) frequencies or TF-IDF scores are not so effective for this problem compared to general document clustering problems. The two-stage clustering approach was not used in any WePS-1 system. Two systems at WePS-2 used two-stage clustering approaches[18][10], but both of them used no weak (term frequency) features.

Liu et al. [14] proposed extracting effective features from the first-stage clustering results and using the features for the second-stage clustering by feature voting. They observed that named entities and term pairs were salient features for identifying documents in the same clusters, and proposed to use them as well as ordinary term frequency features. One contribution by ours is applying a framework of bootstrapping algorithms to model such feature weighting methods via two-stage clustering in an elegant way. Moreover, they used all the features simultaneously. However, our preliminary investigation and previous reports suggest that, as mentioned above, simply blending term frequency features with NE features generally little contribution to person name disambiguation. This means that the strong features and term frequency features should be used separately. Our two-stage algorithm models this hierarchy of document features. Weak features are used only in the second-stage clustering while the first-stage clustering is performed by using strong features only.

4. FIRST-STAGE CLUSTERING

In this and the next section, we describe our *two-stage* clustering algorithm. As mentioned in the introduction, we categorize the features for representing documents into *strong features*, including NEs, CKWs, and URLs, and *weak features*, including single words, where the former have strong discriminative power and the latter do not. Our method uses only strong features to make the first-stage clusters and uses weak features to supplement them, where this use is guided by the first-stage clusters.

The algorithm proceeds as follows: (1) Make clusters on the basis of the similarities calculated by strong features, and (2) Find documents highly related to each cluster through weak features and add them to the cluster.

This section describes the first-stage clustering in detail, and the next section provides the second-stage clustering.

4.1 Preprocessing

We used lxml [1] and a sentence segmenter [2] to convert HTML files to text files that consist of sentences. We next extracted local text around each query string by using the window sizes set as parameters[3]. Tree Tagger [4] was used

[1]http://codespeak.net/lxml/

[2]http://www.answerbus.com/sentence/

[3]We tested four different window-size parameters (50, 100, 200, and all words) in preliminary experiments and chose the best setting for each feature (100 for CKW, all for NE).

[4]http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/

to add part-of-speech tags to each word. We used Stanford NER[5] for identifying person, place, and organization names. In addition, URL strings were extracted from the original HTML files.

4.2 Document Features

In the first stage, three types of features (strings) are extracted from documents: named entities, compound key words, and URLs. In this subsection, we explain these features in detail.

4.2.1 Named Entity Features

We used person names, organization names, and place names as the named entities used as features. They typically represent real-world entities related to the person.

While person names were used as is, some location names and organization names were filtered out by using stop-word lists that list the location/organization names that have high frequencies.

4.2.2 Compound Key Word Features

We also use compound key words as features. We describe how to extract them here.

First, we calculate the importance score for compound words in a document with the method proposed by Nakagawa et al. [16].

The importance score for the compound words is calculated as follows: Let $CW(= W_1W_2\cdots W_L)$ be a compound word, where W_i $(i = 1, 2, \cdots, L)$ is a simple noun. $f(CW)$ is the number of independent occurrences of compound word CW in a document where "independent" occurrence of CW means that CW is not a part of any longer compound nouns. The importance score of compound word CW is

$$Score(CW) = f(CW) \cdot LR(CW), \tag{1}$$

$LR(CW)$ is defined as follows:

$$LR(CW) = \left(\prod_{i=1}^{L}(LN(W_i)+1)(RN(W_i)+1)\right)^{\frac{1}{2L}} \tag{2}$$

$LN(W_i)$ and $RN(W_i)$ are the frequencies of nouns that directly precede or succeed simple noun W_i. We extracted the compound words that have a score higher than the threshold value θ_{CKW} as CKW features.

This score is defined based on the intuition that some words are used as term units more frequently than others, and a phrase that contains such "good" term units is likely to be important. Figure 1 outlines example statistics for the appearance of compound words in a corpus, which include "disaster information", and "information security" three times each, "information system" once, and "information ethics" two times. In this case, for example, $LN(Information) = 3$ and $RN(Information) = 3{+}1{+}2 = 6$.

4.2.3 Link Features

We extract URLs (in `<a>` tags) from each document and use them as *link features*. Link features also include the URL of the document itself. URLs with high frequencies were discarded in the same way as in the location/organization name filtering described in section 4.2.1.

[5]http://nlp.stanford.edu/software/CRF-NER.shtml

4.3 Document Similarities

In this subsection, we describe how to calculate document similarities using the extracted features. We used the *overlap coefficient*[21] defined below to calculate similarities between documents.

$$Overlap(d_x, d_y) = \frac{|\mathbf{f}_x \cap \mathbf{f}_y|}{\max(\min(|\mathbf{f}_x|, |\mathbf{f}_y|), \theta_{overlap})}$$

where $\mathbf{f}_x$ and $\mathbf{f}_y$ are sets of features extracted from documents d_x and d_y, respectively. $\theta_{overlap}$ is a threshold value to avoid too small denominator values in the equation, which is currently set to $\theta_{overlap} = 4$ determined on the training data. Similarities by NEs (sim_{NE}) and by CKWs (sim_{CKW}) are defined by this overlap coefficient, where $\mathbf{f}_x$ is a set of NEs in d_x for sim_{NE}, and a set of CKWs in d_x for sim_{CKW}.

The definition of similarities by URLs (sim_{URL}) is slightly different. Link similarity sim_{URL} is defined as follows.

$$sim_{\mathrm{URL}}(d_x, d_y) = \begin{cases} 1 \text{ if } & d_x \text{ links to } d_y \text{ or vise versa.} \\ \mathrm{Overlap}(d_x, d_y) & \text{otherwise} \end{cases} \tag{3}$$

4.4 Calculation of Merged Similarities

In this subsection, we describe how to merge different similarity scores. First, we define the merged similarity of different types of NEs. Next, we define the merged similarity of NEs, CKWs, and URLs.

4.4.1 Merging Different Similarities: for Named Entities

Different similarity scores are calculated for different types of named entities, namely, person names, location names, and organization names. We take the linear interpolation of these different scores:

$$sim_{\mathrm{NE}}(d_x, d_y) = \alpha_P sim_P(d_x, d_y) + \alpha_L sim_L(d_x, d_y) + \alpha_O sim_O(d_x, d_y)$$

where $\alpha_P + \alpha_L + \alpha_O = 1$. We set these values to be $\alpha_P > \alpha_O > \alpha_L$[6].

4.4.2 Merging Different Similarities: for NE, CKW, and LINK

We define the merged similarity score given the set of these different similarity values. The new similarity score is

[6]Currently, we use the parameters $\alpha_P = 0.78$, $\alpha_O = 0.16$, and $\alpha_L = 0.06$ tuned by the training data.

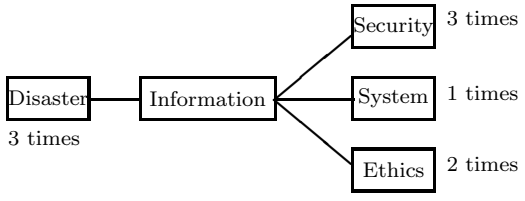

Figure 1: Example of statistics for term unit connections

calculated by taking the maximum of the given similarity values of NE, CKW, and LINK as follows.

$$sim_{\max}(d_x, d_y) = \max(sim_{\mathrm{NE}}(d_x, d_y), sim_{\mathrm{CKW}}(d_x, d_y), sim_{\mathrm{URL}}(d_x, d_y)) \quad (4)$$

4.5 Clustering Algorithm

With the document similarities calculated by the methods described above, the algorithm makes clusters of documents.

We used the standard hierarchical agglomerative clustering (HAC) algorithm for clustering. This algorithm starts from one-in-one clustering (each document is a size-one cluster) and iteratively merges the most-similar cluster pairs. The parameter of HAC is the similarity threshold value and does not need the number of clusters. We used the average-distance approach for defining inter-cluster distances. In this approach, similarity between clusters C_i, C_j is defined as equation (5). $sim_{\max}(d_x, d_y)$ is the above-mentioned similarity scores.

$$sim(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{d_x \in C_i} \sum_{d_y \in C_j} sim_{\max}(d_x, d_y) \quad (5)$$

5. SECOND-STAGE CLUSTERING

This section describes the second-stage clustering. It uses the bootstrapping approach which is achieved by matrix multiplication.

5.1 Second Stage Clustering by Bootstrapping

Our approach is to apply bootstrapping algorithm to the person name disambiguation. Bootstrapping is a method used originally to extract a set of instances (e.g., *country names*) and patterns iteratively. It starts with some seed instances and finds patterns that are useful to extract such seed instances (e.g., "*'s prime minister"). These patterns are in turn used to harvest new instances, and from the harvested new instances new patterns are induced. The algorithm repeats these steps until convergence criteria are fulfilled.

Espresso [20] is a well-known algorithm for information extraction that was proposed for harvesting semantic relations. It finds in documents *word pairs* that have relations similar to the given seed pairs, e.g., finds a pair (Ford, Nixon) given the pair (Bush, Regan).[7] It extracts instances and extraction patterns iteratively, and selects the instances and patterns according to the reliability function defined based on self-mutual-information values. Komachi et al. [13] provided a theoretical analysis for Espresso by representing it as HITS-like matrix multiplication. We hereinafter borrow this matrix representation for the Espresso algorithm from [13]. They represent Espresso algorithm[8] by the matrix multiplication as $\mathbf{i}^{(t+1)} = \frac{1}{|I||T|} \cdot \mathrm{M}^T \mathrm{M} \mathbf{i}^{(t)}$.Here, the matrix M represents strength of connections between instances and patterns, and vector $\mathbf{i}$ represents a cluster of instances where $\mathbf{i}_j$ is the weights (called *reliability*) to the j-th instance. Starting with the initial vector $\mathbf{i}^{(0)}$ which represents a cluster of seed instances, we can calculate the weights of instances obtained by bootstrapping by multiplying M, M^T, and the normalizing factor $\frac{1}{|I||T|}$ to the initial vector repeatedly.

We try to apply it to the person name disambiguation problem. In our problem settings, instances and patterns discussed above correspond to *documents* and *(weak) features*. Given the first-stage clusters, the algorithm regards them as seed instances, finds weak features related to them, and finds new instances (new documents, in our case) by using the weak features (as extraction patterns). One difference between our representation and the above one is that we use *matrix* R instead of *vector* i because we have more than one cluster. In our matrix representation, each column vector in matrix R represents each cluster. We can represent updating of clusters simultaneously by using R.

If we define a feature-document matrix P, which has strength of relations between the ith feature and jth document in its (i,j) element and denote a document-cluster matrix by $\mathrm{R}_D^{(t)} = \{r_{d,C}\}$, our algorithm can be formulated as $\mathrm{R}_D^{(t+1)} = \frac{1}{|D||F|} \cdot \mathrm{P}^{\mathrm{T}} \mathrm{P} \mathrm{R}_D^{(t)}$ ($|D|$ and $|F|$ are constants and do not affect the results in practice.)

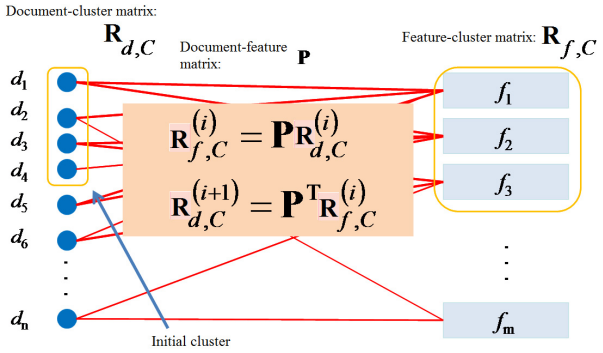

Figure 2: Second-Stage Clustering by Bootstrapping

Figure 2 illustrates the meaning of this matrix multiplication. Here, feature f_i is connected to document d_j if f_i is contained in d_j (i.e., the element $p_{i,j}$ in feature-document matrix P is not zero). Documents d_1, d_2, d_3, and d_4 are in the same initial cluster and this is represented in the document-cluster matrix by setting $r_{1,k}^{(0)} = r_{2,k}^{(0)} = r_{3,k}^{(0)} = r_{4,k}^{(0)} = 1$ if the cluster ID is k. Such clustering information is propagated to the feature-cluster matrix $\mathrm{R}_{f,C}^{(0)}$ through the document-feature matrix P. That is, feature-cluster relation weights are obtained by multiplying $\mathrm{R}_{d,C}^{(0)}$ by P. In resulting weight matrix $\mathrm{R}_{f,C}$, features strongly related to the k-th cluster are given high weights in the k-th column. After that, new document-cluster matrix $\mathrm{R}_{d,C}^{(1)}$ is obtained by multiplying $\mathrm{R}_{f,C}^{(0)}$ by P^T, which propagates the feature-cluster relation weights to the new document-cluster relation weights.

5.1.1 Algorithm

This section describes our bootstrapping algorithm. Here, we assume that first-stage clusters of size 2 or more are *seed instances*. The remaining documents (i.e., documents

[7]Both have the relation "succession".

[8]This representation is for the algorithm they call *simiplfied Espresso* in which some filtering steps after each iteration are omitted from the original Espresso. The bootstrapping algorithm we used is also this *simplified Espresso* algorithm.

Algorithm 1 Bootstrapping Algorithm for Person Name Disambiguation

Procedure: $D, F, \mathrm{R}_D^{(0)}$
Step-1: // Calculation of Feature-Document Matrix P

$$\mathrm{P}[f,d] = \begin{cases} \frac{1}{\max pmi} \log \frac{p(f,d)}{p(f)p(d)} & \text{if } \frac{p(f,d)}{p(f)p(d)} > 1 \\ 0 & \text{otherwise} \end{cases}, (f \in F, d \in D)$$

where $\max pmi = \max(\mathrm{P}[f', d'])$ $(f' \in F, d' \in D)$

for$t \in 0, \cdots, T-1$ // T: Number of Iterations
Step-2: $\mathrm{R}_F^{(t)} = \frac{1}{|D|}\mathrm{P}\mathrm{R}_D^{(t)}$
Step-3: $\mathrm{R}_D^{(t+1)} = \frac{1}{|F|}\mathrm{P}^{\mathrm{T}}\mathrm{R}_F^{(t)}$
endfor

Step-4:
for $C \in \mathcal{C}$ **do** $C_d^{(T)} = \arg\max_C r_{d,C'}^{(T)}$
where $\{C' | (C' \in \mathcal{C} \wedge |C| > 1) \vee C_d^{(0)}\}$
endfor
Define $\mathcal{C}^{(T)}$ based on C_d.
return $\mathcal{C}^{(T)}$

not connected to any other document) are the sources from which the algorithm extracts new instances for each cluster.

Algorithm 1 shows our bootstrapping algorithm. Here, P is a feature-document matrix, $\mathrm{R}_D^{(t)} = \{r_{d,C}\}$ is a document-cluster matrix, and $\mathrm{R}_F^{(t)} = \{r_{f,C}\}$ is a feature-cluster matrix. Our definition of document-feature matrix P is the same as in the Espresso [20] algorithm, which uses the self-mutual information. The algorithm updates $\mathrm{R}_D^{(t)}$ and $\mathrm{R}_F^{(t)}$ iteratively by multiplying P by P^T, resulting in refinement of the document-cluster matrix. The initial document-cluster matrix $\mathrm{R}_D^{(0)}$ is generated by setting $r_{d,C}^{(0)} = 1$ when $d \in C$ in the first-stage clustering result $\mathcal{C}$ and $r_{d,C}^{(0)} = 0$ if otherwise.

The algorithm is explained in detail below.

1. In step 1: A feature-document matrix P is generated. Notice that entries whose corresponding self-mutual information is below zero are set to zero.

2. In step 2: A feature-cluster matrix $\mathrm{R}_F^{(t)}$ is calculated from P and the current document-cluster matrix $\mathrm{R}_D^{(t)}$.

3. In step 3: A document-cluster matrix $\mathrm{R}_D^{(t+1)}$ is calculated from P and the current feature-cluster matrix $\mathrm{R}_F^{(t)}$.

4. In step 4: Find cluster $C' \in \mathcal{C}'$ for each document d that maximizes the relation value $r_{d,C'}$ where $\mathcal{C}' \in \mathcal{C}$, $|\mathcal{C}'| \geq 2$, and $d \in \mathcal{C}'^{(0)}$.

The algorithm iterates steps (2) and (3), and the final result $\mathcal{C}'$ is generated from document-cluster matrix.

6. EXPERIMENTS

In this section, we report our experimental results on the WePS-2 data set.

6.1 Data Sets and Baselines

We used the latest dataset for person name disambiguation: WePS-2 clustering task data set. The WePS-2 test set[9] consists of 30 names, each of which has 150 pages. The *all-in-one* baseline is the result when all documents belong to one cluster. The *one-in-one* baseline is the result when each document is a size-one cluster. The *combined* baseline is a mixture of these two baselines, where each document belongs to one cluster from the *all-in-one* baseline and another cluster from the *one-in-one* baseline. Note that the same document can refer to two or more entities in these data sets.

We used two baselines for the evaluation of the second-stage clustering. The first baseline is the *TOPIC algorithm* proposed by Ono et al.[19]. This algorithm estimate the *latent topic* (e.g., sports, computer science, arts, etc.) for each document by using the probabilistic generative model called the *Dirichlet process unigram mixture* where parameters are initialized by the first-stage clustering results. If two clusters share the same topic[10], they are merged into one cluster.

The second baseline is the *CKW algorithm* proposed by Ikeda et al.[11]. This second-stage clustering algorithm re-extracts CKWs from the resulting *clusters* of the first stage in contrast with the fact that CKWs for the first stage clustering are extracted from each *document*. The concept behind the algorithm is that CKWs extracted from clusters are more reliable than the ones extracted from documents because clusters contain more words than documents, and therefore the former provides more reliable statistics of term frequency for the keyword extraction algorithm than the latter. If documents in small-size clusters share the same re-extracted CKWs with large clusters, these documents are moved to the large cluster.

We used 1-gram and 2-gram features, excluding some stop words, to represent the documents for the second-stage clustering. The weight for each feature was defined by using TF-IDF scores, where the IDF values were estimated by using Web 1T 5-gram[11].

6.2 Evaluation Measures

The extended B-Cubed measure[1] was the measure adopted at the second WePS workshop held in 2008–2009[3].

Assume that $L(e)$ and $C(e)$ correspond to the collect class and machine-predicted class of e. Multiplicity precision and recall between e and e' are calculated as

$$MPrec(e,e') = \frac{Min\left(|C(e) \cap C(e')|, |L(e) \cap L(e')|\right)}{|C(e) \cap C(e')|}$$
$$MRec(e,e') = \frac{Min\left(|C(e) \cap C(e')|, |L(e) \cap L(e')|\right)}{|L(e) \cap L(e')|},$$

Extended B-Cubed precision (BEP) and recall (BER) are calculated as

[9] http://nlp.uned.es/weps/weps-2-data/
[10] The topic of each cluster is defined as the most frequent topic for documents in the cluster.
[11] http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2006T13

Table 1: Results: WePS-2 Data Set

Topic	BEP	BER	F_B
Baseline			
ALL IN ONE	0.43	1.00	0.53
ONE IN ONE	1.00	0.24	0.34
COMBINED	0.43	1.00	0.52
First-Stage Clustering			
ORIGINAL	0.92	0.70	0.78
Second-Stage Clustering			
TOPIC	0.94	0.70	0.79
CKW	0.87	0.77	0.81
BOOTSTRAP			
1-gram, $T = 1$	0.89	0.82	**0.85**
1-gram, $T = 2$	0.66	0.91	0.73
1-gram, $T = 3$	0.53	0.95	0.63
2-gram, $T = 1$	0.92	0.70	0.78
WePS top 3			
1st	0.87	0.79	0.82
2nd	0.85	0.80	0.81
3rd	0.93	0.73	0.81

$$\text{BEP} = \operatorname*{Avg}_{e}\left[\operatorname*{Avg}_{e',\ C(e)\cap C(e')\neq\emptyset}\left[MPrec(e,e')\right]\right]$$

$$\text{BER} = \operatorname*{Avg}_{e}\left[\operatorname*{Avg}_{e',\ L(e)\cap L(e')\neq\emptyset}\left[MRec(e,e')\right]\right]$$

Here, $\operatorname*{Avg}_{e}[\cdot]$ is the average through e. The F-measure is the harmonic mean of the two, $F = \frac{1}{\frac{1}{2}\left(\frac{1}{BEP}+\frac{1}{BER}\right)}$.

6.3 Results

We tested our algorithm on the WePS-2 data set with BEP-BER measures.

We show the results in Table 1. We have a parameter θ_f that determines the threshold value in the HAC algorithm used in the first-stage clustering. The value of θ_f was determined by using the training data.

"ORIGINAL" represents the results when we used the first-stage clustering only. "TOPIC" and "CKW" are baselines for the second-stage clustering. "BOOTSTRAP" represents our bootstrapping-based algorithm. The features used in the second-stage clustering are represented by "1-gram" and "2-gram", and the number of iterations in the second-stage clustering is T.

The two-stage clustering algorithm with CKW improved the results from ORIGINAL. Using topic features also improved the performance but the improvement was small. We believe the reason the TOPIC features did not work well is that TOPIC treats all single words equally without giving any weight to each word, while the bootstrapping approach can give weight to each weak feature through the calculation of reliability scores. BOOTSTRAP with 1-gram features and $T = 1$ performed the best (0.85 in B-Cubed f-measure.) among all the systems. We believe that one of the reasons our method performed well was that the first-stage clustering had already shown good precision values for the WePS-2 data set. This was good for our algorithm, which assumed high-precision clusters at the first stage.

The score of "BOOTSTRAP" was better than that of "CKW" by 0.04 points, which was even higher than the score of top-ranked system at WePS-2.

The use of 2-gram features did not contribute to improving the performance. (We observed no changes in clusters at the first iteration, so the second and third iterations were omitted.) We believe the reason was that the 2-gram features were "sparse" and our bootstrapping algorithm requires a large number of appearances of weak features. Increasing the iteration number for "1-gram" makes the precision values severely worse, and the improvement of recall values could not cover this performance drop. We believe the reason was that, in $T = 2, 3$ cases, weak features were overestimated and documents having only weak relations to each cluster were merged into the cluster.

We also tested the Von Neumann Kernel and Graph Laplacian reported in [13], which models the infinite number of iterations in efficient ways, but no improvement was observed. This result and the previous result that $T = 1$ was better than $T = 2, 3$ suggest that it is sufficient to do the iteration only once, which is the weight propagation from documents to features followed by the weight propagation in the opposite direction (i.e., from features to documents.) Iterating this procedure more than once causes too much overspreading of features, which results in low precision. However, it is possible that there is the optimum solution is in between $T = 0$ and $T = 1$, or between $T = 1$ and $T = 2$. Searching for these "intermediate" states by introducing a method to slow the propagation of weights is an interesting future direction.

7. CONCLUSION

We proposed a new algorithm for person name disambiguation. It consists of two stages where the results of the first-stage clustering are used to extract features for the second-stage clustering by applying the Espresso bootstrapping algorithm. We compared our method with various methods including two baseline two-stage clustering methods and WePS top systems and found that our method outperformed all of them. Future work includes testing our method on a greater variety of domains and improving the preprocessing algorithm to filter out useless documents.

Acknowledgments This work was supported in part by MEXT Grant-in-Aid for Scientific Research on Priority Areas: "Cyber Infrastructure for the Information-explosion Era" and MEXT Grant-in-Aid for Scientific Research (A).

8. REFERENCES

[1] E. Amigo, J. Gonzalo, J. Artiles, and F. Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*, 12(4), 2009.

[2] J. Artiles, J. Gonzalo, and S. Sekine. The SemEval-2007 WePS evaluation: Establishing a benchmark for the web people search task. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 64–69, 2007.

[3] J. Artiles, J. Gonzalo, and S. Sekine. WePS 2 Evaluation Campaign: overview of the Web People

Search Clustering Task. *2nd Web People Search Evaluation Workshop (WePS 2009)*, 2009.

[4] A. Bagga and B. Baldwin. Entity-based cross-document coreferencing using the vector space model. In *Proceedings of COLING-ACL 1998*, pages 79–85, 1998.

[5] K. Balog, L. Azzopardi, and M. de Rijke. Personal name resolution of web people search. In *WWW2008 Workshop: NLP Challenges in the Information Explosion Era (NLPIX 2008)*, 2008.

[6] R. Bekkerman and A. McCallum. Disambiguating web appearances of people in a social network. In *Proceedings of The 14th International World Wide Web Conference (WWW2005)*, pages 463–470, 2005.

[7] D. Bollegala, Y. Matsuo, and M. Ishizuka. Extracting key phrases to disambiguate personal name queries in web search. In *Proceedings of the Workshop: How can Computational Linguistics improve Information Retreival? at COLING-ACL 2006*, pages 17–24, 2006.

[8] R. Bunescu and M. Pasca. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06)*, 2006.

[9] E. Elmacioglu, Y. F. Tan, S. Yan, M.-Y. Kan, and D. Lee. PSNUS: Web people name disambiguation by simple clustering with rich features. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 268–271, 2007.

[10] M. Ikeda, S. Ono, I. Sato, M. Yoshida, and H. Nakagawa. Person Name Disambiguation on the Web by Two-Stage Clustering. *2nd Web People Search Evaluation Workshop (WePS 2009), 18th WWW Conference*, 2009.

[11] M. Ikeda, S. Ono, I. Sato, M. Yoshida, and H. Nakagawa. Person name disambiguation on the web by twostage clustering. In *Proceedings of the 2nd Web People Search Evaluation Workshop (WePS 2009) at WWW-2009*, 2009.

[12] D. Kalashnikov, R. Nuray-Turan, and S. Mehrotra. Towards breaking the quality curse.: a web-querying approach to web people search. In *Proceedings of SIGIR '08*, pages 27–34, 2008.

[13] M. Komachi, T. Kudo, M. Shimbo, and Y. Matsumoto. Graph-based Analysis of Semantic Drift in Espresso-like Bootstrapping Algorithms. *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1010–1019, 2008.

[14] X. Liu, Y. Gong, W. Xu, and S. Zhu. Document clustering with cluster refinement and model selection capabilities. *In Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 191–198, 2002.

[15] G. S. Mann and D. Yarowsky. Unsupervised personal name disambiguation. In *Proceedings of CoNLL2003*, pages 33–40, 2003.

[16] H. Nakagawa and T. Mori. Automatic term recognition based on statistics of compound nouns and their components. *Terminology*, 9(2):201–219, 2003.

[17] C. Niu, W. Li, and R. K. Srihari. Weakly supervised learning for cross-document person name disambiguation supported by information extraction. In *Proceedings of 42nd Annual Meeting of the Association for Computational Linguistics (ACL-2004)*, pages 598–605, 2004.

[18] R. Nuray-Turan, Z. Chen, D. Kalashnikov, and S. Mehrotra. Exploiting web querying for web people search in weps2. *2nd Web People Search Evaluation Workshop (WePS 2009)*, 2009.

[19] S. Ono, I. Sato, M. Yoshida, and H. Nakagawa. Person name disambiguation in web pages using social network, compound words and latent topics. In *Proceedings of the 12th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD2008)*, pages 260–271, 2008.

[20] P. Pantel and M. Pennacchiotti. Espresso: Leveraging Generic Patterns for Automatically Harvesting Semantic Relations. *In Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 113–120, 2006.

[21] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of CSCW 94 Conference on Computer Supported Cooperative Work*, pages 175–186. ACM Press, 1994.

[22] N. Slonim and N. Tishby. Document clustering using word clusters via the information bottleneck method. *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 208–215, 2000.

[23] N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. *Proceedings of the 37-thAnnual Allerton Conference on Communication*, 2000.

[24] X. Wan, M. L. J. Gao, and B. Ding. Person resolution in person search results: WebHawk. In *Proceedings of CIKM2005*, pages 163–170, 2005.