Cross-Domain Entity Resolution in Social Media

W. M. Campbell, Lin Li, C. Dagli, J. Acevedo-Aviles, K. Geyer, J. P. Campbell
Human Language Technology Group
MIT Lincoln Laboratory
Lexington MA
{wcampbell,lin.li,cdagli,joel,kgeyer,jpc}@ll.mit.edu

C. Priebe
Department of Applied Math and Statistics
Johns Hopkins University
Baltimore MD
cep@jhu.edu

Abstract

The challenge of associating entities across multiple domains is a key problem in social media understanding. Successful cross-domain entity resolution provides integration of information from multiple sites to create a complete picture of user and community activities, characteristics, and trends. In this work, we examine the problem of entity resolution across Twitter and Instagram using general techniques. Our methods fall into three categories: profile, content, and graph based. For the profile-based methods, we consider techniques based on approximate string matching. For content-based methods, we perform author identification. Finally, for graph-based methods, we apply novel cross-domain community detection methods and generate neighborhood-based features. The three categories of methods are applied to a large graph of users in Twitter and Instagram to understand challenges, determine performance, and understand fusion of multiple methods. Final results demonstrate an equal error rate less than 1%.

1 Introduction

We consider the problem of associating entities (typically people and organizations) across multiple social media sites. Specifically, given an entity's username `@org12` on Twitter, can we identify, if it exists, the corresponding username `@org123` for the same entity on Instagram. We call this problem *cross-domain entity resolution*.

Strategies for entity resolution are multifold. In a document data set, cross-document entity coreference resolution [Bagga and Baldwin, 1998] seeks to identify text mentions that correspond to the same entity. In another task, entity linking, the goal is to associate entity mentions in text to a knowledge base [Han *et al.*, 2011; Rao *et al.*, 2013]. Entity resolution, also known as record linkage or de-duplication, in relational data and databases is addressed in [Bhattacharya and Getoor, 2007; Christen, 2012].

The cross-domain entity resolution problem is strongly related to the above mentioned methods, but is distinct. In social media, information about an entity is present in multiple forms as profiles, content, and graph structure. Profiles give information about username, full name, profile pictures, links, etc. Content shows the topics and idiolect of a particular entity. Finally, graph structure relates the local communications and interests of the entity.

Prior approaches to the entity resolution problem in social media are given in [Bartunov *et al.*, 2012; Goga *et al.*, 2013; Iofciu *et al.*, 2011; Liu *et al.*, 2013; Lyzinski *et al.*, 2015; Malhotra *et al.*, 2012; Peled *et al.*, 2013; Raad *et al.*, 2010; Tan *et al.*, 2014; Zhang and Philip, 2015; Zhang *et al.*, 2015]. Peled, et. al. is the most complete and explores entity resolution with multiple features, including name, document similarity, and graph features. Our approach focuses on more challenging cross-domain platforms (Twitter and Instagram) at large scale with noisy, incomplete information. In addition, we use more advanced features in each category. We propose multiple normalization methods for profile matching, including a novel application of the Burrows-Wheeler transform, and additional token-based string comparison metrics. On the graph side, we construct a graph using both usernames and hashtags. We then apply a recent cross-domain community detection algorithm to generate additional graph features [Li and Campbell, 2015]. Our extensive experiments on Twitter and Instagram show the best within-category methods and also demonstrate the fusability of the different features.

The outline of this paper is as follows. In Section 2, we outline the basic problem and data setup. In Section 3, we describe profile-based methods and our approximate string matching techniques. In Section 4, we review author identification approaches. Section 5 discusses methods to perform joint community detection and the resulting graph-based features. Finally, Section 6 applies the methods to large Twitter and Instagram data sets and fuses the results.

2 Problem Setup

2.1 Entity Resolution in Cross Media

Our goal is to match entities across different social media platforms. Information from this process is obtained from

[†]This work was sponsored by the Defense Advanced Research Projects Agency under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

three sources. First, profile information from the JSON object for a post is used to obtain attributes of the entity (we use 'post' to describe posts on Instagram and tweets on Twitter). To create a general approach, we use only the username and (full) name of the entity as profile features. A second source of information is the content of the posts. Third, we construct a graph using mentions of hashtags and users and extract graph-based features; see Section 5 for more details.

Our basic approach to the problem is to create comparison scores between multiple features for entities and then fuse the resulting scores. For the graph-based features, we assume the presence of *seeds*. Seeds are known or high-confidence entity matches between different social media platforms.

Multiple problems are encountered in the process of matching. First, Twitter and Instagram are noisy platforms. Information about a user is self-reported in many cases and may be misleading or erroneous. Additionally, posts between users may not indicate a relation between individuals; this process creates a noisy graph. A second related problem is incomplete information. Many profiles for users are unavailable—their accounts are private or have been removed. Also, individuals may leave profile fields blank, post infrequently, or not communicate with others. These behaviors all create incomplete information in the proposed features. A third challenge, which we do not address in this paper, is sampling. Both the graph and content features are dependent on the duration and coverage of the sampling of posts. Overall, we address these challenges through feature robustness and fusion.

2.2 Data Set

Geotagged Twitter and Instagram data from the Boston area was collected for our experiments. Twitter data consist of approximately 4.65 million tweets collected from 1/1/2014 to 10/1/2014. Instagram data consist of 3.71 million posts (and comments) collected between 12/31/2013 to 12/31/2014. For Instagram, some comments on these posts extended into 2015.

3 Profile-Based Methods

Profile-based entity resolution relies upon comparing user attributes and finding measures of similarity. Depending on the social media platform, different profile information is available. For generality, we focus on usernames and full names.

Multiple observations can be made for these attributes. For usernames that may not be available across platforms, a common strategy is for users to add a few extra characters to their username on one platform to obtain a username on another platform. As a result, approximate string matching is an appropriate comparison method. Also, users may rearrange substrings in a username; e.g., change `@bobsmith` to `@smithbob`. For this case, we use a transform to normalize the rearrangement—Burrows-Wheeler. Finally, variations in case, unusual characters, etc., especially in full names, are addressed with text normalization.

3.1 Approximate String Matching Techniques

Our pipeline for approximate string matching is as follows. First, we optionally perform text normalization on the string. Normalization converts any unusual UTF-8 characters to a standardized form, eliminates emojis, and eliminates emoticons. We also remove nonsentential punctuation, markup, and long repeats (e.g., 'cooooool' to 'cool'). Full names are reordered to a standard form (e.g., 'Smith, Bob' to 'Bob Smith'). Second, we optionally lowercase the entire string. Third, we compute a normalized similarity between strings.

For approximate string metrics, we use standard methods—Damerau-Levenshtein, Levenshtein, Jaro, Jaro-Winkler, and Jaro-Winkler with soft-TFIDF; see, for example [Cohen *et al.*, 2003]. To convert Levenshtein and Damerau-Levenshtein to similarity measures with a $[0, 1]$ range, we apply the transform from [Yujian and Bo, 2007]. To obtain a similarity from Jaro and Jaro-Winkler, we use 1 minus the distance.

3.2 Burrows-Wheeler Transform

Users naturally segment and rearrange substrings in usernames to create new usernames. Finding matches in this case is an interesting and challenging problem. A natural solution, which is difficult to implement, is to segment a username into tokens and then perform approximate token matching. For example, a username like `@bobtsmith` might be segmented to 'bob', 't', and 'smith' and then comparisons could be based on the match of these tokens. This process requires a trained system to split the character stream; i.e., how do we infer where spaces have been removed.

As an alternate to a full tokenization approach, we consider only limited movement of tokens within a string. Specifically, only circular shifts of tokens are allowed. For example, `@bobtsmith` to `@smithbobt`, but not to `@tbobsmith`. We use a lossy version of the Burrows-Wheeler (BW) transform [Burrows and Wheeler, 1994] on the input string. We *do not* add a terminate character to the input. We then, as in the standard BW transform, find all circular shifts of the string. Next, the strings are sorted lexicographically. The BW transform is the concatenation of the last character of each string in the sorted list. For instance, in our example, we obtain the same BW transform `@hotmsbtib` for both inputs `@smithbobt` and `@tbobsmith`. Note this avoids comparing all possible shifts of usernames to find the best alignment.

4 Content-Based Methods

For content-based entity disambiguation, we use standard methods for author identification based on idiolect [Campbell *et al.*, 2007]. For each Twitter user U, we collect all tweets. We then perform text normalization to eliminate unusual UTF-8 characters (emojis, emoticons), links, and any repeated characters. Then, counts of the words and hashtags are found, $\mathrm{count}(w_i|U)$, where w_i is the ith word in the dictionary of possible words (including hashtags).

As in standard text classification, the counts are converted to a vector $\mathbf{v}$ with weighted probability entries

$$v_i = c_i p(w_i|U) \qquad (1)$$

where $p(w_i|U) = \frac{\mathrm{count}(w_i|U)}{\sum_V \mathrm{count}(w_i|V)}$. We use a log weighting of $c_i = \log\left(\frac{1}{p(w_i|\mathrm{all})}\right) + 1$, where $p(w_i|\mathrm{all})$ is the probability of word w_i across all Twitter users.

To train author identification models, we use a Support Vector Machine (SVM) one-vs-rest approach; i.e., we train each Twitter user U against the remaining Twitter users to obtain an SVM $f_U(\cdot)$. For our approach, a linear kernel is used. Additionally, models are built only if a minimum number of words (200, empirically determined) was available for that user. This minimum word count ensures that the training process is well-conditioned, but results in many users not having models.

To produce scores, we again find a single vector per user based on the Instagram posts. These vectors are scored with the Twitter SVM models, $f_U(\cdot)$, for all trials. Note that the role of Twitter and Instagram for training and testing can be switched, but we use only one direction for simplicity.

5 Graph-Based Methods

The graph-based approach extracts user features based on the graph structure and computes a similarity between the graph features. Examples of graph features are community membership and (weighted) neighborhood. Before describing the extraction of different graph features, it is important to first construct graphs that capture the richness of both Twitter and Instagram data.

5.1 Content + Context Graph Construction

Graph construction is performed by designating both users (e.g., `@twitter`) and hashtags (e.g., `#fashion`) as vertices in the graph. For Twitter, edge types correspond to multiple categories—user-to-user tweets, user mentions of users or hashtags, retweets, and co-occurrence of hashtags or users. For Instagram, edge types correspond to—user comments on user posts, user mentions of users, user mentions of hashtags, and co-occurrence of users or hashtags. For both Twitter and Instagram, the count of occurrence of various edge types is saved in the graph. For final analysis, counts are summed across edge types, resulting in a weighted undirected graph.

5.2 Cross-Domain Community Detection

Using the technique described above, we can construct a Twitter graph G_{twitter} and an Instagram graph G_{inst}. To extract community features, we perform cross-media community detection to identify communities simultaneously across Twitter and Instagram graphs. The key to achieving this is to align the graphs using *seeds*. Seeds are known vertex matches across graphs (e.g., one obvious choice of seeds is the common hashtags). We use a random walk-based approach to align the graphs to form a single interconnected graph. There are three general strategies: (1) *aggregation* that merges pairs of vertices in the seed set; (2) *linking* that adds links to the seed pairs; and (3) *relaxed random walk* that allows a random walker to switch between graphs with some probability. Once the graphs are aligned and connected, it is straightforward to adapt Infomap [Rosvall and Bergstrom, 2008] for community detection. Infomap is a random walk-based algorithm that partitions the graph by minimizing an information-theoretic based objective function.

For the experiment, we use the aggregation approach with Infomap for community detection across Twitter and Instagram graphs. Prior work [Li and Campbell, 2015] shows that with a sufficient number of seeds, the aggregation approach is the most faithful to the underlying community structure. Specifically, we first associate a Markov transition matrix to the *union* of G_{twitter} and G_{inst}. Each element in the Markov matrix represents the probability of a random walk of length 1 going from one vertex to the other; the Markov transition probability is computed by normalizing the edge weights between a vertex and all of its adjacent vertices. Second, for each vertex pair in seeds, we merge the two vertices and update the transition matrix with probability $p = 0.5$ that a random walk moves to the adjacent vertices in G_{twitter} and probability $1-p$ that a random walk moves to the adjacent vertices in G_{inst}. The resulting aligned and connected graph is denoted as G_{join}; it includes all the vertices from both G_{twitter} and G_{inst} and the edge weights are given by the Markov transition matrix. Additionally, we apply Infomap only on the largest connected component of the aligned graph G_{join}; vertices that are in the largest connected component have a community assignment.

5.3 Graph-Based Features and Similarity Measures

As hinted earlier, we are interested in extracting two classes of graph features: community features and neighborhood features. Note that neighbors of a vertex in a graph are vertices connected by an edge to the specified vertex; they are also referred to as 1-hop neighbors. Generally, for a vertex v in a graph, k-hop neighbors are defined as vertices that are reachable from v in *exactly* k hops.

Community Features and Similarity

The basic idea is to be able to represent the similarity in community membership between users across graphs. First, we perform a cross-media community detection on Twitter and Instagram graphs. One simple way to compare community features of two users is to assign a value '1' to the pair that are in the same community and '0' otherwise. However, this binary-valued similarity score will likely cause confusion because it assigns a similarity score '1' to all users belonging to the same community. To mitigate this problem, we propose to represent a user's community feature via the community membership of all its (k-hop) neighbors in its respective graph. For example, the community feature for a Twitter user U is given by a count vector $\mathbf{c}(U)$ with entries

$$c_i = |\{N | \operatorname{comm}(N) = i,\ N \in \operatorname{nbr}(U|G_{\text{twitter}})\}| \quad (2)$$

where $\operatorname{comm}(\cdot)$ indicates the community assignment of vertex N and $\operatorname{nbr}(\cdot)$ is the set of k-hop neighbors.

For the experiment, we set $k = 1, 2$ and use one of the two methods to measure the similarity in community feature between users. One is to compute the dot product of normalized count vectors, i.e., $\operatorname{sim}(\mathbf{c}(U_i), \mathbf{c}(U_j)) = \frac{\mathbf{c}(U_i)^T \mathbf{c}(U_j)}{\|\mathbf{c}(U_i)\|_2 \|\mathbf{c}(U_j)\|_2}$. The other method is to use an SVM; it is similar to the method for author identification discussed in Section 4, but replacing a dictionary of words with a dictionary of communities.

Neighborhood Features and Similarity

Given the aligned and connected graph G_{join} (i.e., constructed from joining G_{twitter} and G_{inst} using seeds), we seek to compute the similarity between two users by analyzing the proximity of their corresponding vertices in G_{join}. A popular approach is based on vertex neighborhoods [Liben-Nowell and Kleinberg, 2003], e.g., common-neighbors approach and its variants. The approach here is similar. However, instead of simply counting the number of common neighbors, we represent the neighborhood feature using the transition probability of the random walk in G_{join}. Specifically, the neighborhood feature of a Twitter user U is given by $\mathbf{p}(U)$, whose i^{th} entry $p_i = p(U, U_i)$ represents the probability that a random walk of a specified length k starting at U ends at the i^{th} vertex U_i in G_{join}. The neighborhood similarity is given by the normalized dot product of the neighorhood features.

We choose to use $k = 1, 2$ hops for computing the neighborhood similarity. Note that for $k = 1$, the edge probability $p(U_1, U_2) = 0$ if U_1 and U_2 are not connected. Also, isolated vertices are not considered.

6 Experimental Results

6.1 Setup

We use the Twitter and Instagram data sets in Section 2 for our experiments. For both data sets, profiles are extracted from the posts, and graphs are constructed using the method in Section 5.1. This processing results in 141.7K Twitter profiles and 925K Instagram profiles. The Twitter graph has 860.8K vertices (280.5K hashtags, 580.3K users) and 2.56M edges. The Instagram graph has 1.667M vertices (533.7K hashtags, 1.134M users) and 9.706M edges. Note that not all users have profile information because a user may be mentioned, but no post is observed from that user.

We construct a common set of trials to test all systems. The trials consist of user pairs (u_t, u_i) where u_t is a user from Twitter and u_i is a user from Instagram. True trials (i.e., same entity trials) are constructed with two strategies: (1) self-reported Twitter-Instagram links in user profiles and (2) links in tweets to Instagram to associate users across the different sites. False trials are constructed by taking a true trial (u_t, u_i) and randomly sampling Instagram users $v_i \neq u_i$ to produce trials (u_t, v_i). Note that only users with one known account on both Twitter and Instagram are used; users with multiple accounts are discarded.

The resulting number of trials is 25,548 true trials and 255,480 false trials. We use five-fold cross validation (train/test) to score all systems. Results are reported by pooling all test scores across all five splits and evaluating the results. We report results for both all trials, which include all of the trials above, and the non-trivial (NT) trials. Non-trivial trials are trials where the usernames are not an exact string match.

6.2 Evaluation

We use two methods for evaluation in the context of a detection problem. First, we look at miss P_m and false alarm P_{fa} for a threshold T. Each trial results in a score. If that score belongs to a true trial (i.e., should be an actual match) and is below T, this results in a miss. If the trial is a false trial (i.e., not an actual match) and the score is above the threshold, then a false alarm occurs. Sweeping the threshold where P_m equals P_{fa} gives our first method of evaluation, the equal error rate (EER). Our second evaluation method is to show detection error tradeoff (DET) curves. This allows a user to view performance at multiple operating points; e.g., low false alarm probability.

Two issues are addressed in evaluation. First, we use miss and false alarm rate instead of precision and recall so that our evaluation is not dependent on the priors for true and false trials. Second, we note when missing trials occur because of missing content or data. The main impact of missing trials is that cross-feature performance is difficult to compare exactly. We address this by comparing only per feature performance and fusion performance. The relative merit of individual features can be approximately inferred from their impact on fusion.

6.3 Profile and Content-Based Features

We use the pipeline of four stages for comparing profile usernames and full names as described in Section 3: **norm**, string normalization; **lower**, conversion to lower case; **bw**, lossy Burrows-Wheeler transform (non-invertible); and **jw/nl/ndl**, the approximate match method–Jaro-Winkler, normalized Levenshtein, and normalized Damerau-Levenshtein. A 'no' in front of the step indicates it is omitted. All trials specified in Section 2 are scored for both profile features.

A plot comparing a representative set of systems for comparing usernames is shown in Figure 1. We make several observations. First, only minor performance differences are observed between Jaro and Jaro-Winkler. Also, the same is true for Levenshtein and Damerau-Levenshtein. Second, normalization and lower case conversion are not useful (as expected). Third, the biggest performance impact is the use of the Burrows-Wheeler type transform. We observe across many of the cross-validation splits that the BW transform crosses over with the no BW case in the DET curve. Sometimes this cross-over is after the EER (as shown) or before. This demonstrates that the usefulness of the BW transform depends on the operating point.

A plot comparing a representative set of systems for full name is shown in Figure 2. The performance is quite distinct from the username case. The best performing system uses Jaro-Winkler and normalizing transforms. Removing these steps substantially impacts performance. In addition, the BW step substantially lowers performance. This property might be expected since name variation is not easily captured as a circular string rotation. Finally, as before, normalized Levenshtein and Damerau-Levenshtein do not perform as well as the Jaro-type comparisons.

As a summary of performance at EER, see the results for profile and content features in Table 1. As a convention in all tables, we bold the metric per feature of the best performing system. In the table, for the full name feature, we show all of the variations of the pipeline for Jaro-Winkler approximate string comparison. Results for the other cases are comparable. For both username and full name, Jaro-Winkler with normalization and lowercase with no Burrows-Wheeler are best

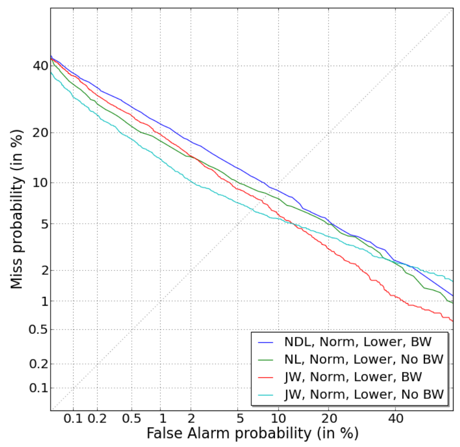

Figure 1: DET curve comparing the performance of multiple approximate string matching methods on the profile username. Note that the curves are for *non-trivial* trials.

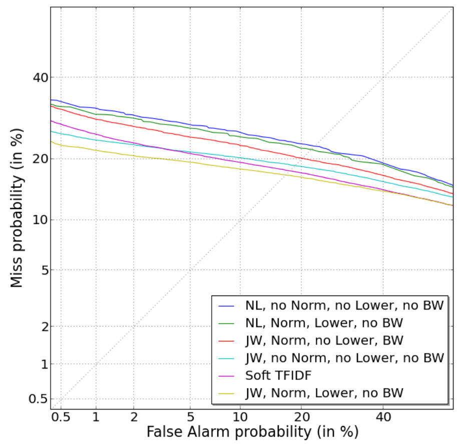

Figure 2: DET curve comparing the performance of multiple approximate string matching methods on the profile full name. Note that the curves are for *non-trivial* trials.

at EER. The table also highlights performance for the content features and the token-based soft-TFIDF method.

6.4 Graph-Based Features

Graph features include community features and neighborhood features (NBR). We use two methods for computing the community similarity: dot product of normalized feature vectors (DP) and SVM. For seed selection, we look at two different choices of seeds to align Twitter and Instagram graphs. One is to use common hashtags, for example, #Boston is present in both graphs and, thus, vertices associated with #Boston can be used as seeds. The performance of the two graph features is summarized in Table 2, showing both the EER for all trials and EER for the non-trivial trials. Note that for community feature using 1-hop neighbors, the SVM method gives better results, while DP gives better results for community feature using 2-hop neighbors.

The other choice of seeds is to use both common hashtags and users with the same usernames across graphs. Table 3 shows the performance of the two graph features. In particular, DP is better than SVM at EER (all trails) for both 1-hop and 2-hop community feature. Note that the EER for all trials is much less than the EER for non-trivial trials. Because seeds are merged in the aligned graph, for users with the same username, their neighborhood features are the same and their community features tend to be similar.

Table 1: Comparison of EERs for various profile and content features and systems.

Feature	System	EER All (%)	EER NT (%)
Full	jw, norm, lower, bw	18.73	16.94
Full	jw, norm, lower, nobw	**16.90**	**14.97**
Full	jw, norm, nolower, bw	20.13	18.69
Full	jw, norm, nolower, nobw	18.41	16.86
Full	jw, nonorm, lower, bw	19.07	17.39
Full	jw, nonorm, lower, nobw	17.26	15.36
Full	jw, nonorm, nolower, bw	20.32	18.95
Full	jw, nonorm, nolower, nobw	18.71	17.13
Full	ndl, nonorm, lower, nobw	21.61	20.04
Full	nl, nonorm, lower, nobw	21.64	20.08
Full	soft-tfidf	17.69	16.24
User	jw, nonorm, lower, bw	2.64	7.54
User	jw, nonorm, lower, nobw	**2.03**	**6.45**
User	ndl, nonorm, lower, bw	3.17	9.06
User	ndl, nonorm, lower, nobw	2.65	8.46
User	nl, nonorm, lower, bw	3.14	9.24
User	nl, nonorm, lower, nobw	2.66	8.52
Content*	SVM	**26.56**	**18.66**

*subset of all trials because of limited content

Table 2: Summary of EERs for various community and neighborhood features and systems. Common hashtags are used as seeds.

Feature	System	EER ALL (%)	EER NT (%)
Comm 1-hop	DP	50.13*	30.22*
Comm 1-hop	SVM	**30.47**	**23.56**
Comm 2-hop	DP	**39.64**	**30.42**
Comm 2-hop	SVM	46.95	45.40
NBR 1-hop	DP	60.04*	39.57*
NBR 2-hop	DP	42.64	34.83

*interpolated values

Table 3: Summary of EERs for various community and neighborhood features and systems. Common hashtags and users with exact usernames are used as seeds.

Feature	System	EER ALL (%)	EER NT (%)
Comm 1-hop	DP	**6.65**	20.90*
Comm 1-hop	SVM	8.33	**17.16**
Comm 2-hop	DP	**36.46**	**29.11**
Comm 2-hop	SVM	43.75	44.92
NBR 1-hop	DP	6.60	29.38*
NBR 2-hop	DP	9.38	28.47

*interpolated values

Table 4: Summary of fusion results. Common hashtags are used as seeds.

Fusion	Model	EER ALL (%)	EER NT (%)
P	Logit	2.07	6.07
P	RandF	1.54	5.74*
P+C	Logit	2.01	5.74
P+C	RandF	1.47	5.18*
N1	Logit	26.74	25.31
N1	RandF	28.44	29.44
N1+N2	Logit	26.97	26.76
N1+N2	RandF	27.43	25.91
P+N1	Logit	1.64	4.82
P+N1	RandF	1.16	3.79
P+C+N1	Logit	1.64	4.87
P+C+N1	RandF	1.13	3.90*
P+N1+N2	Logit	1.68	4.85
P+N1+N2	RandF	1.01	3.18
P+C+N1+N2	Logit	1.68	4.92
P+C+N1+N2	RandF	**1.00**	**3.32**

*interpolated values

6.5 Fusion

The goal for fusion is to combine the similarity scores from different features to obtain a better entity disambiguation system. The models we train are logistic regression and random forest. The former builds a linear model that predicts the probability of an outcome using the logistic distribution function; the latter is an ensemble learning method that constructs a number of decision trees and averages the probabilities of an outcome from these decision trees.

One issue for training the fusion models is the problem of missing data. For example, the content features may be missing for some users due to an insufficient number of posted words. Similarly, community (or neighborhood) features may be missing because users are not in the largest connected component (or they are isolates) in the aligned graph. One simple strategy to address this issue is to train a separate model for all combinations of present features using the training data, and then to test by breaking up the testing set according to the present features and using the respective model to compute the fusion score.

Results for fusion are summarized in Tables 4 and 5. Features used in the fusion are greedily selected, i.e., the best system over all trials per feature type. The variable 'P' denotes the fusion of profile features (for the purposes of this paper, username and full name), 'C' denotes content, 'N1' is the fusion of 1-hop community and 1-hop neighborhood features, and 'N2' is the fusion of 2-hop community and 2-hop neighborhood features. Figure 3 shows the performances of fusing selected features using the random forest model. Observe that the fusion of profile, content and graph features significantly improves the performance as compared to the performance of individual features. Also, the addition of graph features to the fusion model significantly lowers the miss probability, particularly at the low false alarm probability. Another observation is that seeding using common hashtags fuses as well as seeding using both common hashtags and users with the same usernames.

Table 5: Summary of fusion results. Common hashtags and users with exact usernames are used as seeds.

Fusion	Model	EER ALL (%)	EER NT (%)
N1	Logit	6.54	19.93
N1	RandF	6.87	23.11
N1+N2	Logit	6.49	19.76
N1+N2	RandF	6.44	20.43
P+N1	Logit	1.47	4.56
P+N1	RandF	1.08	3.43
P+C+N1	Logit	1.53	4.66
P+C+N1	RandF	1.03	3.30*
P+N1+N2	Logit	1.44	4.44
P+N1+N2	RandF	0.94	3.18
P+C+N1+N2	Logit	1.48	4.56
P+C+N1+N2	RandF	**0.89**	**3.08**

*interpolated values

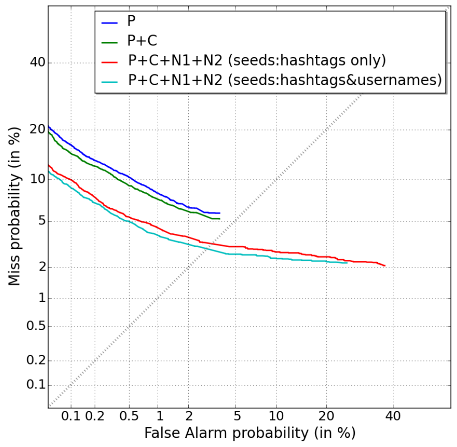

Figure 3: DET curve comparing the performances of fusing selected features using the random forest model. Note that the curves are for non-trivial trials.

7 Conclusions

In this paper, we demonstrated profile-, content-, and graph-based features for cross-domain entity resolution. Novel methods for both profile and graph based features were presented. Fusion of these feature types showed that profile and graph features worked best in combination. Excellent performance was achieved for non-trivial entity resolution. We demonstrated our methods on Twitter and Instagram, and we expect our approach to generalize for other social media platforms, as well.

Future work includes the use of additional features, such as characteristics of profile images (e.g., properties and hashes), patterns of life (e.g., geographic tracks over time and activity over time), and stylometrics (e.g., emojis, emoticons, and unconventional writing).

Acknowledgements

The authors acknowledge inspiration and early collaboration with Daniel Sussman (Harvard University), Vince Lyzinski (JHU HLTCOE), and Jason Matterer (JHU, MIT LL).

References

[Bagga and Baldwin, 1998] Amit Bagga and Breck Baldwin. Entity-based cross-document coreferencing using the vector space model. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 79–85. Association for Computational Linguistics, 1998.

[Bartunov *et al.*, 2012] Sergey Bartunov, Anton Korshunov, Seung-Taek Park, Wonho Ryu, and Hyungdong Lee. Joint link-attribute user identity resolution in online social networks. In *Proceedings of the 6th International Conference on Knowledge Discovery and Data Mining, Workshop on Social Network Mining and Analysis. ACM*, 2012.

[Bhattacharya and Getoor, 2007] Indrajit Bhattacharya and Lise Getoor. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):5, 2007.

[Burrows and Wheeler, 1994] Michael Burrows and David Wheeler. A block-sorting lossless data compression algorithm. In *Digital Equipment Corporation Technical Report*, number 124. Citeseer, 1994.

[Campbell *et al.*, 2007] William M Campbell, Joseph P Campbell, Terry P Gleason, Douglas A Reynolds, and Wade Shen. Speaker verification using support vector machines and high-level features. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(7):2085–2094, 2007.

[Christen, 2012] Peter Christen. A survey of indexing techniques for scalable record linkage and deduplication. *Knowledge and Data Engineering, IEEE Transactions on*, 24(9):1537–1555, 2012.

[Cohen *et al.*, 2003] William W Cohen, Pradeep D Ravikumar, Stephen E Fienberg, et al. A comparison of string distance metrics for name-matching tasks. In *IIWeb*, volume 2003, pages 73–78, 2003.

[Goga *et al.*, 2013] Oana Goga, Howard Lei, Sree Hari Krishnan Parthasarathi, Gerald Friedland, Robin Sommer, and Renata Teixeira. Exploiting innocuous activity for correlating users across sites. In *Proceedings of the 22nd international conference on World Wide Web*, pages 447–458. International World Wide Web Conferences Steering Committee, 2013.

[Han *et al.*, 2011] Xianpei Han, Le Sun, and Jun Zhao. Collective entity linking in web text: a graph-based method. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 765–774. ACM, 2011.

[Iofciu *et al.*, 2011] Tereza Iofciu, Peter Fankhauser, Fabian Abel, and Kerstin Bischoff. Identifying users across social tagging systems. In *ICWSM*, 2011.

[Li and Campbell, 2015] Lin Li and William M. Campbell. Matching community structure across online social networks. In *Proceedings of the NIPS Workshop: Networks in the Social and Information Sciences*, 2015.

[Liben-Nowell and Kleinberg, 2003] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, pages 556–559. ACM, 2003.

[Liu *et al.*, 2013] Jing Liu, Fan Zhang, Xinying Song, Young-In Song, Chin-Yew Lin, and Hsiao-Wuen Hon. What's in a name?: an unsupervised approach to link users across communities. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 495–504. ACM, 2013.

[Lyzinski *et al.*, 2015] Vince Lyzinski, Daniel L Sussman, Donniell E Fishkind, Henry Pao, Li Chen, Joshua T Vogelstein, Youngser Park, and Carey E Priebe. Spectral clustering for divide-and-conquer graph matching. *Parallel Computing*, 47:70–87, 2015.

[Malhotra *et al.*, 2012] Anshu Malhotra, Luam Totti, Wagner Meira Jr, Ponnurangam Kumaraguru, and Virgilio Almeida. Studying user footprints in different online social networks. In *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, pages 1065–1070. IEEE Computer Society, 2012.

[Peled *et al.*, 2013] Olga Peled, Michael Fire, Lior Rokach, and Yuval Elovici. Entity matching in online social networks. In *Social Computing (SocialCom), 2013 International Conference on*, pages 339–344. IEEE, 2013.

[Raad *et al.*, 2010] Elie Raad, Richard Chbeir, and Albert Dipanda. User profile matching in social networks. In *Network-Based Information Systems (NBiS), 2010 13th International Conference on*, pages 297–304. IEEE, 2010.

[Rao *et al.*, 2013] Delip Rao, Paul McNamee, and Mark Dredze. Entity linking: Finding extracted entities in a knowledge base. In *Multi-source, Multilingual Information Extraction and Summarization*, pages 93–115. Springer, 2013.

[Rosvall and Bergstrom, 2008] Martin Rosvall and Carl T Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008.

[Tan *et al.*, 2014] Shulong Tan, Ziyu Guan, Deng Cai, Xuzhen Qin, Jiajun Bu, and Chun Chen. Mapping users across networks by manifold alignment on hypergraph. In *AAAI*, volume 14, pages 159–165, 2014.

[Yujian and Bo, 2007] Li Yujian and Liu Bo. A normalized Levenshtein distance metric. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1091–1095, 2007.

[Zhang and Philip, 2015] Jiawei Zhang and S Yu Philip. Multiple anonymized social networks alignment. *Network*, 3(3):6, 2015.

[Zhang *et al.*, 2015] Yutao Zhang, Jie Tang, Zhilin Yang, Jian Pei, and Philip S Yu. Cosnet: Connecting heterogeneous social networks with local and global consistency. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1485–1494. ACM, 2015.