

# Entity Resolution in Geospatial Data Integration

Vivek Sehgal  
Computer Sc. Dept.  
University of Maryland  
College Park, MD USA  
viveks@cs.umd.edu

\*

## ABSTRACT

Due to the growing availability of geospatial data from a wide variety of sources, there is a pressing need for robust, accurate and automatic merging and matching techniques. *Geospatial Entity Resolution* is the process of determining, from a collection of database sources referring to geospatial locations, a single consolidated collection of 'true' locations. At the heart of this process is the problem of determining when two locations references match—i.e., when they refer to the same underlying location. In this paper, we introduce a novel method for resolving location entities in geospatial data. A typical geospatial database contains heterogeneous features such as location name, spatial coordinates, location type and demographic information. We also construct features using neighbors of a location. We investigate the use of all of these features in algorithms for geospatial entity resolution. Entity resolution is further complicated by the fact that the different sources may use different vocabularies for describing the location types and a semantic mapping is required. We propose a novel approach which learns how to combine the different features to perform accurate resolutions. We present experimental results showing that methods combining spatial, non-spatial (e.g., location-name, location-type, etc.) and neighbor information together outperform methods based on spatial or name information alone.

## Categories and Subject Descriptors

H.4 [Information Systems Applications]: Miscellaneous

## General Terms

Algorithms, Experimentation

\*In collaboration with Lise Getoor and Peter D Viechniki

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ACM-GIS'06, November 10–11, 2006, Arlington, Virginia, USA.  
Copyright 2006 ACM 1-59593-529-0/06/0011 ...\$5.00.

## 1. INTRODUCTION

The rapid growth of geospatial data has fueled significant interest in mining relevant information from them. The availability of geospatial data from multiple sources requires the integration of this information before using it effectively. An important part of integration involves finding matching locations across datasets. In other words, we need to reconcile location references corresponding to the same real-world location entity. Previous work on entity resolution has primarily focused on non-spatial data, such as medical data [12], census data [16], bibliographic data [6, 2], natural language [7, 9] and relational data [13].

The presence of a continuous spatial component in geospatial data makes the problem hard, as the data captured from the spatial domain is often noisy and imprecise. Also, different organizations may record spatial properties of entities using different scale and structure [1]. For example, one organization might represent mountains as points, while another could represent them as regions. In recent work, Beeri et al. [1] use only spatial features to find matches between datasets. Their results show that entity resolution is non-trivial even while using only spatial information as features.

The presence of non-spatial information can be very valuable in improving entity resolution. A typical dataset contains information such as location name, spatial coordinates (i.e., latitude and longitude), location type (e.g. dune, city, river, airport, etc.) and demographic information (e.g. population, etc.). The confidence in matching two locations from different datasets with similar coordinates should increase if non-spatial information such as location name and location type are also similar. The use of non-spatial information may improve performance by identifying matching locations which would otherwise be rejected by traditional spatial-only approaches. This impacts the *recall* of the entity resolution algorithm. Similarly, two locations with similar coordinates but with dissimilar non-spatial features should have a reduced chance of being labeled as matches. This can impact the *precision* of the entity resolution algorithm. In our experiments, we find that using spatial and non-spatial together improves both recall and precision.

In this paper, we develop methods to use both spatial and non-spatial features for entity resolution. Integrating these feature types is non-trivial because of the need to define a similarity metric which combines semantically distinct features. The use of non-spatial data itself is further complicated by the fact that certain features such as location-type may be represented differently in the datasets. For example, one dataset might label a real-world location as an "is-

land”, while the other might classify it as a “hypsographic” location with “hydrographic” locations as neighbors. In addition to mapping feature values between datasets, we also need to find corresponding weights for these feature types to construct a final similarity measure. The choice of the similarity metric and the corresponding weights strongly influence the accuracy of entity resolution. One can design a similarity metric manually, but given the complex relations between different features (e.g. location type with coordinates, etc.), such a naive approach may not be able to model the data well. In this paper, we learn the weights and also the similarity metric from ground truth data describing ‘true’ matches.

Unfortunately, learning is impeded by the fact that the ground truth data contains only positive examples. We can construct a training set consisting of both positive instances (matches) and negative instances (non-matches) from the ground truth data by including as non-matches all pairs of locations not included in the ground truth data. However this naive approach results in a training set overwhelmingly containing negative instances (i.e. pairs of locations which do not match). This skew not only impacts the usefulness (the accuracy of an approach would remain high even if it classifies all pairs of locations as non-duplicates), but it also makes learning highly inefficient due to the large number of negative instances. We explore ways of effectively choosing negative instances for training. Results of our extensive experimentation show how the size and choice of negative instances impact the accuracy in entity resolution.

In the next section, we introduce the problem in more detail. In Section 3, we describe previous approaches. In Section 4, we discuss our approach. Section 6 covers our extensive experimentation. Finally, we end with conclusions and future work.

## 2. THE GEOSPATIAL ENTITY RESOLUTION PROBLEM

In this paper, we explore a framework where we have two geospatial datasets  $A$  and  $B$  developed by independent sources. Each dataset is a collection of location references, each of which corresponds to some real-world location entity. The target of entity resolution is to find pairs of locations  $\{l_i, l_j\}$ , such that  $l_i \in A$  and  $l_j \in B$ , and  $l_i$  and  $l_j$  correspond to the same real-world location entity. As mentioned earlier, each location reference  $l_i$  is described by a set of features  $l_i = [location\ name, spatial\ coordinates, location\ type]$ . The spatial coordinates are defined as the latitude and longitude of the location. The location type is a categorical feature with a relatively small domain size. Figure 1 shows a simple example containing two matching location references obtained from two different data-sources. The first location, Qaryat an Nu‘aymiyah, is from the US Board on Geographic Names - Geographic Namespace Database (GNDB) and the second location, Qaryat an Na‘imiya, is from the 2005 Iraq Atlas published by the National Geospatial Intelligence Agency (NGADB). The matching locations have non-identical names (though they sound quite similar), coordinates and location types.

## 3. PREVIOUS APPROACHES

Commercial geographic-information systems [11] use spatial coordinates to join location references. One-sided near-

est join is the most common approach, where locations references  $l_i \in A$  and  $l_j \in B$  are labeled as duplicates, if  $l_j$  is closest to  $l_i$  given all locations in  $B$ . The asymmetry in the above approach is addressed by Beeri et al. [1]. They modify the definition by adding an extra condition that  $l_i$  is also closest to  $l_j$  given all locations in  $A$ . They show that addition of this condition may help improve precision of results, though the strong conditions may compromise recall. Another recent approach [10] integrates heterogeneous georeferenced data available from the web. Here, they match geospatial objects by comparing their attributes using a tunable string matching algorithm. Other methods in the domain of geospatial data integration such as [4, 8, 14, 15, 3] integrate digital maps. Image processing techniques [8] have also been applied in merging digital image maps. Satellite imagery has been the most popular information used for integrating maps. Though these methods help in a global alignment of geospatial database, the efficacy of these methods strongly depends upon the quality and richness of the satellite image. Also, these methods require that both the datasets have complete information about the region. Since many real-life geospatial integration tasks require integrating a small set of locations recently collected, it is critical to find alternate approaches using incomplete information. Efficiency and scalability are other issues with these approaches.

## 4. FEATURE-BASED MATCHING

In this section, we define matching approaches using each location feature (namely spatial coordinates, location name and location type) independently. Each approach first defines a similarity (or distance) measure and then use it to find matches across datasets.

### 4.1 Location Name Matching

As defined earlier, our problem is to find matching pairs of locations from two different datasets. Here a match is defined as the location  $l_j$  with the most “similar” name. We found that matching locations often had their names slightly misspelled. The similarity between two names can be defined using a variety of string similarity measures. One common measure is the Levenstein edit distance algorithm, where the distance is defined as the number of edit operations (e.g. addition, deletion and change of character) required to convert one string to another. We will refer to this measure as simply “Edit Distance” denoted as  $edit(S_1, S_2)$ . Here the string similarity measures we used give equal cost to each operation. It would be interesting to refine the cost to make them region and language specific. Another popular string similarity measure is the Jaccard similarity. For two strings  $S_1$  and  $S_2$ , let *common* be the set of characters which are present in both  $S_1$  and  $S_2$  and let *total* be the set of characters which are present in either of the strings, then the Jaccard similarity is defined as a ratio between *common* to the *total* characters.

$$Jacc(S_1, S_2) = \frac{|Unique\ common\ characters|}{|Total\ unique\ characters|}$$

A final string similarity measure that we use is the Jaro-Winkler measure which defines the distance between two strings by taking into account the spelling deviations. Let  $S'_1$  be the characters in  $S_1$  that are “common with”  $S_2$ , and



**Figure 1: Potential duplicate entries in separate gazetteer files for Iraq, with similar but not identical feature information. Location A’s (extracted from Geographic Namespace Database - GNDB) name attribute is “Qaryat an Nu’aymiyah”, and its geospatial feature type is ‘Populated Place’. Location B’s (extracted from National Geospatial Intelligence database - NGADB) name attribute is “Qaryat an Na’imiyah”, and its geospatial feature type is ‘Pop. Place’.**

let  $S'_2$  be the characters in  $S_2$  that are “common with”  $S_1$ ; roughly speaking, a character  $a$  in  $S_1$  is “common with”  $S_2$  if the same character appears in about the same place in  $S_2$ . Let  $T_{1,2}$  measure the number of transpositions of characters in  $S'_1$  relative to  $S'_2$ , then the Jaro similarity is defined as follows:

$$J(S_1, S_2) = \frac{1}{3} \left( \frac{|S'_1|}{|S_1|} + \frac{|S'_2|}{|S_2|} + \frac{|S'_1| - T_{1,2}}{2|S'_1|} \right)$$

and the JaroWinkler similarity is defined as:

$$Jw(S_1, S_2) = J(S_1, S_2) + C \times PrefixLength \times (1 - J(S_1, S_2))$$

Here  $C$  is a constant and  $PrefixLength$  is the length of the longest common prefix. For our experiments, we use  $C = 0.1$ . It has been shown to work well with person names. We refer to this measure as “Jw”.

## 4.2 Coordinate Matching

Assuming each location is a point reference, we define the spatial component for the location as latitude and longitude of the point. A naive approach to find the best match for a location  $l_i \in A$  in dataset  $B$  is to find the location  $l_j \in B$  with the closest latitude and longitude. Coordinate similarity is defined as the inverse of the coordinate distance denoted  $dist(l_i, l_j)$

$$CoordSym(l_i, l_j) = \frac{1}{dist(l_i, l_j)}$$

In this approach, locations  $l_i$  and  $l_j$  are mapped to each other if the coordinate similarity is above a threshold. Note that here locations in  $A$  are mapped to locations in  $B$  while the reverse is not guaranteed. We refer to this approach as *NN*.

$$CoordSym(l_i, l_j) = \frac{1}{cord(l_i, l_j)}$$

## 4.3 Location Type Matching

Location type can also aid us in finding matches by rejecting those pairs of locations where the types are not similar. Because location types have some underlying semantics, ideally we would like to make use of this to define for example a ‘river’ type to be more similar to a ‘stream’ type than to a ‘desert’ type. Another interesting challenge here is that the location types may be described using different vocabularies, so that not only do we need to be able to compute a semantically meaningful distance, but we must map between two potentially different vocabularies.

In this work, we do not assume that we are given any additional semantic information and we compute the location type similarity based on co-occurrence. First we build a positive training set containing pairs of location references which are matches. Similarly, we construct a negative training set containing all pairs of locations which are not matches. Now let  $t_i$  and  $t_j$  be the location types for locations  $l_i$  and  $l_j$  respectively, then the similarity between them is defined as

$$TypeSym(l_i, l_j) = n_{t_i, t_j} / (n_{t_i} + n_{t_j})$$

where  $n_{t_i, t_j}$  is equal to the number of positive training samples where types  $t_i$  and  $t_j$  co-occurred, while  $n_{t_i}$  and  $n_{t_j}$  are the total number of instances (both positive and negative) containing these types respectively. Intuitively, this definition captures the co-occurrence probability between any two location-types. If two locations are duplicates they are likely to have similar location types, but the same argument cannot be applied in the complimentary case. Hence, the similarity value between two location types is not reduced if they are seen together in negative instances.

## 5. INTEGRATING SPATIAL AND NON-SPATIAL FEATURES

Integration of location name and coordinates is complicated by the need of different similarity measures for the different feature types. One way to combine the similarity obtained from these two sets of features is by putting a threshold on one, while using the other as a secondary filter. Thus, a match for a location  $l_i \in A$  in dataset  $B$  is defined as the location  $l_j \in B$  with the most similar name and with coordinate similarity above some threshold. A similar definition can be made for the complimentary case where matches are found using coordinates and the name similarity is above some threshold. The extra information, for example coordinate similarity in the first case, helps us in rejecting those locations which have similar names but are too spatially distant to be considered matches. This can help in improving precision of the result set with minimal effect on the recall.

### 5.1 Learning a Combined Similarity Measure

Unfortunately, the above approach does not capture matches which are neither "most" similar in names or coordinates. In these cases, one needs to have a similarity measure which combines both spatial and non-spatial features. A naive choice of such a similarity measure can be a function that is the weighted mean of similarity between coordinates and names, where the weights are chosen manually in an ad-hoc manner. Since domain knowledge is not available for combining the features, deciding on the weights can be very difficult. Also for a more robust similarity measure, the measure should account for the relation between spatial and non-spatial features.

Here, we learn the weights from ground-truth data consisting of matching locations. From this ground-truth set of matches, we can construct a training set in which the positive instances correspond to those pairs of locations that are matches, while the negative instances consist of combinations of places that are not matches. Assuming there are no duplicates in the datasets themselves, two datasets  $A$  and  $B$  can have at most  $\min(|A|, |B|)$  matches. If we consider all non-matches, the number of negative instances is often very large (of the order  $|A| \times |B|$ ). This potential skew in training data may affect recall by misleading the classifier to always classify a location pair as a non-match. Also, the large size of negative instances set can make the training process inefficient.

To solve this problem, we select a smaller set of negative instances from the negative training set. Since misclassification of location pairs is more likely when they have similar features, it may prove beneficial to choose as negative instances those pairs of non-matching locations which have similar features. So we modify our negative instance selection policy to take the above into account. The negative instance set consists of both randomly selected non-matching locations and non-matching locations that have similar features (also referred as "near misses"). In general, the process of finding near misses is non-trivial because of presence of inter-related features. We construct the set of near-miss negative instances by choosing negative instances for every location  $l_i \in A$  as pairs  $(l_i, l_j)$ , where  $l_j \in B$  is not a true match for  $l_i$  and is part of the top  $k$  most similar locations in  $B$  to  $l_i$  using the similarity measure (i.e. Euclidean for spatial coordinates and string similarity for location names)

Table 1: Description of the two real-world dataset used for experimentation.

DATASET	LOCATIONS	LOC-TYPES
NGA	151101	9
PCGN	2096	49

for one of the features. In our experiments, we calculate negative instances using both spatial and non-spatial features independently. Here, the value of  $k$  controls the number of negative instances created using a particular feature and thus affects performance of the training process. Choosing an ideal  $k$  can be tricky given lack of domain knowledge and arbitrary complexity of the training method used. In case one of the features is known to be highly ambiguous in separating negative and positive instances, it might be helpful to choose a higher value of  $k$  for this feature to get a well-defined classification boundary. In our experiments, we vary the value of  $k$  to obtain an optimal set of negative training data.

Once the training data is constructed, each location pair  $(l_i, l_j)$  from the training data is converted into the following feature vector. The feature vector contains a list of spatial and non-spatial similarity values between the two locations  $l_i$  and  $l_j$  is:

$$(CoordSym(l_i, l_j), Jacc(l_i, l_j), Jw(l_i, l_j), SpatialSym(l_i, l_j), TypeSym(l_i, l_j))$$

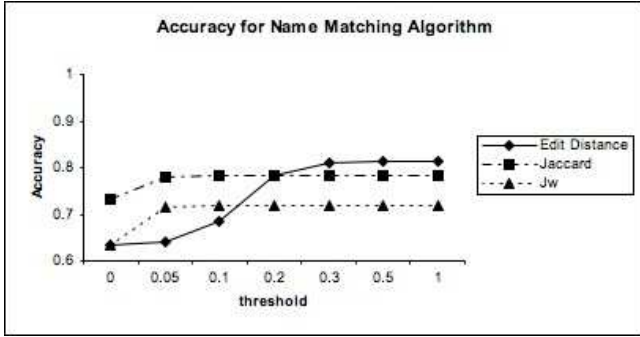
Here,  $edit_{i,j}$ ,  $jacc_{i,j}$ ,  $jaro_{i,j}$  correspond to the Levenstein edit, Jaccard, Jaro-Winkler distance between the names for location  $l_i$  and  $l_j$  respectively.  $SpatialSym_{i,j}$  corresponds to the coordinate similarity.  $TypeSym_{i,j}$  is equal to the similarity of the location types as described earlier.

The final similarity metric is learned using the training data. For our experiments, we use a variety of classifiers including logistic regression, voted perceptron (Neural Network) and support vector machines (SVMs) using SMO [5] for learning the similarity function. Each of these classifier models the similarity measure with varying complexity. For example, a simple logistic regression tries to fit a hyper-plane on the training data. A more complex function may be able to model the training data well, but might not perform well on the test data (i.e. over-fitting). On the other hand, a function like logistic regression might be too simple to model the data correctly.

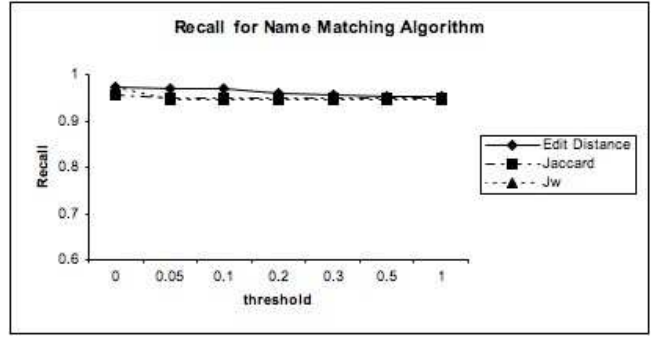
## 6. EXPERIMENTS AND RESULTS

### 6.1 Datasets

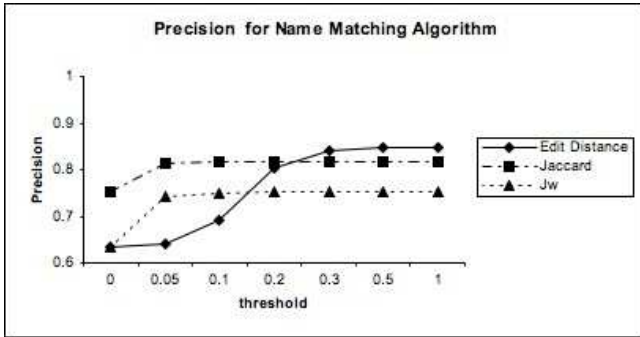
Two real-world datasets were used for the experiments reported below. The first dataset  $A$ , labeled 'NGA', represents all names of locations in Afghanistan taken from the Geographic Names Database of the United States Board on Geographic Names maintained by the National Geospatial-Intelligence Agency. The Geographic Names Database is the official standard for spelling of foreign place names for use throughout the United States Government. The second database  $B$ , labeled 'PCGN', was prepared by the United Kingdom Permanent Committee on Geographic Names. It contains 2,096 records of locations in Helmand Province, Afghanistan.



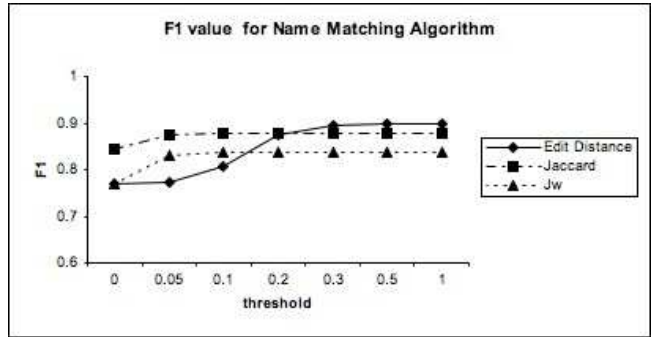
(a) Accuracy



(b) Recall



(c) Precision



(d) F1

**Figure 2: Performance measure for different name matching algorithms (a) Accuracy (b) Recall (c) Precision (d) F1**

Each location has a name (in the BGN-style Romanization format), spatial coordinates (latitude and longitude) and location type. The location types are semantically similar but use different terminologies across datasets. A location type in one dataset can correspond to more than one location type in the other dataset. In total, dataset *B* has 49 location types, while dataset *A* has just 9 location types. The location types in dataset *B* are more specific and thus give richer information about the region; examples of location types are cemetery, airfield, etc. On the other hand, dataset *A* uses general location types which classify locations into broad categories like hydrographic, hypsographic, underwater and so on. The ground truth data covering the actual mappings between *A* and *B* is prepared by the US BGN and UK Permanent Committee on Geographic Names, and it covers most of the locations in dataset *A*. The ground-truth data contains a total of 2006 pairs of matching locations. Table 1 provides a summary of the datasets used.

## 6.2 Evaluation

Each entity resolution approach as defined earlier produces a list of location pairs predicted as matches or non-matches. Accuracy, which is defined as the percent of location pairs correctly predicted, is one method for evaluating approaches. The quality of results is also measured by comparing two standard performance measures, recall and precision.

Recall is defined as the proportion of positive matches which are correctly identified:

$$recall = \frac{\text{Positive instances predicted}}{\text{Total positive instances}}$$

Precision is defined as ratio between the number of correct matches predicted to the total number of matches predicted:

$$precision = \frac{\text{True positive instances predicted}}{\text{Total instances predicted}}$$

One can increase recall by increasing the number of location pairs predicted as matches, by relaxing the threshold criteria. But this often decreases the precision of the results. In general, there is an inverse relationship between recall and precision. An ideal learning model has both high recall and high precision. Sometimes recall and precision are combined together into a single number called F1. F1 is defined as a harmonic mean of recall and precision:

$$F1 = \frac{2 \times recall \times precision}{recall + precision}$$

## 6.3 Name Matching

In this first set of experiments, we found matches for locations based on their name similarity. Since dataset *B* was much smaller and covered a subset of the region described by *A* (as dataset *A* covered the entire Afghanistan region),

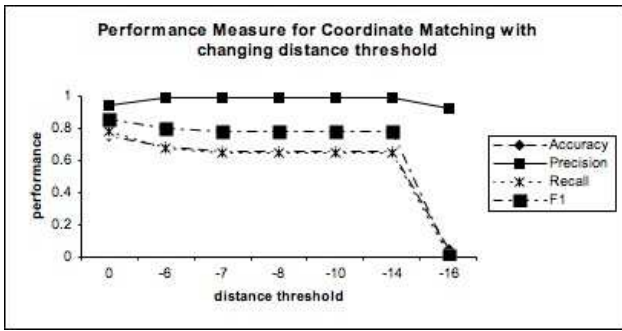


Figure 3: Performance for spatial component matching with changing distance threshold.

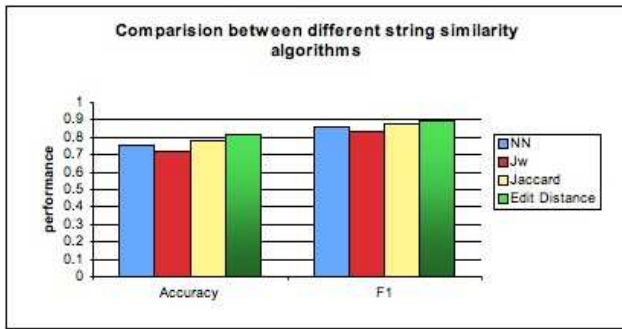


Figure 4: Comparison between algorithms using Edit, Jaccard, Jw for name matching and NN for coordinate matching.

we mapped every location in  $B$  to a location with the most “similar” name in  $A$ . Here similarity is defined using a variety of string distance algorithms such as Levenstein Edit distance, Jaccard similarity, and Jaro-Winkler similarity as explained earlier. Figure 2(a) compares the performance for these algorithms in terms of accuracy. Figures 2(b), 2(c) and 2(d) compare the recall, precision and F1-values respectively. Levenstein Edit distance is found to perform the best both in terms accuracy and F1-value. It can be also seen that as the threshold increases (i.e., tightening the similarity criteria), the recall decreases while the precision increases. The highest F1-value is achieved at a similarity threshold of around 0.5 which corresponds to just 50% similarity. This indicates that many matching locations have significant differences in their names. Intuitively, location name is a weak feature (as the names for matching locations tend to vary a lot across datasets) and it may be beneficial to use it in conjunction with other features.

## 6.4 Coordinate Matching

The next set of experiments use only the spatial component for finding matches. In this case, a location in  $B$  was mapped to the nearest location in  $A$  if the distance between them is below a threshold. The threshold is varied to find the best performance for this algorithm. Figure 3 shows the performance for this method. The x-axis is in logarithmic

Table 2: Change in performance by integrating spatial and non-spatial components. Here, duplicates are found using location names while enforcing the coordinate similarity above a threshold. Below, we threshold the  $\log$  of coordinate distance (inverse of coordinate similarity) below a threshold. Notice, the improvement in precision with stricter coordinate distance threshold. On the other hand, recall reduces.

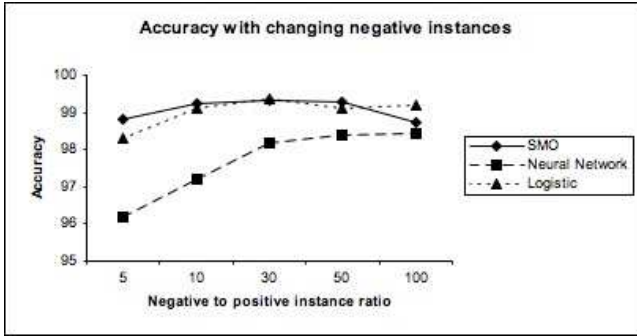
THRESHOLD	ACCURACY	PRECISION	RECALL	F1
0	0.817	0.955	0.848	0.899
-1	0.818	0.957	0.847	0.899
-2	<b>0.820</b>	0.961	0.847	<b>0.900</b>
-5	0.813	0.991	0.812	0.893
-10	0.773	0.991	0.770	0.867
-16	0.585	0.999	0.567	0.724

scale as the actual coordinate distance tends to get arbitrarily small. The performance remains constant for most of the threshold values but tapers in the end when the similarity criteria is made stricter. Figure 4 compares the results with the best results achieved from name matching. One can see that the best name matching algorithm (i.e., edit distance) performs better than the coordinate matching algorithm.

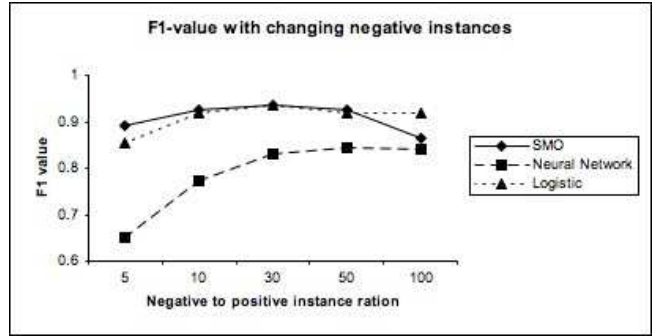
## 6.5 Integrating Spatial and Non-Spatial Components

As a first step towards integrating spatial and non-spatial components, we use both location name and spatial coordinates together. We start with the simple method, where two locations  $l_i \in A$  and  $l_j \in B$  are defined as matches, if the name of  $l_i$  is most similar to  $l_j$  as compared to all locations in  $A$  and the coordinate distance is below a threshold. The distance is measured using the logarithmic scale. Table 2 shows the performance improvement of this method with changing threshold. Both the accuracy and the F1-value increases with decreasing distance threshold (or stricter distance similarity criteria) reaching a maximum at around  $exp^{-2}$ , after which they again taper down. The results indicate that using spatial and non-spatial component together leads to better performance.

A more sophisticated similarity metric is learned using a variety of classifiers including logistic regression (Logistic), support vector machines (SMO) and voted perceptron (Neural Network). We perform two sets of experiments, one where only location name and coordinates are integrated, while in the second, location type is also integrated. Figures 5(a) and 5(b) compare the performance of these learning methods using location name and coordinates with changing ratio between negative and positive instances. As seen from the graph, logistic regression is found to perform slightly better than support vector machines, while the neural network is found to perform the worst. The best performance for both logistic regression and support vector machines is achieved when the ratio between the number of negative and positive training examples is 30:1. Logistic regression is found to be the most robust against changing proportions of negative examples. While the performance of support vector machines reduces considerably after the proportion is increased beyond 30, the performance of logistic regression remains mostly constant. This makes logistic regression

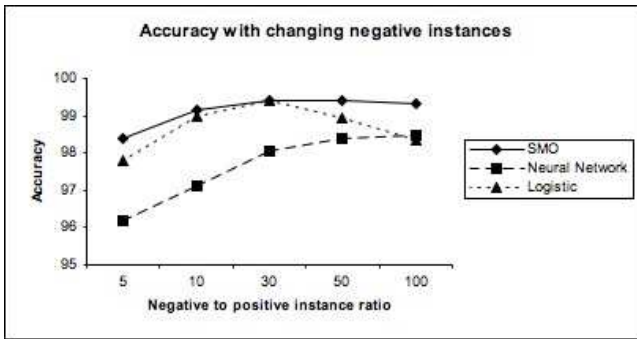


(a) Accuracy

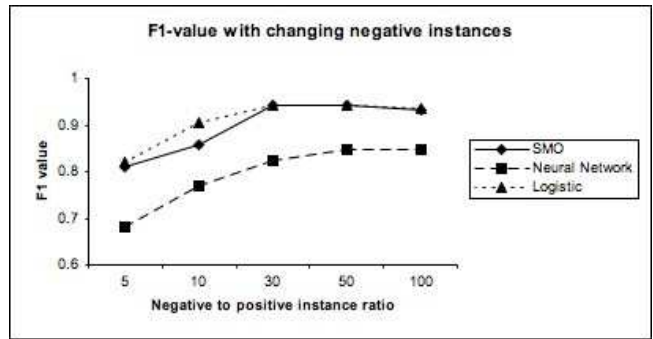


(b) F1

**Figure 5: Comparison of performance between algorithms used for integration with changing ratio between negative and positive instances. The algorithms used all the features except location type. Figure (a) compares accuracy while (b) compares F1 measure.**



(a) Accuracy



(b) F1

**Figure 6: Comparison of performance between algorithms used for integration with changing ratio between negative and positive instances. The algorithms used all the features including name, coordinates and location types. Figure (a) compares accuracy while (b) compares F1 measure.**

scalable to larger datasets while using location name and coordinates as features. The performance for the neural network on the other hand is found to monotonically increase with increasing proportion of negative instances.

Figure 6(a) and 6(b) compare the performance of these training methods using all features with changing ratio between negative and positive instances. Here again, logistic regression is found to perform the best with a ratio of 30:1 between negative and positive training examples. The support vector machine is found to perform comparable to logistic regression, while the neural network gives the worst performance. In this case, the support vector machine was found to be more robust with changing negative samples ratio.

Table 3 compares the best performance between only-name (NAME), only-coordinates (COORD), name and coordinates (LOGNL), and name, coordinates and type (LOGNLT). The training and test data are generated keeping ratio between the negative and positive instances as 30 (as it was found to show the best performance). Note that

the performance measures for only-name and only-coordinate methods were also computed using the same test data. The results show that our approach outperforms the best performances obtained for name and spatial features independently. There is a marked increase in both the recall and precision value. Specifically, there is a significant increase in the recall from 0.4 to 0.921. As expected, our approach also outperforms in terms of F1 value which is a combination of recall and precision. In terms of accuracy too, our approach was found to be more promising. Though both only-name (NAME) and only-coordinates (COORD) performed with a high accuracy (between 96-98%), it is important to note that even an all-negative classifier (which classifies all test instances as negative) will achieve a performance of 96.7% accuracy for our test data (which contains negative instances in the excess ratio of 30:1). This suggests that accuracy is perhaps not the best indicator of the performance in this setup. To conclude, our method using a machine learning algorithms to weight all the three features is found to perform the best, indicating that using spatial and non-spatial

**Table 3: Comparison performance between the best method using only location name (NAME), only coordinates (COORD), location name and coordinates (LOGNL), and location name, coordinates and location type (LOGNLT). The negative instance ratio of 30 was shown to give the best performance. From this figure one can see that as more features are integrated the performance increases both in terms of accuracy and f1-value.**

STATS	NAME	COORD	LOGNL	LOGNLT
ACCURACY	0.964	0.983	0.9935	0.994
PRECISION	0.79	0.95	0.966	0.962
RECALL	0.4	0.88	0.907	0.921
F1	0.531	0.91	0.936	0.941

features together renders better performance.

## 7. FUTURE WORK

In our experiments, we found that Levenstein edit distance was the best similarity metric for comparing location names. As mentioned earlier, it would be interesting to make our name similarity measures language specific. Currently, we give equal cost for each operation corresponding to addition, deletion or change of a literal. For our Afghanistan dataset, we found the names of the locations are in Arabic and language specific information may be used to improve performance. For examples, some river names had “wadi” as suffix or prefix. “Al” was also found a very common prefix. For a better string similarity measure, one might consider ignoring these prefixes or give zero cost for their addition or deletion. It would be interesting to learn language or region dependencies directly from the training data.

There are also opportunities for improving the current entity resolution method by including more semantic information and constructing more sophisticated similarity measures. Making use of additional semantic information in comparing location types is one interesting possibility. Or, a more complex similarity measure can be defined between two locations references using their proximity to an “influential” site, like airport, capital city, etc. We plan to explore these directions in future work.

## 8. CONCLUSION

In this paper, we introduced methods for resolving location references across datasets using both spatial and non-spatial features. We performed an extensive evaluation using spatial and non-spatial components independently and showed how using combined information provides more accurate results. The matching is complicated by the presence of features such as location type which has different vocabularies for different data sources. We propose ways to combine these features by learning a classifier from a sample of resolved locations. Unfortunately, the process of building a classifier using these features is hindered by skew in the training data which contains many more negative examples than positive examples. We have developed an approach to effectively select negative examples which maximizes the performance of the classifier. We show how using this combination of techniques we can construct entity resolution algorithms with both high precision and high recall.

## Acknowledgements

This work was supported by the National Geospatial Agency and the National Science Foundation under NSF #0423845 and NSF #0438866.

## 9. REFERENCES

- [1] C. Beeri, Y. Kanza, E. Safra, and Y. Sagiv. Object fusion in geographic information systems. In *International Conference on Very Large Data Bases*, 2004.
- [2] Indrajit Bhattacharya and Lise Getoor. Iterative record linkage for cleaning and integration. In *SIGMOD Workshop on Data Mining and Knowledge Discovery*, 2004.
- [3] Ching-Chien Chen, Craig A. Knoblock, Cyrus Shahabi, Yao-Yi Chiang, and Snehal Thakkar. Automatically and accurately conflating orthoimagery and street maps. In *Proceedings of the 12th ACM International Symposium on Advances in Geographic Information Systems (ACM-GIS'04)*, 2004.
- [4] Y. Doytsher and S. Filin. The detection of corresponding objects in a linear-based map conflation. *Surveying and Land Information Systems*, 60(2):117–128, 2000.
- [5] G. W. Flake and S. Lawrence. Efficient SVM regression training with SMO. In *Machine Learning*, volume 46, pages 271–290, 2001.
- [6] S. Lawrence, K. Bollacker, and C. L. Giles. Autonomous citation matching. In *Proceedings of the Third International Conference on Autonomous Agents*, 1999.
- [7] X. Li, P. Morie, and D. Roth. Semantic integration in text: From ambiguous names to identifiable entities. *AI Magazine Special Issue on Semantic Integration*, 2005.
- [8] H. Mayer. Automatic object extraction from aerial imagery—a survey focusing on buildings. *Computer Vision and Image Understanding*, 74(2), 1999.
- [9] A. McCallum and B. Wellner. Conditional models of identity uncertainty with application to noun conference. In *Neural Information Processing Systems Conference*, 2004.
- [10] Martin Michalowski, Jos Luis Ambite, Snehal Thakkar, Rattapoom Tuchinda, Craig A. Knoblock, and Steve Minton. Retrieving and semantically integrating heterogeneous data from the web. In *IEEE Intelligent Systems*, 19(3), 2004.
- [11] M. Minami. Using arcmap, 2000. Environmental Systems Research Institute, Inc.
- [12] H. Newcombe, J. Kennedy, S. Axford, and A. James. Automatic linkage of vital records. *Science*, 130:954–959, 1959.
- [13] Parag and P. Domingos. Multi-relational record linkage. In *ACM SIGKDD Workshop on Multi-Relational Data Mining*, 2004.
- [14] A. Saafeld. Conflation-automated map compilation. *International Journal of Geographical Information Systems*, 2(3):217–228, 1988.
- [15] A. Samal, S. Seth, and K. Cueto. A feature based approach to conflation of geospatial sources. *International Journal of Geographical Information*



*Systems*, 18(00):1—31, 2004.

- [16] W. E. Winkler. Methods for record linkage and bayesian networks. Technical report, Statistical Research Division, U.S. Census Bureau, Washington, DC, 2002.