GEM: An Efficient Entity Matching Framework for Geospatial Data

Setu Shah
Arizona State University
Tempe, Arizona, USA
snshah12@asu.edu

Vamsi Meduri
Arizona State University
Tempe, Arizona, USA
vmeduri@asu.edu

Mohamed Sarwat
Arizona State University
Tempe, Arizona, USA
msarwat@asu.edu

ABSTRACT

Identifying various mentions of the same real-world locations is known as spatial entity matching. *GEM* is an end-to-end Geospatial EM framework that matches *polygon* geometry entities in addition to *point* geometry type. Blocking, feature vector creation, and classification are the core steps of our system. *GEM* comprises of an efficient and lightweight blocking technique, *GeoPrune*, that uses the geohash encoding mechanism. We re-purpose the spatial proximality operators from Apache Sedona to create semantically rich spatial feature vectors. The classification step in *GEM* is a pluggable component, which consumes a unique feature vector and determines whether the geolocations match or not. We conduct experiments with three classifiers upon multiple large-scale geospatial datasets consisting of both spatial and relational attributes. *GEM* achieves an F-measure of 1.0 for a *point* × *point* dataset with 176k total pairs, which is 42% higher than a state-of-the-art spatial EM baseline. It achieves F-measures of 0.966 and 0.993 for the *point* × *polygon* dataset with 302M total pairs, and the *polygon* × *polygon* dataset with 16M total pairs respectively.

CCS CONCEPTS

• **Information systems** → **Entity resolution**; **Geographic information systems**.

KEYWORDS

spatial entity matching, spatial blocking, geohash, Apache Sedona

ACM Reference Format:
Setu Shah, Vamsi Meduri, and Mohamed Sarwat. 2021. GEM: An Efficient Entity Matching Framework for Geospatial Data. In *29th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '21), November 2–5, 2021, Beijing, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3474717.3483973

1 INTRODUCTION

As geospatial data flows in from different sources, various hurdles arise such as data inconsistency, data redundancy, discrepancy between old and new data and much more. To make it useful, such data needs to be curated before being passed to applications. Entity matching (EM) is an integration technique with a goal to match different mentions of the same real-world entity [10] across diverse data sources.

Relational EM is a fervent area of research and relational EM systems leverage string similarity functions to derive numerical feature vectors for a textual record pair. Magellan [8] is an end-to-end relational EM system and DeepMatcher [12] applies deep learning techniques for relational EM. We feed Magellan and DeepMatcher with pre-aligned spatial data and compare their performance against our system *GEM*. We implement Magellan as an end-to-end system with its default or best working options - Overlap blocker and Random Forest Classifiers as highlighted by Konda et al. [8]. Whereas Mudgal et al. [12] engages complex neural networks to derive suitable representation for the long text attributes. The major limitation while adapting relational EM to spatial data is that spatial attributes are treated as plain strings, resulting in loss of non-trivial coordinate information. This can lead to substandard performance which we empirically validate in Section 3.

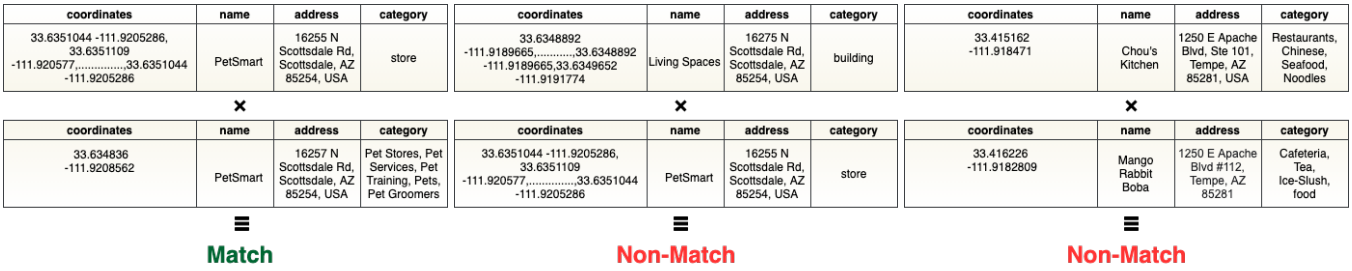
Spatial EM is the task of determining whether the given spatial entities map to the same geolocation [7]. Unlike relational EM, we cannot simply treat spatial coordinates (i.e., latitude and longitude) as strings in spatial EM. Doing so will result in a significant loss of semantic information and poor matching performance. Some state-of-the-art spatial EM systems [6, 7, 11] only match a *point* to a *point*, which falls short in catering to diverse geometric data types of geospatial data. Martins [9] has performed EM across different spatial data types (i.e. points and polygons) by considering simple distance heuristics and functions based on area overlap to determine the similarity across spatial geometries. These distances need to be normalized and are much less advanced than the 7 spatial proximality operators we use in *GEM*.

QuadSky [7] is an end-to-end spatial EM system with support for blocking, feature vector creation and matching. Although, it only matches *point* geometries and hence to adapt it for *polygon* matching, we reduce the polygon to a point i.e., its *centroid*. Doing so leads to poor EM performance which we will show in Section 3. Note that we could not compare *GEM* to GeoBench [11] or GeoAlign [6] due to their restrictive web interfaces and lack of source code. In summary, we note that the relational EM systems need non-trivial adaptation towards solving the spatial EM problem. Although some of the existing works on spatial EM can tackle both relational and spatial attributes, they miss out on catering to the diverse spatial geometry which includes polygons. Moreover the system that works with points and polygons employes a very naive metrics of distance and area overlap for decision making.

Problem Definition: Consider two spatial datasets S_{left} and S_{right}. Tuples in these datasets are well-defined spatial entities that have both spatial and relational (non-spatial) attributes. The EM

SIGSPATIAL '21, November 2–5, 2021, Beijing, China

ACM ISBN 978-1-4503-8664-7/21/11…$15.00
https://doi.org/10.1145/3474717.3483973



(a) Point × Polygon **(b) Polygon × Polygon** **(c) Point × Point**

Figure 1: Illustrative examples of Spatial Entity Matching across diverse geometries

Approaches	Support for spatial data				
	Spatial Blocking	Different geometry support	Feature generation	Training	Inference
Magellan[8]	No	N/A	No	No	No
Deepmatcher[12]	No	N/A	No	No	No
GeoBench[11]	Yes	Point	Yes	N/A	No
GeoAlign[6]	No	Point	Yes	N/A	No
QuadSky[7]	Yes	Point	Yes	N/A	Yes

Table 1: State-of-the-art approaches

task here is to *match* spatial entities from S_{left} to that of S_{right}. These datasets have pre-aligned schema and can have spatial data of either geometry type: *point* or *polygon*.

Our system provides support for three entity matching scenarios: *point*×*point*, *point*×*polygon*, *polygon*×*polygon*. Figure 1 depicts the three scenarios and provides an example for each case. Consider an entity pair $s_1 \in S_{left}$ and $s_2 \in S_{right}$. Each pair contains one spatial and three relational attributes. s_1 and s_2 in fig 1a are classified as a *match* based on the *coordinates* and the similarity between *name and address* . In contrast, the pair in Figure 1b is marked as a *non-match* because all the attributes are different. Even though *address* and *category* are similar in Figure 1c, the entity pair is classified as a *non-match* because of different *name* and *coordinates*. In this paper, we introduce an end-to-end Geospatial Entity Matching system called *GEM*, which can handle all these three EM scenarios. Our blocking mechanism, *GeoPrune*, is similar to how Isaj et al. [7] re-purposes the QuadTree algorithm for blocking, but we leverage a much lighter geohash encoding technique [2]. *GEM* leverages Apache Sedona [1] which is a scalable geospatial data processing engine to create spatial feature dimensions by re-purposing 7 spatial join operators. These 7 spatial proximality operators provide rich semantics about the geometry types using Boolean dimensions which capture containment, equality, intersection, and distance. We utilize the Simmetrics library [5] to create the relational feature dimensions.

GEM encodes the information about both relational similarity and spatial proximality between a spatial entity pair which is converted into a numerical feature vector, that is passed to a binary classifier to generate a match or non-match label for the entity pair. Following are our contributions in this paper:

- *GEM* that can provide seamless support for three spatial EM case.
- Spatial blocking mechanism, *GeoPrune*, that uses the geohash encoding technique.
- We build feature vectors at scale using Sedona and Simmetrics library.
- For classification, we provide 3 pluggable binary ML classifiers - Random decision forests (RF), Support Vector Machines (SVM), and feed-forward neural network (NN).

We will now discuss the system architecture of *GEM* , followed by experimental results.

2 OUR SOLUTION: GEM

The architecture of *GEM* consists of preprocessing, followed by spatial blocking, feature vector creation and lastly classification. Figure 2 gives an overview of the system architecture. The preprocessing step performs a sanity check to ensure that the entity pairs do not have unaligned attributes or null spatial coordinates. We require that the spatial coordinates of each entity pair are non-null, although we allow the non-spatial attributes to have null values. We generate coordinate information for datasets that do not originally have spatial coordinates using the 'googlemaps' package [3]. We label the pairs in the post blocking set in a semi-manual fashion to create the ground truth for datasets that do not have gold labels. We manually generate simple Boolean DNF (Disjunctive Normal Form) rules for matching through trial-and-error method on various samples of the data. These are approximate rules as they are based on manual examination of several samples, which is why we spent extensive manual effort to verify and correct the mis-labelled pairs.

Spatial Blocking: *GeoPrune*, the spatial blocking step prunes away the obviously non-matching pairs from the pool of Cartesian product pairs. It consists of two steps - geohash computation and blocking. Geohash indexing is a flexible and efficient way of encoding spatial coordinates into a 12 character string. The mechanism of *GeoPrune* is similar to prefix based similarity computation in strings. Instead of textual feature strings, here we use geohash codes. Consider a spatial entity pair (s_1, s_2) with geohash codes G_{s_1} and G_{s_2} respectively. We determine a granularity, k, up until which we will be matching the geohash codes of a given pair. For instance, if *k=5* then, every spatial entity pair whose initial 5 out of 12 characters in the geohash codes are same, will qualify to the post-blocking candidate set. For example, (G_{s_1}, G_{s_2}): (**9tbqh**gn36wp7, **9tbqh**1ma6xz5) will qualify, in contrast to another pair (G_{s_1}, G_{s_2}): (**9tbqh**gn36wp7, **9tbq4**1ma6xz5) whose prefixes are different.

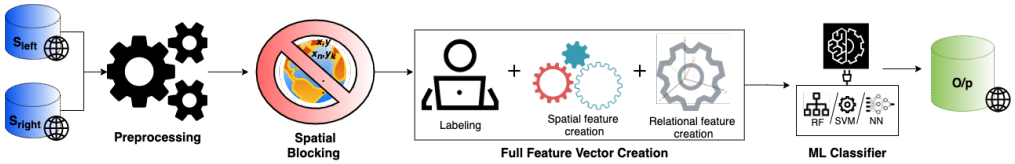


Figure 2: System architecture overview of *GEM*

Dataset		#left	#right	#total pairs	#post-blocking
Point-Point	Fodors-Zagats (FZ)	533	331	176.4k	1314
	Zomato1-Yelp1 (R1)	3013	5882	17.7M	3063
	Zomato2-Yelp2 (R2)	7689	4055	31.1M	6864
	Yelp-Yellow Pages (R3)	9947	28787	286.3M	30912
Polygon-Point	Yelp-OSM	4979	60803	302.7M	229377
Polygon-Polygon	AZ-Maricopa	4979	3357	16.7M	35864

Table 2: Details of the spatial datasets

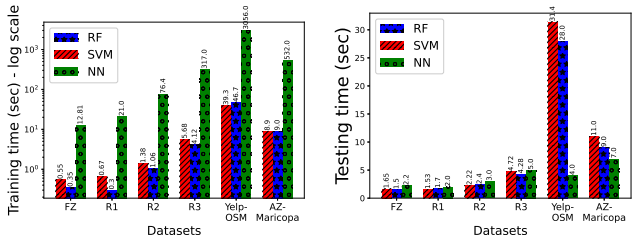
The smaller prefix blocks subsume the larger prefix blocks, i.e. as you increase the granularity measure, k, the size of the post blocking set will become smaller and smaller. This granularity measure, k, is different for different datasets. The discussion about a precise setting for k is deferred to section 3. To the best of our knowledge, we are the first to re-purpose geohash for spatial blocking. Computing the geohash code is fairly straightforward for *points*. In the case of *polygons*, we compute the geohash code of its *centroid*. If the centroids of two polygons are reasonably far, this can be detected by *GeoPrune*, that prunes away such polygon pairs.

Feature Vector Creation: We apply Apache Sedona's seven spatial SQL operators individually to each geometry pair in order to derive their spatial proximity. They provide advanced semantics like containment, equality, intersection, and distance about the spatial entity pair. We put a '1' if the proximal operator evaluates to TRUE and '0' if it evaluates to FALSE. The relational attribute similarity evaluation is done by computing 21 string similarity scores like Jaccard similarity, Cosine similarity, Levenshtein distance and more using the Simmetrics [5] library and normalizing the values to lie between 0 and 1. Since spatial information is vital in matching locations, we replicate the seven spatial feature dimensions to create a ratio of 60:40 for spatial:relational dimensions in the final feature vector. We use simple NN that can benefit by oversampling as it enables quick model training and convergence, making the system time efficient.

Classification: We offer three types of classifiers that are easily pluggable into *GEM*. We borrow the implementations of the classifiers' supervised variant for linear classifier: Support Vector Machine (SVM), tree-based classifier: Random Decision Forests (RF), and non-linear classifier: Neural Nets (NN) from Meduri et al. [10]. They all take in the same unique feature vector generated in the preceding step. The NN is a simple architecture with sigmoid activation, L2 loss function and is trained for 100 epochs. SVM is implemented as a cost-sensitive classifier that can handle the class skew well to produce accurate results.

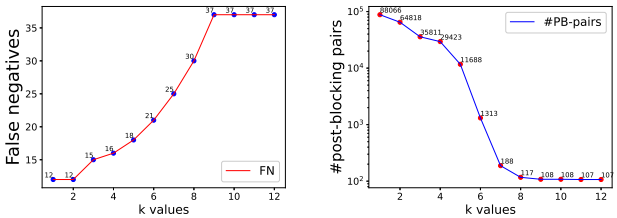
3 EXPERIMENTAL EVALUATION

We ran our experiments on an Intel Xeon E5-2687WV4 CPU (12 cores, 3.0 GHz per core) machine with 100 GB RAM and a 4 TB hard drive. We used Apache Sedona 0.1.0 along with Apache Spark 3.0.1 and Apache Hadoop 2.7.2.



(a) Training latency **(b) Testing latency**

Figure 3: Latency comparisons of the three classifiers across all the datasets



(a) # False Negatives **(b) # Post-blocking Pairs**

Figure 4: Evaluation of RF classifier for various values of k in FZ dataset

As shown in Table 2 we have four restaurant datasets for the *point* × *point*, and one each for the *point* × *polygon* and *polygon* × *polygon* cases. We borrowed R1, R2, and R3 from Konda et al. [8]'s data repository and FZ from [4]. None of these four datasets originally had location coordinates; so we auto-generated them from the given address. The Yelp-OSM dataset contains polygon coordinates and point coordinates for various business establishments in Arizona. Whereas the AZ-Maricopa contains polygon coordinates for buildings in the entire Arizona and Maricopa County in Arizona. While FZ dataset has its own ground truth [4], we determined the ground truth for other datasets. We maintain the spatial equivalence (distribution of tuple pairs in each region) across the 80% train and 20% test sets. Next we discuss the performance of various classifiers, evaluate our GeoPrune blocking mechanism w.r.t. variation in k (the spatial blocking threshold), and we compare *GEM* against the baselines.

Comparison of Classifiers: The training time for RF and SVM are almost the same, while NN takes more time. Figure 3 shows the time taken by each classifier for training and testing on different datasets. Table 3 contains the values of test precision, recall and F1-measure for all datasets over the three classifiers. We observe that F1-measure for the *point* × *point* datasets over all the classifiers

Datasets		Random Forest			SVM			Neural Network		
		Precison	Recall	F1-Measure	Precision	Recall	F1-Measure	Precision	Recall	F1-Measure
Point-Point	FZ	1	1	**1**	1	1	**1**	1	1	**1**
	R1	0.967	0.967	0.967	0.948	1	**0.973**	0.88	1	0.93
	R2	0.99	0.993	**0.992**	0.973	0.993	0.983	0.932	1	0.965
	R3	0.983	0.989	**0.986**	0.969	0.997	0.983	0.908	0.997	0.95
Point-Polygon	Yelp-OSM	0.96	0.972	**0.966**	0.906	0.987	0.944	0.691	1	0.817
Polygon-Polygon	AZ-Marizopa	0.997	0.99	**0.993**	0.971	0.983	0.977	0.901	0.99	0.95

Table 3: Test performance evaluation of *GEM* across various classifiers and datasets

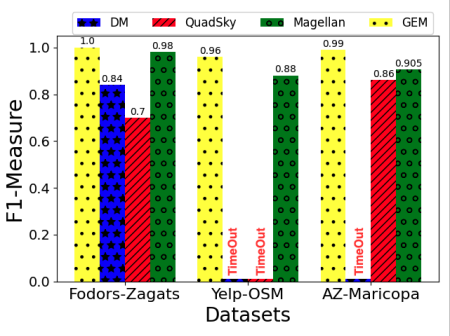


Figure 5: Comparison of *GEM* with baselines across 3 spatial datasets with different geometries

≥ 0.95. Moreover both RF and SVM are highly accurate across all three EM scenarios. RF works with a committee of 20 relatively uncorrelated decision trees that captures the nonlinear dependencies among attributes and can handle non-separable cases effectively. SVM's accurate performance indicates that the data has very less noise possibly due to effective blocking procedure. Even though the NN we used has a simple architecture, it produces competitive scores for most of the datasets and outperforms the more involved NN like Mudgal et al. [12].

Evaluation of blocking: Random Forest classifier proves to be accurate and efficient for majority of the datasets. Hence to study the best blocking threshold k for the FZ dataset we will fix RF as the classifier. We use 112 matches provided with the dataset [4] while labelling the ground truth. Figures 4a and 4b respectively shows the number of FNs and number of post-blocking pairs for all values of k (1-12). As it can be observed, with increase in the value of k the #FNs also increases which leads to decrease in Recall. While the #post-blocking pairs decrease with the increase in k which implies that low values of blocking threshold k will lead to high training and test latencies. Hence we empirically deduce that the most optimal blocking threshold for the FZ dataset is k=6, which produces test F1-measure of 1.0 (see table 3).

Comparison with baselines: While *GEM* performs the best on *point* × *point* dataset, Konda et al. [8], Mudgal et al. [12] and Isaj et al. [7] have an F1-score of 0.983, 0.84 and 0.70 respectively for the FZ dataset. Although, these systems are not able to sustain their performance for the other two spatial EM scenarios. We implement and compare the baselines as end-to-end EM systems. While Magellan [8] is able to scale for all the Cartesian product pairs, due to its coarse post-blocking set and treating spatial attributes as strings, it scores a F1-measure of only 0.88 and 0.905 for the Yelp-OSM and AZ-Maricopa datasets respectively. DeepMatcher [12] being a complex neural network classifier is not able to scale for either *point* × *polygon* or *polygon* × *polygon* datasets, which is illustrated by 'TimeOut' in Figure 5. QuadSky [7], provides support only for the *point* data type and it suffers due to the approximation of a *polygon* to its centroid (point). Doing so in the crucial step of feature vector creation results in significant information loss and compromised F1-scores. Hence for the AZ-Maricopa dataset, QuadSky produces and F1-score of 0.86 while *GEM* performs 15% better with an F1-score of 0.99. The spatial EM system is not able to scale for the Yelp-OSM dataset. *GEM* provides the best F1-scores of 1, 0.966 and 0.993 for the three datasets respectively.

4 CONCLUSION AND FUTURE WORK

GEM achieved F1-scores of 1, 0.96 and 0.99 for FZ, Yelp-OSM and AZ-Maricopa datasets respectively, emphasizing that the system providing native support for diverse geometry types can outperform geometry-agnostic spatial EM baselines. Possible directions for future work include comparing *GeoPrune* to relational blocking methods and evaluating the performance difference of *GEM* with and without oversampling geospatial feature dimensions.

REFERENCES

[1] Apache [n. d.]. *Apache Sedona.* Apache. https://sedona.apache.org/
[2] pubnub [n. d.]. *Geohash.* pubnub. https://www.pubnub.com/learn/glossary/what-is-geohashing/
[3] Python [n. d.]. *Python Client for Google Maps Services.* Python. https://pypi.org/project/googlemaps/
[4] [n. d.]. *Restaurant Dataset.* https://www.cs.utexas.edu/users/ml/riddle/data.html
[5] [n. d.]. *SimMetrics Java Library.* https://github.com/Simmetrics/simmetrics
[6] Nelly Barret, Fabien Duchateau, Franck Favetta, and Ludovic Moncla. 2019. Spatial Entity Matching with GeoAlign (demo paper). In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems.* 580–583.
[7] Suela Isaj, Esteban Zimányi, and Torben Bach Pedersen. 2019. Multi-Source Spatial Entity Linkage. In *Proceedings of the 16th International Symposium on Spatial and Temporal Databases.* 1–10.
[8] Pradap Konda, Sanjib Das, Paul Suganthan GC, AnHai Doan, Adel Ardalan, Jeffrey R Ballard, Han Li, Fatemah Panahi, Haojun Zhang, Jeff Naughton, et al. 2016. Magellan: Toward building entity matching management systems. *Proceedings of the VLDB Endowment* 9, 12 (2016), 1197–1208.
[9] Bruno Martins. 2011. A supervised machine learning approach for duplicate detection over gazetteer records. In *International Conference on GeoSpatial Sematics.* Springer, 34–51.
[10] Venkata Vamsikrishna Meduri, Lucian Popa, Prithviraj Sen, and Mohamed Sarwat. 2020. A Comprehensive Benchmark Framework for Active Learning Methods in Entity Matching. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data.* 1133–1147.
[11] Anthony Morana, Thomas Morel, Bilal Berjawi, and Fabien Duchateau. 2014. Geobench: a geospatial integration tool for building a spatial entity matching benchmark. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems.* 533–536.
[12] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data.* 19–34.