A Supervised Skyline-Based Algorithm for Spatial Entity Linkage

Suela Isaj
Aalborg University
Aalborg, Denmark
suela@cs.aau.dk

Vassilis Kaffes
University of the Peloponnese
Tripoli, Greece
vkaff@uop.gr

Torben Bach Pedersen
Aalborg University
Aalborg, Denmark
tbp@cs.aau.dk

Giorgos Giannopoulos
IMSI/Athena R. C.
Athens, Greece
giann@athenarc.gr

ABSTRACT

The ease of publishing data on the web has contributed to larger and more diverse types of data. Entities that refer to a physical place and are characterized by a location and different attributes are named *spatial entities*. Even though the amount of spatial entity data from multiple sources keeps increasing, facilitating the development of richer, more accurate and more comprehensive geospatial applications and services, there is unavoidable redundancy and ambiguity. We address the problem of spatial entity linkage with SkylineExplore-Trained (*SkyEx-T*), a skyline-based algorithm that can label an entity pair as being the same physical entity or not. We introduce LinkGeoML-eXtended (*LGM-X*), a meta-similarity function that computes similarity features specifically tailored to the specificities of spatial entities. The skylines of *SkyEx-T* are created using a preference function, which ranks the pairs based on the likelihood of referring to the same entity. We propose deriving the preference function using a tiny training set (down to 0.05% of the dataset). Additionally, we provide a theoretical guarantee for the cut-off that can best separate the classes, and we show experimentally that it results in a near-optimal F-measure (on average only 2% loss). *SkyEx-T* yields an F-measure of 0.71-0.74 and beats the existing non-skyline-based baselines with a margin of 0.11-0.39 in F-measure. When compared to machine learning techniques, *SkyEx-T* is able to achieve a similar accuracy (sometimes slightly better one in very small training sets) and more importantly, having no-parameters to tune and a model that is already explainable (no need for further actions to achieve explainability).

1 INTRODUCTION

Web data is growing continuously in size and heterogeneity, providing rich and diverse information about different entities. Nowadays, we can rely on different sources for the same information, making the process of obtaining data more transparent and source-independent. Some of this web data is connected to a location, like a geographical point, or an address, thus, referring to a physical spatial entity. A spatial entity, apart from pointing to a location, is also characterized by an identity, which is constructed by a set of attributes, such as the name, the type, the phone number, reviews of people, etc. For example, “Restaurant Ambiance" is a spatial entity, located at (55.6,7.9), with the phone +4511111111 and the tags “restaurant" and “cosy". Different and independent sources can provide information about the same spatial entity. However, this independence of the sources and the ease of publishing data has also brought redundancy and sometimes even ambiguity. Several records of the same spatial entity might exist in the same source or across different sources. Continuing the previous example, another source might contain similar information about “Restaurant Ambiance", but now located at (55.7,7.8), with a different phone +4522222222 and the tags “restaurant" and “classy". The process of deriving which records belong to the same real-world physical spatial entity is known as *spatial entity linkage*.

The problem of spatial entity linkage is important for the research fields that work with spatial entities such as influence maximization in geo-social graphs, geo-recommender systems, trajectory pattern mining, and for real world applications such as social networks, logistics, geomarketing, cadastral management, tourism, and leisure. Linked spatial entities contain a richer and integrated representation of the real entity. Additionally, spatial entity linkage can also improve the quality of the data for the industries that use spatial entities as the main input for their activities, such as marketing companies, tax offices, etc. Hence, spatial entity linkage is a problem that involves multiple stakeholders in different fields. Unfortunately, research on spatial entity linkage has not progressed at the same rate as the entity resolution research. Many papers focus on entity linkage of human entities [18, 19, 21, 25, 30, 65], while spatial entity linkage has only been superficially addressed, usually contributing only a tool and leaving the decisions to be made by the user, rather than providing an automatic algorithm [6, 34, 42]. These papers give few details about the accuracy of their methods and make arbitrary decisions to choose thresholds and scoring functions. Only a handful of the recent papers in the field [2, 24, 29, 31, 32] provide solid progress on the accuracy of the models. Isaj et al. [29, 31] use preference functions to form skylines and rank the pairs based on the likelihood of referring to the same physical entity, but they choose them in an ad-hoc fashion. On the other hand, [2, 24, 32] propose domain-specific similarity functions and exploit them directly or within machine learning algorithms to solve the task; however, they do not investigate the explainability of the proposed models. In the present paper, combining and extending two different approaches (skyline-based and machine learning), we propose a supervised skyline-based algorithm *SkyEx-T* that exploits domain-specific similarity features, can be successfully trained even on a very small training dataset and produces accurate and easily explainable results. We make the following contributions:

- We propose a supervised skyline-based algorithm *SkyEx-T* that can learn a preference function and a cut-off using a very small training set (down to 0.05% of the full dataset).
- We provide a theoretical guarantee for the cut-off selection in *SkyEx-T*, and we show experimentally that the selected cut-off yields a near-optimal F-measure.
- We incorporate domain knowledge via the domain-specific LGM-X (inter-Linking Geo-spatial entities using Machine learning - eXtended) similarity-based features, increasing the effectiveness of *SkyEx-T*.
- We evaluate *SkyEx-T* on two real-world dataset of spatial entities originating from four different sources and achieve an F-measure that significantly beats the existing baselines (0.11-0.39 margin).

SkyEx-T only shares the ranking of pairs as a common concept with our previous approaches [28, 29, 31], while the underlying algorithms are different. First, *SkyEx-T* is *supervised* and trains a preference function. Second, the selection of the cut-off is entirely novel, differently from *SkyEx-F* [31], which did an exhaustive search for the threshold or *SkyEx-D* [29], which is density-based. Third, the preference function is more expressive, given that the features can be prioritized. Finally, *SkyEx-T* is much more efficient, which results into reducing the runtime from hours to a few seconds or minutes.

We study the related work in Section 2, formulate the problem of spatial entity linkage in Section 3, describe the main components of our solution, introduce *LGM-X*, and propose *SkyEx-T* in Section 4, provide the experimental results in Section 5, and finally, conclude and point to future work in Section 6.

2 RELATED WORK

The entity linkage problem is addressed widely in several papers, across various domains [12, 19, 25, 30, 36, 38, 59]. There are several terms related to this problem, such as data integration, entity resolution, deduplication, etc. The entity linkage process consists of three main steps: *blocking*, *pairwise comparison*, and *pair labeling*. Our work is focused on pairwise comparison and the pair labeling.

Entity resolution. A vast amount of research in entity resolution focuses on humans. Textual attributes [48], profile photos [18], network of friends [36, 39], publications [37, 65, 66], social media posts [25, 30] are used to distinguish the pairs of entities that might refer to the same individual. The spatial entity resolution problem shares only the basics with the human entity resolution problem in terms of comparing attributes such as name, keywords, etc., while the recent advances of the entity resolution research focus entirely on human networks, human activities, temporal and spatial traces of humans, etc. Consequently, they cannot apply to spatial entities.

Spatial data integration. In contrast to the general entity resolution problem, *spatial data integration* specifically matches and merges spatial objects. Spatial objects are fully determined by their geographical coordinates (e.g. maps data) while a spatial entity, besides being geo-located, is identified by a combination of attributes (name, phone, categories, reviews of visitors, address, rating, etc). There are several papers that integrate spatial objects which come from sensors and radars [1, 4, 62, 64] but these solutions are not applicable to spatial entities since they by considering only the geographical information, often end up merging records of different entities into one. Schafers et al [57] produce an integrated view of road segments by matching them in terms of the shape and length and the name of the street if available, but this solution works only roads. Safra et al [54] propose location-based joins for spatial data and focus entirely on the location of the objects. While they discuss the existence of semantics for these spatial objects (hotels, shopping malls), they do not deal with other attributes than locations.

Spatial entity blocking. There are several blocking techniques that use textual attributes to group similar entities [20, 45, 46, 60], but few considering spatial entities and their geographical coordinates [6, 29, 31, 42]. In most of the works, the spatial entities are grouped using user-defined distance thresholds. For example, in the work of Karam et al. [34], the spatial entities that are at most 5 m apart are grouped. Morana et al. [42] defines a threshold based on the type of spatial entities, e.g., 50 m for restaurants, but 500 m for parks. Following the same idea, the authors of [29, 31] propose an algorithm, *QuadFlex*, that is inspired from a quadtree and can adapt the radius based on the density of the spatial entities in an area. In this way, the algorithm will be more restrictive for spatial entities in the city center, setting a smaller radius than spatial entities in the countryside. Spatial blocking techniques are not in the scope of this work, so we use the state of the art *QuadFlex* approach [29, 31] to create our pairs.

Spatial entity pairwise comparison. There is research on spatial data integration, focusing on integrating spatial objects, which, in contrast to spatial entities, are identified only by their geographic coordinates and sometimes their shape [1, 4, 57, 62]. Spatial entities might share the same spatial object while still belonging to different entities, e.g., a restaurant and a hairdresser are located in the same geographical point, but on different floors of the same building. As a result, we need to compare the attribute values of spatial entities in order to decide whether they belong to the same physical entity or not. The more similar two spatial entities are, the more likely they refer to the same physical spatial entity. To measure the similarity, the attribute values of the entities are compared, and similarity features are calculated. For name similarities, one could use Levenshtein distance, Jaccard similarity, cosine similarity [25, 30], or more advanced metrics such as a Soft-TFIDF measure combined with Levenshtein similarity [43].

String similarity based comparison of spatial entities is used by several works in the literature emphasize mainly on the textual aspect of spatial entities to perform linkage, under the assumption of e.g. lack [24] or unreliability [16] of other attributes, including spatial ones. Santos et al. [56] present a thorough overview of 13 different string similarity functions for pairwise comparing pairs of toponyms and deciding whether they correspond to the same entity and perform an extended comparison of these functions on the accuracy achieved. Additionally, the computed similarity scores are used as training features in state-of-the-art supervised machine learning algorithms for classification. However, these metrics are general and not designed explicitly for spatial entities as discussed next.

The metrics proposed in Davis et al. [17] and Deniz et al. [35] are specifically designed for the name attributes of spatial entities that mostly correspond to variations of the procedures used for generic name matching. Davis et al. [17] present a hybrid, three-stages method, i.e., the DAS similarity measure, that combines features from token-based and edit-based approaches. Deniz et al. [35] propose a four-stage process that takes into account accentuation and other language-specific aspects of spatial entities names. Recchia et al. [50] evaluates the set of algorithms presented in Christen et al. [11] on place names listed in the GEOnet

Names Server, that contains romanized entity names from 11 different countries. Based on their study, no similarity measure achieves the highest accuracy in all datasets. A different approach is followed in the problem of business places deduplication by Delvi et al. [16]. In the proposed solution, authors identify words of higher significance (core terms) that use to build a name model. This model is properly combined with a spatial context model using an unsupervised learning algorithm. Although the above methods attempt to incorporate the specifics of spatial entities name attributes, they do not achieve state-of-the-art accuracy results in the entity matching problem.

Deep learning methods for name entity matching are also being proposed in the literature. Santos et al. [55] present a method, based on Siamese RNNs, for addressing the task of entity matching. Their method yields better accuracy results than traditional classifiers on similarity-based training features. Alexis et al. [3] propose an Attention-based network model and a hybrid scheme model that combines individual machine and deep learning approaches and achieves the highest accuracy reported results on the Geonames toponym dataset. However, these methods require large amounts of data to properly engineer and train deep network model architectures, which does not apply in our setting.

In our recent works [24, 32], we propose the LGM-Sim meta-similarity function, i.e. a series of processing and matching steps that can be applied on top of any baseline similarity function, that incorporates domain knowledge for the toponym interlinking problem. We demonstrate that applying our method on top of several baseline similarity functions improves the interlinking effectiveness by a large extent. Moreover, we utilize training features derived from LGM-Sim within classifiers, showing a further increase in accuracy. These works of ours, as well as the baseline work of [56], which we compare against and demonstrate large increase in interlinking accuracy, *consider exclusively name similarity features* and *ignore the spatial features*. Thus, in the current work, we reutilize the LGM-Sim meta-similarity as component of our method, extending its application both in the names and the addresses of POIs.

Spatial entity pair labeling. After having pairwise similarities and features, the pairs need to be labeled as positive if they belong to the same physical entity or as negative, otherwise. Sehgal et al. [58] stand between spatial data integration and spatial entity integration because their entities have names, geographical coordinates, and types but they refer to spatial objects representing landscapes (lakes, hills, rivers, etc.). They used supervised learning to calculate the similarity of the spatial entity/object types and another classifier that learns the weights of each similarity in a training set and assigns the class accordingly. Similarly, Karam et al. [34] and Berjawi et al. [6] assign weights to the similarity scores of the attributes: Berjawi et al. [6] arbitrarily constructs a similarity score based on an average of the attribute similarities, while Karam et al. [34] use the coefficient $\frac{2}{3}$ for the name, the coordinates and the type of the entity and the coefficient $\frac{1}{3}$ for the website, the address and the phone number. Morana et al. [42] uses textual and semantic attributes and takes the labeling decision based on belief theory [49]. Isaj et al. [28, 29, 31] propose a skyline-based labeling method that instead of assigning weights, construct a preference function using the Pareto dominance concept and separate the classes by defining the number of skylines. However, their proposed algorithms need the number of skylines k as a parameter, which has to be found either through exhaustive experiments, or by a computationally heavy unsupervised technique. Moreover, the preference function is chosen heuristically, without having a procedure in place that would choose the best preference function for the underlying dataset. *SkyEx-T* improves this significantly because it is supervised, learns the preference function and the cut-off from a very small training set, and provides guarantees for applying the learned model on the test set. Furthermore, all the research mentioned above [6, 29, 31, 34, 42, 58] uses general similarity metrics, sometimes with slight tuning [58], which are not very effective on spatial entities.

Summary. While we can observe significant advances in the field of entity resolution, the spatial entity linkage has only been explicitly addressed in very few papers. The research in spatial entity pairwise comparison has proposed effective spatial metrics for measuring the similarity between entities [16, 35, 55] and the current state of the art [2, 24, 32] achieves high accuracy without the need to train deep networks. In contrast to the current spatial entity linkage works, our proposed solution *SkyEx-T* uses *LGM-X* features, which extend the state of the art LGM-Sim features used for toponym interlinking [24, 32]. Moreover, for spatial pair labeling, instead of using weights like in [6, 34, 58], we use the Pareto operator similarly to Isaj et al. [28, 29, 31]. However, differently from [28, 29, 31], besides the Pareto operator, we can also prioritize features, and the preference function is not chosen heuristically; instead, we propose a *supervised* algorithm for selecting the preference selection and the cut-off using very little labeled data.

3 PROBLEM DEFINITION

In this section, we introduce definitions and formulate the problem of Spatial Entity Linkage. We define a spatial entity as:

Definition 3.1. A spatial entity s is a uniquely identified entity, located in a geographical point with coordinates $\langle long, lat \rangle$, and described by a set of attributes $A = \{a_1, a_2, ..., a_n\}$ with values $\{v^{a_1}, v^{a_2}, ..., v^{a_n}\}$.

Example 3.2. An example of a spatial entity is a restaurant located in a point $\langle 55.7, 8.9 \rangle$ with the name "Restaurant Amelie", with the phone number of +4511111111, the address "Vestergade 23", and categories such as {food, coffee, cosy}. In this case, the name, the phone number, the address, and the categories are the attributes a_1, a_2, a_3, and a_4, respectively. Similarly, the values of the attributes are v^{a_1} = "Restaurant Amelie", v^{a_2} = +45111111, and v^{a_3} = {food, coffee, cosy}.

To discover which spatial entities belong to the same physical entity, we need to compare them. The spatial entities can be compared to each other in an exhaustive way (Cartesian product), or, to reduce the number of comparisons, they can be grouped in blocks based on one or more attributes [20, 31, 45, 46, 60]. Within the blocks, the spatial entities are compared pairwise with respect to their attribute values. From the pairwise comparison, we obtain *features*, defined as follows:

Definition 3.3. A feature X^{a_n} ($A \times A \mapsto [0, 1]$) is a function that computes the similarity between the values $v_i^{a_n}$ and $v_j^{a_n}$ of the attribute a_n for the spatial entities s_i and s_j, respectively.

Note that from the definition, there can be more than one computed features for the same attribute, for example:

Example 3.4. Let us consider two spatial entities s_1 and s_2 named "Amelie" and "Ami", respectively. Let a_1 be the name attribute. Thus, we have $v_1^{a_1}$ = "Amelie" and $v_2^{a_1}$ = "Ami". We can

compute several similarities on the name attribute, e.g., Jaccard, cosine, Jaro-Winker similarity. Let $X_1^{a_1}$, $X_2^{a_1}$, and $X_3^{a_1}$ be the Jaccard, cosine, and Jaro-Winker computed similarities of $v_1^{a_1}$ and $v_2^{a_1}$. Therefore, we have $X_1^{a_1} = 0.6$, $X_2^{a_1} = 0.6123724$, and $X_3^{a_1} = 0.8333333$.

For two spatial entities, we compute N features for their n attributes and obtain a featured pair of spatial entities:

Definition 3.5. A featured pair of spatial entities $\langle s_i, s_j, X\rangle$ is formed by two spatial entities s_i and s_j and their features $X = \{X_1^{a_1}, X_2^{a_1}, X_3^{a_2}...., X_N^{a_n}\}$, where the N features are computed for their respective pairs of attribute values $\{\langle v_i^{a_1}, v_j^{a_1}\rangle, \langle v_i^{a_2}, v_j^{a_2}\rangle, ..., \langle v_i^{a_n}, v_j^{a_n}\rangle\}$ for the attributes $\{a_1, a_2, ..., a_n\}$.

Note that we need methods to compare the different attributes of the spatial entities. In the next sections, we will discuss similarity metrics and machine-learning techniques to compare the spatial entities (the features). The intuition behind the comparison is that the more similar two spatial entities are regarding their features, the more likely it is that they refer to the same physical spatial entity. Thus, we want to assign a *label* to each pair; a positive label if the entities of the pair have high values of similarities and are the same physical spatial entity, and a negative label if they are different physical spatial entities.

Definition 3.6. A labeled pair of spatial entities $\langle s_i, s_j, X, C_{ij}\rangle$ is formed by a featured pair of spatial entities and their class C_{ij}. The class C_{ij} takes the value 1 ($C_{ij} = 1$) when the pair $\langle s_i, s_j\rangle$ is likely to refer to the same physical spatial entity, or 0 ($C_{ij} = 0$), otherwise.

Problem definition: *For any given pair of spatial entities $\langle s_i, s_j\rangle$, the Spatial Entity Linkage problem determines whether s_i and s_j refer to the same physical spatial entity or not, meaning that $C_{ij} = 1$ or $C_{ij} = 0$, respectively.*

In other words, the Spatial Entity Linkage problem aims to transform unlabeled pairs of spatial entities into labeled ones. In the next sections, we introduce our solution that computes the features of the pairs and labels them based on the similarity of the spatial entities.

4 SKYEX-T WITH LGM-X

In this section, we introduce our approach to solve the spatial entity linkage problem, *SkyEx-T* with *LGM-X* features, which uses domain-specific features for expressing the similarity between spatial entities and a skyline-based algorithm trained on a tiny dataset for labeling the pairs of spatial entities.

4.1 Overview of SkyEx-T with LGM-X

We briefly introduce the main components and the workflow of our proposed approach (Fig. 1). We start with a set of pairs of spatial entities. In order to obtain this set of pairs, we can either construct the full Cartesian product of our spatial entities, or use some of the proposed blocking techniques to reduce the number of candidates. Since the construction of this set of candidate pairs is orthogonal to our method, we use the blocking technique as in Isaj et at. [29, 31]. We apply *LGM-X meta-similarity function* (detailed in Section 4.2) to produce domain-specific features, which better captures the characteristics of spatial entity names (Step 1 in Fig. 1). *LGM-X* produces featured pairs of spatial entities. We start by reducing the dimensionality of the featured pairs, with respect to the distinct features they contain (Step 2 in Fig. 1 and detailed in Section 4.3.1). This process aims to keep only a subset of (most informative with regards to the spatial entities' similarity) features for each pair. From this step we obtain featured pairs but with fewer features. Then, in Step 3, we learn the preference function p (Sections 4.3.2 and 4.3.3) and the cut-off c_t (Section 4.3.4) by training *SkyEx-T*, which are needed to label the pairs. Finally, in Step 4, we use the preference function p to put the pairs in the test set into skylines, and the cut-off c_t to separate the ranked pairs and assign them into a class (Section 4.3.5). This final step produces the labeled pairs.

4.2 Feature extraction with LGM-X

In this section, we present the *LGM-X* (inter-Linking Geo-spatial entities using Machine learning - eXtended), our proposed training features to feed the *SkyEx-T* algorithm for better capturing and exploiting the different attributes of spatial entities and their geographical coordinates. In the following subsections, we present a summary of the LGM-Sim [24] algorithm, a meta-similarity function that aims to capture meaningful entity characteristics and incorporate the specifics of spatial entities attributes related to their name and address. By meta-similarity, we mean a series of processing and matching steps that can be applied on top of any baseline similarity function (e.g. Levenshtein, Jaro-Winkler) and produce a similarity score between two strings. Consequently, we discuss the training features we introduce, i.e. features derived from the proposed meta-similarity and additional entity specific features, that enhance the process of similarity matching.

4.2.1 ***LGM-Sim Overview.*** LGM-Sim aims at properly splitting the strings of compared attributes of the spatial entities into discrete lists of terms, with each list containing terms of different semantics. Each of these lists are assigned individual weighting scores to create an ensemble for deciding whether two entities refer to the same physical entity or not. This approach has the advantage that pays more attention to entity terms that are of greater significance in discovering the correct correlation among the examined entities; correspondingly, less significant terms contribute less to the final decision, i.e., assigned a small weight score, or not at all, e.g. frequent terms like cafe, park or church.

LGM-Sim takes as input the strings of the two name attributes and first pre-processes them in order to remove accentuation and punctuation marks. Next, a process decides whether the terms within the two strings should be alphanumerically sorted. Afterwards, the algorithm initiates the process of splitting the initial strings into three separate lists of terms each, i.e., (i) two *base* lists, (ii) two *mismatch* lists and (iii) two *frequent terms* lists. It firsts identifies frequent terms within the two strings and moves them to the respective lists, i.e., the *frequent terms* lists. This way, it isolates the least significant terms that are the least important in determining whether the two strings refer to the same entity or not. The other two types of lists will contain terms that potentially are of higher significance in determining whether two strings describe the same entity. Specifically, the *base* lists will contain terms from the two strings that (loosely) match to each other and the *mismatch* lists will contain the rest of the terms that failed to match through the two strings. Finally, for each type of list, individual similarity scores are computed which are differently weighted in order to produce the final similarity score of the two spatial entities. All the parameters of LGM-Sim, i.e., comparison thresholds and individual score weighting, are automatically learned on a training dataset. Additionally, the set

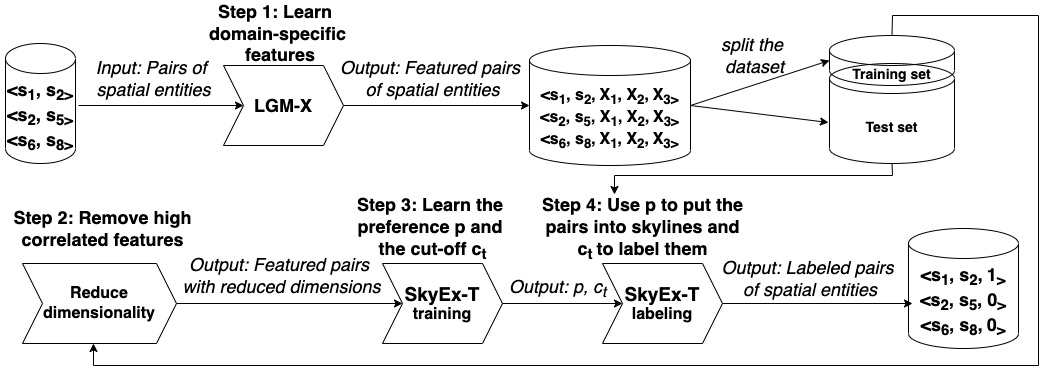

Figure 1: Overview of SkyEx-T with LGM-X

of frequent terms are automatically gathered by the corpus of spatial entities contained in this training dataset.

4.2.2 ***LGM-X Features.*** In this work, we adopt the set of training features presented in Santos et al. [56] and Giannopoulos et al. [24] and further extend it with entity-specific defined features, that concern numerical and spatial aspects of the compared entities. Specifically, we encode several generic similarity measures, like Damerau-Levenshtein, Jaro-Winkler, Jaccard N-Grams etc., as features for the classifiers. Moreover, we consider a set of training features where the custom sorting function of LGM-Sim is performed on the utilized set of generic similarity measures. This function decides whether to sort the terms of each string alphanumerically or not, before computing the similarity score of the entity names. Further, we generate a corresponding set of features by applying LGM-Sim on top of the basic generic similarity measures. Additionally, we consider the three individual similarity scores computed on the three types of individual lists that LGM-Sim splits the two entity names into. Note that the above set of features are applied to the name and address (only the text part of it) attributes of spatial entities. In addition to these, we define features that concern numerical and spatial aspects of the compared entities. Specifically, the numerical distance of the address numbers of the compared spatial entities form one integer feature, whereas the Euclidean distance of their geographical coordinates form one float feature. Eventually, we consider six groups of training features (Table 1): (i) the *basic* similarity features as presented in Santos et al. [56]; (ii) the *sorted* similarity features; (iii) the *LGM-Sim* based similarity features; (iv) the individual scores on the split of the attributes produced by applying LGM-Sim based Damerau-Levenshtein similarity on respective attributes; (v) the numerical feature on address number; and (vi) the spatial feature on geographical coordinates. For the latter two feature types, we consider normalized distances between the numerical values for a POI pais, which can be straightforwardly handled also as a similarity score, in line with the rest of the string similarity features. Note that if an attribute is missing for one or both entities in a given pair, the corresponding LGM-X features would return 0, given that there is no measurable similarity for that specific attribute in the given pair.

We note that, in the current setting, no (hyper)parameter tuning or feature engineering efforts are required for producing the aforementioned features: the parameters (weights, thresholds) of LGM-Sim have been learned in our previous work [24], on a toponyms dataset (Geonames), so LGM-Sim is used in the current dataset "as is"[1]. Further, applying a series of baseline similarity functions on the names and the addresses of POI pairs of our setting is also straightforward. Thus features (i), (ii) and (iii) above can directly be produced without any tuning or feature engineering effort. Similarly, producing distance scores on the street number and the geographical coordinates is also straightforward.

Table 1: The LGM-X training features for spatial entities

Feature Type \ Attribute	Name	Address		Coords
		name	no	
Basic similarities	14	14		
Sorted similarities	13	13		
LGM-Sim based sims.	13	13		
Individual sim. scores	3	3		
Numerical			1	
Spatial				1

4.3 Pair labeling with SkyEx-T

In this section, we present *SkyEx-T* (Skyline Explore - Trained), our skyline-based algorithm which labels the candidate entity pairs as positives when there is a high likelihood that they belong to the same spatial entity, and negative, otherwise. In the following subsections, we will detail how *SkyEx-T* learns the preference function p in a small training set, selects a cut-off ratio c_t, and then ranks the data based on the preference p and separates the classes according to c_t.

4.3.1 ***Reducing the dimensionality.*** Each pair is described as a set of N *LGM-X* features $\{X_1^{a_1}, X_2^{a_1}, X_3^{a_2}, ..., X_N^{a_n}\}$ that are computed on the values of the attributes $\{a_1, a_2, ..., a_n\}$ as described in Section 4.2.2. Note that the *LGM-X* features always return a value for every pair (see above). To have a simpler notation, from now on, we will not use the superscript of the attribute in the features but will simply write $\{X_1, X_2, X_3, ..., X_N\}$. Some of these features are highly correlated, given that they measure the similarity of the same attributes, e.g., measuring Jaccard and cosine similarity on the names of the spatial entities. To remove highly correlated features, we use the *mutual information measure* (MI) [15] to detect the dependency between two variables. MI can detect different relationships, and it is not limited

[1] Actually, we can consider the above procedure as a case of "transfer learning" where a model's hyperparameter tuning is performed on a completely different dataset (in this case, Geonames). As we can see next in the evaluation results, the LGM-Sim features also perform well in the current setting, indicating a successful reutilization of previous findings and data in a different setting

to linear correlation (like Pearson's correlation). MI measures the similarities of the joint distribution of the two variables. If $p_x(x)$ and $p_y(y)$ are the marginal probability density functions of variables x and y, respectively, and $p_{x,y}(x,y)$ is the joint probability function of both x and y, then the MI of x and y is defined as:

$$MI(x,y) = \int_x \int_y p_{x,y}(x,y) log \frac{p_{x,y}(x,y)}{p_x(x)p_y(y)} \quad (1)$$

After the features are pairwise compared using MI, we identify those pairs of features that are highly correlated. Then, we remove one from each of the highly correlated pairs; we choose to remove the feature with the largest mean correlation overall. Using the remaining features, we need to build a model to predict the class of the pairs.

4.3.2 ***Preference functions.*** We propose having a function that is expressed as a preference of feature values; the pairs that are ranked better with respect to the preference function are the pairs that are likely to refer to the same physical entity.

Definition 4.1. A preference function $p : P \mapsto P_k$ is a function that takes as input a set of pairs $P = \{\langle s_i, s_j \rangle\}$ and outputs the partially ranked (some pairs share the same rank) list of pairs $P_k = \{\langle s_i, s_j, k \rangle\}$ with respect to their feature values, where $k \in [1, m]$ is the rank of a pair $\langle s_i, s_j \rangle$, and m is the maximal rank.

The preference function ranks the pairs from the most preferred pairs to the least preferred ones, meaning that the first rank pairs are more likely to correspond to the same spatial entity than the rest. To show that a pair $\langle s_i, s_j \rangle$ is preferred over another pair $\langle s'_i, s'_j \rangle$, we will use the symbol $\succ$, so $\langle s_i, s_j \rangle \succ \langle s'_i, s'_j \rangle$. The definition of the preference is based on the definition of the skylines. We use a definition similar to [29]:

Definition 4.2. A skyline of level k is a set of pairs $Skyline(k) = \{\langle s_i, s_j \rangle\}$ ranked in the k^{th} position according to the preference function p such that for each pair $\langle s_i, s_j \rangle \in Skyline(k)$ and for each pair $\langle s'_i, s'_j \rangle \in Skyline(k')$ where $k' > k$, $\langle s_i, s_j \rangle \succ \langle s'_i, s'_j \rangle$.

Let us now further detail the components of the preference functions: *the preferred feature direction* and *the preference operators*, defined as below:

Definition 4.3. The preferred feature direction $d() : X \mapsto \mathcal{N}$ is a function that takes as input the list of values of a feature X, orders them preferring either high values ($high()$) or low values ($low()$), and outputs the rank for each feature.

$d()$ is a generic way to express the preferred feature direction, but in practice, we will use $high()$ or $low()$ based on the feature. Instead of coefficients, the preference function only specifies if we prefer high or low values of a feature. For example, for a pair to be labeled as positive (they are the same physical spatial entity), we would prefer a high Levenshtein similarity, so $high(X_{levenshtein})$. Note that we can construct a very basic preference function by using the preferred feature direction on only one feature. For example, $p = high(X_{levenshtein})$ will rank the pairs P based on their Levenshtein similarity and assign them to skylines. If we want to include more than one preferred feature direction in a preference function, we will need the preference operators. We introduce two types of preference operators: the *Pareto* operator and the *priority* operator:

Definition 4.4. The Pareto operator $\bigwedge$ is a binary operator connecting two preferred feature directions according to the Pareto optimality concept.

The Pareto optimality [8] concept is widely used in multi-criteria optimization problems. *A combination of* $(x_1, x_2, ..., x_k)$ *is Pareto optimal when there is no other combination that can increase an* x_v *without decreasing at least one* x_w. In our context, the Pareto operator prefers one solution over the rest if it is better in terms of at least one feature value, while for the remaining feature values it is either the same or better. Let us illustrate this concept with an example:

Example 4.5. Let us consider the pair of spatial entities $\langle s_1, s_2 \rangle$, which are compared with regards to their name. We have two computed features that indicate the similarity of the name: X_1=0.7 and X_2=0.3. Since a high similarity is likely to indicate a match of the pairs, we prefer high values. Hence, the preferred feature direction is $high()$ for both features. Given that we have no information on whether we should prefer one feature over another, we choose to connect these features with the Pareto operator, so $high(X_1) \bigwedge high(X_2)$. This means that for another pair $\langle s_3, s_4 \rangle$ to be preferred over $\langle s_1, s_2 \rangle$ ($\langle s_3, s_4 \rangle \succ \langle s_1, s_2 \rangle$), $\langle s_3, s_4 \rangle$ has to be better at least in one of the features values, while not being worse in the other features. For example $\{X_1 = 0.7, X_2 = 0.4\}$, $\{X_1 = 0.9, X_2 = 0.3\}$, $\{X_1 = 0.8, X_2 = 0.4\}$ would be combinations which are preferred over the $\{X_1 = 0.7, X_2 = 0.3\}$.

Besides the Pareto operator, we can also choose to prefer one feature over another. We introduce the *priority* operator defined as follows:

Definition 4.6. The priority operator $\triangleright$ is a binary operator connecting two preferred feature directions such that the feature direction on the left side of the operator is preferred over the one on the right side.

The priority operator expresses an order of the feature importance; for example, if the cosine similarity is more likely to detect matches than the Jaccard similarity, we prioritize those pairs that have high cosine similarity over those with high Jaccard similarity. Let us give an example:

Example 4.7. Let us consider the pair of spatial entities $\langle s_1, s_2 \rangle$ of the previous example, with the similarities of the name: X_1=0.7 and X_2=0.3. Let us suppose that X_2 can detect the class better than $X1$. Even though high values of both features are preferred, we would be more interested in having high values of X_2. Thus, we express the preference as $high(X_2) \triangleright high(X_1)$. This means that for another pair $\langle s_3, s_4 \rangle$ to be preferred over $\langle s_1, s_2 \rangle$ ($\langle s_3, s_4 \rangle \succ \langle s_1, s_2 \rangle$), $\langle s_3, s_4 \rangle$ has to have a higher X_2 (regardless of the values of X_1), or the same X_2 but a better X_1. For example $\{X_1 = 0.8, X_2 = 0.3\}$ and $\{X_1 = 0.6, X_2 = 0.4\}$ would be combinations which are preferred over the $\{X_1 = 0.7, X_2 = 0.3\}$.

After defining the preferred feature directions and the preference operators, we can construct different preference functions. Let us illustrate this with an example:

Example 4.8. Let us consider a pair of spatial entities $\langle s_1, s_2 \rangle$ of the previous example, with the similarities of the name: X_1=0.7 and X_2=0.3 and physical distance (in meters) from each other $X_3 = 10$. Since $X1$ and X_2 express the similarity of the name, we prefer a high value, so the preferred feature direction is $high(X_1)$ and $high(X_2)$. On the contrary, we prefer the spatial entities to be as close as possible to each other, thus, we prefer a low value on the distance, so $low(X_3)$. Let us suppose that we do not have any preferred order of X_1 and X_3, while X_2 can detect the class better than $X1$ and $X3$. In that case, we use the Pareto operator between X_1 and X_3 as in $high(X_1) \bigwedge low(X_3)$ and the priority

operator for X_2 as in $high(X_2) \triangleright (high(X_1) \bigwedge low(X_3))$. Hence, the preference function is $p = high(X_2) \triangleright (high(X_1) \bigwedge low(X_3))$.

When we have few features and some domain knowledge, we can select the preference function ourselves. However, when having many features, there are many possible ways to connect them with operators. In the next subsection, we introduce our algorithm *SkyEx-T* that automatically determines a suitable preference function.

4.3.3 ***Preference Training.*** In this section, we explain the procedure for training the preference function. In our previous work, [29, 31], the preference function was chosen heuristically, and the only operator was Pareto. Instead, in the present paper, *we add expressiveness and agility by introducing a preference function that can be trained on a very small training set*, and thus automatically learn optimal preferences for each dataset. We propose the *SkyEx-T* algorithm (Skyline Explore - Trained) for preference training which further reduces the dimensionality and selects a preference function based on the importance of each feature for classifying a pair. Then, $SkyEx-T$ ranks the pairs according to p and finds the cut-off ratio for separating the classes (this procedure will be detailed in the next section). The pseudocode of *SkyEx-T* is formalized in Algorithm 1.

To learn which remaining features are more suitable for the preference function, we calculate their correlation with the class C. C is 0 if the pairs are not a match and 1 otherwise. We want to find those features whose increase is usually associated with an increase in the class label (C changes from 0 to 1). We use Pearson's correlation because it is capable of detecting these monotonic relationships between variables ($\rho_{X_i} = \frac{cov(X_i,C)}{\sigma_{X_i}\sigma_C}$, where $cov(X_i, C)$ is the covariance of the feature X_i and the class C, and σ_{X_i} and σ_C are the standard deviations of X_i and C, respectively).

After having all ρ_{X_i} values for each X_i, we select those X_i which have high absolute values of ρ_{X_i}. We plot the absolute values of ρ_{X_i} in decreasing order and notice when $|\rho_{X_i}|$ falls considerable. This will be graphically associated with an elbow in the curve. We denote the first elbow as ε_1 and the X_i features with $|\rho_{X_i}| \leq \varepsilon_1$ as X_{ε_1}. X_{ε_1} are showing the strongest monotonic relationships to C, so their increase is closely related to C. Given that these are the most important features, they will be prioritized over the rest of the features while using the Pareto operator amongst X_{ε_1} themselves, treating them equally. In order to make the function more accurate, we can refine it further using the less important features. For that, we will again search for the next elbow in the curve, denoted as ε_2. We denote with X_{ε_2} the features with $|\rho_{X_i}| \leq \varepsilon_2$ and $\rho_{X_i} > \varepsilon_1$. X_{ε_2} will be connected by the Pareto operator themselves, while they will be connected by the priority operator with X_{ε_1}. To sum up, we have two groups of features: X_{ε_1} and X_{ε_2}. We use the Pareto operator among the X_{ε_1} and X_{ε_2} but the priority operator for X_{ε_1} over X_{ε_2}. Let us illustrate this procedure with an example:

Example 4.9. Let us suppose we have two spatial entities s_1 and s_2 with 10 features $\{X_1, X_2, ...X_{10}\}$. After computing ρ_{X_i} for each feature and ordering their absolute values in a descending order, we get these values {0.6, 0.56, 0.55, 0.54, 0.34, 0.33, 0.33, 0.32, 0.11, 0.06}, which are the $|\rho_{X_i}|$ values of $\{X_3, X_4, X_7, X_1, X_9, X_2, X_5, X_8, X_{10}, X_6\}$, respectively. Fig. 2 shows $|\rho_{X_i}|$ values ordered from the highest to the lowest. It is possible to notice the two elbows in the curve, ε_1 and ε_2. Let us suppose that the preferred direction of each feature is $high()$. Then, the preference function will be as follows: $p = (high(X_3) \bigwedge high(X_4) \bigwedge high(X_7)) \triangleright (high(X_1) \bigwedge high(X_9) \bigwedge high(X_2) \bigwedge high(X_5))$.

The above procedure is formalized in lines 1-10 in Algorithm 1. We first initialize R_X in line 1 to store all ρ_{X_i}. Then we calculate ρ_{X_i} for each feature and order R_X in lines 2-3. We graphically identify the elbows ε_1 and ε_2 in line 4. Then, we connect the features within X_{ε_1} and X_{ε_2} by the Pareto operator in lines 5-10. Finally, we prioritize X_{ε_1} over X_{ε_2} in line 11.

4.3.4 ***Skyline ranking and fixing the cut-off ratio.*** After having the preference function, we can apply it to the pairs and rank them accordingly. We use the skylines as in the work of Isaj et al. [31] to first find those pairs who are preferred the most when using preference p. Let us suppose that $Skyline(1)$ contains all the pairs $\{\langle s_i, s_j \rangle\}$ that, given preference p, $\langle s_i, s_j \rangle \succ \langle s'_i, s'_j \rangle$ for each $\langle s'_i, s'_j \rangle \in Skyline(k)$, where $k > 1$. Thus, all pairs in $Skyline(1)$ dominate the rest of the pairs. Then, we need to remove $Skyline(1)$ from the set of pairs. Applying the same preference, we choose the next set of most preferred pairs ($Skyline(2)$). We continue this procedure until there are no more pairs left.

While moving from one skyline to the next, we need to define which cut-off number of skylines best separates the classes. Given that the preferred pairs are likely to indicate a match, we will label as positive the first skylines up to the cut-off and the remaining as negative. Cutting too early will result in high precision, but low recall, while cutting too late will improve the recall, but at the cost of lower precision. Thus, we use a well-known balanced indicator, the F-measure ($F1 = \frac{2*precision*recall}{precision+recall}$). We measure the F-measure for each cut-off and fix the cut-off to k_l where the F-measure is maximized. We express this cut-off as a ratio of $\frac{k_l}{b}$ where b is the maximal level of skylines. Let us provide theoretical guarantees that a cut-off ratio found in a training set can satisfy the F-measure maximization in the whole dataset or the test set.

THEOREM 4.10. *Let us apply the preference function p on two random samples P_1 and P_2 from the pairs P. If P_1^k and P_2^k are the ranked pairs of P_1 and P_2 with regards to p, and $\mathcal{D}_1$ and $\mathcal{D}_2$ are the probability density of P_1^k and P_2^k, respectively, then $\mathcal{D}_1 \sim \mathcal{D}_2$.*

PROOF. Based on the Central Limit Theorem [53], the sample distributions will respect the population distribution, around the same mean μ as the population P but with a smaller standard deviation proportional to the size of the sample $\frac{\sigma}{\sqrt{n}}$. Let us denote by $f(k)$ the probability density function of the pairs with relation to their skyline level k. After applying p on P_1, P_2 and P, the new ranked pairs in P_1^k and P_2^k will respect their order of ranking in each sample. Note that $f(k)$ is deterministic. Let us suppose that we apply the preference p in P and we obtain the ranked

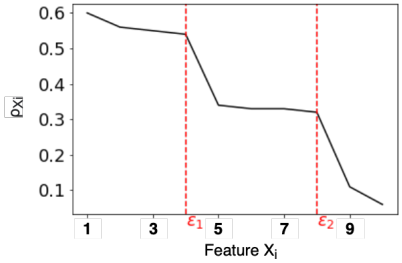

Figure 2: Finding ε_1 and ε_2

Algorithm 1 SkyEx-T training

Input: A set of labeled pairs $P_t = \{\langle s_i, s_j, C_{ij}\rangle\}$
Output: A trained preference function p and cut-off c_t

1: $R_X \leftarrow \emptyset$
2: Calculate ρ_{X_i} for each X_i and add each $|\rho_{X_i}|$ to R_X
3: Order R_X in a descending order
4: Find ε_1 and ε_2 elbows in R_X
5: **for each** X_i that $|\rho_{X_i}| \leq \varepsilon_1$ **do**
6: $\quad p_1 = d(X_1) \bigwedge d(X_2) ... \bigwedge d(X_m)$
7: **end for**
8: **for each** X_i that $|\rho_{X_i}| > \varepsilon_1$ and $|\rho_{X_i}| \leq \varepsilon_2$ **do**
9: $\quad p_2 = d(X_m + 1) \bigwedge d(X_{m+2}) ... \bigwedge d(X_n)$
10: **end for**
11: $p = p_1 \triangleright p_2$
12: $P_k \leftarrow \emptyset$
13: $F \leftarrow \emptyset$
14: **while** $|P_k| < |P|$ **do**
15: $\quad$ Find $Skyline(k) = \{\langle s_i, s_j\rangle\} \mid \forall \langle s', s''\rangle \in P_t - \{\langle s_i, s_j\rangle\}$, $\langle s_i, s_j\rangle \succ \langle s', s''\rangle\}$
16: $\quad$ Remove $Skyline(k)$ from P and add it to P_k
17: $\quad$ Label P_k as positive
18: $\quad$ Calculate $F1(k)$ and add $F1(k)$ to F
19: $\quad P_t = P_t - Skyline(k)$
20: **end while**
21: Find k_l such that $F1(k_l) = max(F1(k))\ \forall k \in \{1, |F|\}$
22: $c_t = \frac{\sum_{i=1}^{k_l} Skyline_i}{\sum_{i=1}^{max(k)} Skyline_i}$
return p, c_t

pairs P^k. For every $\langle s_1, s_2\rangle$ and $\langle s_3, s_4\rangle$ in P^k, if $\langle s_1, s_2\rangle \succ \langle s_3, s_4\rangle$, $\langle s_1, s_2\rangle \in P_1$, and $\langle s_3, s_4\rangle \in P_1$, then we have that $\langle s_1, s_2\rangle \succ \langle s_3, s_4\rangle$ in P_1^k as well. Similarly, For every $\langle s_1, s_2\rangle$ and $\langle s_3, s_4\rangle$ in P^k, if $\langle s_1, s_2\rangle \succ \langle s_3, s_4\rangle$, $\langle s_1, s_2\rangle \in P_2$, and $\langle s_3, s_4\rangle \in P_2$, then we have that $\langle s_1, s_2\rangle \succ \langle s_3, s_4\rangle$ in P_2^k. This means that $f(k)$ will simply, in a deterministic way, reorder the observations of the samples, which already have similar distributions, strictly respecting the order in the in P^k. As a result, the new probability density distributions will be similar, so $\mathcal{D}_1 \sim \mathcal{D}_2$. □

Theorem 4.11. *Let us apply the preference function p on two random samples P_1 and P_2 from P and denote by P_1^k and P_2^k their respective ranked pairs with regards to p. If k_1 is a cut-off in the probability density function $D_1 = \int_1^{b_1} f_1(k)dk$ (limits in $[1, b_1]$) of P_1^k such as the F-measure is maximal, then $\frac{k_1 b_2}{b_1}$ will also be a near-optimal cut-off for P_2^k with probability density function $D_2 = \int_1^{b_2} f_2(k)dk$ (limits in $[1, b_2]$).*

Proof. The probability density distribution for the ranked pairs P_1^k and P_2^k is $\int_1^{b_1} f_1(k)dk$ and $\int_1^{b_2} f_1(k)dk$, respectively, and the skyline levels lie in $[1, b_1]$ for P_1^k and $[1, b_2]$ for P_2^k. Chiu et al. [10] present the preferred solutions of a decision maker in relation to the mean, variance, and third moments of a distribution. In other words, a preferred cut-off in a probability distribution $\mathcal{D}_1$ is a preferred solution for a probability distribution $\mathcal{D}_2$ as long as the mean, variance, and third moments are similar. When sampling P_1 and P_2 from P, we gain a sample of pairs with a mean μ as in P [53]. After ranking, the pairs near μ will be assigned in the k_{μ_1}-th skyline for the pairs P_1^k and in k_{μ_2} skyline for the pairs P_2^k. According to Theorem 1, $\mathcal{D}_1 \sim \mathcal{D}_2$, so $k_{\mu_2} \simeq k_{\mu_1}$. The variances will be $\sigma_1 = \int_1^{b_1} (k - k_{\mu_1})^2 f(k)d(k)$ and $\sigma_2 = \int_1^{b_2} (k - k_{\mu_2})^2 f(k)d(k)$ for P_1 and P_2, respectively. Given that $k_{\mu_2} \simeq k_{\mu_1}$, we have that $\sigma_1 \simeq \sigma_2$. The third moment is an indicator of skewness and it is defined as: $m_1^3 = \frac{\int_1^{b_1} (y - k_{\mu_1})^3 dF(y)}{\sigma_1^3}$ and $m_2^3 = \frac{\int_1^{b_2} (y - k_{\mu_2})^3 dF(y)}{\sigma_2^3}$ for P_1 and P_2, respectively. Given that $k_{\mu_1} \simeq k_{\mu_2}$ and $\sigma_1 \simeq \sigma_2$, then $m_1^3 \simeq m_2^3$. The pairs in the first skylines have a high likelihood to belong to the same physical spatial entity and usually, they form a skewed long-tail distribution [29]. Let us suppose that $k = k_1$ is a cut-off in the probability density function $\int_1^{b_1} f(k)dk$ such that when labeling pairs in a skyline smaller than k_1 as positive and the rest as negative, the F-measure is maximized. In other words, k_1 is the preferred cut-off for the decision maker. According to the skewness condition [10], a similar cut-off in the probability density distribution $\int_1^{b_2} f(k)dk$ of P_2 will yield a near-optimal F-measure value. In order for k_1 to be applicable to P_2, we need to express it as a ratio $\frac{k_1}{b_1}$ and adapt it to P_2 as in $\frac{k_1 b_2}{b_1}$. □

Note that this process is stochastic, meaning that it might not always find the optimal cut-off, but it will always find the near-optimal cut-off value (we also show this experimentally in Section 5). We will learn the cut-off ratio from a training set and then, given Theorem 2, we will apply it to the test set. However, having the cut-off ratio $\frac{k_1}{b_1}$ requires that we rank all the pairs in the training set and moreover, to find $\frac{k_1 b_2}{b_1}$, we need b_2, which again will need the ranking of all the pairs in the test set. For a big dataset, this procedure will be time-consuming. Therefore, we use a simple derivative of Theorem 2, formalized in Lemma 1, where instead of a cut-off in the number of skylines, we use the percentage of the data belonging to the optimal level of skylines.

Lemma 4.12. *If $c_1 = \frac{\sum_{i=1}^{k_1} Skyline_i}{\sum_{i=1}^{b_1} Skyline_i}$ is the ratio of the total pairs in P_1^k that belong to a skyline level of at most k_1, where k_1 is a cut-off in the probability density function $D_1 = \int_1^{b_1} f_1(k)dk$ (limits in $[1, b_1]$) of P_1^k such as the F-measure is maximal, then $c_1 * |P_2^k|$ is a near-optimal cut-off for P_2^k for maximizing the F-measure.*

Proof. The cut-off k_1 in P_1^k corresponds to the skyline level which can best separate the classes. Given that the probability density distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ of P_1 and P_2 are similar (Theorem 1), for each pair $\langle s_i, s_j\rangle$ in P_1^k, the probability of belonging to the positive class is $P(k_{\langle s_i, s_j\rangle} \leq k_1) \sim \frac{\sum_{i=1}^{k_1} Skyline_i}{\sum_{i=1}^{b_1} Skyline_i}$. According to Theorem 2, the cut-off ratio $\frac{k_1}{b_1}$ is near-optimal for P_2^k in order to best separate the classes. Let us denote with $c_2 = \frac{\sum_{i=1}^{k_2} Skyline_i}{\sum_{i=1}^{b_2} Skyline_i}$ the ratio of total pairs in P_2^k that corresponds to skyline levels up to k_2. Given that Theorem 2 and $\mathcal{D}_1 \sim \mathcal{D}_2$, $\frac{\sum_{i=1}^{k_1} Skyline_i}{\sum_{i=1}^{b_1} Skyline_i} \sim \frac{\sum_{i=1}^{k_2} Skyline_i}{\sum_{i=1}^{b_2} Skyline_i}$. Thus, the ratio $c_1 \sim c_2$, so c_1 is the near-optimal for P_2^k. □

The skyline ranking is detailed in lines 12-19 in Algorithm 1. We start with an empty set of explored skylines P_k in line 12. Note that in line 13 we are initializing F, which will keep the F-measure for each cut-off. We will need F later when we learn the cut-off c_t. While there are still pairs that are not ranked, we

find the next skyline that satisfies the preference function (line 15) and remove it from the pairs P_t. We label all pairs in P_k as positive and calculate the F-measure for this cut-off in lines 17-18. Then, we remove the current skyline and continue the ranking. We find the best cut-off k_l that maximizes the F-measure in line 21. We express the cut-off as a data ratio (Lemma 1), which makes it applicable to any sample (line 22). Finally, we return the learned preference function p and the cut-off ratio c_t.

4.3.5 ***Classification of Pairs.*** In the previous subsections, we discussed the training of *SkyEx-T* (Algorithm 1) that determines the preference function, ranks the pairs, and selects the cut-off. In this subsection, we will formalize *SkyEx-T* labeling algorithm for classifying the pairs (Algorithm 2). We start with a set of unlabelled pairs P, a trained preference function p and cut-off ratio c_t from Algorithm 1. We apply the preference function p on P. We rank the pairs in P according to the preference function p and assign them to skylines (1-5). Note that we do not rank all the pairs but only the pairs in the first skylines that constitute the ratio c_t of all pairs, so a total of $c_t|P|$ pairs. We label all the ranked pairs in P_k as positive and the remaining unranked pairs as negative (lines 7-8). Finally, we return the labeled P'.

Algorithm 2 SkyEx-T labeling

Input: A set of unlabeled pairs $P = \{\langle s_i, s_j\rangle\}$, a trained preference function p and a cut-off ratio c_t
Output: A set of labeled pairs $P' = \{\langle s_i, s_j, C_{ij}\rangle\}$

1: $P_k \leftarrow \emptyset$
2: **while** $|P_k| < c_t|P|$ **do**
3: Find $Skyline(k) = \{\langle s_i, s_j\rangle\} \mid \forall\langle s', s''\rangle \in P - \{\langle s_i, s_j\rangle\}$, $\langle s_i, s_j\rangle \succ \langle s', s''\rangle\}$ based on p
4: Remove $Skyline(k)$ from P and add it to P_k
5: $P = P - Skyline(k)$
6: **end while**
7: Set $C_{ij} = 1$ (positive class) for all $\langle s_i, s_j\rangle \in P_k$ and add them to P'.
8: Set $C_{ij} = 0$ (negative class) for all $\langle s_i, s_j\rangle \in P - P_k$ and add them to P'.
return P'

5 EXPERIMENTAL RESULTS

In this section, we describe our datasets and then, report the results of our experiments with *SkyEx-T*.

5.1 Dataset

We are using two datasets of spatial entities for the experiments: *North Denmark spatial entities* (North-DK) and the *Fodor's and Zagat's restaurants* (Restaurants).

North Denmark spatial entities North Denmark dataset contains spatial entities from Denmark, originating from four different sources, Google Places (GP), Foursquare (FSQ), Yelp, and Krak[2]. We refer to as *North Denmark spatial entities* (North-DK). The spatial entities are retrieved by using the seed-driven approach described in Isaj et al. [27]. From the spatial entities we obtained (more than 120,000), we only use the 75,541 spatial entities that have at least a phone number or a website, so we can construct the ground truth from these. 51.50% of the spatial entities originate from GP, 46.22% from Krak, 0.03% from FSQ, and 2.23% from Yelp. The 75,541 retrieved spatial entities were grouped using the spatial blocking techniques [31]. After the spatial blocking, we obtained 777,446 pairs. To establish a ground truth, similarly to [29, 31], we apply a commonly-used rule for generating entity matching ground truths [25, 26, 30, 63]: if the phone number or the website is the same, then we infer that the pair refers to the same physical entity. By using this rule, we obtain 27,102 positive pairs (3.5% of the total pairs). In Table 2, we observe that most of the positive pairs are from different sources, especially from Krak and Google Places (64.2% of the total positive pairs). Even though the spatial entities are uniquely identified within a source, we can still find duplicates within the same source. 28.7% of the total positive pairs originate from same-source pairs, especially in Krak and in Google Places. Given that the phone number and the website are used for generating the ground truth, they are not used in the algorithm and in the pair comparison. The spatial entities have geographical coordinates, which are used in the spatial blocking [31] to generate the pairs, and the pairs are compared against the names and addresses using LGM-X.

Source	Krak	GP	Yelp	FSQ
Krak	3,789	17,405	902	7
GP		3,546	968	13
Yelp			460	12
FSQ				0

Table 2: The sources of the positive pairs

Fodor's and Zagat's restaurants[3] **(Restaurants)** is a dataset of 864 restaurant entities, where 61.69% are extracted from Fodor's Travel[4] and 38.31% from Zagat website[5]. The entities are characterized by the name, the address, the city, the phone number, and the type of the spatial entity. Given that the geographical coordinates are missing in the dataset, we do not perform the spatial blocking but rather use all the pairs. We obtain 372,816 pairs, 112 out of which refer to the same spatial entity (0.03% of the total pairs). These 112 pairs are already given, so we did not have to compute the ground truth for this dataset. However, the phone number attribute has been used to derive the class and form the ground truth, so we exclude the phone number from the pairwise comparison of attributes. The pairs are compared against the names and addresses using LGM-X.

5.2 SkyEx-T cut-off prediction

In this section, we evaluate the prediction of the cut-off c_t for *SkyEx-T*. The cut-off c_t is learned from the training set as in Algorithm 1. For comparison, we found the optimal cut-off c^* that would yield the highest F-measure for the learned preference function. To do so, we exhaustively tried all the cut-offs in the test set, measured the F-measure for each of them, and noted the highest value. In practice, this is not only time-consuming but even impossible since the labels are not present in the test set. By having the optimal cut-off, we can evaluate how accurately *SkyEx-T* can predict the cut-off. We repeated this procedure 10 times on disjoint training sets for each of the training sizes and report the averages in Table 3. We could not train on less than 1% of the data in Restaurants because there are only 112 positive

[2]Krak (www.krak.dk) contains businesses, organizations, companies, etc. with their attributes such as name, location, address, phone, categories, etc. Krak is part of Eniro Danmark A / S., which is responsible for The Yellow Pages
[3]https://www.cs.utexas.edu/users/ml/riddle/data.html
[4]https://www.fodors.com
[5]https://www.zagat.com

Table 3: SkyEx-T F-measure for estimated c_t vs the optimal c^* cut-off in North-DK

Training size	0.05%	0.1%	0.4%	0.8%	1%	4%	8%	12%	16%	20%	80%
SkyEx-T F-measure	0.682	0.690	0.708	0.705	0.706	0.736	0.717	0.718	0.711	0.711	0.727
SkyEx-T F-measure for c*	0.707	0.715	0.714	0.718	0.713	0.740	0.721	0.719	0.712	0.712	0.731
Difference of F-measure	0.025	0.025	0.006	0.014	0.007	0.004	0.004	**0.001**	**0.001**	**0.001**	0.005
Difference of F-measure in %	3.49%	3.46%	0.81%	1.90%	0.95%	0.48%	0.62%	**0.13%**	0.17%	**0.13%**	0.65%

Table 4: SkyEx-T F-measure vs SkyEx-T F-measure for estimated c_t vs the optimal c^* cut-off in Restaurants

Training size	1%	4%	8%	12%	16%	20%	80%
SkyEx-T F-measure	0.782	0.813	0.831	0.823	0.821	0.828	0.820
SkyEx-T F-measure for c*	0.841	0.840	0.840	0.839	0.834	0.839	0.838
Difference of F-measure	0.059	0.027	**0.009**	0.015	0.013	0.011	0.018
Difference of F-measure in %	6.99%	3.26%	**1.07%**	1.84%	1.54%	1.31%	2.18%

pairs in the whole dataset, and less than 1% of that would mean that we would not have any positive pairs on the training set. Thus, we cannot go lower that 1%.

For both datasets, *SkyEx-T* finds a cut-off that is nearly-optimal. For small training sets (0.05% and 0.1% in North-DK and 4% in Restaurants), *SkyEx-T* yields an F-measure that is around only 0.025 smaller than the optimal, corresponding to a 3.4% difference. When using 1% of the data as a training set in Restaurants, *SkyEx-T* predicts a cut-off that results in an F-measure that is 0.059 smaller than the optimal. However, considering that 1% of the training set in the Restaurants dataset contains only 1 or 2 positive pairs, the *SkyEx-T* prediction is in fact impressive. The F-measure values improve rapidly when moving to larger training sets; here, the difference between the F-measures is only 0.01, a small 0.95% difference on average, for 0.04%-8% training sets on North-DK and on average 1.44% for 8%-20% training sets on Restaurants. Furthermore, for 12%-20% of the data as training set, the loss in F-measure in the North-DK dataset is 0.001 (0.13%), which is insignificant. *SkyEx-T predicts a cut-off that is near-optimal, with a difference of only 0.01 in F-measure (1.34%) in North-DK and 0.02 (2.6%) in Restaurants.* Thus, we have experimentally validated our theoretical findings in Theorem 1, Theorem 2, and Lemma 1.

5.3 Comparison with Spatial Entity Linkage Baselines

Even though there are a good deal of papers on general entity matching and on toponym matching [3, 32, 56], only a few consider matching geo-located spatial entities [6, 34, 42]. In this section, we compare against solutions that propose the full pipeline for matching spatial entities (blocking + pairwise comparison + pair labeling). Given that Restaurants are not geo-located, the following related work could not be applied to them (no coordinates for using the blocking techniques, for calculating the Euclidean distance, etc), so we use only the North-DK dataset.

Berjawi et al. [6] compare spatial entities using Euclidean distance on the coordinates and Levenshtein similarity for the rest of the attributes (textual), and finally, add them all together. We exclude the phone number and the website from the similarities because we used them to derive the label (see above). If the total score is higher than 0.75, the authors match the pairs with high confidence. We further try different threshold values and report the best results (with the suffix *-Flex*). Berjawi et al. [6] consider two different versions: name + address + geographic coordinates (V1) and name + geographic coordinates (V2). Morana et al. [42] group together entities that share a token in the name or the category, and compare them using Euclidean distance for the geographical coordinates, Levenshtein distance for their name and address and a semantic similarity using Wordnet (Resnik) for the category. Regarding the weights, they use $\frac{1}{3}$ for address, phone, etc., and $\frac{2}{3}$ for the name, category, and geographic similarity. Finally, the top k candidates for each entity are considered for merging (we play with this parameter and report the best results). Karam et al. [34] group together entities that are 5 m apart, compare their names with Levenshtein distance, Euclidean distance for the coordinates and the categories semantically. Finally, belief theory is used to derive the label [49]. We additionally compare to our earlier paper [29]: we use the same blocking technique *QuadFlex*, and then, for labeling the pairs, we use a threshold-based (*SkyEx-F*) and an unsupervised skyline-based algorithm (*SkyEx-D*). *SkyEx-F* tries different thresholds for separating the classes (we report the best values in the table), and *SkyEx-D* separates the classes by considering the density of the pairs in the skylines. Instead, our proposed supervised *SkyEx-T* in the present paper uses the LGM-X features and learns the preference function and cut-off from a small training set.

The results are presented in Table 5. Berjawi et al.(V1)[6] yields the highest precision of 0.93 but a very low recall of 0.26, while Karam et al [34] yield the highest recall of 0.73 but a very low precision of 0.23. Berjawi et al.(V1)-Flex [6], Berjawi et al.(V2)[6], and Berjawi et al.(V2)-Flex[6] achieve a relatively good F-measure of 0.63, but are still outperformed by the versions using *QuadFlex* and *SkyEx-D* [29], *SkyEx-F* [29] and *SkyEx-T*. These three methods yield similar results, since they use the same spatial blocking technique and skylines to rank the pairs. However, there is an increase of 0.03 and 0.02 in F-measure, respectively, when using *SkyEx-T* instead of *SkyEx-D* and *SkyEx-F*.

However, besides the small improvement in F-measure (0.02-0.03 in F-measure), there are some important advantages of *SkyEx-T* over *SkyEx-D* and *SkyEx-F*. First of all, *SkyEx-F* requires the cut-off k. Given that k (the number of skylines) is not an intuitive parameter, the user might struggle to find a good threshold. Second, *SkyEx-T* uses the LGM-X features and learns a suitable preference function; thus, the improvement in F-measure when compared to *SkyEx-F* is related to the use of better features and a trained preference function. Finally, the greatest advantage of *SkyEx-T* over both *SkyEx-D* and *SkyEx-F* is its superior runtime. The runtime in the case of *SkyEx-D* and *SkyEx-F* could go up to 2 hours, while *SkyEx-T* are in the range of seconds or at most a few minutes (see Section 5.5). This is explained by the fact that *SkyEx-F* has to search for a good threshold in the whole dataset,

Table 5: Comparison with the Spatial Entity Linkage Baselines on North-DK

Approach	Prec.	Rec.	F1
Berjawi et al.(V1)[6]	**0.93**	0.26	0.41
Berjawi et al.(V1)[6]-*Flex*	0.87	0.50	0.63
Berjawi et al.(V2)[6]	0.73	0.56	0.63
Berjawi et al.(V2)[6]-*Flex*	0.73	0.56	0.63
Morana et al.[42]	0.39	0.60	0.47
Karam et al.[34]	0.23	**0.73**	0.35
QuadFlex + SkyEx-D [29]	0.85	0.62	0.71
QuadFlex with *SkyEx-F*[29]	0.87	0.60	0.72
QuadFlex + SkyEx-T	0.88	0.63	**0.74**

and *SkyEx-D* has to loop through the skylines while calculating the density between them, while *SkyEx-T* learns the optimal parameters from a very small training set.

5.4 Comparing SkyEx-T to Machine Learning Solutions

Let us now consider if and how *SkyEx-T* can be compared to general machine learning techniques. First, we note that related work has not used machine learning for the problem of spatial entity linkage, but only for toponym linking. Second, if we were to successfully apply standard machine learning techniques for spatial entity linkage, it would would require a) large amounts of labeled data, which are unavailable in practice because most of these entities originate from different (not interconnected) online sources and there is no indication of linkage (no labels); b) further tuned LGM-X features which again would require a large training set. Thus, standard machine learning techniques are infeasible in practice, while *SkyEx-T* is designed to deal with very small training sets of spatial entities and use pre-trained domain-specific features (no further tuning needed). However, we still make a comparison for completeness, but maintain that machine learning techniques cannot be considered as baselines for the spatial entity linkage problem.

We compare to Support Vector Machine (SVM) [14], Decision Trees [5], Random Forest [7], Extremely Randomized Trees (Extra Trees) [23], Extreme Gradient Boosted Trees (XGBoost) [9], Multi Layer Perceptron (MLP) [13]. We trained on 0.05%, 0.1%, 0.4%, 0.8%, 1%, 4%, 8%, 12%, 16%, and 20% of the North-DK dataset, and 1%, 4%, 8%, 12%, 16%, and 20% of the Restaurants dataset. For each training size, we repeated the experiment 10 times with disjoint training sets, and we report the averages F-measure in Table 6, *The choice of* LGM-X *features proved highly effective for all methods*, even on small training sets, which are atypical for machine learning, a finding also shown experimentally for *LGM-Sim* in [24]. *LGM-X* was used on a previously selected set of hyperparameters (tuned in on slightly different entities (toponyms) [32]) and was still able to generalize and achieve good effectiveness on various dataset sizes and configurations.

We can notice a slight advantage of *SkyEx-T* over the machine learning techniques for small training sets. *SkyEx-T* has the highest F-measure for 0.05%, 0.1%, 0.4% (together with MLP), and 4% Overall, *SkyEx-T* is in the top 3 best methods in terms of F-measure for North-DK in small training sets). For Restaurants, *SkyEx-T* is in the top 3 best methods for 1% and 4%. More importantly, *SkyEx-T* starts at a very high F-measure when most of the machine learning techniques fail Even though the training of *SkyEx-T* is significantly lighter compared to machine learning techniques (we only use the label for checking correlations), it still achieves similar F-measure values as the machine learning techniques, especially on small training sets. If we observe the difference in percentage from the maximal F-measure for each training size (Table 6), *SkyEx-T* differs on average by only 1.04% for training sets up to 20% of the data. In Restaurants (Table 7), *SkyEx-T* yields, on average, an F-measure of 4.85% less than the maximal on training sets up to 20%. Furthermore, on training sets less than 8%, this difference is only 3.63%, the second-best after ExtraTrees, while SVM, MLP and XGBoost fail when trained on 1% of the data. Additionally, it is important to note that there was no single best machine learning model that would give the best F-measure consistently for all the training sizes; apart from *SkyEx-T*, four different methods (RandomForest, ExtraTrees, XGBoost, and MLP) competed for the highest F-measure; thus, there was no obvious winner.

Additionally to the results in terms of F-measure, we can compare the methods in terms of other criteria. Considering that spatial entities are important in many business applications, the models used for spatial entity linkage need to be explainable for the end-user, especially when these decisions have a high impact on the business. Moreover, “the right to explanation" is now officially a requirement of EU’s General Data Protection Regulation (GDPR) [33]. In the case of *SkyEx-T*, the preference model is human-readable and easily explainable. An example of a preference function is: *(high(SimName)* $\bigwedge$ *high(LGM_baseScore)* $\bigwedge$ *high(SimAddress))* $\triangleright$ *(high(Sorted_Dice_bigrams)* $\bigwedge$ *high(Dice_bigrams)* $\bigwedge$ *high(Sorted_Soft_Jaccard)* $\bigwedge$ *high(LGM_Dice_bigrams)).* It is simple to understand which s_1 and s_2 matched noticing what features the model prefers overall and which features over others. As for the machine learning techniques, their complexity usually comes with a lack of explainability.

For the linear SVM, we can use the proposed probability estimates by Platt [47] to derive individual feature effects on the model outcome by perturbing one feature at a time through a range of values, whilst keeping the other features fixed. The Decision Trees model interpretation is rather simple, in theory by measuring the amount of “impurity", i.e., the reduced variance or Gini index compared to the parent node.However, in practice, this is feasible only in cases where the tree depth is rather small. In our case, the depth of Decision Trees in Restaurants was 2-3 for training sets of 0-10%, 4-6 for 10-20%, 7-9 for 80% but very large in North-DK, 28-34 for 0-10%, 30-42 for 10-20%, 40-45 for 80%, which makes the model unexplainable in practice. Nevertheless, both SVM and Decision Trees are outperformed, in terms of F-measure, in most scenarios by other algorithms, as presented in Tables 6. The interpretation gets even harder in tree-based models, e.g., Random Forest, Extra Trees and XGBoost, which consist of a large number of deep trees. Complex techniques, like Impurity Feature Importance or Conditional Permutation Feature Importance [61], are required to gain insights for the importance of the training features where each approach has problems and drawbacks and makes it difficult to apply in every case. Finally, the MLP (MultiLayer Perceptron) model is far from explainable since it is a type of a basic artificial neural network and, thus, it is treated as a black-box in most cases.

There are several works that explicitly address explainability and interpretability of machine learning techniques. [22, 40, 44, 51]. LIME [51] can explain any classifier by approximating it locally with an interpretable model. SHAP values [40] are additive feature importance methods that use conditional expectations to reach interpretability. DICE [44] propose a framework that works with the importance of necessary tradeoffs, causal implications,

Table 6: SkyEx-T versus ML Techniques on North-DK

Training size	0.05%	0.10%	0.40%	0.80%	1%	4%	8%	12%	16%	20%	80%
	F-measure										
SVM	0.655	0.653	0.683	0.692	0.694	0.708	0.713	0.715	0.718	0.719	0.723
DecisionTree	0.596	0.589	0.609	0.613	0.612	0.622	0.632	0.634	0.641	0.644	0.667
RandomForest	0.678	0.682	0.696	0.702	0.700	0.715	0.721	0.725	0.727	0.730	**0.749**
ExtraTrees	0.670	0.676	0.693	0.700	0.699	0.710	0.717	0.721	0.723	0.726	0.744
XGBoost	0.673	0.679	0.700	0.705	0.704	0.717	**0.724**	**0.728**	0.731	**0.733**	0.747
MLP	0.678	0.688	**0.708**	**0.719**	**0.709**	0.719	0.719	0.724	**0.731**	0.724	0.727
SkyEx-T	**0.682**	**0.690**	**0.708**	0.705	0.706	**0.736**	0.717	0.718	0.711	0.711	0.727
	Difference from Max F-measure in %										
SVM	3.96%	5.36%	3.53%	3.76%	2.12%	3.80%	1.52%	1.79%	1.78%	1.91%	3.47%
DecisionTree	12.61%	14.64%	13.98%	14.74%	13.68%	15.49%	12.71%	12.91%	12.31%	12.14%	10.95%
RandomForest	0.59%	1.16%	1.69%	2.36%	1.27%	2.85%	0.41%	0.41%	0.55%	0.41%	0.00%
ExtraTrees	1.76%	2.03%	2.12%	2.64%	1.41%	3.53%	0.97%	0.96%	1.09%	0.95%	0.67%
XGBoost	1.32%	1.59%	1.13%	1.95%	0.71%	2.58%	0.00%	0.00%	0.00%	0.00%	0.27%
MLP	0.59%	0.29%	0.00%	0.00%	0.00%	2.31%	0.69%	0.55%	0.00%	1.23%	2.94%
SkyEx-T	0.00%	0.00%	0.00%	1.95%	0.42%	0.00%	0.97%	1.37%	2.74%	3.00%	2.94%

Table 7: SkyEx-T versus ML Techniques on Restaurants

Training size	1%	4%	8%	12%	16%	20%	80%
	F-measure						
SVM	0.196	0.777	0.847	0.846	0.858	0.875	0.889
DecisionTree	0.818	0.798	0.796	0.810	0.831	0.816	0.875
RandomForest	0.743	0.830	0.843	0.844	0.843	0.859	0.879
ExtraTrees	**0.823**	**0.836**	**0.857**	0.853	0.860	**0.885**	0.904
XGBoost	0.000	0.724	0.823	0.827	0.847	0.870	**0.910**
MLP	0.077	0.789	0.837	**0.877**	**0.870**	0.874	0.871
SkyEx-T	0.782	0.813	0.831	0.823	0.821	0.828	0.820
	Difference from Max F-measure in %						
SVM	76.23%	7.13%	1.23%	3.46%	1.41%	1.07%	2.30%
DecisionTree	0.56%	4.59%	7.16%	7.61%	4.56%	7.79%	3.84%
RandomForest	9.74%	0.77%	1.66%	3.67%	3.14%	2.88%	3.41%
ExtraTrees	0.00%	0.00%	0.00%	2.65%	1.18%	0.00%	0.66%
XGBoost	100.00%	13.46%	4.06%	5.61%	2.72%	1.70%	0.00%
MLP	90.62%	5.62%	2.40%	0.00%	0.00%	1.26%	4.22%
SkyEx-T	4.98%	2.78%	3.12%	6.09%	5.70%	6.41%	9.92%

and optimization issues in generating counterfactuals. Probabilistic contrastive counterfactuals in Lewis [22] are used to compute explanations at local, global, and contextual levels, that can be of use to any user, regardless of their depth of knowledge. The advances of these works make it possible for any machine learning model to be somehow explainable. However, these are several steps, sometimes complex and labor intensive to use, which need to be taken in order to achieve explainability, and moreover, they do not generate a one-to-one mapping with the actual model. In contrast, SkyEx-T is already explainable out-of-the-box without the need for extra, labor-intensive steps; one can see which features are important and in which order; one can see the whole model and there is nothing else except what is defined in the model (no other weights, hidden layers, etc.).

To sum up, *all the machine learning techniques will need additional steps to improve their explainability and maybe not always succeed, while SkyEx-T's model is already human-readable and straightforward explainable.* Moreover, *SkyEx-T* has a very light configuration, meaning no coefficients or layered structures. The preference model and cut-off are chosen in a straightforward manner in the training step and later applied to the test set. Thus, there is no parameter tuning, hyperparameters, etc, very differently from machine learning. This makes *SkyEx-T* more easily deployable compared to machine learning. All in all, *SkyEx-T achieves an accuracy similar to machine learning, while having a straightforward explainability and no need for parameter/hypermater tuning.*

5.5 Implementation and Runtime

SkyEx-T is implemented in R using the `rpref` package [52] for skyline ranking, `infotheo` [41] for correlation calculations, and `skyex` [28] for calculating the optimal cut-off. *SkyEx-T* has a quadratic complexity $O(n^2)$. We report the average runtime of 10 samples for each training percentage in Fig. 3 in our larger dataset, North-DK. We measured the *preference training time*, which includes the calculation of correlations and constructing the preference function, and *skyline ranking time*, which ranks the training set based on the preference function and chooses the best cut-off. For up to 1% of training set size, the total training time is less than 7 seconds, up to 4% of training set size less than a minute (Fig. 3). *Since SkyEx-T needs a very small training dataset to be trained adequately, the typical runtime will be in the range of seconds or minutes.*

6 CONCLUSION AND FUTURE WORK

We addressed the problem of spatial entity linkage with a skyline-based approach, *SkyEx-T*, which, in contrast to the previous works [29, 31], trains the preference function and learns the cut-off for separating the classes, requiring only a small training set. We incorporated advanced similarity features, computed with *LGM-X*, which specifically capture the characteristics of spatial entities. We showed that *SkyEx-T* significantly outperforms the spatial entity linkage baselines with a margin of 0.11-0.39 in F-measure. When compared to machine learning techniques, *SkyEx-T* provides a similar accuracy on small training sets while having a straightforward explainability and easily deployable deployment model. As future work, we plan to improve the scalability of *SkyEx-T* and adapt it to other classification problems.

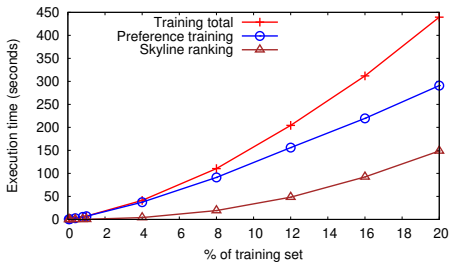

Figure 3: North-DK 0.05%-20%

REFERENCES

[1] Rifaat Abdalla. 2016. Geospatial data integration. In *Introduction to Geospatial Information and Communication Technology (GeoICT)*. Springer, 105–124.

[2] Konstantinos Alexis, Vassilis Kaffes, and Giorgos Giannopoulos. 2020. Boosting toponym interlinking by paying attention to both machine and deep learning. In *Proceedings of the Sixth International ACM SIGMOD Workshop on Managing and Mining Enriched Geo-Spatial Data*. 1–5.

[3] Konstantinos Alexis, Vassilis Kaffes, and Giorgos Giannopoulos. 2020. Boosting Toponym Interlinking by Paying Attention to Both Machine and Deep Learning *(GeoRich '20)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3403896.3403970

[4] Sandrine Balley, Christine Parent, and Stefano Spaccapietra. 2004. Modelling geographic data with multiple representations. *International Journal of Geographical Information Science* 18, 4 (2004), 327–352.

[5] William A Belson. 1959. Matching and prediction on the principle of biological classification. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 8, 2 (1959), 65–75.

[6] Bilal Berjawi, Elisabeth Chesneau, Fabien Duchateau, Franck Favetta, Claire Cunty, Maryvonne Miquel, and Robert Laurini. 2014. Representing Uncertainty in Visual Integration.. In *DMS*. 365–371.

[7] Leo Breiman. 2001. Random Forests. 45, 1 (2001). https://doi.org/10.1023/A:1010933404324

[8] Yair Censor. 1977. Pareto optimality in multiobjective problems. *Applied Mathematics and Optimization* 4, 1 (1977), 41–59.

[9] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[10] W Henry Chiu. 2010. Skewness preference, risk taking and expected utility maximisation. *The Geneva Risk and Insurance Review* 35, 2 (2010), 108–129.

[11] P. Christen. 2006. A Comparison of Personal Name Matching: Techniques and Practical Issues. In *Sixth IEEE International Conference on Data Mining - Workshops (ICDMW'06)*. 290–294. https://doi.org/10.1109/ICDMW.2006.2

[12] Peter Christen, Tim Churches, and Markus Hegland. 2004. Febrl–a parallel open source data linkage system. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 638–647.

[13] Ronan Collobert and Samy Bengio. 2004. Links between Perceptrons, MLPs and SVMs. In *Proceedings of the Twenty-First International Conference on Machine Learning (ICML '04)*. Association for Computing Machinery, 23. https://doi.org/10.1145/1015330.1015415

[14] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning* 20, 3 (1995), 273–297.

[15] Thomas M Cover and Joy A Thomas. 1991. Information theory and statistics. *Elements of Information Theory* 1, 1 (1991), 279–335.

[16] Nilesh Dalvi, Marian Olteanu, Manish Raghavan, and Philip Bohannon. 2014. Deduplicating a Places Database. In *Proceedings of the 23rd International Conference on World Wide Web (WWW '14)*. Association for Computing Machinery, New York, NY, USA, 409–418. https://doi.org/10.1145/2566486.2568034

[17] Clodoveu Davis Jr and Emerson Salles. 2007. Approximate String Matching for Geographic Names and Personal Names. 49–60.

[18] Matthew Edwards, Stephen Wattam, Paul Rayson, and Awais Rashid. 2016. Sampling labelled profile data for identity resolution. In *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 540–547.

[19] Julia Efremova, Bijan Ranjbar-Sahraei, Hossein Rahmani, Frans A Oliehoek, Toon Calders, Karl Tuyls, and Gerhard Weiss. 2015. Multi-source entity resolution for genealogical data. In *Population reconstruction*. Springer, 129–154.

[20] Vasilis Efthymiou, Kostas Stefanidis, and Vassilis Christophides. 2015. Big Data Entity Resolution. In *2015 IEEE International Conference on Big Data (IEEE BigData 2015)*.

[21] Donatella Firmani, Barna Saha, and Divesh Srivastava. 2016. Online entity resolution using an oracle. *Proceedings of the VLDB Endowment* 9, 5 (2016), 384–395.

[22] Sainyam Galhotra, Romila Pradhan, and Babak Salimi. 2021. Explaining black-box algorithms using probabilistic contrastive counterfactuals. In *Proceedings of the 2021 International Conference on Management of Data*. 577–590.

[23] Pierre Geurts, Damien Ernst, and Louis Wehenkel. 2006. Extremely randomized trees. *Machine learning* 63, 1 (2006), 3–42.

[24] Giorgos Giannopoulos, Vassilis Kaffes, and Georgios Kostoulas. 2020. Learning Advanced Similarities and Training Features for Toponym Interlinking. In *European Conference on Information Retrieval*. Springer, 111–125.

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

[26] Oana Goga, Patrick Loiseau, Robin Sommer, Renata Teixeira, and Krishna P Gummadi. 2015. On the reliability of profile matching across large online social networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1799–1808.

[27] Suela Isaj and Torben Bach Pedersen. 2019. Seed-driven geo-social data extraction. In *Proceedings of the 16th International Symposium on Spatial and Temporal Databases*. 11–20.

[28] Suela Isaj and Torben Bach Pedersen. 2020. skyex: an R Package for Entity Linkage. In *International Conference on Extending Database Technology*. 587–590.

[29] Suela Isaj, Torben Bach Pedersen, and Esteban Zimányi. 2020. Multi-Source Spatial Entity Linkage. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[30] Suela Isaj, Nacéra Bennacer Seghouani, and Gianluca Quercini. 2019. Profile reconciliation through dynamic activities across social networks. In *International Conference on Advanced Information Systems Engineering*. Springer, 126–141.

[31] 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.

[32] Vassilis Kaffes, Giorgos Giannopoulos, Nikos Karagiannakis, and Nontas Tsakonas. 2019. Learning domain specific models for toponym interlinking. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 504–507.

[33] Margot E Kaminski. 2019. The right to explanation, explained. *Berkeley Tech. LJ* 34 (2019), 189.

[34] Roula Karam, Franck Favetta, Rima Kilany, and Robert Laurini. 2010. Integration of similar location based services proposed by several providers. In *International Conference on Networked Digital Technologies*. Springer, 136–144.

[35] Deniz Kılınç. 2016. An accurate toponym-matching measure based on approximate string matching. *Journal of Information Science* 42 (2016), 138–149. https://doi.org/10.1177/0165551515590097

[36] JooYoung Lee, Rasheed Hussain, Victor Rivera, and Davlatbek Isroilov. 2018. Second-level degree-based entity resolution in online social networks. *Social Network Analysis and Mining* 8, 1 (2018), 19.

[37] Furong Li, Mong Li Lee, Wynne Hsu, and Wang-Chiew Tan. 2015. Linking temporal records for profiling entities. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*. 593–605.

[38] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep entity matching with pre-trained language models. *Proceedings of the VLDB Endowment* 14, 1 (2020), 50–60.

[39] Li Liu, William K Cheung, Xin Li, and Lejian Liao. 2016. Aligning Users across Social Networks Using Network Embedding.. In *IJCAI*. 1774–1780.

[40] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*. 4768–4777.

[41] Patrick E Meyer and Maintainer Patrick E Meyer. 2009. Package 'infotheo'. *R Packag. version* 1 (2009).

[42] 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.

[43] Erwan Moreau, François Yvon, and Olivier Cappé. 2008. Robust similarity measures for named entities matching. In *COLING 2008, 22nd International Conference on Computational Linguistics, Proceedings of the Conference, 18-22 August 2008, Manchester, UK*. 593–600.

[44] Ramaravind K Mothilal, Amit Sharma, and Chenhao Tan. 2020. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. 607–617.

[45] G. Papadakis, E. Ioannou, C. Niederée, T. Palpanas, and W. Nejdl. 2012. Beyond 100 million entities: large-scale blocking-based resolution for heterogeneous data. In *WSDM*.

[46] G. Papadakis, J. Svirsky, A. Gal, and T. Palpanas. 2016. Comparative analysis of approximate blocking techniques for entity resolution. *PVLDB* (2016).

[47] John Platt. 2000. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. *Adv. Large Margin Classif.* (2000).

[48] Gianluca Quercini, Nacéra Bennacer, Mohammad Ghufran, and Coriane Nana Jipmo. 2017. LIAISON: reconciLIAtion of Individuals Profiles Across SOcial Networks. In *Advances in Knowledge Discovery and Management*. Springer, 229–253.

[49] Ana-Maria Olteanu Raimond and Sébastien Mustière. 2008. Data matching–a matter of belief. In *Headway in spatial data handling*. Springer, 501–519.

[50] Gabriel Recchia and Max Louwerse. 2013. A Comparison of String Similarity Measures for Toponym Matching. *COMP 2013 - ACM SIGSPATIAL International Workshop on Computational Models of Place* (11 2013), 54–61. https://doi.org/10.1145/2534848.2534850

[51] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1135–1144.

[52] Patrick Roocks. 2016. Computing pareto frontiers and database preferences with the rPref Package. (2016).

[53] Murray Rosenblatt. 1956. A central limit theorem and a strong mixing condition. *Proceedings of the National Academy of Sciences of the United States of America* 42, 1 (1956), 43.

[54] Eliyahu Safra, Yaron Kanza, Yehoshua Sagiv, Catriel Beeri, and Yerach Doytsher. 2010. Location-based algorithms for finding sets of corresponding objects over several geo-spatial data sets. *International Journal of Geographical Information Science* 24, 1 (2010), 69–106.

[55] Rui Santos, Patricia Murrieta-Flores, Pável Calado, and Bruno Martins. 2018. Toponym matching through deep neural networks. *International Journal of Geographical Information Science* 32 (2018), 324–348. https://doi.org/10.1080/

13658816.2017.1390119

[56] Rui Santos, Patricia Murrieta-Flores, and Bruno Martins. 2018. Learning to combine multiple string similarity metrics for effective toponym matching. *International Journal of Digital Earth* 11 (2018). https://doi.org/10.1080/17538947.2017.1371253

[57] Michael Schäfers and Udo W Lipeck. 2014. SimMatching: adaptable road network matching for efficient and scalable spatial data integration. In *Proceedings of the 1st ACM SIGSPATIAL PhD Workshop*. 1–5.

[58] Vivek Sehgal, Lise Getoor, and Peter D Viechnicki. 2006. Entity resolution in geospatial data integration. In *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*. 83–90.

[59] Kai Shu, Suhang Wang, Jiliang Tang, Reza Zafarani, and Huan Liu. 2017. User identity linkage across online social networks: A review. *Acm Sigkdd Explorations Newsletter* 18, 2 (2017), 5–17.

[60] Liangcai Shu, Aiyou Chen, Ming Xiong, and Weiyi Meng. 2011. Efficient spectral neighborhood blocking for entity resolution. In *2011 IEEE 27th International Conference on Data Engineering*. IEEE, 1067–1078.

[61] Carolin Strobl, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn. 2007. Bias in Random Forest Variable Importance Measures: Illustrations, Sources and a Solution." BMC Bioinformatics, 8(1), 25. *BMC bioinformatics* 8 (2007).

[62] PG Tabarro, Jacynthe Pouliot, Richard Fortier, and Louis-Martin Losier. 2017. A WebGIS to support GPR 3D data acquisition: A first step for the integration of underground utility networks in 3D city models. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 42 (2017), 43.

[63] Sheila Tejada, Craig A Knoblock, and Steven Minton. 2001. Learning object identification rules for information integration. *Information Systems* 26, 8 (2001), 607–633.

[64] Volker Walter and Dieter Fritsch. 1999. Matching spatial data sets: a statistical approach. *International Journal of geographical information science* 13, 5 (1999), 445–473.

[65] Qing Wang, Dinusha Vatsalan, and Peter Christen. 2015. Efficient interactive training selection for large-scale entity resolution. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 562–573.

[66] Yutao Zhang, Jie Tang, Zhilin Yang, Jian Pei, and Philip S Yu. 2015. 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*. 1485–1494.