STEM: stacked threshold-based entity matching for knowledge base generation

Semantic Web 1 (2016) 1–5
IOS Press

STEM: Stacked Threshold-based Entity Matching for Knowledge Base Generation

Enrico Palumbo [a,b,c,*], Giuseppe Rizzo [a] and Raphaël Troncy [b]
[a] *Istituto Superiore Mario Boella, Turin, Italy*
[b] *EURECOM, Sophia Antipolis, France*
[c] *Politecnico di Torino, Turin, Italy*

Abstract. One of the major issues encountered in the generation of knowledge bases is the integration of data coming from a collection of heterogeneous data sources. A key essential task when integrating data instances is the entity matching. Entity matching is based on the definition of a similarity measure among entities and on the classification of the entity pair as a match if the similarity exceeds a certain threshold. This parameter introduces a trade-off between the precision and the recall of the algorithm, as higher values of the threshold lead to higher precision and lower recall, and lower values lead to higher recall and lower precision. In this paper, we propose a stacking approach for threshold-based classifiers. It runs several instances of classifiers corresponding to different thresholds and use their predictions as a feature vector for a supervised learner. We show that this approach is able to break the trade-off between the precision and recall of the algorithm, increasing both at the same time and enhancing the overall performance of the algorithm. We also show that this hybrid approach performs better and is less dependent on the amount of available training data with respect to a supervised learning approach that directly uses properties' similarity values. In order to test the generality of the claim, we have run experimental tests using two different threshold-based classifiers on two different data sets. Finally, we show a concrete use case describing the implementation of the proposed approach in the generation of the 3cixty Nice knowledge base.

Keywords: knowledge base generation, entity matching, link discovery, stacking, FEIII, DOREMUS, 3cixty, OAEI

1. Introduction

In the last decade, we have witnessed to the generation of several knowledge bases that grant access to an enormous amount of structured data and knowledge. However, the generation of knowledge bases has required a tremendous manual effort to overcome several challenges. One of the typical issues in the generation of knowledge bases that integrate data from a collection of heterogeneous sources is that of automatically detecting duplicate records. Entity matching (also known as instance matching, data reconciliation or record linkage) is the process of finding non-identical records that refer to the same real-world entity among a collection of data sources [1]. Entity matching allows to identify redundant data, remove them (*deduplication*) and obtain unambiguous entities. Entity matching is rendered troublesome by the different data models used by the data providers, by possible misspellings, errors and omissions in data descriptions, by the use of synonyms, as well as the presence of implicit semantics in the textual descriptions. Consider as an example the case in which one record is named "Black Diamond BGWB14 Inc." and the second record is named "Black Diamond f.s.b.". In order to understand whether the two records correspond to the same real world entity, in addition to taking into account other properties such as the address, the state or the geographical position, it is clearly necessary to have expertise in the domain and to be able to understand the meaning of the abbreviations, as well as to rule out evident misspellings or mistakes. Nevertheless, a manual comparison from human experts

*Corresponding author. E-mail: palumbo@ismb.it, Tel. +39 011 227 62 27

is in most cases unfeasible, as matching entities requires a quadratic computational time (e.g. matching $\sim 10^3$ entities requires $\sim 10^6$ comparisons). Thus, entity matching systems typically define a metric to measure a similarity between entities. This metric can be defined through knowledge of the domain and a trial-and-error process, in a top-down manner [2,3], or can be learned from annotated examples, in a bottom-up way [4,5]. Then, the similarity is turned into a confidence score, which represents the degree of confidence in asserting that the pair of entities is a match. Finally, a threshold has to be specified, in order to convert the confidence score into a decision, namely classifying the pair as a match or not. This decision threshold introduces a trade-off between the precision, i.e. the capacity of discriminating false positives, and the recall, i.e. the capacity of individuating true positives, of the algorithm. Indeed, higher values of the threshold lead to a more selective classifier, which tends to incur in false negatives, reducing the recall of the algorithm, while lower values of the threshold produce the opposite effect. Thus, the user typically attempts to find a balance between these two measures, either manually or using more sophisticated approaches that are able to learn a configuration from annotated examples. Independently from the strategy chosen to set the final threshold, state-of-the-art systems typically rely on a single decision threshold. In this paper, we show that the combination of the predictions of an array of thresholds using ensemble learning, and, more in detail, stacked generalization, is able to break the trade-off between the precision and the recall of the algorithm, increasing both at the same time, and consequently the F-score of the algorithm. We propose a general approach called STEM (Stacked Threshold-based Entity Matching), which can be applied on top of any numerical threshold-based entity matching system. STEM is based on the principle of *Stacking* (or Stacked Generalization) [6], which consists in training a meta-learner to combine the predictions of a number of base classifiers. STEM creates several instances, corresponding to different values of the final decision threshold, of a threshold-based classifier. Then, the classification outputs of this ensemble of classifiers are used as a binary feature vector for a supervised learner, which is trained on a set of manually annotated data. The main goal of STEM is that of enhancing the *efficiency* of a threshold-based entity matching system, namely the capability of generating high-quality matches improving the precision and the recall of the classifier at the same time [14]. In order to test the generality of our claim, we run experimental tests using two different unsupervised threshold-based classifiers. The first is a Naive Bayes classifier [7,8], which follows the approach popularized by the Paul Graham's spam filter[1] and is implemented by the open source deduplication framework Duke.[2] The second is a linear classifier, implemented by the open source framework Silk,[3] [9] which is currently quite widespread in the Semantic Web and Linked Data communities. The first dataset used for the experimental evaluation is that released by the organizers of the Financial Entity Identification and Information Integration (FEIII) Challenge. The second dataset is taken from the DOREMUS project, released by the instance matching track of the Ontology Alignment Edition Initiative (OAEI). In addition to these datasets, we further validate STEM by describing its implementation in a concrete use case, represented by the 3cixty project.[4] In this context, STEM is used on top of a Naive Bayes threshold-based classifier to match entities and remove duplicates representing places and events coming from a number of heterogeneous local and global data sources in order to create a cleaner and of better quality knowledge base, which is used to support the planning of tourist visits and to offer a digital guide for tourists when exploring the city. The novel contributions of this paper are:

- We design a generic framework based on stacked generalization that is able to improve the efficiency of threshold-based entity matching systems;
- We provide empirical evidence of this claim by testing it with two different threshold-based entity matching systems, showing that efficiency gain can be up to 44% of F1 from a threshold-based classifier;
- We show that STEM applied on top of a Naive Bayes classifier performs better and has a weaker dependence on the amount of manually annotated entity pairs with respect to pure machine learning approaches;
- We describe the implementation of the framework in the generation of the 3cixty knowledge base, providing evidence of its efficiency on a newly generated gold standard data set;

[1] http://www.paulgraham.com/spam.html
[2] https://github.com/larsga/Duke
[3] https://github.com/silk-framework/silk
[4] https://www.3cixty.com

The remainder of the paper is structured as follows: in Sec. 2 we describe the relevant related work in entity matching, in Sec. 3 we describe the problem of entity matching and of the trade-off between precision and recall, in Sec. 4 we describe the STEM approach and the theoretical background of the base classifiers utilized in the experimental part, in Sec. 5 we describe the experimental setup and the configuration process, in Sec. 6 we show the experimental results, in Sec. 7 we describe the implementation of STEM in the 3cixty project and in Sec. 8 we conclude the paper.

2. Related Work

Entity matching is a crucial task for data integration [12] and probabilistic approaches able to handle uncertainty have been proposed since the 60s [13].
A survey of frameworks for entity matching is reported by Köpcke in [14], where a classification of several entity matching frameworks is done by analyzing the entity type, i.e. how the entity data is structured, blocking methods, i.e. the strategy employed to reduce the search space and avoiding the comparison of each possible pair of records, the matching method, i.e. the function utilized to determine if a pair of records represents the same real world entity and the training selection, i.e. if and how training data is used. By taking into account the matching method, entity matching frameworks may be divided in frameworks without training [2,3,15], in which the model needs to be manually configured, training-based frameworks [16,18], in which several parameters are self-configured through a learning process on an annotated training set, and hybrid frameworks, which allow both manual and automatic configuration [19,20].
The authors of the survey thoroughly compare different frameworks on a set of key performance indicators and highlight a research trend towards training-based and hybrid approach, which, in spite of the dependence on the availability, size and quality of training data, significantly reduce the effort of manual configuration of the system. The most commonly used supervised learners are Decision Trees and SVM. Training can help for different purposes, such as learning matching rules or in which orders matchers should be applied, automatically setting critical parameters and/or determining weights to combine matchers similarity values [23,17,18,19]. In [21], a comparison among the most common supervised (training-based) learning models is reported together with an experimental evaluation. The authors report a high degree of complementarity among different models which suggests that a combination of different models through ensemble learning approaches might be an effective strategy. The idea of ensemble learning is to build a prediction model by combining the strengths of a collection of simpler base models. Ensemble learning can be broken down into two tasks: developing a population of base learners from the training data, and then combining them to form the composite predictor [22]. In [23] the authors report that an ensemble of base classifiers built through techniques such as bagging, boosting or stacked generalization (also known as stacking) generally improves the efficiency of entity matching systems. Another evidence of the efficiency of ensemble approaches to entity matching is reported in [24].

In the past years, the Linked Data [25] research community has shown a great deal of interest for Entity Matching. More specifically, Entity Matching (or Instance Matching) can be seen as a part of the process of Link Discovery. Link Discovery has the purpose of interlinking RDF data sets that are published on the Web, following the evidence of recent studies that show that 44% of the Linked Data datasets are not connected to other datasets at all [26]. Link Discovery can be seen as a generalization of Entity Matching, because it can be used to discover other properties than an equivalence relation between instances. Moreover, as remarked in [27], in Link Discovery resources usually abide by an ontology, which describes the properties that resources of a certain type can have as well as the relations between the classes that the resources instantiate. The authors of [27] report a comprehensive survey of Link Discovery frameworks, which shows that modern framework such as Silk [28,4], LIMES [29], EAGLES [5] combine manually defined match rules with genetic programming and/or active learning approaches to automatize the configuration process. A different approach is proposed by WOMBAT [30], which relies on an iterative search process based on an upward refinement operator. WOMBAT learns to combine atomic link specifications into complex link specifications to optimize the F-score using only positive examples. Another line of work is that of collective entity matching (or resolution) systems, which are not based on pairwise similarity comparison as STEM, but rather on the attempt to capture the dependencies among different matching decisions [31,32,33,34,35,36].

STEM is a general approach to improve the effi-

ciency of an entity matching system, which is in principle applicable to any pairwise numerical threshold-based classifier to enhance its efficiency.

3. Problem Formulation

The problem of entity matching can be defined as follows [37]:

Definition 1 *Given two datasets A and B, find the subset of all pairs of entities for which a relation $\sim$ holds:* $M = \{e_1 \in A, e_2 \in B, (e_1, e_2) \in AxB : e_1 \sim e_2\}$

In this formulation, we assume that the schema mapping problem is solved, and thus:

Definition 2 *Given a property π of the schema of A and a property ρ of the schema of B, we assume that a set of **mapped properties** has been defined $m_i = \{(\pi_i, \rho_i)\}$ with $i = 1..K$. In the following, when we refer to the properties $i = 1..K$, we refer to the components of the mapping m_i, i.e. to π_i for $e_1 \in A$ and ρ_i for $e_2 \in B$.*

We assume that the comparison between a pair of entities e_1 and e_2 is carried out on a set **literal values** v_i of properties $i = 1..K$. We assume that the entity matching system is a pairwise numeric threshold-based binary classifier acting on the properties $i = 1..K$. Thus:

Definition 3 *We define the **linkage rule** as a Boolean function $\hat{f} : (e_1, e_2) \in AxB \rightarrow \{0, 1\}$, where $\hat{f} = 1$ indicates that the pair is deemed a match and $\hat{f} = 0$ indicates that the pair is not deemed to be a match.*

Definition 4 *The comparison of two entities is carried out by means of a **comparison vector** $s(e_1, e_2) = \{s_1(e_1, e_2), s_2(e_1, e_2)..s_K(e_1, e_2)\}$, where the components $s_i \in \mathbb{R}^+$ represent atomic similarities defined over a number of literal values $s_i(e_1, e_2) = s(v_i(e_1), v_i(e_2))$ of the properties $i = 1..K$*

Definition 5 *The comparison vector is then turned into a **confidence vector** $c(e_1, e_2) = \{c_1(s_1(e_1, e_2)), c_2(s_2(e_1, e_2))..c_K(s_K(e_1, e_2))\}$, where each component c_i represents the degree of confidence in stating that the pair of entities is a match given by the similarity score $s_i(e_1, e_2)$*

Definition 6 *We define a **confidence function** $f : c(e_1, e_2) \rightarrow [0, 1]$ which maps the confidence vector $c(e_1, e_2)$ into a final score representing the confidence of the entity matching system in stating that the pair of entities is a match.*

Definition 7 *We define as **threshold-based classifier** a linkage rule $\hat{f}$ that depends on the confidence function f in the following way:*

$$\hat{f}(e_1, e_2; t) = \theta(f(c(e_1, e_2)) - t) \quad (1)$$

where θ is the Heaviside step function and $t \in [0, 1]$ is a given threshold.[5] The linkage rule of a threshold-based classifier has a very intuitive interpretation. A pair of entities is considered to be a match if the degree of confidence f that the pair is a match is above a certain threshold t. The threshold t can be defined experimentally or can be learned from a set of examples. Independently from the strategy through which it is set, the threshold t introduces a trade-off between the rate of false positives and false negatives that the algorithm will accept. To see why this is the case, let us first start from the definition of the two types of errors defined in [37], considering the variations of the confidence value $f \in [0, 1]$. The first error, which corresponds to the probability of having false positives, occurs when an unmatched comparison is considered as a match:

$$p_{fp} = P(\hat{f} = 1 | e_1 \neq e_2) = \int_0^1 P(\hat{f} = 1 | f) P(f | e_1 \neq e_2) df \quad (2)$$

Given that we consider a binary threshold-based classifier, it follows from Eq. 1 that:

$$P(\hat{f} = 1 | f) = \theta(f - t) \quad (3)$$

which leads to:

$$p_{fp} = P(\hat{f} = 1 | e_1 \neq e_2) = \int_t^1 P(f | e_1 \neq e_2) df \quad (4)$$

The second error, which corresponds to the probability of false negatives, occurs when a matched comparison is not considered to be a match:

$$p_{fn} = P(\hat{f} = 0 | e_1 = e_2) = \int_0^1 P(f | e_1 = e_2) P(\hat{f} = 0 | f) df \quad (5)$$

[5] We assume that both f and t are normalized in the [0,1] interval, but it is intuitive to see that the same argument holds for any closed interval $[a, b] \in \mathbb{R}$ with $a < b$, as Eq. 1 is invariant to any multiplying factor within the θ function.

From Eq. 3, it follows that:

$$P(\hat{f} = 0|f) = 1 - \theta(f - t) \quad (6)$$

and thus:

$$p_{fn} = P(\hat{f} = 0|e_1 = e_2) = \int_0^t P(f|e_1 = e_2)df \quad (7)$$

The probability of true positives is similarly measured as:

$$p_{tp} = P(\hat{f} = 1|e_1 = e_2) = \int_t^1 P(f|e_1 = e_2)df \quad (8)$$

Let us now consider the graphic illustration provided in Fig. 1. Assuming that f is a meaningful confidence function, the probability density function of the values of f under the condition that $e_1 = e_2$, i.e. $P(f|e_1 = e_2)$, has a higher mean and is located to the right, while $P(f|e_1 \neq e_2)$, conditioned by $e_1 \neq e_2$, is located to the left. Let also N_+ be the total number of positive pairs, i.e. matching entities, and N_- the total number of negative pairs, i.e. non matching entities, in the data. In this graphical representation, the linkage rule of Eq. 1 implies that the area of $P(f|e_1 = e_2)$ situated to the left of the vertical line (in yellow) corresponds to the probability of classifying a true match as a non matching pair, i.e. to the probability of producing false negatives p_{fn}. The number of false negatives FN is then given by $FN = p_{fn}N_+$. On the other hand, the area of $P(f|e_1 \neq e_2)$ situated to the right of the vertical line (in orange) corresponds to the probability of classifying a false match as a match, i.e. the probability of producing false positives p_{fp}. Similarly to the previous case, we have that $FP = N_- p_{fp}$. Finally, we also have that the grey area in the graph is the probability of true positives p_{tp}. The number of true positives is then given by: $TP = N_+ p_{tp}$. From Fig. 1 we can see that p_{fn}, and consequently FN, is increasing when the threshold t increases, and at the same time p_{fp}, and consequently FP, is decreasing when the threshold t increases. p_{tp} is also decreasing, but at a slower pace. Now, if we recall the definition of precision and recall [38]:

$$p = \frac{TP}{TP + FP} \quad (9)$$

$$r = \frac{TP}{TP + FN} \quad (10)$$

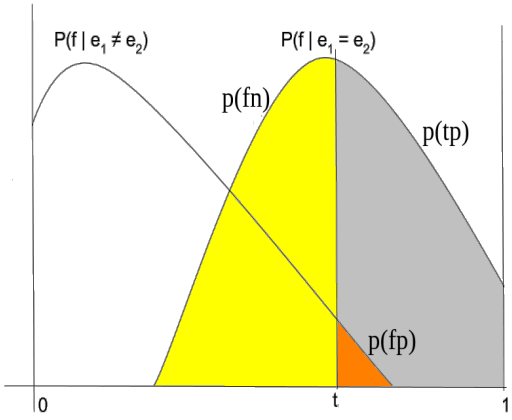

Fig. 1. Graphical depiction of p_{fn}, p_{fp} and p_{tp} under the linkage rule Eq. 1. The vertical line represents the decision threshold t. The shape of the probability distribution has illustrative purposes.

we can see that, when t increases, $FP \to 0$ faster than TP, and p increases. At the same time, FN is growing and r decreases. Conversely, when t decreases FP grows and FN decreases, increasing r and decreasing p. Thus, the threshold t introduces a trade-off between the precision and the recall of the algorithm (we provide experimental evidence of this heuristic argument in Sec. 6). Note that this trade-off is not limited to Entity Matching and is well known by the Information Retrieval and Statistical Learning community, where precision-recall curves obtained through variations of the decision threshold are often used as a measure of an overall algorithm's efficiency [39,38,40].

4. Stacked Threshold-based Entity Matching

In this work, we show that stacking can break the trade-off by raising both precision and recall at the same time through supervised learning. Stacking [6] (also known as stacked generalization), is based on the idea of creating an ensemble of base classifiers and then combining them by means of a supervised learner, which is trained on a set of labeled examples. In this case, the base classifiers correspond to a single threshold-based classifier $\hat{f}(e_1, e_2; t)$ with a set of different decision thresholds $t_1, t_2.., t_N$. The supervised learner is a binary classifier whose features are the match decisions of the base classifiers $F : \{\hat{f}(e_1, e_2; t_1), \hat{f}(e_1, e_2; t_2)...\hat{f}(e_1, e_2; t_N)\} \to \{0, 1\}$

and whose output is a binary match decision, which represents the final decision of the system. Stacking requires the creation of a gold standard G containing correctly annotated entity pairs, which are used as a training set for the supervised learning approach. More in detail, the Stacking Threshold-based Entity Matching approach (Fig. 2) works as follows:

1. **Blocking** (optional): although not strictly needed, it is in practice necessary for large datasets to find good candidates e_1, e_2 using a blocking strategy, avoiding a quadratic comparison of entities (see Section 5.3 and Section 5.4).
2. **Threshold-based classifier**: start from a linkage rule based on a threshold-based classifier $\hat{f}(e_1, e_2; t)$ such as the one defined in Eq. 1
3. **Threshold perturbation**: generate an ensemble of N linkage rules $\hat{f}(e_1, e_2; t_i)$ where t_i are linearly spaced values in the interval $[t - \frac{a}{2}, t + \frac{a}{2}]$ and $0 < \frac{a}{2} < min\{t, 1 - t\}$ is the perturbation amplitude
4. **Stacking**: combines the predictions corresponding to different thresholds using supervised learning.
   Features: use the predictions $x_i = \hat{f}(e_1, e_2; t_i)$ as features for a supervised learner $F(x; w)$ where w are parameters that are determined by the learning algorithm.
   Training: train the supervised learner $F(x; w)$ on the gold standard G, determining the parameters $\hat{w}$. This typically involves the minimization of an error function $E(x, w, G)$:

$$\hat{w} = \min_{w} E(x, w, G) \tag{11}$$

   The shape of the error function $E(x, w, G)$ and how the optimization is solved depends on the particular supervised learner that is chosen. In this paper, we use an SVM classifier and we thus rely on the SVM training algorithm [22]. Note that the training process only needs to be done once per dataset, as the learned model can be stored and loaded for testing.
   Testing: generate the final prediction $F(x; \hat{w})$.

The intuition behind this approach is that using stacking the space of features is no longer the confidence score f as for the threshold-based classifier. The supervised learner F uses as features the set of matching decisions of the base classifiers and, by combining them in a supervised way, it is no longer tied to the trade-off introduced by the threshold t that we have described in Sec. 3. We show experimental evidence of the effectiveness of stacking in increasing both the precision and the recall of a threshold-based classifier in Sec. 6.

We now provide the descriptions of the two threshold-based entity matching systems that are used in this paper, showing that they both constitute particular cases of Eq. 1.

4.1. Linear Classifier

One of the simplest models for the confidence function $f(e_1, e_2)$ of a pair of entities e_1 and e_2 is that obtained by the linear combination of the components of the confidence vector $c(e_1, e_2)$ introduced in Sec. 3. Given the set of properties $j = 1..K$ and their respective values $v_j(e_1)$ and $v_j(e_2)$ for both entities, property-wise similarities are functions that yield a vector of similarity scores $s_j = s_j(v_j(e_1), v_j(e_2))$, where typically $s_j \in [0, 1]$ with $s_j = 1 \iff v_j(e_1) \equiv v_j(e_2)$. At this point, similarity scores s_i are normally turned into the property-wise confidence scores $c_i = c_i(s_i)$, which are then linearly combined through the confidence function f. This is the case of Silk[6] [28], which is a popular Link Discovery framework, specifically built to generate RDF links between data items within different Linked Data resources. More specifically, Silk works with distances d_i rather than with similarities s_i and different comparators can be selected to define the distances d_i, such as Levehnstein, Jaro-Winkler, exact comparators, Jaccard [41]. Then, distance scores $d_i > 0$ are turned into confidence scores c_i according to the rule[7] (Fig. 3):

$$c_i = c(d_i) = \begin{cases} -\frac{d_i}{\tau_i} + 1 & 0 \leq d_i < 2\tau_i \\ -1 & d_i \geq 2\tau_i \end{cases}$$

where τ_i are property-specific thresholds. Note that c_i is a monotone decreasing function, as it depends on distances d_i rather than on similarities s_i values. In this way, for each property used for the comparison, a confidence score $c_i \in [-1, 1]$ is obtained. Silk allows to combine these confidence scores in multiple ways,

[6] http://silkframework.org
[7] https://github.com/silk-framework/silk/blob/master/doc/LinkageRules.md

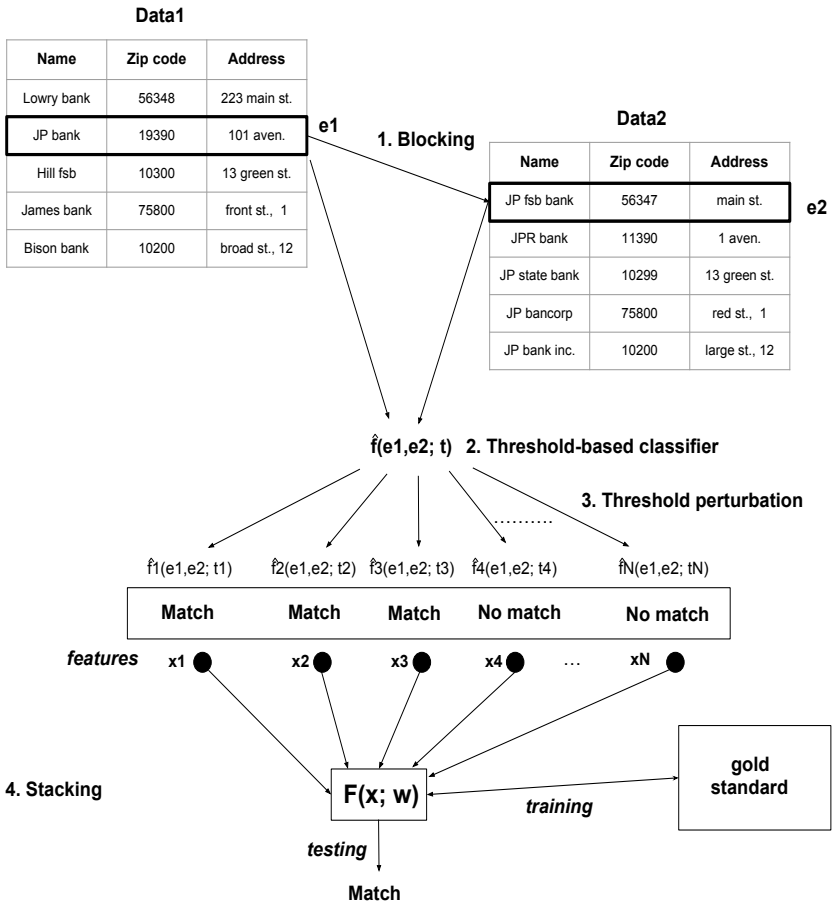

Fig. 2. Global architecture of the STEM framework.

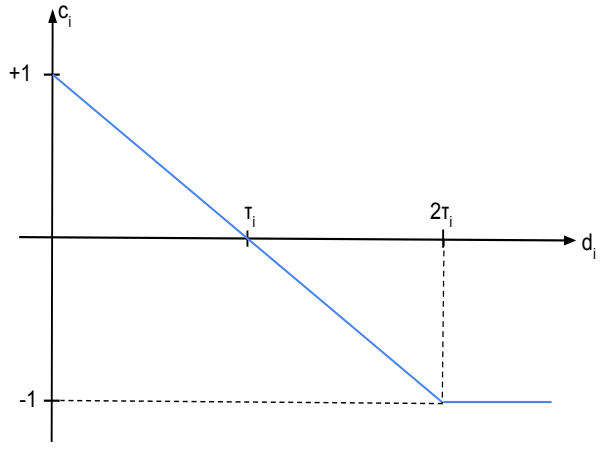

Fig. 3. Silk function to compute property-wise confidence scores from distance values

among which the linear combination, which is the one that has been utilized in this work:

$$\hat{f} = \theta(\sum_{i=1}^{K} w_i c_i - t) \tag{12}$$

which corresponds to the linkage rule Eq. 1 with:

$$f(c(e_1, e_2)) = \sum_{i=1}^{K} w_i c_i \tag{13}$$

The final decision threshold t corresponds to the parameter 'minConfidence' in Silk configuration file. This parameter, together with all the others such as property-wise thresholds or comparators, can be manually set through a trial-and-error process or they can be learnt through an active learning algorithm that is based on the approach of letting users annotate matches that produce the utmost information gain [42].

4.2. Naive Bayes Classifier

Naive Bayes is a term used to describe a family of classifiers that are based on the Bayes theorem and on a particular assumption of independence among the components of the evidence vector [7,43]. In this paper, we use the formulation of the Naive Bayes classifier that has been popularized by Paul Graham's Bayesian spam filter.[8] We now want to show that the Naive Bayes classifier can be considered as a threshold-based classifier, obeying to the decision rule of Eq. 1.

Given a set of classes X_i with $i = 1..N$ and a vector of observations $x = \{x_1, x_2..x_K\}$, the Naive Bayes classifier aims to estimate the probability of a class given a set of observed data $P(X_i|x)$ by applying the Bayes theorem and the conditional independence condition (from this assumption comes the adjective 'Naive'):

$$P(X_i|x) = \frac{P(x|X_i)P(X_i)}{P(x)} = \frac{P(X_i)\prod_{j=1}^{k} P(x_j|X_i)}{P(x)} \tag{14}$$

In our case, we have a binary classification problem, where the decisions $X_1 = 1 =$ 'Match' and $X_2 = 0 =$ 'No Match' and the observations are represented by the comparison vector s. Eq. 14 for $X_1 = 1$ becomes:

$$P(1|s) = \frac{P(1)\prod_{i=1}^{k} P(s_i|1)}{P(s)}$$

Since $P(s) = P(s|1)P(1) + P(s|0)P(0)$ the denominator can be rewritten as:

$$P(1|s) = \frac{P(1)\prod_{i=1}^{k} P(s_i|1)}{P(s|1)P(1) + P(s|0)P(0)} \tag{15}$$

and then, using again the conditional independence hypothesis, factorized as:

$$P(1|s) = \frac{P(1)\prod_{i=1}^{k} P(s_i|1)}{P(1)\prod_{i=1}^{k} P(s_i|1) + P(0)\prod_{i=1}^{k} P(s_i|0)} \tag{16}$$

Now, by applying Bayes theorem $P(s_i|1) = \frac{P(1|s_i)}{P(1)}P(s_i)$ and $P(s_i|0) = \frac{P(0|s_i)}{P(0)}P(s_i)$, denoting with $x = P(1)$ and $1 - x = P(0)$, we have:

$$P(1|s) = \frac{\frac{1}{x^{k-1}}\prod_{i=1}^{k} P(1|s_i)}{\frac{1}{x^{k-1}}\prod_{i=1}^{k} P(1|s_i) + \frac{1}{(1-x)^{k-1}}\prod_{i=1}^{k} P(0|s_i)} \tag{17}$$

Finally, assuming that, *a priori*, $P(1) = P(0)$ and thus $x = 1 - x$, we can remove the coefficients and by denoting with $c_i = P(1|s_i)$ we obtain:

$$P(1|s) = \frac{c_1 c_2..c_k}{c_1 c_2..c_k + (1-c_1)(1-c_2)..(1-c_k)} \tag{18}$$

Details of the derivation can be found in [44]. Note that $c_i = P(1|s_i)$ exactly represents the confidence score derived from the similarity value s_i. It is also important to notice that this simplifying assumption $P(1) = P(0)$, although widespread in practice and used in the implementation of Duke, is not necessary for a Naive Bayes classifier and can be modified with any other strategy to estimate prior probabilities.

At this point, it is necessary to specify a decision rule, that is a rule to turn the probability evaluation into a decision. A common approach is the Maximum a Posteriori (MAP) Estimation [45], namely selecting the class that maximizes the posterior probability:

$$X = \arg\max_{X_i} P(X_i|x) \tag{19}$$

which allows to define a binary linkage rule as:

$$\hat{f} = 1 \iff P(1|s) > P(0|s) \tag{20}$$

which can easily be rewritten as:

$$\hat{f} = 1 \iff \frac{P(1|s)}{P(0|s)} > 1 \tag{21}$$

Now, by adopting a decision-theoretic notion of cost, we can turn Eq. 21 into [46]:

$$\hat{f} = 1 \iff \frac{P(1|s)}{P(0|s)} > \lambda \tag{22}$$

where λ is a value that indicates how many times false positives are more *costly* than false negatives. From Eq. 22, it is clear that if $\lambda > 1$, we require that $P(1|s)$ is λ times greater than $P(0|s)$ in order to consider the pair to be a match, and thus we are more keen to accept false negatives than false positives. Vice versa, if

[8] http://www.paulgraham.com/spam.html

$\lambda < 1$, the algorithm will tend to have more false positives than false negatives. Finally, by considering that $P(0|s) = 1 - P(1|s)$ and by using Eq. 18 we obtain the decision rule:

$$\hat{f} = 1 \iff \frac{c_1 c_2..c_k}{c_1 c_2..c_k + (1-c_1)(1-c_2)..(1-c_k)} > t \tag{23}$$

where $t = \frac{\lambda}{1+\lambda}$. It is now easy to see that Eq. 23 can be rewritten as:

$$\hat{f} = \theta(\frac{c_1 c_2..c_k}{c_1 c_2..c_k + (1-c_1)(1-c_2)..(1-c_k)} - t) \tag{24}$$

which has the same form of Eq. 1, where the combination of confidence scores c_i has the role of the global 'confidence function':

$$f(c(e_1, e_2)) = \frac{c_1 c_2..c_k}{c_1 c_2..c_k + (1-c_1)(1-c_2)..(1-c_k)} \tag{25}$$

We have thus shown that from a Naive Bayes classifier we can obtain a threshold-based classifier abiding by Eq. 1. As we have argued in Sec. 3, the threshold t rules the trade-off between the rate of false positives and false negatives that the algorithm will accept. This is evident by its relation with λ:

$$\lambda \to \infty \Rightarrow t \to 1 \tag{26}$$

$$\lambda \to 0 \Rightarrow t \to 0 \tag{27}$$

Thus, the higher the value of t, the higher needs to be the probability that the pair is a match for the algorithm to consider it a match. Thus, we are less likely to have false positives and more likely to have false negatives.

In the past years, Naive Bayes classifiers have been utilized in a large number of fields, such as spam filtering [8], document and text classification [47], information retrieval [7], entity matching [48] and so on. Duke[9] is a popular open-source deduplication engine, which implements Naive Bayes classification. Duke is a flexible tool, which accepts different formats of input data, and is easy to configure through a simple XML file. For each field of each data source, the user can choose a number of string cleaners, such as functions that remove abbreviations or normalize lower/upper cases. For each property, Duke allows to select a comparator among popular string similarity measures such as Levensthein, Jaro-Winkler, exact comparators and so on [41]. The comparators thus compute, for each property, a normalized similarity score s_i. Then, in order to turn similarity scores into a confidence score c_i, Duke uses the heuristic function:

$$c_i = P(1|s_i) = \begin{cases} low_i & s_i \leq 0.5 \\ (high_i - 0.5)s_i^2 + 0.5 & s_i \geq 0.5 \end{cases}$$

where low_i and $high_i$ are parameters that the user can configure for each property. The rationale behind this formula of $P(1|s_i)$ is that $P(1|s_i = 0) = low$ and $P(1|s_i = 1) = high$, and, as Duke's users were finding the algorithm to be too strict, a quadratic instead of a linear trend has been chosen when s_i is larger than 0.5. After that c_i is computed for each property, the overall $P(1|s)$ is calculated through Eq. 18 and the decision is taken through Eq. 23. Similarly to the case of Silk, the final decision threshold t is a parameter that can be configured in a XML file. Duke also includes a genetic algorithm that automatizes the configuration process and in general represents a valid alternative to the manual configuration. Through an active learning approach, Duke asks to the user in an interactive way if a pair of entities should be a match or not, selecting the most informative pairs, i.e. the ones with utmost disagreement among the population of configurations [5].

4.3. Computational complexity

A crucial point for entity matching systems, which are often used to find matching entities among datasets with large numbers of instances, is their computational complexity. The stacking approach introduced by STEM adds an overhead to the runtime performance of the base classifier due to the generation of the ensemble of predictions and to the learning process. More in detail, the computational complexity of STEM in the current sequential implementation can be roughly approximated with:

$$T_{STEM} \approx N * T_{baseclassifier}(n, m) + T_{stacking}(N, g, k)$$

where N is the number of features, n is the number of instances of the first dataset, m is the num-

[9] https://github.com/larsga/Duke

ber of instances of the second dataset, g is the size of the training set and k is the size of the test set. $T_{baseclassifier}(n,m)$ depends on the nature and the configuration of the base classifier, especially on the blocking strategy. Let us assume that the base classifier adopts a smart blocking strategy such as that of Silk, we can assume that $T_{baseclassifier}(n,m) = O(n+m)$ [9]. $T_{stacking}(N,g,k)$ depends on the supervised learner that is used and depends on the training and the testing time $T_{stacking}(N,g,k) = T_{train}(N,g) + T_{test}(N,k)$. The training and testing processes can easily be decoupled, for instance saving the trained model to a file and loading it subsequently for testing. However, in this section, we discuss the worst case in which we need to first train and then test the model. In this paper, we use a kernel SVM[10] classifier as a supervised learner, whose complexity may vary depending on a number of practical factors depending on the specific implementation. However, as a rule of thumb, it is reasonable to assume [10,11] that: $O(N*g^2) < T_{train}(N,g)(N,g) < O(N*g^3)$ and $T_{test}(N,k) < O(N*k)$. To summarize, we can then say that:

$$T^{train}_{STEM} < N*O(n+m) + O(N*g^3)$$

$$T^{test}_{STEM} < N*O(n+m) + O(N*k)$$

In our experiments, we have that $g <= n$, $g <= m$ and, by using cross-validation, we have that $k \approx g$. In practice, we observe that when the number of features N grows, the time for generating the predictions quickly surpasses the time of stacking, i.e. $N * T_{baseclassifier}(n,m) > T_{stacking}(N,g,k)$ (see Section 6 for an example). Note that the time could be reduced to $T_{STEM} \approx T_{baseclassifier}(n,m) + T_{stacking}(N,g)$ by parallelizing the generation of the predictions of the N base classifiers, which we leave as a future work.

5. Experimental setup

As we have explained in Sec. 4, the STEM approach is general and can be utilized on top of any threshold-based entity matching system. In this paper, we have implemented it and evaluated through two different open source frameworks, Duke and Silk, which are based respectively on a Naive Bayes and on a linear classifier. In Sec. 5.3 and in Sec. 5.4, we describe the configuration process of these frameworks inside STEM. The software implementation of STEM, the configuration files and the data used for the experiments are publicly available on github.[11]

5.1. Datasets

The first dataset utilized for the evaluation of the proposed approach is that released by the organizers of the Financial Entity Identification and Information Integration challenge of 2016 (FEIII2016).[12] The purpose of the challenge is that of creating a reference financial-entity identifier knowledge base linking heterogeneous collections of entity identifiers. Three datasets have been released:

- FFIEC: from the Federal Financial Institution Examination Council, provides information about banks and other financial institutions that are regulated by agencies affiliated with the Council.
- LEI: contains Legal Entity Identifiers (LEI) for a wide range of institutions.
- SEC: from the Securities and Exchange Commission and contains entity information for entities registered with the SEC.

In this paper, we focus on the Entity Matching of entities of the FFIEC database and the SEC database, as it proved to be the most challenging one. The gold standard, which can be seen as a benchmark for the evaluation of the systems as well as a set of annotations to create a supervised system, has been created by a panel of experts of the field. The gold standard contains 1428 entity pairs, with 496 positive and 932 negative examples. The dataset is available online.[13]
A second evaluation of the STEM approach is performed on the dataset released by the DOREMUS project[14] in the context of the instance matching track of the Ontology Alignment Evaluation Initiative 2016 (OAEI2016[15]). The Instance Matching Track of the OAEI 2016 aims at evaluating the efficiency of matching tools when the goal is to detect the degree of similarity between pairs of items/instances expressed in the form of OWL Aboxes. The DOREMUS datasets contain real world data coming from two major French cultural institutions: the French National

[10] http://scikit-learn.org/stable/modules/svm.html

[11] https://github.com/enricopal/STEM

[12] https://ir.nist.gov/dsfin/index.html

[13] https://ir.nist.gov/dsfin/data/feiii-data-2016-final.zip

[14] http://www.doremus.org/

[15] http://islab.di.unimi.it/im_oaei_2016/

Library (BnF) and the Philharmonie de Paris (PP). The data is about classical music works and is described by a number of properties such as the name of the composer, the title(s) of the work, its genre, instruments and the like. We focused our evaluation on two tasks, whose description we report here:

- **Nine heterogeneities**: "This task consists in aligning two small datasets, BnF-1 and PP-1, containing about 40 instances each, by discovering 1:1 equivalence relations between them. There are 9 types of heterogeneities that data manifest, that have been identified by the music library experts, such as multilingualism, differences in catalogs, differences in spelling, different degrees of description."
- **Four heterogeneities**: "This task consists in aligning two bigger datasets, BnF-2 and PP-2, containing about 200 instances each, by discovering 1:1 equivalence relations between the instances that they contain. There are 4 types of heterogeneities that these data manifest, that we have selected from the nine in task 1 and that appear to be the most problematic: 1) Orthographical differences, 2) Multilingual titles, 3) Missing properties, 4) Missing titles."

Data is accessible online.[16] To the reference links provided by the organizers, we add 20 and 123 false links respectively for the nine heterogeneities and the four heterogeneities gold standards, to enable the supervised learning approach implemented by STEM that necessitates both positive and negative entity pairs.

A third dataset used in the experimentation of the STEM approach is derived from the 3cixty Nice Knowledge Base. This knowledge base contains Nice cultural and tourist information (such as Place-type entities) and it is created with a multi datasource collection process, where numerous entities are represented in multiple sources leading to duplicates. This creates the need of matching and the resolution of the entities. Further details of the making of the knowledge base with the selection of the gold standard is detailed in Sec. 7, while in this section we report the statistics of the gold standard that drove the entity matching task.

For the FEIII and the DOREMUS datasets, we assume that the Unique Name Assumption is true, meaning that two data of the same data source with distinct references refer to distinct real world entities. For 3cixty this is clearly not the case, as we are matching the dataset with itself to detect duplicates.

A summary of the datasets statistics is reported in Tab. 1 and of the matching tasks in Tab. 2.

[16] http://islab.di.unimi.it/im_oaei_2016/data/Doremus.zip

5.2. Scoring

To evaluate the efficiency of the algorithm we have used the standard precision p, recall r and f measures [38]. These measures, if not specified otherwise, have been evaluated through a 4-fold cross validation score process. Given the ambiguity of the definition of p, r and f when performing cross validation [49], we hereby specify that we have used:

$$p = \frac{1}{4}\sum_{i=1}^{4} p_i \tag{28}$$

$$r = \frac{1}{4}\sum_{i=1}^{4} r_i \tag{29}$$

$$f = \frac{1}{4}\sum_{i=1}^{4} f_i \tag{30}$$

$$\tag{31}$$

where $i = 1..4$ are the four folds. The computation of the precision score depends on the *closed/open world assumption*, i.e. whether we assume that all correct matches are annotated in the gold standard or not. In practice, it is normal to have in the gold standard only a fraction of all the real matching entities and we thus follow, by default, the open world assumption. In this case, when a pair of entity is considered to be a match by STEM and is not present in the gold standard, we are unable to determine whether this is due to a false positive or to a missing annotation and we simply ignore it in the scoring. In the case of the experiments with DOREMUS datasets, where the organizers of the challenge claim to have annotated all the true matches, we follow the closed world assumption. In this case, every match that is not annotated in the gold standard is considered as a false positive.

5.3. Duke

Entity format: Duke is able to handle different formats for input data, such as .csv (comma separated value) or .nt (n-triples). In the first case, an entity is represented by a record in a table. In the second case,

Dataset	Domain	Provider	Number of entities	Format
FFIEC	Finance	Federal Financial Institution Council	6652	CSV
SEC	Finance	Security and Exchange Commission	129312	CSV
BnF-1	Music	French National Library	32	RDF
PP-1	Music	Philharmonie de Paris	32	RDF
BnF-2	Music	French National Library	202	RDF
PP-2	Music	Philharmonie de Paris	202	RDF
3cixty Nice places	Culture and Tourism	3cixty Nice Knowledge Base	336,900	RDF

Table 1
Datasets summary

Dataset 1	Dataset 2	Gold standard pairs	Challenge	Task
FFIEC	SEC	790	FEIII2016	FFIEC-SEC
BnF-1	PP-1	52	OAEI2016	DOREMUS - 9 heterogeneities
BnF-2	PP-2	325	OAEI2016	DOREMUS - 4 heterogeneities
3cixty Nice places	3cixty Nice places	756	-	-

Table 2
Matching tasks summary

an entity is a node in a Knowledge Base.

Blocking method: we reduce the search space for the entity matching process from the space of all possible pairs of entities AxB using an inverted index, in which property values are the indexes and the tuples are the documents referred by the indexes. The lookup of a tuple given a value has, therefore, a unitary cost. We reduce the search space to a small subset of the most likely matching entity pairs that satisfy a given Damerau-Levenshtein distance [50] for each value pair of the tuples, and we considered the first m candidates.[17]

Configuration: the first step of the implementation consists in configuring Duke. Duke is built by default on top of a Lucene Database,[18] which indexes the records through an inverted index and does full-text queries to find candidates, implementing the blocking strategy. The Lucene Database can be configured in Duke by setting a number of parameters such as the **max-search-hits**, that is the maximum number of candidate records to return or **min-relevance**, namely a threshold for Lucene's relevance ranking under which candidates are not considered. Duke then allows to select a number of properties to be taken into account to establish if a pair of entities match, such as name, address, zip code. Duke requires to specify a mapping between the fields of the data sources and those on which the comparison has to be performed, e.g. "LegalName → NAME, LegalEntityAddress → ADDRESS, LegalEntityCode → ZIPCODE". In this case, we have manually configured Duke during the participation to the FEIII2016 challenge and the choice of cleaners, comparators is reported in [51].

5.4. Silk

Entity format: Silk is specifically built to deal with RDF formats, such as .ttl (turtle) or .nt (n-triples), where entities are represented as nodes in a Knowledge Graph. However, it allows to convert data from a variety of formats, such as .csv (comma separated values).

Blocking method: Silk implements a multidimensional blocking system, called MultiBlock [52], which is able to not lose recall performance. Differently from most blocking system that operates on one dimension, MultiBlock works by mapping entities into a multidimensional index, preserving the distances between entities.

Configuration: Silk can easily be configured through an XML file. To configure the blocking algorithm, it is sufficient to specify the number of blocks, which

[17] We empirically set the distance to 2 and the number of potentially retrievable candidates to 1,000,000 (conservative boundary).
[18] `https://lucene.apache.org`

we have empirically set to 100. A set of properties $i = 1..K$ onto which the matching is based needs to be specified and then, for each of them, the user can select among a large number of 'transformators' (comparable to Duke's cleaners) to pre-process and normalize strings. The choice of transformators and comparators has been based on the result obtained with Duke in the participation to the FEIII challenge and a similar configuration file has been produced for Silk. A manual configuration to optimize the f score has been used also for the DOREMUS data in the context of the OAEI challenge [53].

5.5. Stacking

Differently from the previous steps, which are mainly based on low-level string similarity measures, the supervised learner can implicitly learn semantic similarities from the human annotations of the gold standard. The stacking process is implemented through a Python script that executes Duke or Silk a number N of times, editing the threshold t through uniform perturbations of amplitude a, automatically modifying Duke's or Silk's configuration file. Then, the script saves Duke's or Silk's outputs and turns them into a training set for a supervised learner with $id1, id2$ pairs on the rows and N features on the columns.
The user may choose different supervised learners for the stacking layer. What we have experimentally found to work better, given the small number of features, is an SVM with a RBF kernel [22], which is the default. In many cases, such as the default one, the learning algorithm leaves a number of parameters (so-called "hyper parameters") to be determined. Let $F(x; \hat{w}, \theta)$ be a supervised learner where θ is the vector of hyper parameters (C and γ in the case of SVM with RBF kernel). In order to optimize the efficiency of the algorithm with respect to these hyper parameters, we have trained the algorithm on an array of possible values of θ and selected $\hat{\theta}$ as the vector that optimized 4-fold cross validation score (grid search cross validation [54]).
For what concerns the number of features N, it is reasonable to expect that higher values tend to increase the efficiency of the algorithm up to a saturation point, where no further predicting power is added by an additional instance of the base classifier. Actually, we observe that increasing the number of features can also lead to efficiency decrease, as a typical overfitting problem. This saturation point will typically depend on the amplitude a of perturbation, as with small intervals $-a/2, a/2$ we expect it to occur earlier. This will also depend on the size of the datasets and its complexity, so no one-fits-all solution has been individuated. As we will see in the experiments though, $a = 0.25$ and $N = 5$ appears to be a good rule of thumb.

6. Results

6.1. STEM vs threshold-based classifiers

In this section, we first provide evidence of the trade-off between precision and recall introduced by the decision threshold and then we show that STEM is able to increase the precision and the recall of the base classifiers at the same time. In the following, we refer to the STEM approach implemented on top of Duke as STEM-NB and to that implemented on top of Silk as STEM-LIN.
The premise of this work is that the threshold t in decision rule Eq. 1 introduces a trade-off between precision and recall. In Sec. 3 we have provided a heuristic argument of why this should be the case and now we provide experimental results. In Fig. 4, we report the precision and recall obtained by running Duke on the FFIEC-SEC dataset for a set of 20 equally spaced threshold values $t \in [0.05, 0.9]$. The graph clearly shows the trade-off between precision and recall of the algorithm ruled by the threshold t. The trend for both curves is non-linear, with moderate changes in the central part and sudden variations at the sides. The typical configuration process of a threshold-based classifier attempts to find a balance between the two metrics, in order to maximize the F-score of the algorithm. Then, using STEM, both metrics can be increased at the same time using stacking. In Tab. 3 and Tab. 4 we support this claim by reporting respectively the results obtained using STEM-NB and STEM-LIN on FFIEC-SEC, DOREMUS 4-heterogeneities, DOREMUS 9-heterogeneities tasks, varying the number of features N. The value of the perturbation amplitude a has been fixed to $a = 0.25$ following the analysis reported in Fig. 5, which shows that this value allows to reach $f = 0.95$ with only 10 configurations and limits the dependence on the value of N. The plot also shows that the saturation effect tends to occur sooner when a is small, as this corresponds to a denser and therefore less informative sampling of the interval.
In Tab. 3, we can observe that, for the FFIEC-SEC task, even with a small number of features $N = 5$, stacking leads to a significant increase of the F-score

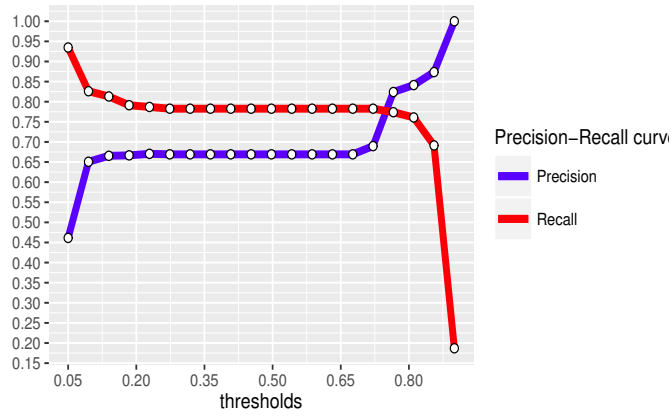

Fig. 4. Precision and recall curves as functions of the threshold t for Duke on the FEIII dataset. It clearly shows the trade-off between $p(t)$ and $r(t)$ introduced by the Naive Bayes classifier decision rule Eq. 23

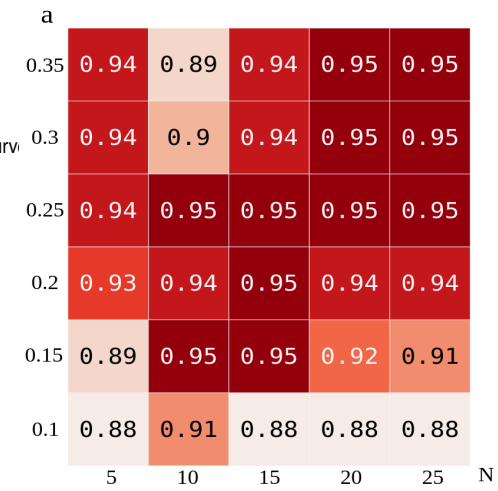

Fig. 5. F1 score for different combinations of a and N on the FFIEC-SEC dataset for STEM-NB

of the algorithm (12%), obtained by increasing both precision and recall at the same time. Increasing the number of features N tends to increase the efficiency, with a saturation effect as the number gets larger. Indeed, going from $N = 5$ to $N = 10$ only grants a 1% gain and no difference of efficiency is observed from $N = 10$ to $N = 20$. A similar behavior is observed for the DOREMUS tasks, where the big efficiency leap is given by the introduction of stacking, whereas raising the number of features N grants small improvements in the case of 4-heterogeneities and decreases the efficiency in the case of 9-heterogeneities. A similar behavior is observed for all the experiments that we have done.

To show that the increase of efficiency is not dependent on the particular threshold-based classifier, we have run the same experiments using STEM-LIN and reported the results in Tab. 4. In this case, we can observe that, although absolute values are lower, the increase in efficiency given by the stacking layer is equally or even more important, achieving a $+20\%$ on the F-score with only $N = 5$ in the FFIEC-SEC task and a $+43\%$ and $+36\%$ for DOREMUS tasks. Also in this case, both precision and recall are increased at the same time and a saturation effect can be detected as N grows. In general, it seems that the saturation effect occurs earlier for DOREMUS tasks, as in three cases out of four with $N = 5$ we already reach the peak efficiency and the fourth case the efficiency only increases by 1%. This is probably due to the fact that DOREMUS datasets are smaller and thus a model with too many features tends to overfit the data.

6.2. *STEM vs supervised learning on similarity values*

In this section, we discuss the second claim of the paper, namely the comparison of a hybrid approach such as STEM with a system that performs machine learning 'from scratch'. More in detail, we have compared STEM to a number of commonly used machine learning algorithms, using similarity values s_i as features. In addition to verifying whether STEM performs better than the other systems in absolute, the intent is also to see whether it is less dependent on the amount of annotated training data. Indeed, given the quadratic nature of the entity matching problem, in most real usage scenarios, annotating a comprehensive gold standard (such as those of FEIII and DOREMUS) is an extremely time consuming endeavour and the user is able to annotate just a small fraction of all possible entity pairs. Therefore, it is interesting to see how an entity matching system performs with a small amount of annotated training pairs. To this end, we have studied how STEM performs at the variation of the amount of training data with respect to an SVM classifier with a RBF kernel, a random forest and a logistic classifier. In order to avoid possible size effects on the scores, we have split the FEIII data in two halves, according to the stratified sampling technique, i.e. keeping constant the proportion of matching and non matching pairs in the two parts. The first half is used as training data and the second half is used as test data. Then, we randomly extract a fraction z of training data from 0.1 to 0.9, train the systems and score them on the test set,

		FFIEC-SEC				DOREMUS 4-het				DOREMUS 9-het			
Base classifier	**N**	**p**	**r**	**f**	**δf**	**p**	**r**	**f**	**δf**	**p**	**r**	**f**	**δf**
Duke	n/a	0.88	0.77	0.82	0	0.46	0.57	0.51	0	0.48	0.66	0.55	0
STEM-NB	5	0.90	0.98	0.94	12%	0.89	0.98	0.93	42%	**0.97**	**1.0**	**0.99**	**44%**
STEM-NB	10	**0.93**	**0.97**	**0.95**	**13%**	0.89	0.98	0.93	42%	0.94	1.0	0.97	42%
STEM-NB	20	0.94	0.97	0.95	13%	**0.89**	**0.99**	**0.94**	**43%**	0.87	1.0	0.93	35%

Table 3
Results of STEM-NB vs Duke for $a = 0.25$ and different values of N across different datasets

		FFIEC-SEC				DOREMUS 4-het				DOREMUS 9-het			
Base classifier	**N**	**p**	**r**	**f**	**δf**	**p**	**r**	**f**	**δf**	**p**	**r**	**f**	**δf**
Silk	n/a	0.57	0.67	0.59	0	0.45	0.43	0.43	0	0.46	0.81	0.58	0
STEM-LIN	5	0.77	0.81	0.79	20%	**0.82**	**0.93**	**0.86**	**43%**	**0.89**	**1.0**	**0.94**	**36%**
STEM-LIN	10	0.78	0.83	0.80	21%	0.75	0.66	0.69	28%	0.89	1.0	0.94	36%
STEM-LIN	20	**0.77**	**0.84**	**0.81**	**22%**	0.75	0.60	0.64	21%	0.87	1.0	0.93	35%

Table 4
Results of STEM-LIN vs Silk for $a = 0.25$, different values of N across different datasets

which remains the same. For each value of z, we repeat the extraction 50 times and we compute the average value. Using the FEIII datasets and the STEM-NB implementation, values of s_i have been computed using the same comparators with the same configuration of STEM-NB. The configuration procedure of the machine learning classifiers is the same as that described in Sec. 5.5, namely a grid search hyper parameters optimization has been used to maximize 4-fold cross validation scores, setting C and γ for SVM, 'n_estimators' for the random forest and the regularization constant C for logistic regression.[19] The result of the experiment is depicted in Fig. 6. We can see that STEM-NB performs better than any other classifier in absolute terms, reaching a peak of 0.931 when 90% of the training data is used. Moreover, it shows little dependency on the amount of training data, producing 0.914 with only 10% of the training data. SVM performs better than the other pure machine learning approaches when 90% of training data is used, but decreases fast when annotated examples are reduced. In Tab. 5, we report, for each classifier, the quantitative estimation of the dependency of f from the fraction of training data z, obtained through the statistical estimation of the angular coefficient m of a linear fit of the points (i.e. the straight lines of Fig. 6). What we can observe is that more complex models such as SVM and Random Forest tend to depend more on the amount of training

[19] `http://scikit-learn.org/stable/user_guide.html`

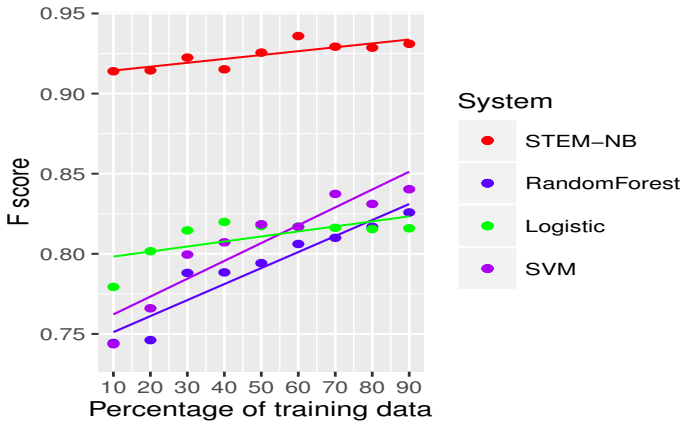

Fig. 6. F-score at the variation of the percentage of training data used. STEM-NB is compared to an SVM classifier, a Random Forest and a logistic classifier

data, while a simple linear model such as logistic regression is performing well even with a small amount of training data. The logistic model is even less dependent on the training data than a hybrid approach such as STEM, but it is not comparable in terms of absolute efficiency. STEM thus represents a model that is complex enough to achieve good efficiency in absolute terms and it is also able to maintain it with a little amount of training data.

6.3. Runtime performance

In Sec. 4.3 we have discussed the computational complexity of STEM, which can be summarized as

Classifier	min	max	m
STEM-NB	0.91	0.93	0.015 ± 0.008
SVM	0.74	0.84	0.09 ± 0.01
Random Forest	0.74	0.83	0.09 ± 0.01
Logistic	0.78	0.82	0.002 ± 0.006

Table 5

Dependency on the amount of training data. 'Min' and 'Max' represent respectively the minimum and maximum F-score and 'm' represents the angular coefficient of a straight line interpolating the points of Fig. 6

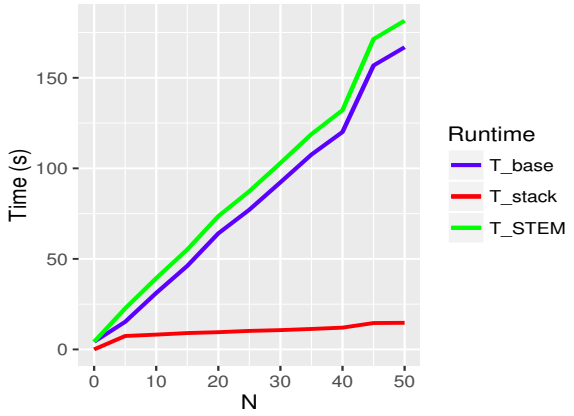

Fig. 7. Runtime for STEM-NB on DOREMUS 4-heterogeneities task with increasing number of features N.

$T_{STEM} \approx N * T_{baseclassifier}(n,m) + T_{stacking}(N,g,k)$. We have also argued that we observed that the time required to run the ensemble of base classifiers is longer than the time required for the stacking layer. In this section, we show an example of such a behavior measuring the runtime of STEM-NB on the DOREMUS 4-heterogeneities dataset with increasing number of features N. In Fig. 7, we can see that the time required to generate the features $T_{base} = N * T_{base\_classifier}$ quickly becomes much more significant than the time required for stacking $T_{stacking}$. A similar behavior has been observed for all the datasets under consideration, suggesting that a parallelization of the feature generation process would greatly improve the runtime performance of STEM as a whole. In general, we also observe a linear trend in both the components $T_{base}(N)$ and $T_{stacking}(N)$, which implies a linear trend for the total time T_{total}, consistently with what we expect from Sec. 4.3.

7. 3cixty Knowledge Base Generation

In this section, we describe the implementation of STEM in the generation of the 3cixty Nice knowledge base, introducing first the key components of the 3cixty data chain.

OVERVIEW 3cixty is a semantic web platform that enables to build real-world and comprehensive knowledge bases in the domain of culture and tourism for cities. The entire approach has been tested first for the occasion of the Expo Milano 2015 [55], where a specific knowledge base for the city of Milan was developed, and is now refined with the development of knowledge bases for other cities, among those Nice. These knowledge bases contain descriptions of events, places (sights and businesses), transportation facilities and social activities, collected from numerous static, near- and real-time local and global data providers, including Expo Milano 2015 official services in the case of Milan, and numerous social media platforms. The generation of each city-driven 3cixty knowledge base follows a strict data integration pipeline, that ranges from the definition of the data model, the selection of the primary sources used to populate the knowledge base, till the entity matching used for distilling the entities forming the final stream of cleaned data that is then presented to the users via multi-platform user interfaces. A detailed description of the technology leading to the creation of the knowledge base is summarized in [59]. In the remainder of this section we introduce the data model and the data sources used in 3cixty. We then detail the entity matching process, which is performed using STEM, describing the experimental setup, gold standard, and results.

DATA MODEL The 3cixty ontology design principle has focused on optimizing the coverage of the terminology in the context of culture and tourism. For each entity to model, such as Place-type and Event-type, we looked for existing knowledge resources (keyword

search) in LOV,[20] Swoogle,[21] Watson,[22] and the Smart City catalogue.[23] We established a rigid search mechanism where two domain experts analyzed the ontology resources that resulted from the search. In detail, they first agreed on a set of keywords to be utilized in the search. The set contains: Event, Place, Transportation, Tourism, Culture, City, Smart City. Then, each expert analyzed the results of the search through the different platforms noting in a spreadsheet the classes and related properties most valuable for the vertical domain of the 3cixty knowledge bases. The two experts compared manually the two spreadsheets and resolved any conflict of representation by optimizing the selection criteria are:

- the popularity of classes/properties based on usage data,
- clarity of descriptions,
- and favoring schema.org when suitable.

Once consensus was reached, classes and predicates were taken and added to the 3cixty data model, which therefore consists in a constellation of existing ontologies. We re-used some classes and properties from the following ontologies: `dul`,[24] `schema`,[25] `dc`,[26] `lode`,[27] `geo`,[28] `transit`,[29] and `topo`.[30] A few additional classes and properties have been created to describe travel distances: we defined origin, distance, travel time, the nearest metro station and bike station. Details are available in [56].

DATA SOURCES The 3cixty knowledge bases contain information about places, events, artists, transportation means, and user-generated content such as media and reviews. The knowledge bases are built using three types of data sources:

- local sources usually offered by city open data portals,
- global sources such as social media platforms,
- editorial data generated by experts of the domain.

The selection of the sources follows a strict protocol that involves two teams of experts who analyze and rank the data sources at disposal to decide which ones were important to be selected for being included in the knowledge base. The teams are both composed of two researchers in the field of data integration and the Semantic Web and two local data experts, knowledgeable of technicalities concerning data publishing, integration and data exploitation. The experts are asked to maximize a 3-objective function: data semantics, instance coverage, and real-time update. The output of such an investigation leads to a survey, which is cross-validated by two domain experts who decide by consensus and iterate in checking existing and new data sources according to the aforementioned objectives, updating the list continuously.

DATA RECONCILIATION The data reconciliation problem is addressed via both category reconciliation and instance reconciliation. The rationale of having both types of reconciliation is to improve the data consumer perceived data quality by removing both category and instance duplicates. In both cases, the reconciliation processes have been applied to the two main topical types of entities in the knowledge base: Events and Places. Such a stage is at the core of the knowledge base creation, since the consumption of the data from the knowledge base is highly polarized by a clean feed of data where no duplicates or near-duplicates are shown.

Reconciling categories has the objective to reduce sparsity in the use of different labels for the same category groups. We addressed the process by using two category thesauri (implemented in `skos`) as pivots: the Foursquare taxonomy[31] for Places, and the taxonomy used in [58] for Events. The alignment, led by two experts of the domain, has established a set of links from the gathered categories, using `skos:closeMatch` and `skos:broadMatch`. An automatic process is then used to identify links according to the exact match of the found categories with the alignment defined by the experts.

Given two data sources, namely A and B, an instance reconciliation process looks at identifying data instances that are similar according to their semantics and thus linking them with *sameAs* links. In 3cixty, we have implemented the instance reconciliation task us-

[20] `http://lov.okfn.org/dataset/lov`
[21] `http://swoogle.umbc.edu`
[22] `http://watson.kmi.open.ac.uk/WatsonWUI`
[23] `http://smartcity.linkeddata.es`
[24] `http://www.loa-cnr.it/ontologies/DUL.owl`
[25] `https://schema.org`
[26] `http://purl.org/dc/elements/1.1/`
[27] `http://linkedevents.org/ontology`
[28] `http://www.w3.org/2003/01/geo/wgs84_pos`
[29] `http://vocab.org/transit/terms`
[30] `http://data.ign.fr/def/topo`
[31] `https://developer.foursquare.com/categorytree`

ing STEM. To do so, we have first generated a gold standard for training the STEM stacked machine learning (Sec. 7.1), and then validated its efficiency (Sec. 7.2).

Using the findings reported in [58] we have listed the instance fields used for the entity matching process, in detail: for Place-type instances the set $P = (label, geo, address)$, where *label* is the place name, *geo* are the geographic coordinates according to a fixed bounding box, and *address* in plain text. For Event-type instances the set $E = (label, geo, time)$, where *label* is the place name, *geo* are the geographic coordinates according to a fixed bounding box, and *time* when the event starts and ends.

7.1. Gold Standard Creation

Given the generality of the STEM approach and the data model of the different 3cixty knowledge bases, we have generated a gold standard from the 3cixty Nice KB, i.e. the knowledge base built for the Nice area, to be used to benchmarking the performance of STEM and to be utilized as training set for the other city knowledge bases.

The gold standard has gone through a process of identifying, with a random sampling, a small portion of Place-type pairs[32] to match, totaling 756 pairs. This accounts to a tiny fraction of the entire set of possible pairs (order of 10^9 possible pairs); then, two human experts rated each as a match or as no-match. The annotation process was divided in two steps: `i)` individual annotation, i.e. each expert performed annotations separately; `ii)` adjudication phase, i.e. the two experts compared the annotations and resolved eventual conflicts.

This has prompted the creation of a gold standard that accounts 228 match and 528 no-match pairs.[33]

7.2. Experimental Results

Similarly to what has been done in Sec. 6.1, we compared STEM with Duke. In order to put Duke in the best conditions, we let it learning the best configurations using the active learning built-in function, just giving as input the instance fields to be utilized in the matching task and the gold standard created by the two experts.

The built-in active learning function works as follows: it iterates multiple times changing the configurations of the comparators aiming to minimize the matching error rate. Such a process prompts the creation of a configuration file summarizing the best Duke settings for the dataset used.

Having observed that it performs better than STEM-LIN (Sec. 6), we have then deployed STEM-NB using Duke configured as above and we conducted a 4-fold cross validation. Table 6 shows the results of the experiments. We can observe how STEM with five classifiers holds better results than a single run of Duke with a $\boldsymbol{\delta}$**f** of 20%. We can also observe how the boost STEM introduces is slightly reduced with an increasing number of Duke instances N, similarly to what observed for DOREMUS data. As we mentioned earlier in the paper, this is the typical overfitting problem, where introducing additional complexity in the model does not provide better learning. As a general suggestion, $N = 5$ seems to be enough to obtain a consistent increment of efficiency with respect to the baseline without overfitting the data. The matching process with $N = 5$ took approximately 3 hours on a laptop with 4 cores and 12GB of RAM.

8. Conclusion

In this paper, we have proposed a framework for stacking threshold-based entity matching algorithms. We have argued and then shown empirically that the final decision threshold, which converts the confidence score of a threshold-based classifier into a decision, introduces a trade-off between the precision and the recall of the algorithm. Using stacking, we have demonstrated that this trade-off can be broken, as the combination of the predictions of an ensemble of classifiers with different threshold values can raise both metrics at the same time, resulting in a significant enhancement of the matching process. This enhancement is not bound to the type of classifier nor to the dataset used, as we observe consistent results for both a linear and a Naive Bayes classifier on three different datasets. Generally, using five classifiers is enough to obtain a consistent increase in performance and increasing the number of classifiers N can easily lead to overfitting, providing small improvements or even decreasing the accuracy of the predictions.

STEM is independent from the configuration pro-

[32] For the sake of brevity we report the entity matching process of the Place-type entities

[33] We aim to share the Gold Standard once the paper is published to foster the reuse and experimental reproducibility.

Base classifier	**N**	**p**	**r**	**f**	**δf**
Duke	n/a	0.76	0.65	0.70	0
STEM-NB	**5**	**0.90**	**0.92**	**0.90**	**20%**
STEM-NB	10	0.76	0.81	0.78	8%
STEM-NB	20	0.79	0.81	0.79	9%

Table 6
Results of STEM-NB vs Duke on the 3cixty Nice dataset for $a = 0.25$ and different values of N.

cess of the threshold-based classifier. Indeed, we have provided three experimental evaluations and in two of them we have manually configured the system, whereas in the third we have used an active learning approach. A further advantage of the STEM approach is the little dependency on the amount of training data, as we have shown that it can reach high levels of performance even with a small fraction of annotated data. STEM has allowed to greatly improve the data reconciliation process of the generation of the 3cixty knowledge base, proving to be accurate, reliable and scalable, as the reconciliation process lasted roughly 3 hours. STEM is computationally more expensive than a single threshold-based classifier, as it involves running several instances of the threshold-based classifier and then training a supervised learner on top of their matching decisions. Among these two components of the total STEM runtime, we observe that for the datasets that we have used in this work, the first is the most expensive step. Thus, as a future work, we plan to improve the computing time of the software using a parallel and/or distributed implementation to allow the simultaneous execution of processes. We also plan to extend the ensembling process to other relevant parameters and to other threshold-based entity matching systems and to experiment different supervised learners.

9. Acknowledgments

This work was partially supported by the innovation activity 3cixty (14523) and PasTime (17164) of EIT Digital Cities.

References

[1] J. Euzenat and P. Shvaiko, *Ontology Matching*, Springer, 2013
[2] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S.E. Whang, J. Widom, *Swoosh: a generic approach to entity resolution*, The VLDB Journal - The International Journal on Very Large Data Bases, vol. 18, pp. 255-276, 2009.
[3] A. Thor and E. Rahm , *MOMA - A Mapping-based Object Matching System.*, CIDR, pp. 247-258, 2007
[4] R. Isele, C. Bizer, *Learning linkage rules using genetic programming*, Proceedings of the 6th International Conference on Ontology Matching-Volume 814, pp. 13-24, 2011
[5] A.C.N. Ngomo, K. Lyko, *Eagle: Efficient active learning of link specifications using genetic programming*, Extended Semantic Web Conference, pp. 149-163, 2012
[6] D.H. Wolpert, *Stacked generalization*, Neural Networks, vol. 5, pp. 241-259, 1992
[7] D.D. Lewis, *Naive (Bayes) at forty: The independence assumption in information retrieval*, Machine learning: ECML-98, 4-15, 1998
[8] V. Metsis, I. Androutsopoulos, and G. Paliouras, *Spam filtering with naive bayes-which naive bayes?*, CEAS, pp. 27-28, 2006
[9] J. Volz, C. Bizer, M. Gaedke and G. Kobilarov, Silk *A link discovery framework for the web of Data*, 2nd Workshop about Linked Data on the Web, Madrid, Spain, 2009
[10] Chih-Chung Chang and Chih-Jen Lin., Libsvm: a library for support vector machines., ACM transactions on intelligent systems and technology (TIST), 2(3):27, 2011.
[11] Léon Bottou and Chih-Jen Lin., Support vector machine solvers., Large scale kernel machines, 3(1):301–320, 2007.
[12] W.W. Cohen, H. Kautz, and D. McAllester, *Hardening soft information sources*, Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, 255-259, 2000
[13] H.B. Newcombe, and J.M. Kennedy, *Record linkage: making maximum use of the discriminating power of identifying information*, Communications of the ACM, vol. 5, 563-566, 1962
[14] H. Köpcke and E. Rahm, *Frameworks for entity matching: A comparison*, vol. 69, pp. 197-210, 2010
[15] Leitão, Luís and Calado, Pável and Weis, Melanie, *Structure-based inference of XML similarity for fuzzy duplicate detection*, Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, pp. 293-302, 2007
[16] Mikhail Bilenko and Raymond J Mooney., Adaptive duplicate detection using learnable string similarity measures., Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 39–48. ACM, 2003.
[17] S. Tejada, C.A. Knoblock and S. Minton, *Learning domain-independent string transformation weights for high accuracy object identification*, Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 350-359, 2002
[18] S. Tejada, C.A. Knoblock, and S. Minton, *Learning object identification rules for information integration*, Information Systems, vol. 26, 8, 607-633, 2001

[19] M.G. Elfeky, V.S. Verykios and A.K. Elmagarmid, *TAILOR: A record linkage toolbox*, Proceedings. 18th International Conference on Data Engineering, 2002, pp. 17-28, 2002

[20] Hanna Köpcke and Erhard Rahm., *Training selection for tuning entity matching.*,QDB/MUD, pages 3-12, 2008.

[21] T. Soru and A.C.N. Ngomo, *A comparison of supervised learning classifiers for link discovery*, 10th International Conference on Semantic Systems, pp. 41-44, 2014

[22] T. Hastie, R. Tibshirani, J. Friedman and J. Franklin, *The elements of statistical learning: data mining, inference and prediction*, The Mathematical Intelligencer, vol. 27, 83-85, 2005

[23] H. Zhao, S. Ram, *Entity identification for heterogeneous database integration - a multiple classifier system approach and empirical evaluation*, Information Systems, vol. 30, 119-132, 2005

[24] Z. Chen, D.V. Kalashnikov and S. Mehrotra, *Exploiting context analysis for combining multiple entity resolution systems*, Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, 207-218, 2009

[25] C. Bizer, T. Heath, and T. Berners-Lee, *Linked data - the story so far*, Semantic Services, Interoperability and Web Applications: Emerging Concepts, 205-227, 2009

[26] M. Schmachtenberg, C. Bizer and H. Paulheim, *Adoption of the linked data best practices in different topical domains*, International Semantic Web Conference, 245-260, 2014

[27] M. Nentwig, M. Hartung, A.C.N. Ngomo, and E. Rahm, *A survey of current Link Discovery frameworks*, Semantic Web Journal, pp. 1-18, 2015

[28] J. Volz, C. Bizer, M. Gaedke and G. Kobilarov, *Discovering and maintaining links on the web of data*, International Semantic Web Conference, 650-665, 2009

[29] A.C.N. Ngomo and S. Auer, *Limes - a time-efficient approach for large-scale link discovery on the web of data*, Integration, vol. 15, 2011

[30] Sherif, Mohamed Ahmed and Ngomo, Axel-Cyrille Ngonga and Lehmann, Jens, *WOMBAT–A Generalization Approach for Automatic Link Discovery*, European Semantic Web Conference, pp. 103-119, 2017

[31] L. Zhu, M. Ghasemi-Gol, P. Szekely, A. Galstyan and C.A. Knoblock, *Unsupervised Entity Resolution on Multi-type Graphs*, International Semantic Web Conference, 649-667, 2016

[32] H. Pasula, B. Marthi, B. Milch, S. Russell, I. Shpitser, *Identity uncertainty and citation matching*, Advances in neural information processing systems, pp. 1401-1408, 2002

[33] P. Cudré-Mauroux, P. Haghani, M. Jost, and K. Aberer, H. De Meer,*idMesh: graph-based disambiguation of linked data*, Proceedings of the 18th international conference on World wide web, 591-600, 2009

[34] X. Dong, A. Halevy, and J. Madhavan, *Reference reconciliation in complex information spaces*, Proceedings of the 2005 ACM SIGMOD international conference on Management of data, pp. 85-96, 2005

[35] Mustafa Al-Bakri, Manuel Atencia, Jérôme David, Steffen Lalande, and Marie-Christine Rousset., *Uncertainty-sensitive reasoning for inferring sameas facts in linked data.*, In 22nd european conference on artificial intelligence (ECAI), pp 698-706. IOS press, 2016.

[36] Fatiha Saıs, Nathalie Pernelle, and Marie-Christine Rousset., L2r: A logical method for reference reconciliation., In Proc. AAAI, pp 329-334, 2007.

[37] I.P. Fellegi, and A.B. Sunter, *A theory for record linkage*, Journal of the American Statistical Association, vol. 64, 1183-1210, 1969

[38] D.M. Powers, *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation*, Journal of Machine Learning Technologies, 2011

[39] C.D. Manning and H. Schütze, *Foundations of statistical natural language processing*, vol.999, 1999

[40] V. Raghavan, P. Bollmann and G.S. Jung, *A critical investigation of recall and precision as measures of retrieval system performance*, ACM Transactions on Information Systems (TOIS), vol. 7, 205-229, 1989

[41] W. Cohen, P. Ravikumar and S. Fienberg, *A comparison of string metrics for matching names and records*, KDD Workshop on data cleaning and object consolidation, vol. 3, 73-78, 2003

[42] R. Isele, C. Bizer, *Active learning of expressive linkage rules using genetic programming*, Web Semantics: Science, Services and Agents on the World Wide Web, vol.23, pp. 2-15, 2013

[43] I. Rish, *An empirical study of the naive Bayes classifier*, IJCAI Workshop on Empirical Methods in Artificial Intelligence, pp. 41-46, 2001

[44] E. Croot, *Bayesian spam filter*, `http://people.math.gatech.edu/~ecroot/bayesian_filtering.pdf` - Last visited: 10 January 2017

[45] K.P. Murphy, *Machine learning: a probabilistic perspective*, MIT press, 2012

[46] I. Androutsopoulos, J. Koutsias, K.V. Chandrinos, and C.D. Spyropoulos, *An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages*, 23rd International Conference on Research and development in Information Retrieval (SIGIR), 160-167, 2000

[47] A. McCallum, K. Nigam and others, *A comparison of event models for naive bayes text classification*, AAAI Workshop on Learning for Text Categorization, vol. 752, pp. 41-48, 1998

[48] S. Sarawagi, A. Bhamidipaty, *Interactive deduplication using active learning*, 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), vol.3, pp. 269-278, 2003

[49] G. Forman, M. Scholz, *Apples-to-apples in cross-validation studies: pitfalls in classifier performance measurement*, ACM SIGKDD Explorations Newsletter, vol.12, pp. 49-57, 2010

[50] G.V. Bard, *Spelling-error Tolerant, Order-independent Pass-phrases via the Damerau-levenshtein String-edit Distance Metric*, 5th Australasian Symposium on ACSW Frontiers, 2007

[51] E. Palumbo, G. Rizzo and R. Tröncy, *An Ensemble Approach to Financial Entity Matching for the FEIII 2016 Challenge*, DSMM'16: Proceedings of the Second International Workshop on Data Science for Macro-Modeling, 2016

[52] R. Isele, A. Jentzsch, and C. Bizer, *Efficient Multidimensional Blocking for Link Discovery without losing Recall*, WebDB, 2011

[53] M. Achichi, M. Cheatham, Z. Dragisic, J. Euzenat, D. Faria, A. Ferrara, G. Flouris, I. Fundulaki, I. Harrow, V. Ivanova and others, *Results of the Ontology Alignment Evaluation Initiative 2016*, 11th ISWC workshop on ontology matching (OM), pp. 73-129, 2016

[54] C.W. Hsu, C.C. Chang, C.J. Lin and others, *A practical guide to support vector classification*, `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf` - Last visited: 10 January 2017, 2003

[55] G. Rizzo, R. Troncy, O. Corcho, A. Jameson, J. Plu, J.C. Ballesteros Hermida, A. Assaf, C. Barbu, A. Spirescu, K. Kuhn, I. Celino, R. Agarwal, C.K. Nguyen, A. Pathak, C. Scanu, M. Valla, T. Haaker, E.S. Verga, M. Rossi and J.L. Redondo Garcia, *3cixty@Expo Milano 2015: Enabling Visitors to Explore a Smart City*, 14^{th} International Semantic Web Conference (ISWC), Semantic Web Challenge, 2015

[56] G. Rizzo, O. Corcho, R. Troncy, J. Plu, J.C. Ballesteros Hermida, A. Assaf, *The 3cixty Knowledge Base for Expo Milano 2015: Enabling Visitors to Explore the City*, $8^{t}h$ International Conference on Knowledge Capture (K-CAP), 2015

[57] N. Mihindukulasooriya, G. Rizzo, R. Troncy, O. Corcho and R. Garcia-Castro, *A Two-Fold Quality Assurance Approach for Dynamic Knowledge Bases: The 3cixty Use Case*, International Workshop on Completing and Debugging the Semantic Web, 2016

[58] H. Khrouf, V. Milicic and R. Troncy, *Mining events connections on the social web: Real-time instance matching and data analysis in EventMedia*, Web Semantics: Science, Services and Agents on the World Wide Web, vol. 24, pp. 3-10, 2014

[59] R. Troncy, G. Rizzo, A. Jameson, O. Corcho, J. Plu, E. Palumbo, J. C. Ballesteros Hermida , A. Spirescu, K. Kuhn, C. Barbu, M. Rossi, I. Cellino, R. Agarwal, C. Scanu, M. Valla, T. Haacker, *3cixty: Building comprehensive knowledge bases for city exploration*, Web Semantics: Science, Services and Agents on the World Wide Web, 2017