A Comparison of Supervised Learning Classifiers for Link Discovery

Tommaso Soru
AKSW, Department of Computer Science
Augustusplatz 10
D-04109 Leipzig, Germany
tsoru@informatik.uni-leipzig.de

Axel-Cyrille Ngonga Ngomo
AKSW, Department of Computer Science
Augustusplatz 10
D-04109 Leipzig, Germany
ngonga@informatik.uni-leipzig.de

ABSTRACT

The detection of links between resources is intrinsic to the vision of the Linked Data Web. Due to the mere size of current knowledge bases, this task is commonly addressed by using tools. In particular, manifold link discovery frameworks have been developed. These frameworks implement several different machine-learning approaches to discovering links. In this paper, we investigate which of the commonly used supervised machine-learning classifiers performs best on the link discovery task. To this end, we first present our evaluation pipeline. Then, we compare ten different approaches on three artificial and three real-world benchmark data sets. The classification outcomes are subsequently compared with several state-of-the-art frameworks. Our results suggest that while several algorithms perform well, multilayer perceptrons perform best on average. Moreover, logistic regression seems best suited for noisy data.

Categories and Subject Descriptors

M.0 [**Knowledge Management**]: Knowledge Acquisition; H.4 [**Information Systems Applications**]: Miscellaneous; I.2.6 [**Computing Methodologies**]: Learning

General Terms

Semantic Web, Linked Data, Link Discovery, Instance Matching, Machine Learning, Benchmarks

1. INTRODUCTION

Over the last years, the development of tools to support the implementation of all Linked Data principles has been regarded as central for the uptake of Linked Data [1]. In particular, several tools have been developed to support the fourth Linked Data principle, i.e., the creation of links between datasets. Formally, the general aim of link discovery is to find the set of resource pairs $(s, t) \in S \times T$ such that $R(s, t)$ holds, where R is a given relation such as `owl:sameAs` or `dbp:near`[1] and S and T are sets of resources. Finding this set is a non-trivial task which is commonly approximated by finding the set $M \subseteq S \times T$ of resource pairs such that $\forall(s,t) \in M : sim(s,t) \geq \theta$, where $\theta \in [0,1]$ is a similarity threshold and *sim* is a complex similarity function; *sim* consists of a combination of atomic similarity functions σ_i, each of which compares the property values p_i of s (denoted $s.p_i$) and q_i of t (denoted $t.q_i$).

Overall, two main categories of learning algorithms have been implemented so far to support the computation of the set M: *Supervised algorithms* for linking take two sets of resources (called the *source set* and the *target set*) as well as positive and negative examples for links as input. These algorithms then return a set of pairs of resources that they assume to be links. *Unsupervised algorithms* on the other hand only take two sets of resources (i.e., need no training data) and return links. To this end, most of the unsupervised approaches [20, 19] implement an objective function that they aim to optimize. While unsupervised approaches have been shown to perform well on datasets where 1-1 mappings exist between the source and the target sets, most of the current tools implement supervised approaches for link discovery. Still, the performance of the different supervised machine-learning paradigms for link discovery has not been systematically analyzed so far.

We address this research gap by comparing ten different supervised machine-learning paradigms on six different datasets. The goal of this evaluation is to determine the answer to the following questions:

Q_1: Which of the paradigms achieves the average best F-measure?

Q_2: Which of the paradigms is most robust against noise?

Q_3: Which of the methods is the most time-efficient?

To this end, we use the well-known n-fold cross-validation experimental setting to evaluate commonly used machine-learning paradigms and present the precision, recall and F-measure they achieve. Moreover, we evaluate their time-efficiency by measuring their runtime over all six datasets. Surprisingly, our results suggest that multilayer perceptrons perform best on the link discovery tasks at hand. Although they have been already used for performing inductive semantic classification [7], to the best of our knowledge they have not been tested before on link discovery. From our

[1] We use the prefixes defined at `http://prefix.cc`

SEM '14 September 04 - 05 2014, Leipzig, AA, Germany
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2927-9/14/09 ...$15.00.
http://dx.doi.org/10.1145/2660517.2660532

results, multilayer perceptrons perform best on average of clean data. Logistic regression on the other hand performs best on noisy data. The two approaches are followed by decision-table-based algorithms.

2. EVALUATION PIPELINE

We now show each step of the evaluation pipeline. Starting from a source and a target dataset containing individuals, we aim at comparing the performance of different classifiers on the link discovery task. First, the join set between the two dataset is created. Thereafter, the alignment among properties and the perfect mapping among the datasets are loaded to label each instance in the join set and to compute its similarity. Subsequently, a 10-fold cross-validation is applied. Ten supervised machine-learning classifiers are trained on the training sets. Finally, these classifiers are evaluated on the test sets and their scores averaged.

A join between two datasets is basically composed by their Cartesian product, i.e. by all pairs $(s,t) \in S \times T$, where $s \in S$ and $t \in T$. As previously mentioned, each pair in the join represents the input for the similarity function *sim*. After the join creation, which is a preprocessing phase, property alignments and the perfect mapping are imported. Being $P = \{p_i, \dots p_n\}$ a set of source properties and $Q = \{q_i, \dots q_m\}$ a set of target properties, a property alignment set is an ordered set $A \subseteq P \times Q$. In other words, a source property can be aligned to an arbitrary number of target properties, and vice-versa. For instance, given the property alignments $\{(p_1,q_1),(p_1,q_2)\}$, the overall atomic similarity value is the mean value of atomic similarities $\sigma(s.p_1, t.q_1)$ and $\sigma(s.p_1, t.q_2)$. Property alignments are usually carried out by domain experts, as well as the perfect mapping. For each example (s,t), we evaluate the atomic similarities among datatype values $(s.p, t.q)$ for all $(p,q) \in A$. In case properties (p,q) are object properties, the alignment holds among the properties connected to the respective object nodes. String values were compared using weighted trigram and cosine similarity, as well as weighted edit distance. In this paper, we refer to the definition of weighted edit distance proposed in [24] using error probabilities on confusion matrices as weights (see [25]). Numerical values were compared using a logarithm-based similarity. After being computed, all similarity values are normalized into the interval $[0,1]$.

3. EVALUATION

3.1 Experimental Setup

We evaluated ten machine learning techniques against six different datasets. Four classifiers belong to the linear non-probabilistic class (*linear* and *polynomial support vector machines*, with and without *sequential minimal optimization (SMO)* and *linear regression*), four of them rely on probability theory (*logistic regression*, *naïve Bayes*, *random tree* and *J48*), one of them is based on neural networks (*multilayer perceptron*) and one is rule-based (*decision table*). The first three datasets D1, D2 and D3 belong to the Ontology Alignment Evaluation Initiative (OAEI) 2010 Benchmark[2] and are synthetic in nature. In order to verify the scalability of the algorithms as well as their performance on real data, we evaluated them on three larger datasets which contained real data. Dataset D4, which contains over 6 million joins, belongs to the bibliographic domain, while D5 and D6 belong to the e-commerce domain. The datasets D4, D5 and D6 are part of the Benchmark for Entity Resolution[3] [11]. The link discovery community relies on the information retrieval concepts of precision, recall and F_1-measure. We will thus report the F_1-*measure* (or *F-score*) in the following.

The Java source code used within this evaluation is available online[4]. Among others, we utilized LibSVM[5] for the linear and polynomial support vector machines and the Weka Java APIs[6] for the remaining eight machine learning techniques. For SVMs, we found the following values using a grid search: $C = 10^6$ for D1:D4, $C = 10^3$ for D5:D6, $\epsilon = 10^{-3}$, $\gamma = 1$. Moreover, we chose a third-degree polynomial kernel ($d = 3$). We used a 10-fold cross-validation across all experiments and set the test significance level to .05. All experiments were carried out on a 64-bit Ubuntu Linux machine with 8GB of RAM and a 2.5 GHz CPU.

3.2 Experimental Results

The F-measures achieved by all approaches at hand are shown in Table 1.[7] As seen, all classifiers performed excellently on dataset D1, achieving high F-scores above 97%. All classifiers except naïve Bayes (∼35%) performed well on dataset D2. Noteworthy is the perfect classification (100%) carried out by Linear SMO on D1 and the decision table algorithm on D2. The same excellent result has been achieved on dataset D3 by multilayer perceptrons, linear regression and decision tables. Here, only the random tree classifier remains below 90% F-score. Dataset D4 reflects the results obtained on D2: While the naïve Bayes classifier plays as lonely outlier (∼29%), all other classifiers reach an F-score between 92% and 98%. Results on both e-commerce datasets (D5 and D6), allow to analyse the behaviour of the algorithms considered herein on noisy datasets that are very hard to learn [11, 24]. The fifth dataset highlights a drawback on the classification task for naïve Bayes classifiers (∼3%). Interestingly, while basic linear SVMs perform poorly (∼27%), SVMs with SMO outperform the other classifiers. Being SVM and SMO two approaches to the same problem, we consider the different implementation from the respective libraries as the reason of this result. The last dataset follows the average trend, except for the decision table classifier, which ranks second-last with less than 30% F-score. Multilayer perceptron sets the highest performance result with ∼43%, tailed by logistic regression (∼42%) and random tree (∼41%). The difficulty to linearly separate few and sparse positives among a much higher number of negatives led many classifiers to fail on D5-D6.

Linear and polynomial SVM appeared to be one order of magnitude faster than the other approaches on D1-D2. The same performance was achieved by random trees and linear SVM on D4 and by J48 on D5, while polynomial SVM was found to be significantly slower than the average trend on

[2] http://oaei.ontologymatching.org/2010/benchmarks

[3] http://goo.gl/bvWBjA

[4] https://github.com/mommi84/BALLAD

[5] http://www.csie.ntu.edu.tw/~cjlin/libsvm

[6] http://www.cs.waikato.ac.nz/ml/weka

[7] Further results were collected and inserted into a technical report at http://mommi84.github.io/BALLAD

Table 1: F$_1$-scores for ten different classifiers on six test datasets are shown, ranked by average F$_1$-score.

Rank	Classifier	D1	D2	D3	D4	D5	D6	Average
1.	Multilayer Perceptron	99.50%	99.50%	**100.00%**	97.43%	35.58%	**43.49%**	**79.25%**
2.	Logistic Regression	99.90%	98.12%	96.67%	97.71%	40.64%	41.92%	79.16%
3.	Linear SMO	**100.00%**	98.73%	**100.00%**	92.58%	**46.63%**	31.39%	78.22%
4.	Decision Table	97.98%	**100.00%**	**100.00%**	97.66%	42.44%	29.66%	77.96%
5.	J48	99.50%	95.56%	98.29%	97.66%	44.28%	31.53%	77.80%
6.	Linear Regression	99.30%	96.92%	**100.00%**	96.36%	37.06%	36.84%	77.75%
7.	Random Tree	97.45%	99.24%	89.89%	96.82%	39.38%	41.03%	77.30%
8.	Linear SVM	99.40%	98.99%	97.75%	**97.81%**	27.06%	39.18%	76.70%
9.	Polynomial-3 SVM	99.40%	93.76%	98.29%	97.67%	37.28%	31.69%	76.35%
10.	Naïve Bayes	97.75%	35.05%	95.19%	29.47%	2.92%	11.90%	45.38%

`D5-D6`. The computation runtime for the fastest classifier obtained an order of magnitude of 1s (`D3`), 10s (`D1-D2-D6`) and 100s (`D4-D5`) respectively. A detailed runtime table is shown in the technical report stated above.

Table 1 is sorted by average F-score, which can be seen in the last column. Multilayer perceptrons performed best among all, reaching the top place both including (79.25%) and excluding (99.11%) the noisier datasets `D5` and `D6`. Again excluding these datasets, decision tables ranked second (98.91%). However, logistic regression outperformed decision tables on overall average (79.16% against 77.96%), as well as linear SMO (78.22%). Linear regression obtained almost the same results as linear SMO, decision tables and random trees, although random trees achieved 3% less if we exclude noisy datasets. On average, linear SVM (76.70%) carried out a better classification than polynomial SVM (76.35%). Naïve Bayes had the worst results in both cases, reaching only an average F-score of 45.38%. The dependence among property values is likely one of the reasons of such failure, especially on real datasets. In fact, a Naïve Bayes classifier considers all features as independent from each other.

Overall, our results hint that while some average trends for machine-learning-based link discovery can be suggested, no algorithm outperforms all other significantly. The results on the non-noisy datasets suggest that multilayer perceptrons should be used as default learning approach for most datasets. Should they fail, we suggest using linear SVM and then decision tables. This answers Q_1. The results on the noisy datasets suggest that random trees and multilayer perceptrons are most robust against noise and answer Q_2. Thus, we suggest using these two approaches first when confronted with noise before contemplating other approaches. Finally, while all approaches scale well and random trees are fastest on 3 of the datasets we considered, we still suggest using multilayer perceptrons as they achieve acceptable runtimes overall. This answers Q_3.

3.3 Comparison with existing frameworks

We compared our results with state-of-the-art frameworks which have been run on the six test datasets. The first three evaluation results belong to the OAEI2010 initiative, wherein the instance-matching implementation of the refalign [22] framework won the *benchmark* contest carrying out a perfect matching on the synthetic datasets `D1:D3`. ASMOV [9] and AgreementMaker [5] achieved excellent results (100%; 99%) on `D1` and good results (94%; 89%) on `D2`, however the latter had a poor accuracy on the third dataset (70%). The unsupervised version of the EAGLE approach [18] yields state-of-the-art results on the synthetic datasets and performs best on `D4`. MARLIN [2] yielded better results using SVM than using AD-Tree on all real datasets. Although their scores are satisfiable, none of these approaches achieved state-of-the-art accuracies. ACIDS [24], COSY (an undisclosed commercial system for entity resolution), and FEBRL SVM [3] currently represent the state of the art for `D4`, `D5`, and `D6` respectively. Our results showed that even using simple settings, some classifiers presented in this paper achieved state-of-the-art results on `D1:D4`. Moreover, while single machine-learning approaches perform well on single tasks, the combination of several classifiers, kernels and techniques can achieve optimized results and better accuracies.

4. RELATED WORK

Overall, two main problems need to be tackled to address the tool support of link discovery. First, naïve implementations of link discovery approaches have a quadratic *time complexity*. This problem has been addressed in manifold ways: An extensive amount of literature has been published by the database community (see [10, 6] for surveys). In particular, time-efficient deduplication algorithms such as PPJoin+ [28], EDJoin [27], PassJoin [12] and TrieJoin [26] were developed over the last years. Several of these were then integrated into the hybrid link discovery framework LIMES [15]. Moreover, dedicated time-efficient approaches were developed for link discovery. For example, RDF-AI [23] implements a five-step approach that comprises the preprocessing, matching, fusion, interlink and post-processing of data sets. [16] presents an approach based on the Cauchy-Schwarz inequality that allows discarding a large number of unnecessary computations. The approaches HYPPO [13] and $\mathcal{HR}^3$ [14] rely on space tiling in spaces with measures that can be split into independent measures across the dimensions of the problem at hand. Standard blocking approaches were implemented in the first versions of SILK and later replaced with MultiBlock [8], a lossless multi-dimensional blocking technique. KnoFuss [20] also implements blocking techniques to achieve acceptable runtimes.

In addition to addressing the runtime of link discovery, several machine-learning approaches have been developed to learn link specifications (also called linkage rules) for link discovery. For example, machine-learning frameworks such as FEBRL [3] and MARLIN [2] rely on models such as Support Vector Machines [4] and decision trees [21] to detect classifiers for record linkage. RAVEN [17] combines active learning version and perceptron learning to detect linear or Boolean classifiers. The EAGLE approach [18] combines active learning and genetic programming to detect link specifications. KnoFuss [20] goes a step further and presents an unsupervised approach based on genetic programming for finding accurate link specifications.

5. CONCLUSION

In this paper, we presented a systematic evaluation of ten different classifiers for supervised learning on the link discovery task. We use three real and three artificial datasets to compare the approaches. On average, all approaches apart from Naïve Bayes perform well. Still, the results show that multilayer perceptrons are best on average. This result is interesting because to the best of our knowledge, multilayer perceptrons have not been used for link discovery so far. We will thus integrate them into the LIMES framework and make them available via the SAIM interface. Moreover, the performance on linear regression on real datasets make it a viable alternative to multilayer perceptrons when dealing with real-life problems. Interestingly, the behavior of the different approaches on the datasets used in our evaluation suggests that they are complementary. Thus, in order to achieve even better accuracy, we aim to combine them by using ensemble learning techniques. We will investigate how good the same techniques perform in case of small training sets, eventually utilizing a larger amount of similarity measures. Finally, an avenue of interest would be to incorporate a component based on Statistical Relational Learning to exploit information about the RDF graph structure while linking.

6. REFERENCES

[1] S. Auer, J. Lehmann, and A.-C. N. Ngomo. Introduction to linked data and its lifecycle on the web. In *Reasoning Web*, pages 1–75, 2011.

[2] M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *SIGKDD*, KDD '03, pages 39–48, New York, NY, USA, 2003. ACM.

[3] P. Christen. Febrl -: an open source data cleaning, deduplication and record linkage system with a graphical user interface. In *KDD*, pages 1065–1068, 2008.

[4] N. Cristianini and E. Ricci. Support vector machines. In *Encyclopedia of Algorithms*. 2008.

[5] I. F. Cruz, C. Stroe, M. Caci, F. Caimi, M. Palmonari, F. P. Antonelli, and U. C. Keles. Using agreementmaker to align ontologies for oaei 2010. *Ontology Matching*, page 118, 2010.

[6] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Eng.*, 19(1):1–16, 2007.

[7] N. Fanizzi, C. d'Amato, and F. Esposito. Inductive classification of semantically annotated resources through reduced coulomb energy networks. *Int. J. Semantic Web Inf. Syst.*, 5(4):19–38, 2009.

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

[9] Y. R. Jean-Mary, E. P. Shironoshita, and M. R. Kabuka. Asmov: Results for oaei 2010. *Ontology Matching*, 126, 2010.

[10] H. Köpcke and E. Rahm. Frameworks for entity matching: A comparison. *Data Knowl. Eng.*, 69(2):197–210, 2010.

[11] H. Köpcke, A. Thor, and E. Rahm. Evaluation of entity resolution approaches on real-world match problems. *PVLDB*, 3(1):484–493, 2010.

[12] G. Li, D. Deng, J. Wang, and J. Feng. Pass-join: a partition-based method for similarity joins. *Proc. VLDB Endow.*, 5(3):253–264, Nov. 2011.

[13] A.-C. Ngonga Ngomo. A Time-Efficient Hybrid Approach to Link Discovery. In *OM*, 2011.

[14] A.-C. Ngonga Ngomo. Link Discovery with Guaranteed Reduction Ratio in Affine Spaces with Minkowski Measures. In *ISWC*, pages 378–393, 2012.

[15] A.-C. Ngonga Ngomo. On link discovery using a hybrid approach. *J. Data Semantics*, 1(4):203–217, 2012.

[16] A.-C. Ngonga Ngomo and S. Auer. LIMES - A Time-Efficient Approach for Large-Scale Link Discovery on the Web of Data. In *IJCAI*, pages 2312–2317, 2011.

[17] A.-C. Ngonga Ngomo, J. Lehmann, S. Auer, and K. Höffner. RAVEN – Active Learning of Link Specifications. In *OM*, 2011.

[18] A.-C. Ngonga Ngomo and K. Lyko. EAGLE: Efficient Active Learning of Link Specifications Using Genetic Programming. In *ESWC*, pages 149–163, 2012.

[19] A.-C. Ngonga Ngomo and K. Lyko. Unsupervised learning of link specifications: deterministic vs. non-deterministic. In *Proceedings of the International Ontology Matching Workshop at ISWC*, 2013.

[20] A. Nikolov, M. d'Aquin, and E. Motta. Unsupervised learning of link discovery configuration. In *ESWC*, pages 119–133, 2012.

[21] S. R. Safavian and D. Landgrebe. A survey of decision tree classifier methodology. *Systems, Man and Cybernetics, IEEE Transactions on*, 21(3):660–674, 1991.

[22] F. C. Schadd and N. Roos. Improving ontology matchers utilizing linguistic ontologies: an information retrieval approach. In *Proceedings of the 23rd Belgian-Dutch Conference on Artificial Intelligence (BNAIC 2011)*, 2011.

[23] F. Scharffe, Y. Liu, and C. Zhou. Rdf-ai: an architecture for rdf datasets matching, fusion and interlink. In *IR-KR at IJCAI 2009*, 2009.

[24] T. Soru and A.-C. Ngonga Ngomo. Active learning of domain-specific distances for link discovery. In *Proceedings of JIST*, 2012.

[25] T. Soru and A.-C. Ngonga Ngomo. Rapid execution of weighted edit distances. In *Proceedings of the Ontology Matching Workshop*, 2013.

[26] J. Wang, G. Li, and J. Feng. Trie-join: Efficient trie-based string similarity joins with edit-distance constraints. *PVLDB*, 3(1):1219–1230, 2010.

[27] C. Xiao, W. Wang, and X. Lin. Ed-Join: an efficient algorithm for similarity joins with edit distance constraints. *PVLDB*, 1(1):933–944, 2008.

[28] C. Xiao, W. Wang, X. Lin, and J. X. Yu. Efficient similarity joins for near duplicate detection. In *WWW*, pages 131–140, 2008.