Similarity Features, and their Role in Concept Alignment Learning

Shenghui Wang*, Gwenn Englebienne†, Christophe Guéret*, Stefan Schlobach*, Antoine Isaac*, Martijn Schut*
*Department of Computer Science, Vrije Universiteit Amsterdam, The Netherlands
†Informatics Institute, Universiteit van Amsterdam, The Netherlands
Email:{swang,cgueret,schlobac,aisaac}@few.vu.nl, schut@cs.vu.nl, G.Englebienne@uva.nl

***Abstract*—Finding mappings between compatible ontologies is an important and difficult open problem. Instance-based methods for solving this problem have the advantage of focussing on the most active parts of the ontologies and reflect the semantics of the ontologies as they are used in the real world. We evaluate how the feature representation of the instances is representative of the corresponding concepts, investigate how this corresponds with the domain characteristics of the data and which role it plays in the task of instance-based ontology mapping. We use two different competitive classifiers and a standard feature selection to identify important features, and study the effect of those different classifiers in the concept alignment context.**

***Keywords*-Instance-based Ontology Matching, Semantic Interoperability, Machine Learning**

I. INTRODUCTION

Motivation: The problem of semantic heterogeneity and the resulting problems of interoperability and information integration have been studied for well over 40 years now. It is at present an important hurdle to the realisation of the Semantic Web. Solving matching problems is one step to the solution of the interoperability problem. Semantic Web community has invested significant efforts over the past few years [1].

Solving the matching problem requires to assess the conceptual similarity between elements of two separate ontologies in order to determine relationships (mappings) such as equivalence or subsumption between them. One way of judging whether two concepts from different ontologies are semantically equivalent is to observe their *extensional* information, that is, the instance data they classify [2], [3], [4]. However, it is not always easy to identify identical instances in many applications. Therefore, a robust instance-based mapping technique should cope with the case when there are no explicitly common instances.

Problem Description: This paper focus on instance-based mapping technique only. In [5], we formulated the matching problem as a classification problem, where a mapping can be predicted from the similarity between the extensional information of two concepts.

As in many other application contexts, the instances are described and can be compared according to many dimensions (*features*). Knowing which of these features play the most important role during the classification is important as to optimise the quality of meta-data. Important features would be taken more care of. It is thus interesting to look for a way of assessing the relative importance of the features. In this paper, we use two different automated methods, namely Markov Random Field (MRF) and Evolution Strategy (ES) to investigate this importance. Concept mapping can be seen as a side effect of these methods, and the quality of the method can be assessed by the quality of the concept mapping it produces. We therefore also compare the concept-mapping performance of our methods to a state-of-the-art, off-the-shelf classifier: the Support Vector Machine (SVM).

Research Questions: Our aim is to answer the following research questions:

- What are the benefits of using a machine learning algorithm to determine the importance of features?
- Are there regularities *wrt.* the relative importance given to specific features for similarity computation? Are these weights related to application data characteristics?
- How do different classifiers perform on this instance-based mapping task?

Findings: The two classifiers provide largely consistent, sensible and valuable insight in the importance of the instance features. As evaluated against a human golden standard, they also outperform the SVM on the concept mapping task, thereby indicating that the highlighted features are indeed important.

II. PROBLEM STATEMENT

Our task is to match two thesauri, GTT and Brinkman, which are used to annotate different book collections at the National Library of the Netherlands (Koninklijke Bibliotheek or KB). In order to improve the interoperability between these collections, for example, using GTT concepts to search books annotated only with Brinkman concepts, we need to find mappings between these two thesauri.

As investigated in [3], books annotated by a concept can be treated as instances of this concept. Using shared book instances has already provided interesting mappings. However, many books are not used, because they are not dually annotated. In this paper, we further our investigation in [5], focus on finding mappings directly using book meta-data, no matter the books are dually annotated or not.

Books are described by their title, author, abstract, *etc.* These features together represent an individual book instance. For each concept, all its instances are grouped into an integrated representation of this concept, feature by feature. For example, all titles of these books are put together as a "bag of words." Term frequencies are measured within bags, so that a concept is represented by a high dimensional vector where each element represents the frequency of a term. The similarity between two concepts is calculated with respect to each feature, using the cosine similarity between the term frequencies in these bags.

The similarity between the two elements of a pair of concepts i is therefore measured and represented by a high dimensional vector F_i. The similarity between feature j of the concepts is indicated by F_{ij}. These similarity vectors can be treated as points in a space. In this "similarity space," each dimension corresponds to the similarity in terms of one feature. As we know, some points (*i.e.*, some pairs of concepts) are real mappings but some are not. Our hypothesis is that the *label* of a point — whether it represents a mapping or not — is correlated with the position of this point in this space.

Given some existing mappings, *e.g.*, from a manual effort, our goal is to *learn* this correlation. Therefore the mapping problem is transformed into a classification problem. With already labelled points and the actual similarity values of concepts involved, it is possible to classify a point — *i.e.*, to decide whether the pair represents a mapping — based on its location in the similarity space. One baseline method is to apply a standard support vector machine (SVM) to find a hyperplane which separates classes with different labels. Another option is to look for a direct correlation between labels and similarities. Here we adopt two classifiers: one based on a graphical Markov Random Field [6] and the other using multi-objective evolution strategy [7].

III. METHODS

All three methods assume that mappings are independent. This is a simplifying assumption (since if a term A maps to B, the probability that A maps to any $C \neq B$ clearly decreases), but it is necessary because explicitly modelling the dependencies between all possible mappings is intractable.

A. Markov Random Field (MRF)

Let $T = \{ (F_i, L_i) \}_{i=1}^N$ be the training set of mappings, with, for each given pair of concepts i, a feature vector $F_i \in \mathbb{R}^K$, where K is the number of features, and an associated label L_i.

We consider a simple graphical model, consisting of an observed multivariate input F and a latent variable L which represents the label. We assume that the mappings are identically distributed conditionally on the observations, and model the conditional probability of a mapping given the input, $p(L_i|F_i)$, using a probability distribution from the exponential family. That is:

$$p(L_i|F_i) = \frac{1}{Z(F_i)} \exp\big(\sum_{j=1}^{K} \lambda_j f_j(L_i, F_i)\big), \tag{1}$$

where $\Lambda = \{ \lambda_j \}_{j=1}^K$ are the weights associated to the potential function and $Z(F_i)$, called the partition function, is a normalisation constant ensuring that the probabilities sum to 1. It is given by

$$Z(F_i) = \sum_{L \in \{0,1\}} \exp\big(\sum_{j=1}^{K} \lambda_j f_j(L, F_i)\big). \tag{2}$$

Because of our assumption that mappings are independent, the likelihood of the data set for given model parameters $p(T|\Lambda)$ is given by:

$$p(T|\Lambda) = \prod_{i=1}^{N} p(L_i|F_i) \tag{3}$$

During learning, our objective is to find the most likely values for Λ. We assume a *prior probability* distribution on Λ which favours small values, assigning a normal distribution with zero mean and covariance σ^2 for each λ_i. The *posterior* probability of Λ is then given by

$$p(\Lambda|T) = p(T|\Lambda)p(\Lambda)/p(T), \tag{4}$$

where $p(T)$ is a normalisation term which does not depend on Λ and can therefore be ignored during optimisation. Moreover, since the logarithm is a monotonically increasing function, we can optimise $\log p(\Lambda|T)$ rather than $p(\Lambda|T)$; this turns out to be easier. Ignoring constants, the function we optimise is thus:

$$\ell(\Lambda) = \sum_{i=1}^{N} \left[\sum_{j=1}^{K} \lambda_j f_j(L_i, F_i) - \log Z(F_i) \right] - \sum_{j=1}^{K} \frac{\lambda_j^2}{2\sigma^2}. \tag{5}$$

This is equivalent with logistic regression, where we assume a linear function for the discriminant and introduce regularisation on the model parameters. The result is a convex function which can easily be optimised using any variation of gradient ascent. We used the L-BFGS [8] for the results presented here. The first derivative of $\ell(\Lambda)$ is given by

$$\sum_{i=1}^{N} \left[f_j(L_i, F_i) - \sum_{L \in \{0,1\}} f_j(L, F_i) p(L|F_i, \Lambda) \right] - \frac{\lambda_j}{\sigma^2} \tag{6}$$

The variance of the prior, σ, is a parameter that has to be set by hand and can be seen as a regularisation parameter which prevents overfitting of the training data. The decision criterion for assigning a label to a new pair of concepts is then given by:

$$L_i^P = \begin{cases} 1 & \text{if } p(L_i = 1|F_i) > 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

B. Multi-Objective Evolution Strategy

The evolutionary computing paradigm consists of a number of algorithms (genetic algorithms, evolutionary programming, and others) that are based on, among others, natural selection and genetic inheritance; these algorithms are used for optimisation, modelling and simulation. For the purpose of this paper, we decided to use evolutionary strategies (ES). Evolutionary strategies have two characteristic properties: firstly, they are used for *continuous value* optimisation, and, secondly, they are *self-adaptive*. The first property is desirable for our problem at hand, because we are dealing with real-valued representations. The second property makes the search strategy adaptive, *i.e.*, it dynamically changes search parameters if necessary. Such self-adaptation is shown to be highly effective in complex search processes where it is difficult to tune the parameters manually.

As compared with the genotype/phenotype solution encoding used in Genetic Algorithm, an ES individual is a direct model of the searched solution. That is, an individual is defined by Λ and some evolution strategy parameters:

$$\langle \Lambda, \Sigma \rangle \leftrightarrow \langle \lambda_1, \ldots, \lambda_K, \sigma_1, \ldots, \sigma_K \rangle \tag{8}$$

Then, a metric for the quality of individuals — a fitness function — is established. The fitness function is related to the decision criterion for the ES, which is sign-based:

$$L_i^{ES} = \begin{cases} 1 & \text{if } \sum_{j=1}^{K} \lambda_i F_{ij} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

From 9, we can see that maximising the number of positive results and negative results are two opposite goals. Those goals can be expressed as a multi-objective fitness function using a first component f_1 for the number of true positives matches and the other one f_2 for the number of true negatives.

$$f_1(\Lambda \mid F, L) = \#\{F_i \mid \sum_{j=1}^{K} \lambda_i F_{ij} > 0 \wedge L_i = 1\} \tag{10}$$

$$f_2(\Lambda \mid F, L) = \#\{F_i \mid \sum_{j=1}^{K} \lambda_i F_{ij} \leq 0 \wedge L_i = 0\} \tag{11}$$

Instead of searching for one global optimum, this definition allows the finding of best compromises between errors made on positive and negatives matches.

The evolution process itself essentially consists of three operators: the recombination, mutation and survivor selection operators.

- Recombination is applied on two parent individuals $\langle \lambda_1^1, \ldots, \lambda_K^1, \sigma_1^1, \ldots, \sigma_K^1 \rangle$ and $\langle \lambda_1^2, \ldots, \lambda_K^2, \sigma_1^2, \ldots, \sigma_K^2 \rangle$. From an arithmetic recombination weighted by a coefficient γ, a first new individual $\langle \lambda_1', \ldots, \lambda_K', \sigma_1', \ldots, \sigma_K' \rangle$ is created:
$$\lambda_j' = (1-\gamma_j)\lambda_j^1 + \gamma_j \lambda_j^2, \quad j = 1, \ldots, K \tag{12}$$
$$\sigma_j' = (1-\gamma_j)\sigma_j^1 + \gamma_j \sigma_j^2, \quad j = 1, \ldots, K \tag{13}$$
similarly, an second child $\langle \lambda_1'', \ldots, \lambda_K'', \sigma_1'', \ldots, \sigma_K'' \rangle$ is created with $\sigma_j'' = \gamma_j \sigma_j^1 + (1-\gamma_j)\sigma_j^2$ and $\lambda_j'' = \gamma_j \lambda_j^1 + (1-\gamma_j)\lambda_j^2$. The value of γ is drawn from a uniform distribution on $[0, 1]$.
- Mutation is applied on one parent individual $\langle \lambda_1, \ldots, \lambda_K, \sigma_1, \ldots, \sigma_K \rangle$. It results in the creation of one child $\langle \lambda_1', \ldots, \lambda_K', \sigma_1', \ldots, \sigma_K' \rangle$.
$$\lambda_j' = \lambda_j + \sigma_j' \mathcal{N}_j(0,1), \quad j = 1, \ldots, K \tag{14}$$
$$\sigma_j' = \sigma_j \exp^{\tau' \mathcal{N}(0,1) + \tau \mathcal{N}_j(0,1)}, \quad j = 1, \ldots, K \tag{15}$$
with $\mathcal{N}(0,1)$ being a random number drawn from a "standard" normal distribution (i.e. with mean equal to 0 and standard deviation of 1). The notation $\mathcal{N}_j(0,1)$ denotes the use of a different value for every j^{th} strategy parameter. The two τ parameters are used to define a learning rate. Following conventions, we set them to be inversely proportional to the square root of problem size $\tau = 1/\sqrt{2\sqrt{K}}$ and $\tau' = 1/\sqrt{2K}$.
- Survivor selection is performed using the NSGA2 [9] strategy. The parent population and the offspring solution are joined into one unique, temporary, population. Those individuals are sorted into different fronts according to Pareto optimality. Starting form the best non dominated front of solution, each successive front is made of next non dominated solution that are not yet in a front. Those fronts are used to generate the new parent population. When not all the elements in a front can be picked up, the selection between the individuals in such a way it preserves diversity.

During one loop of the algorithm, new candidate solutions are created using recombination and/or mutation until an oversize criterion is reached. Then, survivor operator is applied to lower the number of individuals to the original population size. The final result of the learning process is the set of best solutions found, according to Pareto optimality. An expert can use the system, stop it at any time and pick up a solution among the best ones found so far. In the absence of an expert, a simple heuristic is used: The winner is the individual whose positive score is the closest to the average of positives scores for all the population. We implemented the ES classifier using OpenBeagle [10], keeping a population of 30 individuals at each iteration.

C. Support Vector Machine

Support vector machines (SVMs) are a set of machine learning algorithms classically used for classification and regression problems [11]. Our work concerns the assessment of a mapping for a given similarity vector. That is, binary classification. In this context, SVM can be used as a maximum margin classifier whose task consists in finding an hyperplane h, with parameters $\omega \in \mathbb{R}^K$ and $b \in \mathbb{R}$, separating the two classes. A sign-based criterion allows the attribution of a class $c_i \in \{-1, +1\}$ to a data vector i.

$$c_i = \begin{cases} +1 & \text{if } \langle \omega \cdot F_i \rangle + b > 0 \\ -1 & \text{if } \langle \omega \cdot F_i \rangle + b \leq 0 \end{cases} \quad (16)$$

The objective is to maximise the margin separating the two classes whilst minimizing classification error risk. Classification is expressed as a constraint. The decision rule from the equation 16 can be changed into the constraint in equation 17 (where N is the number of elements in the training dataset).

$$c_i(\langle \omega \cdot F_i \rangle + b) \geq 1, \quad i = 1, \ldots, N \quad (17)$$

The margin to maximize separates each class set of points closest to the hyperplane. Those support vectors satisfy the condition $|| \langle \omega \cdot F_i \rangle + b ||_2 = 1$. It can be shown that maximizing this margin is equivalent to minimizing the quantity $\frac{1}{2}\langle \omega \cdot \omega \rangle$.

We now have an objective to minimize and some constraints. Next step of SVM formulation is to take the Lagrangian $\mathcal{L}(\omega, \alpha, b)$ of this optimisation problem. This notation introduces a set of Lagrange coefficients $\alpha_i \in \mathbb{R}^+$.

$$\mathcal{L}(\omega, \alpha, b) = \frac{1}{2}\langle \omega \cdot \omega \rangle - \sum_{i=1}^{N} \alpha_i [c_i(\langle \omega \cdot F_i \rangle + b) - 1] \quad (18)$$

This formulation is only able to deal with data that is strictly linearly separable. In order to deal with non linearly separable datasets, the scalar product $\langle F_i \cdot F_j \rangle$ is replaced by a kernel function $K(F_i, F_j)$. The expected outcome of this so called "kernel trick" is to map the data from $\mathbb{R}^K$ to a higher dimension space were they will be linearly separable. Moreover, a tolerance for error is added by setting a maximum boundary C for the α_i. The final optimization problem is:

$$\begin{array}{ll} Max. & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j c_i c_j K(F_i, F_j) \\ with & \sum_{i=1}^{N} \alpha_i c_i = 0 \\ and & 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, N \end{array} \quad (19)$$

And the final decision criterion for the SVM is:

$$L_i^{SVM} = \begin{cases} 1 & \text{if } \sum_{l=1}^{N} \alpha_l c_l K(F_l, F_i) + b \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad (20)$$

The choice of the kernel function has a sensitive impact on the performance of the classifier. Practically, it dictates the shape of the surface that will surround the two classes. We decided to use the commonly used Radial Basis Function (RBF) to get "potato-shaped" classes. This kernel is expressed as

$$K(F_i, F_j) = \exp(-\gamma || F_i - F_j ||_2). \quad (21)$$

We used the implementation of libSVM for the results reported here, with $\gamma = 0.5$ and $C = 8$.

λ_j	Feature	λ_j	Feature	λ_j	Feature
1	Lexical	11	author	21	issued
2	Jaccard	12	contributor	22	language
3	Date	13	creator	23	mods:edition
4	ISBN	14	dateCopyrighted	24	publisher
5	NBN	15	description	25	refNBN
6	PPN	16	extent	26	relation
7	SelSleutel	17	hasFormat	27	spatial
8	abstract	18	hasPart	28	subject
9	alternative	19	identifier	29	temporal
10	annotation	20	isVersionOf	30	title

Table I
LIST OF THE FEATURES

IV. EXPERIMENTS

We match the GTT and Brinkman thesauri, which contain 35K and 5K concepts respectively. They are used to annotate two book collections of the KB, containing 2M books of which nearly 1M books were annotated, including 307K books with GTT concepts only; 490K with Brinkman concepts only; 222K with both.

A. Feature selection for similarity calculation

On top of the similarity calculated using book metadata, as introduced in Section II, we also measured the relative edit distance as the lexical distance between two concepts. The Jaccard similarity measure used in [3] is also included. Note that the Jaccard measure is calculated from dually annotated books only. If two concepts are never used to annotate dually indexed books, we set the Jaccard measure to be the average of all calculated Jaccard measures. The features used are listed in Table I and all similarity values are normalised to have zero mean and unit variance in order to make comparison of λ_i meaningful.

The lexical and Jaccard similarity are of course strong indicators of concept mappings, and may seem to give artificially high results for our instance-based method. However, it is a great advantage that we can include any information in the features, and let the machine decide on their relative importance. For reference, Figure 1 includes how the MRF performs when these two features are removed ("MRF 3-30"). It shows that we still obtain quite good results from the instances only, although the best results are obtained with the combination ("MRF 1-30").

B. Control-Experiment: Quality of Learning

First, we used human labelled pairs to carry on 10-fold cross validation in order to check validity of our learned mappings. These pairs of concepts were judged by a human evaluator who assigned a "mapping" or "non-mapping" label to each pair of concepts. The similarity between these pairs of concepts were calculated as introduced above. The whole data set was divided into 10 folds, each time using 9 folds to train the probabilistic model and the remaining fold to test the model.

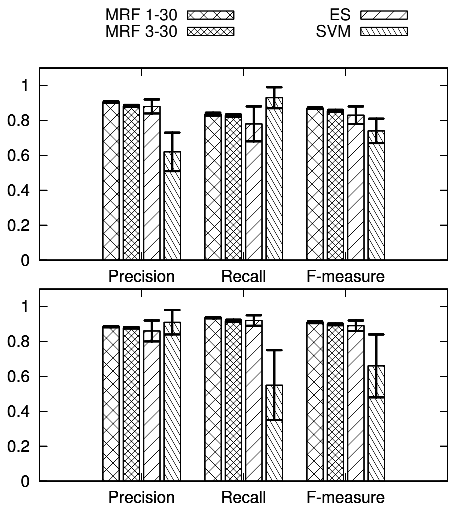

Figure 1. Precision, recall and F-Measure for mappings with a positive label (top) and a negative label (bottom). Error bars indicate one standard deviation over the 10 folds of cross-validation.

In the testing step, the predicted mappings were compared with the real mappings. The positive precision is the proportion of real mappings among all predicted positive mappings, and the positive recall is the proportion of true predicted mappings among all real mappings. The negative precision is the proportion of the non-mappings among all predicted negative mappings and the recall is the proportion of the predicted non-mappings among all non-mappings. Figure 1 shows the performance of the three classifiers. These show to be generally quite good for the MRF and ES methods, with performances comparable to the results of state-of-the-art mappers [12]. Our deployment of SVM generally performs worse than MRF and ES. One possible reason for this may be the tuning of the parameters γ and C. Another reason may be our choice of the RBF kernel which is perhaps not optimal for this problem. However, those results clearly show that our chosen classifier are highly competitive and perform favorably wrt. state of the art matching tools.

C. Relative importance of features

An important benefit of our first two methods is that the solutions are interpretable by humans. In an attempt to work out which features of our instances are important for mapping, we explored whether the value of λ_i reflects the intuitive importance of feature i. Figure 2 depicts how the weights (the values of λ) varied over the 10 folds of cross-validation for the MRF and ES classifiers, as well as the mutual information between the mapping label and each similarity feature.

A first observation is that ES lambdas are not really conclusive: the 10 solutions are much less consistent than MRF ones. Reassuringly, however, ES lambdas that are most inconclusive correspond to the least informative features (as shown by the mutual information). Focusing on the MRF, then, we can observe that apart from a few exceptions,

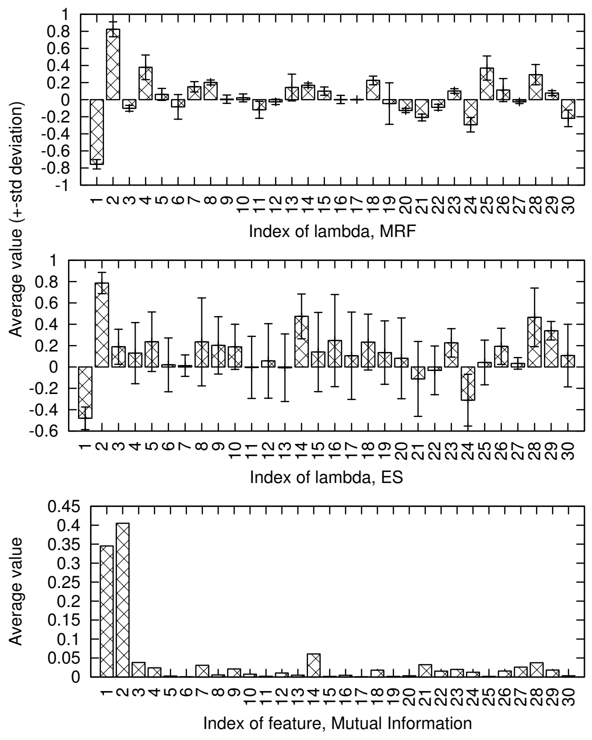

Figure 2. Values of Λ and mutual information between features and labels

important features in terms of mutual information are associated to large weights, while unimportant features are normally associated to small weights. Notice in that respect that feature 1 is a distance measure, while all other features are similarity measures. Some less informative features still have large weights (*e.g.*, feature 25), however. This may be explained by the fact that the mutual information was computed independently for all features. A feature may be completely random overall, yet be informative conditionally on some other feature. The combination of such features will still be informative and result in larger weights. Similarly, a feature may be very informative by itself, yet not provide any supplementary value (and may even be detrimental) if another feature already provides the same information, thus explaining some features with high mutual information have low weights.

A more detailed examination of the weights allows us to compare the learnt importance of features with the intuitions provided by the application context. A first set of features has large weights as expected, such as the similarity between the concept labels (feature 1), their co-occurrence in the set of dually-annotated books (feature 2) and the subject (feature 28). A few features are expected not to play a significant role and have indeed low importance: size of the book (16), (rare) format description (17) and language (22), for instance.

Some features, more surprisingly, were given an importance level that conflicts with what one could have anticipated: description (15) and abstract (8), which give readable descriptions of book content, happen to be only marginally important, less than for example the date of copyright (14). The latter, for instance, may mirror phenomena like the publication of a number of books on the same subject in short periods of time or, perhaps, that some concepts are used a lot for a short period, and much less before and after that period.

This last category illustrates how learning can help making decisions in dubious cases. For instance, it is well known that book titles (30) do not always cover their subject entirely. Our experiments demonstrate that similarity between these rather hints at conceptual dissimilarity — even though this is less clear for the alternative titles (9). Similarly, two books may refer to different subjects while being written by the same author(s). This is especially true when homonymy is not dealt with. — creator, author, contributors, respectively 11, 12, 13 — or published by the same publisher (24).

This observation tends to show that when many different description features interact, there is no systematic correlation between what a learning method could find and what an application expert may anticipate. And in such cases it is highly valuable, for tuning mappers exploiting instance similarities, to apply learning techniques instead of relying solely on human judgement.

V. CONCLUSION

In this paper, we take the instance-based mapping technique one step further and investigate what instance features are important in this context. Our analysis has shown that the overall similarity of instances is too coarse a measure: the similarity of some features is very indicative of a valid mapping while some are not and, even worse, the similarity of some instance features actually indicates concept dissimilarity.

Two different machine learning techniques are used to automatically identify meaningful features. Both methods assign mostly consistent importance to the features, which agrees with the domain characteristics of the data.

The two classifiers we propose, the MRF and the ES, result in a performance in the neighbourhood of 90%, showing the validity of the approach. Their performance is not significantly different, but both significantly outperform the SVM, an off-the-shelf classifier.

In the future, we would like to investigate how instance similarity can be used to infer multi-concept mappings (n to m mappings). We would also like to learn the type of mapping (for example "broader than," "narrower than," as defined in the SKOS standard [13]), using multiple labels in the classification process.

REFERENCES

[1] J. Euzenat and P. Shvaiko, *Ontology Matching*. Springer Verlag, 2007.

[2] R. Ichise, H. Takeda, and S. Honiden, "Integrating multiple internet directories by instance-based learning," *Proceedings of the eighteenth International Joint Conference on Artificial Intelligence*, 2003.

[3] A. Isaac, L. van der Meij, S. Schlobach, and S. Wang, "An empirical study of instance-based ontology matching," in *Proceedings of the 6th International Semantic Web Conference*, Busan, Korea, 2007.

[4] C. Wartena and R. Brussee, "Instanced-based mapping between thesauri and folksonomies," in *Proceedings of the 7th International Semantic Web Conference*, Karlsruhe, Germany, 2008.

[5] S. Wang, G. Englebienne, and S. Schlobach, "Learning concept mappings from instance similarity," in *Proceedings of the 7th International Semantic Web Conference*, Karlsruhe, Germany, 2008.

[6] R. Kindermann and J. L. Snell, *Markov Random Fields and Their Applications*. AMS, 1980.

[7] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies: A comprehensive introduction," *Journal Natural Computing*, vol. 1, no. 1, p. 352, 2002.

[8] D. C. Liu and J. Nocedal, "On the limited memory method for large scale optimization," *Mathematical Programming*, vol. 45, pp. 503–528, 1989.

[9] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *Proceedings of the Parallel Problem Solving from Nature VI Conference*. Paris, France: Springer. Lecture Notes in Computer Science No. 1917, 2000, pp. 849–858.

[10] C. Gagn and M. Parizeau, "Genericity in evolutionary computation software tools: Principles and case-study," *International Journal on Artificial Intelligence Tools*, vol. 15, no. 2, pp. 173–194, April 2006.

[11] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, March 2000.

[12] C. Caracciolo, J. Euzenat, L. Hollink, R. Ichise, A. Isaac, V. Malaisé, C. Meilicke, J. Pane, P. Shvaiko, H. Stuckenschmidt, O. Sváb-Zamaza, and V. Svátek, "Results of the ontology alignment evaluation initiative," Tech. Rep., 2008.

[13] A. Isaac and E. Summers, "Skos primer," Working Draft, W3C, Tech. Rep., March 17 2009. [Online]. Available: http://www.w3.org/TR/skos-primer/