Meta-learning beyond classification: A framework for information extraction from the Web

Georgios Sigletos[1, 2], Georgios Paliouras[1],
Constantine D. Spyropoulos[1], Takis Stamatopoulos[2]

[1] Institute of Informatics and Telecommunications, NCSR "Demokritos",
P.O. BOX 60228, Aghia Paraskeyh, GR-153 10, Athens, Greece
{sigletos, paliourg, costass}@iit.demokritos.gr
[2] Department of Informatics and Telecommunications, University of Athens,
TYPA Buildings, Panepistimiopolis, Athens, Greece
{sigletos, takis}@di.uoa.gr

Abstract. This paper proposes a meta-learning framework in the context of information extraction from the Web. The proposed framework relies on learning a meta-level classifier, based on the output of base-level information extraction systems. Such systems are typically trained to *recognize* relevant information within documents, i.e., streams of lexical units, which differs significantly from the task of classifying feature vectors that is commonly assumed for meta-learning. The proposed framework was evaluated experimentally on the challenging task of training an information extraction system for multiple Web sites. Three well-known methods for training extraction systems were employed at the base level. A variety of classifiers were comparatively evaluated at the meta level. The extraction accuracy that was obtained demonstrated the effectiveness of the proposed framework of collaboration between base-level extraction systems and common classifiers at meta-level.

1 Introduction

One common *meta-learning* approach, known as *stacked generalization* [18], deals with the task of learning a meta-level module to combine the predictions of multiple base-level learners. Base learners are treated as "black boxes", i.e., only their output predictions are used, without considering the details of their functionality. The meta-level module is expected to achieve performance superior to each of the base learners, on unseen data.

Current work in meta-learning of this type focuses on the *classification* problem, i.e. learn how to assign the correct class value to each one of a set of different events, where each event is described by a *vector* of predefined *features*. Various studies, e.g. [1] and [16], have investigated which classifiers and data representations, either at the base or the meta level, and under which strategies, can lead to better classification results over unseen events.

In this paper, we attempt to drive the meta-learning framework outside the common feature-vector representation, employed in classification tasks. Our motivation is the

information extraction (IE) task, which can be defined as the process of directly extracting relevant text fragments from collections of documents and filling the slots of a predefined template. In particular, information extraction from Web pages is a simplified version of the harder free-text information extraction examined by the Message Understanding Conferences [11]. Despite its simplicity, though, it has gained popularity in the past few years, due to the proliferation of online sources, and the need to recognize useful pieces of information inside the Web chaos.

IE can be formulated as a regular-expression matching process within a document that is modeled by a sequence of lexical units (tokens). Learning a classifier to perform this task is unnatural and as a result specialized systems, like STALKER [12] and SoftMealy [10], learn extraction rules in the form of special types of regular expressions. However, there is a small number of approaches, which enumerate the possible text fragments that can be found within a document and then model the task as a binary classification one [5], [6]. In this case, the task is to learn whether or not a candidate fragment fills some template-slot. There is a number of problems associated with this approach, such as the exponential number of candidate fragments and the disproportionately large number of "negative" events. Therefore, it is particularly desirable to design an alternative framework that will use common IE systems.

Thus, the main contribution of this paper is a novel meta-learning framework that removes the constraint of employing classifiers at the base level, accommodating IE systems that *recognize* relevant text instances within documents, rather than classifying text fragments. The prediction output of the base IE systems is appropriately transformed into vectors of features, to be used for training a common meta-level classifier. We have experimented with three algorithms at the base level: two deterministic (STALKER [12] and $(LP)^2$ [2]) and a stochastic finite-state approach (Hidden Markov Models (HMMs) [13]). Four classifiers were evaluated at the meta level.

Section 2 reviews some basic theory in meta-learning for classification tasks. Section 3 illustrates our proposed framework. Section 4 presents the experimental results. Finally, the basic conclusions of this work are presented in Section 5.

2 Building a meta classifier – Basic theory

Wolpert [18] introduced an approach for constructing ensembles of classifiers, known as *stacked generalization* or *stacking*. A *classifier ensemble*, consists of a set of *n* classifiers $C_1, \ldots C_n$, called *base-level* classifiers and a *meta-level* classifier C_{ML} that learns how to combine the predictions of the base-classifiers. The base-classifiers are generated by applying *n* different classification algorithms on a labeled dataset $L_B = \{(x_k, y_k)\}$, where x_k and y_k are the features and the class value for the *k*-th instance vector respectively. The individual predictions of the base-classifiers on a different labeled dataset L_M, are used to train the meta-classifier C_{ML}. The predictions of the base-classifiers on L_M are then transformed into a meta-level set of classification vectors. At runtime, C_{ML} combines the predictions $P_M(x) = \{P_i(x), i = 1 \ldots n\}$ of the *n* base-classifiers on each new instance *x*, and decides upon the final class value *y(x)*. The

predictions of the base-classifiers on *x* are transformed into a single vector representation, which is then classified by C_{ML}.

3 Building a meta-classifier for information extraction

The majority of IE systems that use machine learning, e.g. [10], [12], represent the acquired knowledge in the form of finite-state automata (FSA) or stochastic FSA [13]. Thus, IE becomes a task of matching a set of regular expressions within each document. We further assume a *single-slot* approach to IE that deals with extracting instances of isolated *facts* (i.e. *extraction fields*), whereby a different automaton is induced for each fact. For example, in a Web page describing CS courses, one automaton has to be induced for extracting instances of the "course number" fact, while a different one is required for extracting instances of the "course Title" fact.

3.1 Preliminaries

Our goal is to incorporate single-slot IE systems, into a meta-learning framework and thus exploit the advantages provided by the meta-learning theory, aiming at higher extraction accuracy. We make the following assumptions:

1. Let D be the sequence of a document's tokens, and $T_i(s_i, e_i)$ a fragment of that sequence, where s_i , e_i are the *start* and *end* token bounds respectively.
2. Let $E_B = \{E_k \mid k = 1\ldots n\}$ be the set of *n* single-slot IE systems, generated by *n* different learning algorithms.
3. Let $I_k = \{i_j : T_j \rightarrow fact_j\}$ be the set of instances extracted by E_k, and $fact_j$ the predicted fact associated with the text fragment T_j.

3.2 The proposed framework

We suggest a novel framework for combining the IE systems at base-level with a common classifier at meta-level, which is graphically illustrated in Figure 1.

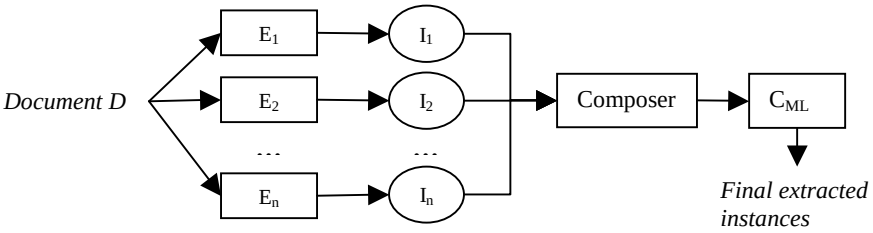

Fig. 1. Combining extraction systems and a common classifier at runtime

The starting point of the architecture depicted in Figure 1 is a *document D*, which is the input to each extraction system E_k, which generates a set of extracted instances I_k, over D. In contrast, the input to each classifier C_i in the common stacking framework, is an instance *vector x*, while the output is the predicted class value $P_i(x)$.

The combination of the base-level IE systems with the meta-level classifier C_{ML} depends on a *composer* module. At runtime, the input to the composer comprises the *n* sets of extracted instances $I_1,\ldots,I_n$. The output of the composer must be a set of vectors, to be finally classified at meta-level by C_{ML}. Similarly in the training phase of C_{ML}, the output of the composer must be a set of *classified* vectors, based on information from the hand-labeled data. In order to translate the output of the IE systems to a fixed-length vector of events for C_{ML}, we make the following assumptions, affecting the functionality of the *composer* module:

1. Each event corresponds to a *distinct* text fragment T(s, e) among all predicted instances in $\cup I_k$, $k = 1\ldots n$. Note that two text fragments $T_1(s_1, e_1)$ and $T_2(s_2, e_2)$ are different, if either $s_1 \neq s_2$ or $e_1 \neq e_2$.
2. The features of the new vector, associated with the text fragment T, are based on the predicted facts for T by the base-level IE systems. Note that for each distinct T among all instances in $\cup I_k$, $k = 1\ldots n$., there exists at least one instance $i_k : T \rightarrow fact_k$.
3. At runtime, each vector associated to a fragment T is to be classified into one of a set of nominal values, corresponding to the *f* different facts in the domain of interest, plus an additional value "false" if the text fragment is judged as not being an interesting fact.
4. During the training of C_{ML}, each vector associated to a T, is augmented with a class value, corresponding to the hand-labeled fact of the fragment. If T is not labeled, the new vector is assigned to the "false" class.

Consider the token table in Table 1(a), which is part of a page describing computer science courses. Table 1(b) shows the extracted instances by 2 base-level IE systems over the token table in Table 1(a). Note that the first system has not predicted a fact for the text fragment $T_2(27, 28)$.

Table 1. (a) Part of a token table for a page describing computer science courses. (b) Extracted instances by two base-level IE systems E_1, E_2. (c) The distinct text fragments and the information associated to each T for constructing the new vectors

…	*25*	*26*	*27*	*28*	…
…	CS414	:	Operating	Systems	…

(a)

T(s, e)	*E_k*	*Fact*
$T_1(25, 25)$	1	Course Number
$T_1(25, 25)$	2	Course Number
$T_2(27, 28)$	2	Course Title

(b)

T(s, e)	*Information for meta-level vectors*
$T_1(25, 25)$	(1, Course Number), (2, Course Number)
$T_2(27, 28)$	(2, Course Title)

(c)

Table 1(c) shows the two *distinct* text fragments, each associated with a set of pairs $<E_k, fact_k>$, where $fact_k$ is the predicted fact by the *k*-th base-level IE system. The

information in those pairs will be used for building the two meta-level vectors – one for each distinct T.

3.3 Meta-level data representation

In this paper we experiment with two different vector representations:

1. *Numeric-feature* representation: each distinct text fragment *T* is modeled by a vector of *f* numeric features, each one corresponding to a fact of interest, e.g. *Course Number*, *Course Title*. During the training of the meta-classifier, the vector is augmented with an additional class feature, which is the true fact of T, according to the labeled document. For each fact predicted by an IE system, the corresponding feature value is incremented by one, starting from zero.
2. *Binary-feature* representation: each distinct text fragment T is modeled by a vector of $n*f$ binary features. The output of each of the *n* base-level IE systems is a set of f binary features. For each predicted fact, the corresponding feature is set to one. In case of ambiguous facts, more that one features will have the value one. All other features are set to zero.

The advantage of the first representation is that the number of features depends only on the number of the facts of interest, and remains fixed, independently of the IE systems employed at base-level. The advantage of the second representation is that the predicted facts of the base-level IE systems are modelled separately. The numeric and binary representations for each of the two distinct text fragments of Table 1(c) are depicted in Table 2.

Table 2. Numeric (a) and binary (b) feature representation for the text fragments of Table 1(c), f_1 = *course number*, f_2 = *course title*

	f_1	f_2	…
T(25, 25)	2,	0,	…
T(27, 28)	0,	1,	…

(a)

	E_1			E_2		
	f_1	f_2	…	f_1	f_2	…
T(25, 25)	1,	0,	…	1,	0,	…
T(27, 28)	0,	0,	…	0,	1,	…

(b)

4 Experiments

Our goal is to empirically evaluate the proposed architecture in the context of single-slot IE from the Web. For this purpose, we conducted experiments on the task of IE from pages across multiple Web sites, which exhibit multiple formats, including tables, nested tables and lists, thus making the extraction task more difficult.

4.1 Base-learners and meta-learners employed

At base level, we experimented with three learning algorithms for performing IE: STALKER [12], $(LP)^2$ [2] and Hidden Markov Models (HMMs) [13]. In this paper we used STALKER in a single-slot mode, as described in [15]. For the HMMs, we adopted the approach proposed in [7] and [14]. For the $(LP)^2$ system, we used the default settings of the *Amilcare* [3] environment[1], in which the $(LP)^2$ is embedded.

At meta level, we experimented with four classification algorithms, all implemented in the WEKA environment [17]. The first one is *j48*, a reimplementation of the C4.5 decision-tree learning algorithm. The next two belong in the family of boosting algorithms: *AdaBoost.M1* [8], with *j48* as a weak classifier, and *LogitBoost* [9]. The last one is the *IB1*, an implementation of the 1-nearest-neighbor algorithm.

4.2 Dataset description

Experiments were conducted on a collection of 101 Web pages describing CS courses, collected from four different university sites in the context of the *WebKB* project [4]. Three facts were hand-tagged for this domain: *course number*, *course title*, and *course instructor*. All pages were pre-processed by a *tokenizer* module, using *wildcards* [12][2].

This corpus was selected due to the fact that it has been used in the past and results are reported in [5]. The approach in [5] is a multi-strategy one, but it does not involve the learning of a meta-classifier. Simple regression models are learned to *map* the relationship between confidence values in the predictions of the base classifiers to true probabilities. At runtime, they rely on a voting scheme, to decide upon the prediction with the highest true probability.

4.3 Results

In order to evaluate our approach we employed a 5-fold *double* cross-validation procedure, known as *cross-validation stacking* [18] and we used *micro-average* recall and precision over all facts. Table 3 shows the base-level experimental results for the CS courses domain. Results for the *F1* metric are also provided, which is the harmonic mean of the recall and precision metrics.

Table 3. Base-level results for the CS courses domain

Macro (%)	*Prec.*	*Recall*	*F1*
HMMs	60,85	62,06	61,45
$(LP)^2$	69,74	62,12	65,71
STALKER	19,77	49,55	28,26
Best Indiv. [5]	**74,37**	**59,47**	**66,08**

[1] The pattern-length was set to 5.

[2] For the $(LP)^2$, pages were preprocessed by a POS tagging module and a Stemming module.

In the CS courses domain, the results of the $(LP)^2$ are comparable to the best individual learner's results, reported in [5]. $(LP)^2$ and HMMs share the same recall, however $(LP)^2$ achieves a higher precision. STALKER does not perform well in this domain. However, we rely on the diversity in the results of the three systems, aiming at higher performance at meta-level.

Tables 4(a) and 4(b) show the meta-level results, using the *numeric-feature* and *binary-feature* representation respectively. The results are micro-averages of the corresponding results for the three facts. The false class is excluded as it is of no particular interest.

Table 4. Meta-level results using (a) the *numeric-feature* and (b) *binary feature* representation for the CS courses domain

Macro (%)	*Prec.*	*Rec.*	*F1*
J48	83,80	59,59	69,65
AdaBoostM1	83,80	59,59	69,65
LogitBoost	84,59	58,93	69,46
KNN (k=1)	84,33	58,75	69,25
Average	**84,13**	**59,22**	**69,50**
M/strategy [5]	**N/A**	**N/A**	**66,9**

(a)

Macro (%)	*Prec.*	*Rec.*	*F1*
J48	86,35	56,45	68,27
AdaBoostM1	86,92	56,15	68,22
LogitBoost	87,48	56,03	68,31
KNN (k=1)	85,19	57,60	68,73
Average	**86,49**	**56,56**	**68,38**
M/strategy [5]	**N/A**	**N/A**	**66,9**

(b)

A clear conclusion from the above results is that the differences between the meta-level classifiers –in each vector representation- are negligible. The *numeric-feature* representation led to slightly better results than the *binary-feature* one, but the difference is too small to lead to an interesting conclusion.

Comparing the meta-level results of Table 4 against the base-level results of Table 3 and the results reported in [5] we note a small decrease in recall, accompanied by a substantial improvement in precision. The meta-level classifiers exploited the diversity in the predictions of the three systems and achieved an overall performance higher than the individual IE systems. The overall conclusion is that the proposed meta-learning framework helps to improve the extraction performance of a series of base-level IE systems.

5 Conclusions

We presented and evaluated a meta-learning framework in the context of IE from the Web. The proposed framework is independent of the employed IE systems at base-level that are not required be classifiers. The presented results are encouraging, showing that the proposed approach improves the precision and overall performance of the IE systems, while outperforming also the state-of-the-art reported in the literature.

Plans for future work include experiments with more complex meta-level vector representations. Additional sources of information (e.g. DOM-based information) will also be investigated. Finally, we plan to experiment with more IE systems and more extraction tasks, in order to evaluate the proposed framework more thoroughly.

References

1. Chan P. K., Stolfo S. J., On the Accuracy of Meta-Learning for Scalable Data Mining. *Journal of Intelligent Information Systems* 8(1): 5-28, (1997).
2. Ciravegna, F., Adaptive Information Extraction from Text by Rule Induction and Generalization. *In Proceedings of the 17th IJCAI Conference.* Seattle (2001).
3. Ciravegna, F., Amilcare: adaptive IE tool, http://nlp.shef.ac.uk/amilcare/ .
4. Craven, M., DiPasquo, D., Freitag, D., McCallum, A.K., Mitchell, T., Nigam, K., Slattery, S., Learning to extract symbolic knowledge from the World Wide Web, *19th AAAI* (1998).
5. Freitag, D., Machine Learning from Informal Domains, *PhD Thesis*, CMU, (1998).
6. Freitag, D., Kushmerick N., Boosted Wrapper Induction, *17th AAAI Conference*, (2000).
7. Freitag, D., McCallum, A.K., Information Extraction using HMMs and shrinkage, *AAAI-99 Workshop on Machine Learning for Information Extraction*, pp.31-36 (1999).
8. Freund, Y., Shapire, R.E., A Decision-theoretic Generalization of online Learning and an Application to Boosting, *Journal Of Computer and System Sciences*, 55(1), 119-139 (1997)
9. Friedman, J., Hastie, T., Tibshirani, R., Additive Logistic Regression: a Statistical View Of Boosting. *Technical Report*, Stanford University (1999).
10. Hsu, C., Dung, M., Generating Finite-state Transducers for Semi-structured Data Extraction from the Web, *Journal Of Information Systems*, Vol 33 (1998).
11. MUC 7, *http://www.itl.nist.gov/iaui/894.02/related_projects/muc*.
12. Muslea, I., Minton, S., Knoblock, C., Hierarchical Wrapper Induction for Semistructured Information Sources, *Aut/mous Agents and Multi-Agent Systems*, 4:93-114, (2001).
13. Rabiner, L., A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77-2 (1989).
14. Seymore, K., McCallum A.K., Rosenfeld, R., Learning hidden Markov model structure for Information Extraction. *Journal of Intelligent Information Systems* 8(1): 5-28, (1999).
15. Sigletos, G. Paliouras G., Spyropoulos C.D., Hatzopoulos M., Mining Web sites using using wrapper induction, named entity recognition and post-processing, *1st European Web Mining Forum*, Cavtat (Dubrovnik) Croatia, September 2003 (to appear).
16. Todorovski, L., Džeroski, S., Combining Classifiers with Meta Decision Trees, *Machine Learning Journal*, Kluwer Academic Publ., Volume 50-(3), p.223-249, (2003).
17. Witten, I., Frank, E., Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, *Morgan Kaufmann Publishers* (2000).
18. Wolpert, D., Stacked Generalization, *Neural Networks*,5(2): 241-260 (1992).

ACKNOWLEDGEMENTS

This work has been partially funded by a research grant -provided by the NCSR "Demokritos"- and CROSSMARC, a EC-funded research project.