Journal of Machine Learning Research 6 (2005) 1751-1782 Submitted 2/05; Revised 9/05; Published 11/05

Combining Information Extraction Systems Using Voting and Stacked Generalization

Georgios Sigletos SIGLETOS@IIT.DEMOKRITOS.GR
Georgios Paliouras PALIOURG@IIT.DEMOKRITOS.GR
Constantine D. Spyropoulos COSTASS@IIT.DEMOKRITOS.GR
Institute of Informatics and Telecommunications
National Centre for Scientific Research (NCSR) "Demokritos"
Aghia Paraskeyh, 153 10, Athens, Greece

Michalis Hatzopoulos MIKE@DI.UOA.GR
Department of Informatics and Telecommunications
University of Athens
Panepistimiopolis, 157 71, Athens, Greece

Editor: William Cohen

Abstract

This article investigates the effectiveness of voting and stacked generalization -also known as stacking- in the context of information extraction (IE). A new stacking framework is proposed that accommodates well-known approaches for IE. The key idea is to perform cross-validation on the base-level data set, which consists of text documents annotated with relevant information, in order to create a meta-level data set that consists of feature vectors. A classifier is then trained using the new vectors. Therefore, base-level IE systems are combined with a common classifier at the meta-level. Various voting schemes are presented for comparing against stacking in various IE domains. Well known IE systems are employed at the base-level, together with a variety of classifiers at the meta-level. Results show that both voting and stacking work better when relying on probabilistic estimates by the base-level systems. Voting proved to be effective in most domains in the experiments. Stacking, on the other hand, proved to be consistently effective over all domains, doing comparably or better than voting and always better than the best base-level systems. Particular emphasis is also given to explaining the results obtained by voting and stacking at the meta-level, with respect to the varying degree of similarity in the output of the base-level systems.

Keywords: stacking, voting, information extraction, cross-validation

1 Introduction

One of the most interesting topics in supervised machine learning is learning how to combine the individual predictions of multiple classifiers. The motivation derives from the opportunity of obtaining higher prediction accuracy at meta-level, while treating classifiers as *black boxes*, i.e., using only their output, without considering the details of their implementation. *Stacked generalization* or *stacking* (Wolpert, 1992) is a common scheme that deals with the task of learning a meta-level classifier to combine the predictions of multiple base-level classifiers. The success of stacking arises from its ability to exploit the diversity in the predictions of base-level classifiers and thus predicting with higher accuracy at meta-level. In contrast, no learning takes place when *voting* on the predictions of multiple classifiers. Voting is typically used as a baseline against which the performance of stacking is compared.

Research on voting and stacking has primarily focused on classification. Each training instance in the domain of interest is represented by a vector $< x_1 ... x_n , y >$, where $x_1 ... x_n$ is a set of attribute values or features, and y is the class value describing a particular event in the domain, which is to be recognized at runtime. In order to classify a new vector $< x_1 ... x_n >$, the predictions of the base-level classifiers form a new feature vector, which is assigned the class value y either by the meta-level classifier or by voting. Cross-validation in the base-level set of feature vectors is required by stacking, in order to create the entire set of meta-level vectors by the predictions of the base-level classifiers, and thus train the meta-level classifier.

In this article we investigate the effectiveness of voting and stacking on the task of Information Extraction (IE). IE is a form of shallow text processing that involves the population of a predefined template with relevant fragments extracted from a text document. The proliferation of the Web and the other Internet services in the past few years intensified the need for developing systems that can effectively recognize relevant information in the enormous amount of text that is available online. A variety of systems have been developed in the context of IE from online text (e.g. Freitag and Kushmerick, 1999; Sonderland, 1999; Freitag, 2000; Ciravegna, 2001; Califf and Mooney, 2003). The key idea behind combining a set of IE systems through stacking is to learn a common meta-level classifier, such as a decision tree or a naive-Bayes classifier, based on the output of the IE systems, towards higher extraction performance. On the other hand, a simpler approach is to vote on the predictions of different IE systems.

In order to apply voting and stacking to IE, the base-level classifiers should be normally replaced by systems that model IE as a classification task. The main problem, however, is that IE is not naturally a classification task (Thompson et al., 1999). A typical IE system is trained using a set of sample documents, paired with templates that are filled with relevant text fragments from the documents. IE could be mapped to a common classification problem by classifying almost every possible unbroken sequence of tokens (usually up to a predefined maximum length) that can be found within a document, as relevant or not (Freitag, 2000). This way of modelling the IE task, however, results in an enormous increase in the number of candidate text fragments, where only the small number of annotated fragments is considered as positive examples, while all the other fragments are considered as negative examples during training. Table 1 shows the examples that are constructed from a hypothetical text fragment within a page describing laptop products.

Text Fragment:	…processor <b> **256 MB** SDRAM…	
Positive Examples:	256 MB	
Negative Examples:	processor	256
	processor 	MB
	….	256 MB SDRAM
	processor <b> 256 MB	MB SDRAM
	…	…

Table 1. An indicative example of recognizing an instance of the field *ram* (highlighted in bold) from a page that describes a laptop product and the set of examples it generates.

Although the size of the candidate text fragments can be somehow reduced by using various heuristics (Freitag, 2000), modelling IE in this manner, does not seem natural.

Alternative approaches of modelling the IE task exist in the literature. Systems like BWI (Freitag and Kushmerick, 1999), $(LP)^2$ (Ciravegna, 2001) and STALKER (Muslea et al., 2001), model IE as a boundary detection task. A boundary is the virtual space between two adjacent tokens. The task here is to recognize starting and ending token boundaries of relevant fragments within a document and then extract the enclosed content. In Table 1, the boundary between "<b>" and "256" and the one between "MB" and "SDRAM" are the starting and ending index respectively, of the fragment "256 MB". Some approaches (Freitag and McCallum, 1999, 2000; McCallum et al., 2000; Lafferty et al., 2001) model IE as the task of labelling the linear sequence of tokens that a text document is parsed into. Fragments consisting of (contiguous) tokens that have been marked as relevant for a field (e.g. “256“, “MB”) are extracted. A variety of other approaches (e.g. Sonderland, 1999; Califf and Mooney, 2003) induce matching rules that extract whole fragments from a text document at runtime and fill the corresponding slots in the template.

This article initially introduces the idea of merging the templates filled by different IE systems into a single *merged* template, which facilitates the application of voting and stacking to IE. The merged template contains those text fragments that have been identified by at least one IE system, along with the individual predictions by the systems. Various voting schemes are then presented that rely either on the nominal or the probabilistic predictions of the IE systems that are available at the base-level.

A new stacking framework is then introduced that combines a wide range of base-level IE systems with a common classifier at the meta-level. Only the output of the IE systems is combined, i.e., the filled templates, which are *merged* into a single template, independently of how the instances that populate the templates were identified. In the new framework, only the meta-level data set consists of feature vectors that are constructed by the predictions of the IE systems, while the base-level data set consists of text documents, paired with filled templates. In contrast, both base-level and meta-level data sets in stacking for classification consist of feature vectors. An extension of the stacking framework for IE is also proposed that is based on using probabilistic estimates of correctness in the predictions of the IE systems.

Extensive experiments were conducted for comparing voting against stacking. Particular emphasis was given to analyzing the results obtained by voting and stacking with respect to how the base-level IE systems correlate in their output. Three well known IE systems were employed at the base-level, each drawn from a different learning paradigm: $(LP)^2$, a sequential covering rule-induction algorithm, Hidden Markov Models (HMMs), a finite-state approach to IE, and Boosted Wrapper Induction (BWI) that introduces the application of boosting to IE. A diverse set of classifiers were comparatively evaluated at the meta-level. Experiments were conducted on five collections of pages from five different domains.

The remainder of this article is structured as follows: Section 2 presents some background in the areas of voting, stacking and IE. Section 3 introduces the concept of the *merged* template and describes various voting schemes for IE. Section 4 describes the new stacking framework for IE. Section 5 describes the experimental design. Section 6 presents the results obtained by voting and stacking, and compares all IE systems at both base-level and meta-level. Section 7 explains the results obtained at the meta-level, with respect to the varying degree of correlation in the output of the base-level systems. Section 8 presents our conclusions, discussing potential extensions.

2 Background

Sections 2.1 to 2.3 provide background in the areas of voting, stacking and information extraction respectively.

2.1 Voting

The simplest way to combine the output of multiple classifiers is within a voting framework. Let $C^1 ... C^N$ be the set of classifiers that are induced by training N different learning algorithms $L^1 ... L^N$ on a data set D consisting of feature vectors. To classify a new instance at runtime, the classifiers $C^1 ... C^N$ are queried for a class value and the class with the highest count is finally selected. This scheme is known as majority (or plurality) voting. Variations include weighted majority voting and voting using class probability distributions (Dietterich, 1997). In the former approach, each classifier’s vote is weighted by its accuracy, as measured by either using a holdout data set or the entire training data set by cross-validation. In the probabilistic approach, each classifier outputs a probability distribution vector over all relevant classes. For each class, the individual probability values are averaged (or summed) by all classifiers, and the class with the maximum value is finally selected.

Note that methods like *boosting* (Freund and Schapire, 1996) and *bagging* (Breiman, 1996) vote on a set $C^1 ... C^N$ of classifiers that are generated by applying a single learning algorithm to N different versions of a given data set, rather than training N different algorithms.

2.2 Stacking

Section 2.2.1 presents stacking, while Section 2.2.2 describes some related work.

2.2.1 Definition

Wolpert (1992) introduced a novel approach for combining multiple classifiers, known as stacked generalization or stacking. The key idea is to learn a meta-level (or level-1) classifier based on the output of base-level (or level-0) classifiers, estimated via cross-validation as follows:

Define D a data set consisting of feature vectors, also referred to as level-0 data, and $L^1 ... L^N$ a set of N different learning algorithms. During a J-fold cross-validation process, D is randomly split into J disjoint parts $D^1 ... D^J$ of almost equal size. At each jth fold, $j = 1..J$, the $L^1 ... L^N$ learning algorithms are applied to the training part $D \setminus D^j$ and the induced classifiers $C^1(j) ... C^N(j)$ are applied to the test part D^j. The concatenated predictions of the induced classifiers on each feature vector x_i in D^j, together with the original class value $y_i(x_i)$, form a new set MD^j of meta-level vectors.

At the end of the entire cross-validation process, the union $MD = \cup MD^j$, $j = 1..J$ constitutes the full meta-level data set, also referred to as level-1 data, which is used for applying a learning algorithm L^M and inducing the meta-level classifier C^M. The learning algorithm L^M that is employed at meta-level could be one of the $L^1 ... L^N$ or a different one. Finally, the $L^1 ... L^N$ learning algorithms are applied to the entire data set D inducing the final base-level classifiers $C^1 ... C^N$ to be used at runtime. In order to classify a new instance, the concatenated predictions of all base-level classifiers $C^1 ... C^N$ form a meta-level vector that is assigned a class value by the meta-level classifier C^M. Figure 1(a) illustrates the cross-validation methodology, while Figure 1(b) illustrates the stacking framework at runtime.

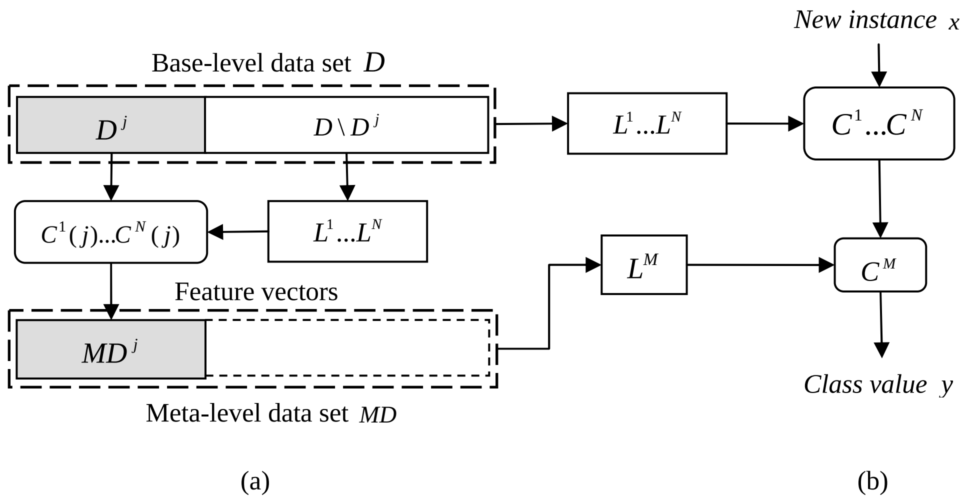

Figure 1. (a) Illustration of the *J*-fold cross-validation process for creating the meta-level data set. (b) The stacking framework at runtime.

2.2.2 Related Work

Research on stacking concerns two major issues, initially described as *black art* by Wolpert (1992). The first is the choice of classifiers at both base-level and meta-level that will lead to the best empirical results. The second issue, which has generally received more attention in the literature, concerns the combination of the predictions of the base-level classifiers and their mapping to attributes for the features vectors at the meta-level. Typical attributes that are used at the meta-level are the class predictions of the N base-level classifiers.

Chan (1996) experimented with various representations including the *class-attribute-combiner* scheme, where the class predictions of the base-level classifiers are appended with the attributes of the base-level vectors, together with the correct class for each vector. Chan (1996) also experimented with an *arbiter* scheme, where a meta-classifier is only trained on a subset of the *base-level* vectors, in which the base-level classifiers disagree in their predictions. A *hybrid* scheme was also evaluated, in which a meta-classifier is only trained on a subset of the meta-level data set that follows the *class-attribute-combiner* scheme, where the base-level classifiers disagree in their predictions. Experimental results showed that the *class-attribute-combiner* is the best scheme. A slight improvement in the accuracy was obtained at meta-level over the best base-level results, but the differences were not measured as statistically significant.

Ting and Witten (1999) introduced a variant of stacking where each base-level classifier predicts a probability distribution vector over all classes, instead of predicting a single nominal value. The individual vectors by the N classifiers are concatenated, thus resulting in $N*Q$ attributes at meta-level, where Q the number of relevant classes. Ting and Witten (1999), suggested also the use of *multi-response linear regression* (MLR) for meta-level learning that proved to be highly effective. MLR is an adaptation of linear regression (Breiman, 1996a) which transforms the classification problem into Q different binary prediction problems: for each relevant class, a linear equation is constructed to predict one if the class value equals the class under consideration or zero otherwise.

Seewald (2003) suggested a modification of the approach described by Ting and Witten (1999), where different sets of meta-level features should be used for each of the Q binary prediction problems. In particular, only the probability values for the class under consideration should be used at meta-level, instead of concatenating the probability distributions of all classifiers, and thus reducing the number of meta-level attributes to N. Experimental results showed an improvement over stacking with probability distributions.

Džeroski and Ženko (2004) investigated the use of MLR in conjunction with class probability distributions augmented with an additional set of attributes that is based on the entropies of the class probability distributions and the maximum probability returned by each classifier. This scheme was found to perform better than using only probability distributions.

Stacking typically outperforms voting. However, voting does not involve cross-validation and the training of a meta-level classifier, and thus it is computationally cheaper than stacking.

2.3 Information Extraction

Sections 2.3.1 and 2.3.2 provide background on the task of Information Extraction (IE), while Section 2.3.3 describes an existing framework for combining multiple IE systems.

2.3.1 Definition

Let $\{f^1 ... f^Q\}$ be a set of Q extraction *fields* for a particular domain of interest, and d a text document annotated by the domain experts with *instances* of those fields. A field *instance* is a pair $< t(s,e), f >$, where $t(s,e)$ is a text fragment, with *s* and *e* be the boundaries of the fragment in a document's token table and $f \in \{f^1 ... f^Q\}$ the associated field. A boundary has been defined above as the virtual space between two adjacent tokens. Define T a template that is filled with pairs $< t(s,e), f >$. A field is typically a *target-slot* in template T, while $t(s,e)$ is a *slot-filler*. A field may also have multiple or no instantiations within a document. Table 2(a) shows a part of a Web page describing laptop products where the relevant text is highlighted in bold. Table 2(b) shows the hand-filled template for this page.

The Information Extraction (IE) task can be defined as follows: *given a new document* d, *find all possible instances for each relevant field within* d *and populate a template* T. This definition states that each field learning problem is considered in isolation, and thus modelled as a binary learning task: given a learning algorithm designed for IE, then for each relevant field $f \in \{f^1 ... f^Q\}$, a *target concept* is learned that identifies relevant instances $< t(s,e), f >$ within text. At runtime, all target concepts are applied separately to d and used to populate T.

An extended approach to IE is to study interactions among relevant fields, and thus grouping field instances into higher-level concepts, also referred to as *multi-slot* extraction (Sonderland, 1999). In this article we handle the simpler single-slot approach, which covers a wide range of IE tasks and motivated the development of a variety of learning algorithms (e.g. Freitag and Kushmerick, 1999; Freitag, 2000; Ciravegna, 2001; Califf and Mooney, 2003).

…**TransPort ZX** <br> <font size="1"> <b> **15"** XGA **TFT** Display </b> <br> **Intel <b> Pentium III 600 MHZ** </b> 256k Mobile processor <br> <b> **256 MB** SDRAM up to 1GB </b> <br> <b> **40 GB** hard drive </b> (removable) <br> …

(a)

T			Short description for field f
$t(s,e)$	s,e	Field f	
TransPort ZX	47, 49	model	Name of the laptop's model
15"	56, 58	screenSize	Size of the laptop's screen
TFT	59, 60	screenType	Type of laptop's screen
Intel<b>Pentium III	63, 67	procName	Name of the laptop's processor
600 MHZ	67, 69	procSpeed	Speed of the laptop's processor
256 MB	76, 78	ram	The RAM capacity of the laptop
40 GB	86, 88	HDcapacity	The hard disk capacity of the laptop

(b)

Table 2. (a) Part of a Web page describing laptop products (b) The hand-filled template for this page.

2.3.2 Related Work

The IE task from free text has been the focus of the Message Understanding Conferences (e.g. DARPA 1995, 1996). On the other hand, the advent of the Web intensified the need for developing systems that help people to cope with the large amount of text that is available online. Systems that perform IE from online text, should generally meet the requirements of low cost and high flexibility in development, and adaptation to new domains. MUC-level systems fail to meet those criteria, in addition to the fact that the linguistic analysis performed for free text does not exploit the extra-linguistic information (e.g. HTML/XML tags, layout format) that is available in online text. Therefore, this type of system has not found wide applicability in the context of online sources.

As a result, less linguistically intensive approaches have been developed for IE on the Web using *wrappers*, which are sets of highly accurate rules that extract a particular resource's content. The manual development of wrappers (Chawathe et al., 1994) has proved to be a time-consuming task, requiring a high-level of expertise. Machine-learning techniques that learn wrappers for IE, either using supervised learning (e.g. Kushmerick, 1997; Muslea et al., 2001; Cohen et al., 2002) or unsupervised learning (e.g. Crescenzi et al., 2001; Chang and Lui, 2001), have been designed to handle highly structured collections of Web pages, such as telephone directories and product catalogues. Those approaches, however, fail when the text type is less-structured, which is also common on the Web.

Recent effort on *adaptive* IE (Ciravegna, 2001; Ciravegna and Lavelli, 2003), motivates the development of IE systems that can handle different text types, from rigidly structured to almost free text -where common wrappers fail- including mixed types. For example, the algorithms presented in (Sonderland, 1999; Ciravegna, 2001; Califf and Mooney, 2003) learn IE rules that exploit shallow natural language knowledge and thus can be applied to less structured text. The BWI algorithm (Freitag and Kushmerick, 1999) relies on a method called *boosting* (Freund and Schapire, 1996) for improving the extraction performance of the learned IE rules, which allows the applicability of the algorithm to less structured text. Hidden Markov modelling (Rabiner, 1989) is a powerful statistical learning technique that has found wide applicability in IE from both structured and unstructured text (Seymore et al., 1999; Freitag and McCallum, 1999, 2000).

In this article we focus on adaptive IE systems and investigate how their performance can be further improved, by combining their output at meta-level. The presence of the token boundaries s and e is essential, as we will show, for combining different IE systems. In some cases (e.g.

Ciravegna, 2001), information about token boundaries is implicitly represented by inserting appropriate start and end XML tags within documents, i.e. $<f>$ and $</f>$ tags for each relevant field $f \in \{f^1 ... f^Q\}$. A template such as the one in Table 2(b) can then be easily constructed. Thus, we assume in the remaining of this article that we deal with templates that include information about token boundaries, as in Table 2(b), which is a realistic assumption, since such information is typically available.

2.3.3 Multistrategy Learning

Despite the growing interest in combining machine learning algorithms and the application to some natural language parsing tasks such as part-of-speech tagging (Halteren et al., 2001) and word-sense disambiguation (Florian et al., 2002), this topic has received little attention by the IE community. The only relevant work is described in (Freitag, 2000), where the IE task was modelled as a classification one, using a set of four base-level systems, which were then combined by multistrategy learning.

The term *multistrategy learning* generally refers to the combination of multiple learning approaches under a single algorithm and was mainly used for combining inductive with analytical learning (Michalski and Tecuci, 1994). Domingos (1996) used the term *empirical multistrategy learning* for distinguishing the case where all learning components are inductive. The basic notion behind empirical multistrategy learning was to design a new complex algorithm that heuristically combines a set of learning components by requiring, however, implementation details of each individual component. For example, the RISE algorithm in (Domingos, 1996) unifies an instance-based learner with a rule-based learner under a new implementation, aiming to overcome the limitations of both approaches.

Stacking combines a set of multiple learning components in a more loose fashion, by only learning to integrate their output at meta-level, and thus treating them as *black boxes*. Despite its simplicity, stacking offers the advantage of being highly extensible to more algorithms at both base-level and meta-level, regardless of their internal structure. On the other hand, voting is the simplest form of loosely integrating the output of multiple components.

Freitag (2000) followed the voting paradigm in the context of IE, which he called "multistrategy learning for IE" and it is based on using probabilistic estimates produced by the IE systems. Specifically, for each relevant field $f \in \{f^1 ... f^Q\}$, the confidence scores that are produced by the base-level systems are mapped into probabilistic estimates of correctness in the predictions of the systems. In case of more than one predictions of the field f for a fragment $t(s,e)$, a combined probability is estimated for $<t(s,e), f>$ using (1).

$$P^C = 1 - \prod_j (1 - p^j), \tag{1}$$

where P^C is the combined probabilistic estimate that the text fragment $t(s,e)$ belongs to the field f, and p^j the probabilistic estimate that some IE system E^j has predicted the field f for $t(s,e)$. Actually, P^C measures the probability that at least one of those IE systems that have predicted the field f for $t(s,e)$, has predicted correctly, which equals the probability that not all predictions for f are wrong.

Finally, the constraint of "one per document" (OPD) is imposed on some fields. This means that only one instance of OPD fields is allowed for each page. For example, a page in the domain of computer science (CS) courses should describe only one course, and thus should contain only one instance of the field *course title*. Therefore, if more are identified, then the one with the highest (combined) probability is selected, while all others are rejected. The example in Table 3 illustrates the usefulness of the OPD constraint.

Assuming in Table 3 that one hypothetical IE system predicts "256 MB" and "1 GB" as ram instances, while another system predicts only "256 MB" as *ram*. The combination of the two systems, under the OPD constraint, produces a single instance $<\text{"256 MB"}, ram>$, the one that has the highest combined probability.

$t(s,e)$	Probability by the first system	Probability by the second system	Combined probability
256 MB	0.4	0.5	0.7
1GB	0.6	-	0.6

Table 3. Combining the predictions of two hypothetical IE systems for a "ram" instance. Supposing that the correct instance is $<''256\ MB'', ram>$

The OPD field constraint, though useful in certain cases, is restrictive for IE in general and does not hold for all relevant fields. For example, a Web page may describe more than one laptop products, and thus more than one *ram* instances may exist. The OPD constraint was also applied for fields that allow many instances per page, without significant loss of performance[1]. For example, a page may rarely describe more than one CS courses. This approach is still restrictive for IE, since a Web page very often describes more than one laptop products.

Finally, converting confidence scores to probabilistic estimates takes place by validating the performance of each IE system on a hold-out set, or by cross-validation in the entire training data set and using a form of regression modelling which is described in more detail by Freitag (2000). The motivation behind mapping confidence scores to probabilistic estimates is that confidence scores are not always reliable, since incorrect matches may be assigned high scores. Thus voting on different IE systems using confidence scores may not always be reliable. The correlation among confidences and probabilities has been also investigated by Kauchak et al. (2004) in the context of the BWI algorithm for IE, and found to be weaker for more difficult IE tasks, e.g. from free-text domains.

In the remainder of this article, the term "multistrategy learning" will be used to refer to the work in (Freitag, 2000).

3 Voting for Information Extraction

Section 3.1 presents an example of combining IE systems. The concept of the *merged* template is introduced, which is important for combining different IE systems either through voting or stacking. Various voting schemes for IE are then presented in Sections 3.2 and 3.3, against which the performance of stacking for IE will be compared.

3.1 Example of Combining Different Systems – The Merged Template

Let $L^1 ... L^N$ be a set of N learning algorithms, designed for IE, which are given a corpus D of training documents, annotated with relevant field instances. The algorithms $L^1 ... L^N$ typically generalize from the training corpus, towards a set of pattern-matching extraction rules. Define $E^1 ... E^N$ the corresponding set of IE systems that exploit the acquired knowledge, to identify relevant instances in new documents. Each trained IE system consists of a set of target concepts that have been learned for the relevant fields. Finally, define $T^1 ... T^N$ a set of templates for a document d, populated by $E^1 ... E^N$ respectively with relevant field instances.

We suggest in this article that a *merged* template can be constructed from $T^1 ... T^N$ as follows: all text fragments $t(s,e)$ identified by $E^1 ... E^N$ in $T^1 ... T^N$ are inserted to an initial pool. Duplicate fragments are removed: two fragments differ if either their start or end boundary differs. For the remaining *distinct* fragments, the fields predicted by $E^1 ... E^N$ are collected and inserted together with the correct field in the template. If some IE system does not predict a field for a text fragment, then the corresponding cell in the merged template is empty. If a text fragment does not exist in the hand-filled template, then the corresponding cell in the last column is also empty. Table 4 shows an illustrative example of a merged template that has been constructed by the output T^1, T^2 of two IE systems E^1, E^2, for the page of Table 2(a).

[1] This is a remark by Dayne Freitag, based on personal contact.

s,e	$t(s,e)$	*Output by* E^1	*Output by* E^2	*Correct field*
47, 49	TransPort ZX	model	manuf	model
56, 58	15"	screenSize	-	screenSize
59, 60	TFT	screenType	screenType	screenType
63, 66	Intel<b>Pentium	-	procName	-
63, 67	Intel<b>Pentium III	procName	-	procName
67, 69	600 MHz	procSpeed	procSpeed	procSpeed
76, 78	256 MB	ram	ram	ram
81, 83	1 GB	ram	HDcapacity	-
86, 88	40 GB	-	HDcapacity	HDcapacity

Table 4. Merged template, based on the output of two IE systems. Each entry corresponds to a text fragment that has been identified by at least one system.

Examining Table 4, we note some disagreement in the predictions of the two systems. For two text fragments (“TransPort ZX”, “1GB”) the predicted fields by E^1 and E^2 differ. Comparing to the hand-filled template of Table 2(b), we conclude that “TransPort ZX” has been correctly identified as *model* only by E^1, while E^2 identified the same fragment as *manuf* (the manufacturer of the laptop). On the other hand, the fragment “1GB” does not exist in the hand-filled template. Therefore, the fields predicted by the two systems for this fragment are false. Furthermore, some text fragments have been identified by only one of the two IE systems. The fragment “15"” has been identified only by E^1, while the fragment “40 GB” has been identified only by E^2. The fields predicted for both fragments are correct.

Examining again Table 4, we wonder whether we can exploit, at some higher level, the disagreement in the predictions of the different IE systems, aiming to achieve superior extraction performance. The desirable result is to automatically fill the last column in the merged template of Table 4 with the correct fields. In other words, we would like to assign the correct field to each text fragment that has been identified by at least one base-level system.

3.2 Majority Voting

A simple idea for combining the predictions of different IE systems is to use majority voting: for each entry in the merged template, we count the predicted fields by the available systems and select the field with the highest count. In the case of a tie, a random selection is typically performed among even fields.

Note that Table 4 contains missing values, reflecting the natural fact that some system may not have predicted a field for a text fragment that has been identified by another system. The significance of missing values has to be carefully considered. For example, if some system predicts an incorrect field f for a text fragment $t(s,e)$, while the remaining systems do not predict any field at all, then ignoring missing values during voting harms precision, since the incorrect field is returned. An alternative is to record a missing value as “false”, providing evidence that no field should be predicted for $t(s,e)$. If the value with the highest count is “false” then no field is assigned to $t(s,e)$. If, however, f is the correct field for $t(s,e)$, interpreting the missing predictions by the remaining systems as “false” values harms overall extraction performance, since the correct field is rejected.

Therefore, two different settings of majority voting are defined, depending on whether missing values are ignored or encoded as “false” values that indicate rejection of prediction.

3.3 Voting Using Probabilities

The voting with probabilities scheme that is presented in this section shares many features with multistrategy learning, as described in (Freitag, 2000) and was briefly outlined in Section 2.3.3. Both schemes share the same method for mapping confidence scores to probabilistic estimates and the same Equation (1) that estimates the combined probability of correctness for an instance $< t(s,e), f >$. However, the two schemes differ in how they model the IE task.

Multistrategy learning considers each field in isolation during combination and relies on the OPD constraint for improving the extraction accuracy, as demonstrated by the example of Table

3. On the other hand, voting using probabilities takes place on a merged template, like the one in Table 4, while no OPD assumption is required for any relevant field. This allows the case of contradictory field predictions among different systems during combination, as demonstrated in Table 4, where the fragment "1GB" has been identified as *ram* by the first system and as *HDcapacity* by the second one. The field with the highest probability should be selected. Multistrategy learning for IE ignores contradictory field predictions.

In Section 3.2 two different settings of majority voting were defined, depending on whether absence of prediction by some system for a text fragment, i.e. a missing value in the merged template, is ignored or encoded as "false" that indicates rejection of prediction. The problem here is that there is no probability for "false". If we assume that "false" corresponds to probability 1, then voting will lead to spurious results. Thus, two different settings for voting using probabilities are defined as follows:

In the first setting, missing values are ignored, similar to the first setting of majority voting. Given a fragment $t(s,e)$, the field f with the highest probabilistic estimate by those systems that have predicted a field for $t(s,e)$ is returned. In the second setting, however, a constraint is imposed on whether f should be accepted or not. If the probability that is attached to f is less than 0.5, then f is rejected. Otherwise, f is returned, as in the first setting. The motivation behind using this constraint, is that if f has been predicted with low degree of confidence by the base-level systems, then it should not be accepted. The value of 0.5 is the natural choice of a threshold for deciding whether f should be accepted or not.

4 Stacked Generalization for Information Extraction

This section starts with the motivation for performing learning, rather than simply voting. Then a new stacking framework for IE is presented, along with an extension that relies on using probabilistic estimates on the output of the base-level systems.

4.1 Motivation for Performing Learning

Examining the merged template of Table 4, we wonder whether we can *learn* to predict the correct field, based on the fields predicted by the available systems, rather than simply voting. A simple motivation for preferring learning, rather than voting, is that the latter cannot handle situations where most of the systems make an error. For example, if a system correctly predicts *ram* for the hypothetical fragment "1,5 GB", while the other systems erroneously predict *HDcapacity*, then voting chooses the latter value. Therefore, it would be desirable to perform learning in order to induce a rule of the form: *if the first IE system predicts "ram" and the other systems predict "HDcapacity", then the correct field is "ram".*

In order to train a common classifier, a set of feature vectors should be provided as training data. The idea suggested in this article is to create a feature vector for every row entry of the merged template, i.e. for each text fragment that has been identified by at least one base-level system. Table 5 shows the new feature vectors created by the merged template of Table 4.

s,e	$t(s,e)$	*Feature vectors*		
		Output by E^1	Output by E^2	Class
47, 49	TransPort ZX	model,	manuf,	model
56, 58	15"	screenSize,	?,	screenSize
59, 60	TFT	screenType,	screenType,	screenType
63, 66	Intel<b>Pentium	?,	procName,	false
63, 67	Intel<b>Pentium III	procName,	?,	procName
67, 69	600 MHz	procSpeed,	procSpeed,	procSpeed
76, 78	256 MB	ram,	ram,	ram
81, 83	1 GB	ram,	HDcapacity,	false
86, 88	40 GB	?,	HDcapacity,	HDcapacity

Table 5. Feature vectors created by the merged template of Table 4.

Absence of prediction by an IE system is indicated by "?". If a text fragment does not exist in the hand-filled template, the class attribute of the corresponding vector takes the value "false" that indicates rejection of prediction. At runtime, the class value is to be assigned by the classifier.

Having specified the format of the feature vector, the remaining issue is to construct the full set of vectors for training the meta-level classifier using cross-validation, as described for the stacking framework in Section 2.2. However, in IE we deal with collections of text documents, annotated with relevant instances, rather than feature vectors. This disparity between base-level and meta-level data sets is handled by sampling from documents during cross-validation, rather than from feature vectors as in stacking for classification.

4.2 Stacking Using Nominal Values

The key idea behind stacking for IE, is to learn a meta-level classifier based on the output of base-level systems via cross-validation as follows:

At the *j*th fold, $j = 1..J$, of cross-validation, the N learning algorithms $L^1...L^N$ are trained on the document set $D \setminus D^j$ and the induced IE systems $E^1(j)...E^N(j)$ are applied to the test set D^j. For each document d in D^j, let $T^1...T^N$ be the populated templates by $E^1(j)...E^N(j)$ respectively. A *merged* template MT is assembled from $T^1...T^N$, as shown in Section 3.1. A new feature vector is produced for each entry in the merged template, which is added to the meta-level data set MD^j. At the end of the cross-validation process, the union $MD = \cup MD^j$ constitutes the full meta-level data set, which is used by a learning algorithm L^M to train the meta-level classifier C^M. Finally, the N learning algorithms are applied to the entire data set D, inducing the base-level systems $E^1...E^N$ to be used at runtime.

Figure 2 presents an algorithmic description of the new stacking framework for IE. The vectors in the new meta-level data set MD belong to $Q+1$ classes, where Q is the number of relevant fields in the domain of interest plus the value "false". A vector classified as "false" indicates that the corresponding fragment $t(s,e)$ does not exist in the hand-filled template, and thus the available base-level systems should not have predicted a field for it (for example the fragment "1 GB" in Table 5).

```
procedure stacking_for_IE ($D$, $J$, $L^1...L^N$, $L^M$) begin
        $D^1...D^J$ = partition of $D$ into $J$ document collections of almost equal size
        for $j$ = 1 to $J$ do begin
          $MD^j$ = {}
          for $i$ = 1 to $N$ do $E^i(j)$ = the system obtained by training $L^i$ on $D \setminus D^j$
          foreach document $d$ in $D^j$ do begin
               for $i$ = 1 to $N$ do $T^i$ = the template, populated by applying $E^i(j)$ to $d$
               $MT$ = create_merged_template ($d$, $T^1...T^N$)
               foreach entry, i.e., for each distinct $t(s,e)$, in $MT$ do begin
                      for $i$ = 1 to $N$ do $f^i \in \{f^1,...f^Q,"?"\}$ = the field by $E^i(j)$ for $t(s,e)$
                      $f \in \{f^1,...f^Q, false\}$ = the correct field for $t(s,e)$
                      $MD^j = MD^j \cup$ vector $< f^1,...f^N, f >$
               end
          end
        end  // end of cross validation
        $MD = \cup MD^j$, $j = 1...J$
        $C^M$ = meta-level classifier obtained by applying $L^M$ on $MD$
        // Train the base-level IE systems
        for $i$ = 1 to $N$ do $E^i$ = the base-level system obtained by training $L^i$ on $D$
end
```

Figure 2. The new stacking framework for information extraction.

The key difference among stacking for IE and common stacking is that cross-validation operates on text documents paired with hand-filled templates, instead of feature vectors labelled with class values. This removes the constraint of performing common classification at base-level, thus allowing the application of stacking to IE. The base-level algorithms $L^1 ... L^N$ are designed for IE, while the learning algorithm L^M that is applied at meta-level is designed for classification, and thus cannot be one of $L^1 ... L^N$ as in stacking for classification.

The size of the meta-level data set is not a-priori known in stacking for IE, unlike common stacking where there is a one-to-one correspondence between base-level and meta-level vectors. Here, the size of the meta-level data set is determined by the output of the base-level systems, i.e. by the individual templates $T^1 ... T^N$ that assemble a merged template, as shown in Figure 2.

The one-to-one correspondence between base-level and meta-level features vectors in common stacking, results in identical class values at both base-level and meta-level. In IE, however, a text fragment that is relevant to our task may not have been identified by any of the available base-level systems. In that case, there is no possibility of identifying that fragment at meta-level, since no feature vector is created and thus resulting in loss of information. This observation suggests that IE systems biased towards *recall* (percentage of the annotated field instances that were identified), should be generally preferred for combination, expecting to reduce the loss of information at meta-level.

Moreover, a meta-level vector in common stacking for classification does not contain missing values, since each base-level classifier predicts a nominal class value or a probability distribution over the relevant classes. On the other hand, a meta-level vector in stacking for IE may contain missing values, as shown in Table 5, since some system may not have predicted a field for a fragment that has been identified by another system. In correspondence to majority voting, missing values in Table 5 can be handled by recording them as "false", which indicates rejection of prediction. In that case, all attribute values, should share the same set of values $\{f^1 ... f^Q, \mathit{false}\}$, thus replacing the set $\{f^1, ... f^Q, "?"\}$ in Figure 2.

Another issue in stacking for IE is the choice of J in the J-fold cross-validation process depicted in Figure 2. The choice of J usually depends on the size of the training data and the computational cost of training. The choice of J also affects the difference in the number of documents for training the base-level systems and the meta-level classifiers. According to Figure 2, the base-level systems $E^1 ... E^N$ that will be used at runtime are retrained on the entire collection D of training documents. On the other hand, the meta-level vectors on each *j*th fold are created by the predictions of the systems $E^1(j) ... E^N(j)$, as trained on a lower proportion of the training documents $D \setminus D^j$. The larger the value of J, the smaller the size of D^j and therefore the smaller the difference between the size of the training data set $D \setminus D^j$ for $E^1(j) ... E^N(j)$ and the complete data set D.

To illustrate this point, for a collection of 40 training documents, the final base-level systems $E^1 ... E^N$ are trained on the full data set. If we assume a five-fold cross validation process, then on each fold j, the $E^1(j) ... E^N(j)$ are trained on 32 documents. An alternative is to use a higher value for J, suffering however a higher computational cost.

4.3 Stacking at Runtime

Given a new document *d*, the systems $E^1 ... E^N$ are used to identify relevant field instances and fill the corresponding templates $T^1 ... T^N$. A merged template is then created from $T^1 ... T^N$. For each row entry in the merged template, i.e., for each distinct $t(s,e)$, a feature vector is created by the predicted fields of $E^1 ... E^N$ for $t(s,e)$ (absence of prediction by an IE system is indicated by "?" or "false"). The new vectors are finally classified by the meta-level classifier C^M into $Q+1$ predefined categories $\{f^1 ... f^Q, \mathit{false}\}$. If a vector is classified into one of the relevant fields $f^1 ... f^Q$ then the corresponding instance $< t(s,e), f >$ is inserted in the final template for d. Otherwise ("false" prediction) the entry is excluded from the final template. For example, if we assume that the class column in Table 5 has been filled by C^M, then for the two vectors that have been classified as "false", the corresponding entries will be excluded from the final template. At runtime, the stacking framework for IE is graphically depicted in Figure 3.

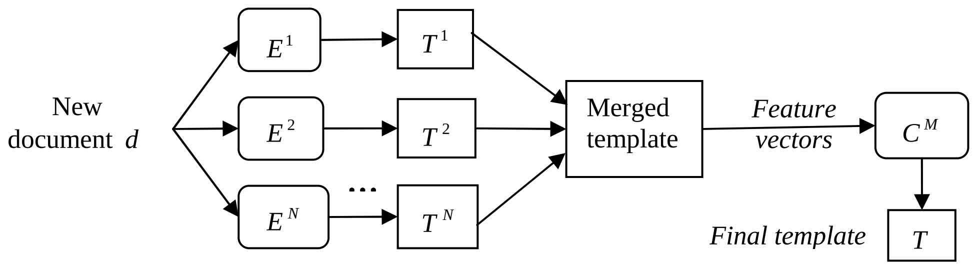

Figure 3. The stacking framework for information extraction at runtime.

According to Figure 3, the input to the systems is a new text document while the output is the corresponding filled template. In contrast, both input and output in the runtime stacking for classification architecture of Figure 1(b), consist of a single feature vector.

4.4 Stacking Using Probabilities

A straightforward extension of stacking with nominal values is to rely on the confidence scores by the base-level systems that have been converted into probabilistic estimates of correctness. The new framework is described as follows:

- Instead of predicting one of the Q relevant fields for each fragment $t(s,e)$, each system generates a confidence score c^k for the predicted field f^k. This is modelled by a Q-element vector that contains zero values, except for the *k*th position where c^k appears, i.e., $<0,...,c^k,...0>$. If a system does not predict a field, all elements are zero.
- Each vector is converted to a new one $<0,...p^k,...0>$, where p^k is a probabilistic estimate that corresponds to c^k and reflects the probability of correctness of the prediction. The conversion process is described in more detail in (Freitag, 2000).
- Finally, the output vectors by $E^1...E^N$ for $t(s,e)$ form a single vector of $N*Q$ elements, appended by the correct field, according to the hand-filled template.

Table 6 shows an illustrative example of the new feature vectors at the meta-level, using probabilistic estimates of correctness.

s,e	$t(s,e)$	*Feature vectors using probabilistic estimates*		
		Output by E^1	Output by E^2	Class
47, 49	TransPort ZX	0, 0, 0.92, 0, 0, 0, 0, 0,	0, 0.34, 0, 0, 0, 0, 0, 0,	model
56, 58	15"	0, 0, 0, 0, 0, 0, 0.83, 0,	0, 0, 0, 0, 0, 0, 0, 0,	screenSize
59, 60	TFT	0, 0, 0, 0, 0, 0, 0, 0.85,	0, 0, 0, 0, 0, 0, 0, 0.91,	screenType
63, 66	Intel<b>Pentium	0, 0, 0, 0, 0, 0, 0, 0,	0, 0, 0, 0.61, 0, 0, 0, 0,	false
63, 67	Intel<b>Pentium III	0, 0, 0, 0.67, 0, 0, 0, 0,	0, 0, 0, 0, 0, 0, 0, 0,	procName
67, 69	600 MHz	0, 0, 0, 0, 0.82, 0, 0, 0,	0, 0, 0, 0, 0.79, 0, 0, 0,	procSpeed
76, 78	256 MB	0, 0, 0, 0, 0, 0.91, 0, 0,	0, 0, 0, 0, 0, 0.77, 0, 0,	ram
81, 83	1 GB	0, 0, 0, 0, 0, 0.55, 0, 0,	0.89, 0, 0, 0, 0, 0, 0, 0,	false
86, 88	40 GB	0, 0, 0, 0, 0, 0, 0, 0,	0.65, 0, 0, 0, 0, 0, 0, 0,	HDcapacity

Table 6. The new meta-level vectors using probabilistic estimates of correctness.

The same vector representation used in Table 6 was also used in the extension of stacking for classification proposed in (Ting and Witten, 1999). The only difference is that class (or field) probability distributions, as they suggest, are not typically produced by IE systems. Given a text fragment $t(s,e)$, either a field f is predicted for $t(s,e)$ or no field is predicted at all. Thus, except for the vector elements that correspond to the predicted fields, all other values are set to zero, as shown in Table 6.

In stacking with probabilities, a missing value is indicated by a vector of which all elements take zero values. Missing values can be handled by augmenting the output of each of the N base-level systems with an additional attribute, indicating the probability for "false". The total number of meta-level attributes will be then $N(Q+1)$. The probability value of the extra attribute will be complementary to the value of the non-zero element of the vector. For example, in the first row entry of Table 6, the value of the extra attribute for the system E^1 will be 0.08, while for each absent prediction the extra attribute takes the value "1". The significance of handling the missing values in stacking is empirically evaluated by comparing the performance of the trained classifiers over the new vectors against the trained classifiers over the vectors with missing values.

5 Experimental Setup

We have performed extensive experiments in five real-world domains, using well-known algorithms at both base-level and meta-level. The primary target was to determine whether stacking provides added value over the base-level systems and voting in the examined domains. Therefore, we comparatively evaluated all combination methods (voting and stacking) for IE, as described in Sections 3 and 4, while also comparing against the best base-level system for each domain of interest. Since the success of stacking relies on the disagreement in the output of the base-level components, we were particularly interested in how stacking behaves with respect to the diversity in the output of the base-level systems, and how that compares to voting.

5.1 Domains

Experiments were conducted using five collections of text documents from five different domains. The first two collections consist of 101 Web pages describing computer science (CS) courses and 96 Web pages describing research projects respectively, and were constructed in the context of the WebKB project (Craven et al., 1998). Both collections were hand-filled for three and two extraction fields respectively: *crsNumber*, the official number of the course (for example, "CS 305"), *crsTitle*, the title of the course (for example, "Operating Systems"), *crsInst*, the name of the course instructor, *projTitle*, the title of the project (for example, "WebKB") and *projMember*, the name of a project member.

The third collection consists of 50 Web pages, describing laptop products that were collected from various vendor sites. A total of 19 fields were hand-filled, including the manufacturer of the laptop, model name, processor name, speed, ram, hard disk capacity, etc. This collection was constructed in the context of building a shopping comparison agent[2] that visits various vendor sites, extracts laptop descriptions and presents the results to user.

The last two collections consist of 300 pages describing job announcements from the *austin.jobs* newsgroup at Austin and 485 pages describing seminar announcements from the Carnegie Mellon University, respectively. Both collections were obtained from RISE (1998) and have been widely used in information extraction research. A total of 17 fields were hand-filled for job announcements, including the title of the available position, the salary, the name of the company, the identifier code of the announcement, etc. Four fields were hand-filled for seminar announcements: *stime*, the starting time of the seminar, *etime*, the ending time of the seminar, *speaker*, the speaker's name and *location*, the location of the seminar.

Note that the available hand-filled templates for job announcements (Califf and Mooney, 2003) do not contain information about the starting and ending token boundaries of the annotated instances, which is however essential for combining different IE systems. Therefore, the entire corpus was re-annotated, using the available hand-filled templates as a guide, so that the new templates include token boundary information, as the one in Table 2(b).

Finally, the HTML tags in the three Web domains were not omitted during the training of the base-level systems, but appropriately tokenized, including their attributes and values. For example, the stream *<td valign="top">* corresponds to the subsequence "*td_start_tag*", "*attrib_valign*", "*value_top*" in the token table of a Web page.

[2] CROSSMARC, R&D project, IST-2000-25366, *http://www.iit.demokritos.gr/skel/crossmarc*

5.2 Base-level Information Extraction Systems

At the base-level we employed three systems that are well known in the literature for information extraction: the $(LP)^2$ system, the BWI system and a HMM-based IE system.

The *Learning Pattern by Language Processing* or $(LP)^2$ system implements a sequential covering algorithm that learns symbolic pattern-matching rules for IE. For each interesting field, a set of start-rules and another one of end-rules are induced that identify the starting and ending boundaries respectively of the relevant instances. Shallow natural language knowledge is used during the induction process such as lexical information (e.g. *capitalized*, *numerical*), part-of-speech tagging (for example, *noun*) and stemming information. Additional *contextual* and *correction* rules are learned that improve the performance of the previously induced rules. Each instance $< t(s,e), f >$ that is recognized at runtime is assigned a confidence score $LS = wrong / matched$, where $wrong$ is the number of erroneous matches, as estimated during training, of the rules that matched $t(s,e)$, and $matched$ is the total number of matches by the rules. The lower this score, the higher the confidence attached to the instance. A detailed description of $(LP)^2$ can be found in (Ciravegna and Lavelli, 2003).

The *Boosted Wrapper Induction* or BWI system learns also symbolic starting and ending pattern-matching rules for each relevant field. Each rule is assigned a confidence score according to a boosting methodology, which is described in more detail in (Freitag and Kushmerick, 1999). The induced rules exploit lexical information (e.g. *capitalized*, *numerical*). Each instance $< t(s,e), f >$ recognized at runtime is assigned the product of confidences of the start and end rules that match $t(s,e)$. The more rules match $t(s,e)$, the higher is the value of the score that is assigned to the instance. A comprehensive analysis of the performance of BWI in a variety of IE tasks can be found in (Kauchak et al., 2004).

Finally, our HMM-based IE is inspired by work described in (Freitag and McCallum 1999; Seymore et al., 1999), whereby a separate HMM is trained for each relevant field. The entire training page is probabilistically modelled by the HMM, by assuming that the first token of the page is emitted by the initial state of the model, then transitioning to the next state that emits the second token, etc. until the ending token of the page is emitted. Special *prefix*, *suffix* and *target* states model the immediate prefix, suffix and the internal structure of the relevant instances respectively. Inducing a HMM for each field involves the calculation of the state-transition and token-emission probabilities over all training pages, based on simple ratios of counts. The *Viterbi* algorithm is used at runtime to identify relevant instances and assign to them a confidence score. More details on how HMMs assign confidence scores for IE can be found in (Sigletos, 2005). An excellent tutorial on HMMs can be found in (Rabiner, 1989).

5.3 Meta-level Algorithms for Classification

At meta-level, we employ the following algorithms, implemented in the WEKA data mining platform (Witten and Frank 2000):

- *J48*, the well known C4.5 (Quinlan, 1993) decision tree algorithm.
- *NaiveBayes*, the well known Naïve Bayes classifier (John and Langley, 1995).
- *IB1*, the 1-nearest-neighbor algorithm.
- *Multi-response linear regression (MLR)*, an adaptation of least-squares linear regression (Breiman, 1996a).
- *SMO*, a fast implementation of Support Vector Machines (Platt, 1999).
- *LogitBoost*, an implementation of the corresponding algorithm (Friedman et al., 1999), using decision stumps as weak-classifier.

Most of these algorithms have already been evaluated as meta-level classifiers in recent studies for stacking (Ting and Witten, 1999; Seewald, 2003; Džeroski and Ženko, 2004).

5.4 Evaluation Methodology and Metrics

For the evaluation, cross-validation was used to obtain an unbiased estimate of performance over unseen data. For the domains of laptop products and job announcements, the corpus was randomly split into 5 equally populated parts. At each fold, a different part of pages was kept for evaluation, and the pages in the remaining four parts were used in order to induce the base-level systems and the meta-level classifiers. Results on the test parts were averaged over all 5 folds. Notice that within the training part at each of the 5 folds, a separate 5-fold cross-validation procedure was used in order to create the meta-level set of feature vectors and thus train the classifiers, as described in Section 4.2.

For the two WebKB domains and seminar announcements, a different evaluation methodology was followed that was also applied in (Freitag, 2000). Each corpus was randomly split into 2 parts of almost equal size. The first part was used to induce the base-level systems and the meta-level classifiers, while the second part was used for evaluation. An internal 3-fold cross-validation process was followed in the training part in order to collect the meta-level set of feature vectors and then train the classifiers. The whole process was repeated 5 times, averaging the results at the end. Moreover, the constraint of "one instance per document" (OPD) was applied for the fields *crsNumber* and *crsTitle* in CS courses, for the field *projTitle* in research projects, and for all four fields in seminar announcements, towards an objective comparison against multistrategy learning for information extraction and the results presented in (Freitag, 2000). Therefore, whenever two or more instances of an OPD field were present within a page, only the instance with the highest score was selected.

Three metrics were used for measuring the performance: *precision (P)*, the percentage of the identified field instances that are correct, *recall (R)*, the percentage of the annotated field instances (in the hand-filled templates) that were identified, and finally $F1$, the harmonic mean of recall and precision defined as $F1 = 2RP/(R+P)$. Our definition of recall and precision is equivalent to *micro-average recall* and *micro-average precision* (Sebastiani, 2002), formally defined as:

$$P = \frac{TP}{TP+FP} = \frac{\sum_{I=1}^{Q} TP^I}{\sum_{I=1}^{Q} (TP^I + FP^I)}, \qquad R = \frac{TP}{TP+FN} = \frac{\sum_{I=1}^{Q} TP^I}{\sum_{I=1}^{Q} (TP^I + FN^I)},$$

where TP^i is the number of instances of the field $f^i \in \{f^1, ... f^Q\}$ that have been correctly identified (*true positive*), FP^i is the number of f^i instances that have been incorrectly identified (*false positive*), and finally FN^i is the number of f^i instances that were not identified (*false negative*). Choosing micro-average metrics allows for an objective overall comparison among different systems, by considering all target instances by all relevant fields. Statistical significance in the conducted comparisons was evaluated using the well-known paired *t*-test (Dietterich, 1998) with a significance level of 95%.

6 Results and Comparisons

The results obtained by all base-level systems in the domains of interest are initially presented in this section, while also investigating whether any improvement in the best results for each domain is possible at meta-level. Then, the meta-level data is analyzed, in order to determine whether and how the predictions of the base-level systems are correlated. This study is intended to serve as a basis for a comparative evaluation of voting against stacking. Then all combination methods are comparatively evaluated, while also comparing against the best base-level results. More detailed analysis of the experimental results is provided in Section 7.

6.1 Results of Base-level

Table 7 shows the $F1$ scores (%) obtained by the base-level systems in the domains of interest. Only the highest $F1$ score for research projects was measured as statistically insignificant. Appendix B.1 shows the scores obtained in all three measures of performance.

	CS courses	Projects	Laptops	Jobs	Seminars
BWI	51.30	60.75	62.26	80.01	83.09
HMM	59.39	61.64	63.81	75.71	79.20
(LP)^2	65.73	58.82	61.26	83.22	86.23

Table 7. Base-level $F1$ scores (%) for the five domains.

The simple choice is to select the best base-level system for each domain. On the other hand, a more desirable approach is to try to exploit the diversity in the output of all systems, hoping to improve the best base-level results. Note that there is no generally accepted measure for diversity, but a variety of measures exist in the literature (Kuncheva and Whitaker, 2003). Ali and Pazzani (1996) define the similarity between two classifiers, as the conditional probability that both classifiers make an error, given that either of them makes an error.

Tables 8(a) to 8(e) are instances of the "contingency table" that was introduced by Freitag (2000) for measuring the similarity in the output of pairs of systems, and inspired by Ali and Pazzani's measure. Each cell in a contingency table measures the conditional probability that the row system makes the correct prediction, given that the column system also predicts correctly. Tables 8(a) to 8(e) suggest that there is space for improving the best base-level system in each domain. For example, in CS courses we notice that only in 69% of the meta-level instances where HMMs yield a correct prediction, $(LP)^2$ also predicts correctly. Thus, in the remaining 31%, where HMMs predict correctly, $(LP)^2$ either predicts an incorrect field, or does not predict any field. Therefore, the performance of $(LP)^2$, which is the best system for this domain, can be further improved.

	BWI	HMM	$(LP)^2$
BWI	1	0.46	0.44
HMM	0.70	1	0.52
$(LP)^2$	0.89	0.69	1

(a) Courses

	BWI	HMM	$(LP)^2$
BWI	1	0.82	0.79
HMM	0.90	1	0.79
$(LP)^2$	0.69	0.62	1

(b) Projects

	BWI	HMM	$(LP)^2$
BWI	1	0.73	0.78
HMM	0.89	1	0.83
$(LP)^2$	0.87	0.76	1

(c) Laptops

	BWI	HMM	$(LP)^2$
BWI	1	0.83	0.86
HMM	0.91	1	0.85
$(LP)^2$	0.93	0.85	1

(d) Jobs

	BWI	HMM	$(LP)^2$
BWI	1	0.87	0.83
HMM	0.91	1	0.84
$(LP)^2$	0.95	0.92	1

(e) Seminars

Table 8. Contingency tables, measuring the agreement in the predictions of the base-level systems. Each cell is the probability that the row system makes a correct prediction, given that the column system makes a correct prediction.

6.2 Analysis of the Meta-level Instances

Each meta-level instance corresponds to a text fragment $t(s,e)$ that has been identified by at least one base-level system, together with the predicted fields for $t(s,e)$ by the base-level systems, the associated probabilistic estimates, and finally the correct human-annotated field for $t(s,e)$. Figure 4 shows a partition of the meta-level instances in the testing corpus, according to whether all systems agree on the same field for a fragment $t(s,e)$ or not.

The leftmost column for each domain in Figure 4 shows that there are regularities in the text documents that can be easily recognized by all available IE systems. For example, the fragment "TFT" is a typical instance of the field *screen type* in the domain of laptops that commonly appears in both training and testing corpus and thus easily detected by all systems.

Observing the rightmost column for each domain in Figure 4, leads to the interesting conclusion that situations where at least two base-level systems predict different fields for a text fragment are not frequent. In order to explain that, one should note that the IE systems exploit both the target content and the surrounding context of the relevant instances, and thus are capable of disambiguating among field instances with similar content. For example, instances of the fields *cdromSpeed* and *dvdspeed* contain similar content, e.g. "24x". A system that simply memorizes field instances verbatim predicts both fields for "24x". Our base-level systems, on the other hand, are capable of examining surrounding tokens such as "cd" or "dvd", and thus distinguish among the two fields. In some cases, however, those regularities in the context were difficult to find, either because they are less apparent, or due to limitations in the context that the base-level systems search for, thus resulting in contradictory fields.

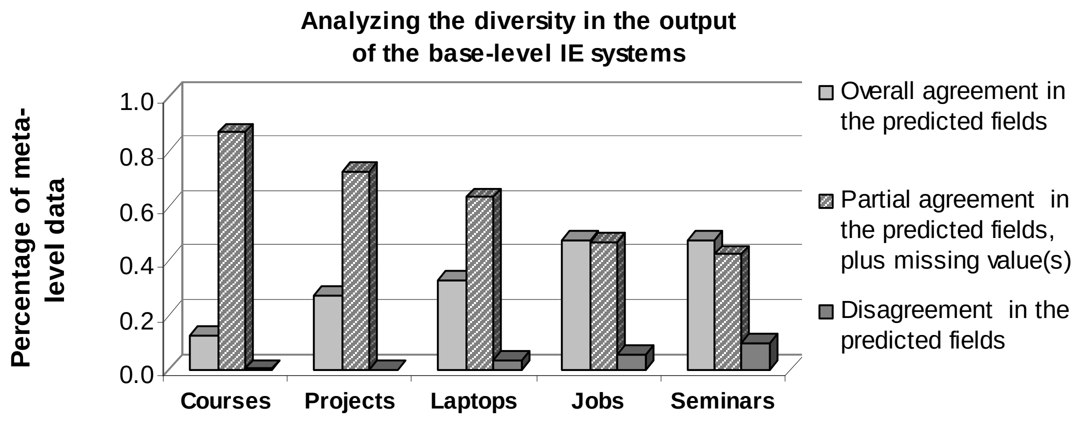

Figure 4. Partitioning the meta-level instances for each domain into three disjoint sets, according to whether all systems agree on the same field for a text fragment (left column), or some system(s) predict the same field while the other(s) abstain from prediction (middle column), or there are at least two contradictory predictions (right column).

For example, (LP)^2 explores a window of w tokens to the left and w tokens to the right of the starting and ending boundaries of the annotated field instances, where the value of w is predefined. Also in HMMs, a predefined number of *prefix* and *suffix* states model the immediate prefix and suffix respectively of the annotated field instances. The largest rate of disagreement appears in seminars (9.9%). This is because the ending time (field *etime*) of a seminar was many times confused with the starting one (field *stime*), thus increasing the size of the rightmost column in Figure 4 for this domain. In CS courses and research projects, contradictory predictions do not exceed 0.5% of all meta-level data, in laptops they are 3.7%, while for jobs they are 5.5%.

Since differences in the predicted fields for a text fragment are not frequent, the interesting question is what kind of disagreement can both voting and stacking exploit in pursuit of improved performance at meta-level? The answer lies in the middle column for each domain, which indicates that the majority of the meta-level instances derive from text fragments that have been assigned identical fields by some, but not all, system(s), while the remaining system(s) abstain from prediction. Since we deal with three systems, this corresponds to situations where either two systems predict the same field, plus a missing prediction by the third system or only one system predicts a field, plus two missing predictions. Therefore, we expect voting and stacking to exploit this kind of disagreement, leading to better results at meta-level. It is also interesting to observe the behaviour of stacking when the predictions by all systems are identical, according to the left column for each domain in Figure 4.

Note finally that the partition of meta-level data as shown in Figure 4, will allow for a more comprehensive comparison of stacking against voting, while also exploring the various aspects of the behaviour of stacking and voting, with respect to the varying degree of correlation in the output of the base-level systems. On the other hand, Table 8 shows a quantitative analysis of the disagreement among pairs of different IE systems that helps in determining whether there is space for improving the best system for each domain.

6.3 Results of Meta-level and Comparisons

Let *MVotM* and *MVotF* be the two majority voting settings, as defined in Section 3.2. The former setting ignores missing field prediction by some system, while the latter setting records missing prediction as "false". Let also *PVotM* and *PVotF* be the two corresponding settings of voting using probabilities, as defined in Section 3.3. In *PVotM*, missing predictions are ignored. In *PVotF*, if the highest probability for a field f is less than 0.5, then f is rejected. Table 9 shows the best $F1$ scores obtained by all voting settings, stacking with nominal values, stacking with probabilities, and by the best base-level systems. Appendix A summarizes again all combination methods, along with a short description for each method.

	Base	MVotM	MVotF	PVotM	PVotF	Stacking Nominal	Stacking Probs
Courses	65.73	65.59	60.29	65.65	70.64	63.92	71.93
Projects	61.64	60.71	67.39	60.75	65.75	66.05	70.66
Laptops	63.81	62.37	67.60	62.76	71.03	68.46	71.55
Jobs	83.22	79.90	83.85	79.99	83.15	85.67	85.94
Seminars	86.23	86.87	87.13	86.90	88.02	88.48	90.03

Table 9. Best $F1$ scores (%) obtained by all combination methods (voting and stacking) and by the best base-level system for each domain of interest. For fair comparisons, the results of a single classifier (*LogitBoost*) were used by both stacking settings.

Tables 10 to 12 compare all combination methods and the best base-level system, based on statistically significant *wins* against *losses*, in the five examined domains. Appendices B.2 to B.4 show detailed numerical values for the three measures of performance by all combination methods in each domain of interest. Detailed results per field can be found in (Sigletos, 2005).

	Best Base	MVotM	MVotF	PVotM	PVotF	Stacking Nominal	Stacking Probs
Best Base		5\0	0\5	5\0	2\2	0\5	0\5
MVotM	0\5		0\5	0\1	0\5	0\5	0\5
MVotF	5\0	5\0		5\0	5\0	2\0	3\1
PVotM	0\5	1\0	0\5		0\5	0\5	0\5
PVotF	2\2	5\0	0\5	5\0		0\5	0\5
Stacking Nominal	5\0	5\0	0\2	5\0	5\0		0\3
Stacking Probs	5\0	5\0	1\3	5\0	5\0	3\0	

Table 10. Statistically significant *wins* against *losses* in the five domains, based on *precision*, of the row system against the column one.

	Best Base	MVotM	MVotF	PVotM	PVotF	Stacking Nominal	Stacking Probs
Best Base		0\5	4\0	0\5	0\4	2\2	2\3
MVotM	5\0		5\0	0\1	5\0	5\0	5\0
MVotF	0\4	0\5		0\5	0\5	0\4	0\4
PVotM	5\0	1\0	5\0		5\0	5\0	5\0
PVotF	4\0	0\5	5\0	0\5		4\0	4\0
Stacking Nominal	2\2	0\5	4\0	0\5	0\4		0\3
Stacking Probs	3\2	0\5	4\0	0\5	0\4	3\0	

Table 11. Statistically significant *wins* against *losses* in the five domains, based on *recall*, of the row system against the column one.

	Best Base	MVotM	MVotF	PVotM	PVotF	Stacking Nominal	Stacking Probs
Best Base		2\0	1\2	1\0	0\4	0\2	0\5
MVotM	0\2		1\3	0\1	0\5	0\3	0\5
MVotF	2\1	3\1		3\1	1\3	0\3	0\5
PVotM	0\1	1\0	1\3		0\5	0\3	0\5
PVotF	4\0	5\0	3\1	5\0		2\2	0\3
Stacking Nominal	2\0	3\0	3\0	3\0	2\2		0\4
Stacking Probs	5\0	5\0	5\0	5\0	3\0	4\0	

Table 12. Statistically significant *wins* against *losses* in the five domains, based on $F1$, of the row system against the column one.

6.4 Discussion

We observe that stacking with probabilities obtains a higher $F1$ score than the best base-level system for each domain. Precision is also improved (substantially for the domains of research projects and laptop products) in all five domains. Recall is improved in three domains but harmed in the remaining two. Stacking with simple nominal values, on the other hand, outperforms the best base-level $F1$ score in only two of the five domains. Note that the large improvement obtained by stacking with nominal values in projects and laptops is not consistent across all folds during evaluation, and thus measured as statistically insignificant. Overall, stacking with probabilities outperforms simple stacking with nominal values. Only for job offers, the obtained improvement in $F1$ against simple stacking was measured as statistically insignificant. Handling missing values in stacking did not significantly influence the results.

Regarding voting, *PVotF* performs comparably or better than the best base-level system for each domain. Recall is improved by *PVotF* at meta-level in most domains. Precision is however improved only in two domains (projects and laptops). Among all voting settings, *PVotF* is best for courses, laptops and seminars, while *MVotF* is slightly better only for jobs. For projects, the improvement obtained by *MVotF* against *PVotF* was measured as statistically insignificant. Overall, *PVotF* is the best among all voting settings.

By comparing voting against stacking, we observe that stacking with probabilities outperforms *PVotF* in all five domains, although the difference is statistically significant only in three domains. Unlike *PVotF*, stacking with probabilities is also consistently effective across all five domains, always outperforming the best base-level system for each domain. Moreover, stacking with probabilities always obtains more precise results than *PVotF*, at the cost of somewhat lower recall. Overall, stacking with probabilities achieves the best results among the combination methods that were evaluated.

6.4.1 Best Classifiers at Meta-level

An interesting result in the stacking with probabilities setting is that the highest $F1$ scores in all domains were obtained by the same classifier: *LogitBoost*. Only for projects, the classifier *j48* obtained a higher, but statistically insignificant, $F1$. On the other hand, *LogitBoost* using nominal values has not been consistently effective over all five domains. The best classifier using nominal values in CS courses was *IB1*, achieving however an insignificantly higher $F1$ score than the best base-level system for this domain.

Table 13 compares the six classifiers in stacking with probabilities, the setting that leads to the best meta-level results, based on statistically significant *wins* against *losses*, in the five examined domains. Appendix B.5 shows the $F1$ scores obtained for the five domains by all classifiers in both simple stacking with nominal values and stacking with probabilities.

	IB1	J48	LogitBoost	MLR	NaiveBayes	SMO
IB1	■	0\3	0\5	1\1	3\0	0\2
j48	3\0	■	0\3	2\0	5\0	1\0
LogitBoost	5\0	3\0	■	5\0	5\0	5\0
MLR	1\1	0\2	0\5	■	3\1	0\2
NaiveBayes	0\3	0\5	0\5	1\3	■	0\2
SMO	2\0	0\1	0\5	2\0	2\0	■

Table 13. Stacking with probabilities. Statistically significant *wins* against *losses* in the five domains, based on $F1$**,** of the row classifier against the column one.

Clearly a variety of other classifiers could also be evaluated at meta-level. The aim of our experiments was to demonstrate the effectiveness of utilizing a common classifier in the context of combining multiple IE systems. Most of the employed classifiers have been also evaluated in recent studies for stacking, with *MLR* to be a state-of-the art approach (Ting and Witten, 1999; Seewald, 2003; Džeroski and Ženko, 2004). *MLR* did not prove to be that effective in our experiments for IE as in the cited studies for common classification. Nevertheless, all meta-level classifiers in the cited studies, including *MLR*, have access to full probability distributions over all relevant classes, produced by the base-level classifiers. Such distributions are not typical for IE systems.

On the other hand, Table 13 shows that boosting simple decision stumps, through *LogitBoost*, was particularly effective at meta-level. Only for projects and seminars, the difference among *LogitBoost* and *j48* was measured as statistically insignificant.

6.4.2 Majority Voting

Tables 10 and 11 show that *MVotM* improves recall but hurts precision, as compared to the best base-level system for each domain, while *MVotF* improves precision but hurts recall. This contradicting behaviour is mainly due to situations where only one system predicts a field f for a text fragment, while the remaining two systems abstain. In such cases, if f is not the correct field for the fragment, this harms precision for *MVotM*, which always accepts f. If f is the correct field, then this harms recall for *MVotF*, which always returns “false”, which is the value with the highest count. Recall is harmed the most in CS courses by *MVotF*, thus also harming $F1$, as it is shown in Appendix B.2, which is due to the fact that single predictions are mostly correct for this domain.

Note that since contradictory field predictions for a text fragment are not frequent, according to Figure 4, *MVotM* does not actually behave like voting. In the overwhelming majority of meta-level data, only one field f participates in the vote counting process. The more systems predict f the higher the count is for f, which is then returned. However, if f is not the correct field, then there is no way to reject it, since missing predictions are ignored and thus precision is harmed. Nevertheless, the recall obtained by *MVotM* approximates the maximum recall that can be obtained at meta-level for each domain, since each base-level system also contributes its uniquely identified correct instances. On the other hand, *MVotF* does behave more like voting, since each missing prediction for a fragment is encoded with the value “false”, which then participates in the vote counting.

The overall conclusion is that neither majority voting setting has been consistently effective, based on $F1$, over all five domains. *MVotF* achieves a higher $F1$ score at meta-level that is statistically significant only for projects and seminars, while *MVotM* does not significantly improve the best base-level $F1$ score in any of the five domains. The large improvement that *MVotF* obtains for laptops is not consistent over all folds during evaluation, and thus measured as statistically insignificant. In addition, we would like to *learn* when a field f is correct, instead of accepting f by *MVotM*, if it has the highest count, or rejecting f by *MVotF* , if the value with the highest count is “false”. Stacking using probabilities achieves this goal, outperforming both settings of majority voting in the five examined domains.

6.4.3 Voting Using Probabilities

Tables 9 to 12 show that the performance of *PVotM* is similar to *MVotM*. Since the overwhelming majority of meta-level instances contain no contradictory field predictions for a text fragment, according to Figure 4, the additional use of probabilities has not proved particularly useful for *PVotM*. The higher the number of votes for f, the higher is the combined probabilistic estimate for the field. Only for laptops, the slight improvement in $F1$ by *PVotM* is statistically significant. As a result, *PVotM* approximates, slightly better than *MVotM*, the maximum recall that can be obtained at meta-level for each domain.

Note that *PVotF* performs an additional test to the field f that is returned by *PVotM*, by examining whether or not the probability associated with f is greater than 0.5, and thus accepts or rejects f. This leads to more precise results for *PVotF*, as compared to *PVotM*, suffering a lower recall. The improvement in precision is however substantial and leads to higher $F1$ score for *PVotF*, since most incorrect predictions are associated with a probability that is less than 0.5. Moreover, if the value with the highest count is “false”, i.e. two systems abstain from prediction, *PVotF* examines the probability of the field f with the next highest count, i.e. the prediction of the remaining system. This explains both the higher recall and lower precision for *PVotF*, against

MVotF that always returns "false" in this case. Although *PVotF* does significantly better than *MVotF* in three domains, the decrease in precision by *PVotF* is higher than the increase in recall for the remaining two (projects and jobs), thus resulting in a comparable or better $F1$ score for *MVotF*. This is due to more situations for the two domains where single predictions are both incorrect and assigned a high probability, and thus correctly rejected by *MVotF*, while incorrectly accepted by *PVotF*.

Although the calibrated probabilities of correctness are more meaningful and consistent than confidence scores, choosing 0.5 as a threshold below which to reject predictions may not always be an optimal choice, as showed in Figure 5 for CS courses.

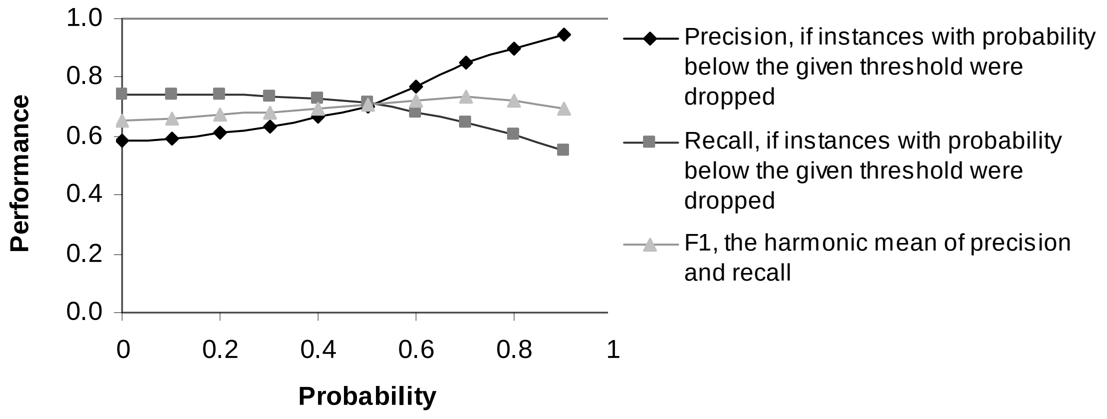

Figure 5. Precision, recall and $F1$ by *PVotF* for the domain of CS courses, regarding the threshold below which, the predicted instances are removed.

The performance of *PVotF* for a probability threshold zero equals that of *PVotM*, where no prediction is rejected. Increasing the threshold, below which predictions are rejected, a tradeoff between recall and precision is observed, reflecting the natural fact that a rejected prediction may be either a correct one, thus harming both precision and recall, or incorrect and thus improving precision. Nevertheless, incorrect predictions are mostly rejected as threshold increases, since most correct predictions are assigned a high probability.

As already mentioned in Section 2.3.3, the mapping of confidences to probabilities takes place by validating the performance of each base-level system on a hold-out set or by cross-validation. This only approximates the *true* probability distribution in the predictions of an IE system over the space of all possible relevant pages. As a result, the optimal threshold below which to reject predictions may vary for different collections of pages. In Figure 5, despite the fact that both precision and recall are almost equally balanced at threshold 0.5, the optimal $F1$ score (73.5%) is achieved at threshold 0.7. For jobs, the optimal $F1$ score (84.08%) is also achieved at threshold 0.7. All differences were measured as statistically significant. For research projects, seminars and laptops, the best $F1$ score remains at threshold 0.5.

However, instead of optimizing the threshold empirically, e.g. in a jack-knifing procedure, it is much more interesting to try to *learn* whether or not to accept a prediction. In other words, we would like to learn the correlation among the probabilistic estimates, returned by the individual base-level systems, towards better meta-level results. Stacking using probabilities achieves this goal in most domains by outperforming *PVotF*, while obtaining more precise results even when both settings perform comparably.

6.4.4 Multistrategy Learning for Information Extraction

Table 14 compares the $F1$ scores obtained by the multistrategy learning setting, as described in Section 2.3.3, against the best obtained scores at the base-level, voting using probabilities, and the results presented in (Freitag, 2000). Results are only presented for the domains of CS courses, research projects and seminars, since these domains were used by Freitag (2000).

	Best Base	PVotM	PVotF	Multistrategy	Multistrategy (Freitag, 2000)
crsNumber	94.46	95.72	95.92	94.91	88.9
crsTitle	70.05	73.50	72.34	73.29	62.0
crsInst	48.21	50.81	57.76	50.81	49.8
projMember	65.00	63.16	68.94	63.09	45.5
projTitle	39.66	34.26	32.88	35.65	34.1
stime	99.09	99.51	99.51	99.42	99.3
etime	97.62	89.09	96.68	67.15	94.3
speaker	73.41	75.88	75.40	75.58	66.2
location	77.43	81.82	81.83	81.82	79.7

Table 14. Best per field $F1$ scores (%) by base-level, voting and multistrategy learning for the domains of CS courses, research projects and seminar announcements.

Our results for multistrategy perform comparably or better for most fields against the ones in (Freitag, 2000), which is partially due to the higher performance of our systems. Multistrategy learning and *PVotM* behave similarly for all fields but for *etime*. Instances of *etime* are many times confused with instances of *stime* by our IE systems, thus resulting in contradictory field predictions. Multistrategy, however, considers each field in isolation and thus fails to distinguish among the two fields. Voting handles better field ambiguities, by selecting the one with the highest probability, thus leading in significantly higher $F1$ score than multistrategy for ambiguous fields. In some cases the highest probability does not correspond to a correct prediction, which justifies why the $F1$ for *etime* is not improved monotonically at meta-level.

Contradictory field predictions do not occur frequently in most fields, and thus *PVotM* mostly handles each field in isolation, as multistrategy does by default. In such cases, and just like *PVotM*, the success of multistrategy strongly depends on how the incorrect predictions by all base-level systems correlate. The more incorrect instances each system uniquely identifies, the higher the decrease in precision is, since missing predictions are ignored and there is no correct field to contradict the incorrect one during voting. This may lead to a worse $F1$ score, compared to the best base-level system, as in research projects. Overall, *PVotF* is the best of the voting settings that were evaluated in this article.

7 Explaining the Results

In Section 6.2, we partitioned the meta-level instances according to how the base-level systems correlate in their output. In this section we compare stacking against voting with respect to this diversity analysis. The aim is to provide useful insight into the behaviour of voting and stacking by comparatively studying their performance, based on the varying degree of disagreement in the output of the base-level systems. In the interest of conducting a fair analysis, the results of a single classifier *(LogitBoost)* are used for stacking.

7.1 Analyzing Cases of Complete Agreement in the Output of the Base-level Systems

Figure 6 compares all combination methods, based on the number of correctly classified meta-level instances, when all base-level systems agree. These instances correspond to the left column in Figure 4 for each domain. Recall that each meta-level instance can be classified into one of the values $\{f^1 ... f^Q, \mathit{false}\}$, where $\{f^1 ... f^Q\}$ the relevant fields and *false* a special value indicating that the text fragment, which corresponds to the meta-level instance, is irrelevant, and thus none of the base-level systems should have predicted a field for it.

Stacking with probabilities performs slightly better than all other combination methods, while *PVotF* follows. In other words, stacking proved to be slightly more useful even when the predictions by the base-level systems are identical. Note that since all three systems agree on the same field, all voting settings except *PVotF* return the same value. Only *PVotF* has the additional option of rejecting a prediction by all systems, if the combined estimate is less than a given threshold, which has proven slightly useful only for projects and laptops.

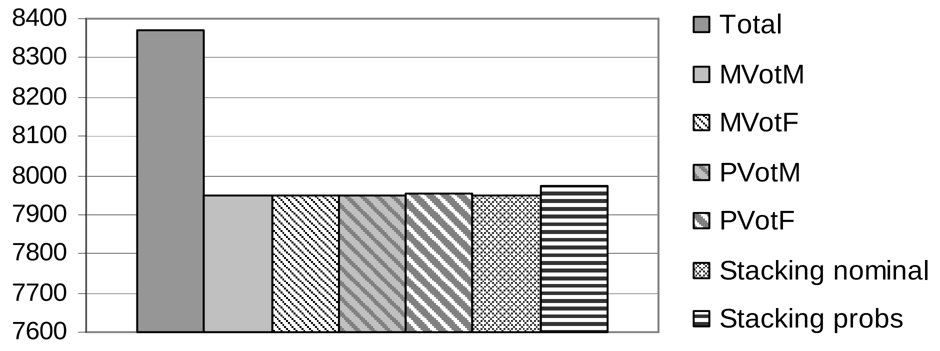

Figure 6. Comparing all combination methods, when all three base-level systems agree on the same field. Sum of correctly classified meta-level instances over all five domains.

Comparing the total number of meta-level instances where all three base-level systems agree (leftmost column in Figure 6) and the results obtained by all combination methods, we conclude that the returned field is not always correct. This was mostly observed in the Web domains and it is partly due to errors during annotation, and mainly due to captured regularities in the text that are not always reliable. For example, "TFT" may be the type of a separate screen product that is described in the same page with a laptop. Similarly for projects, some faculty or student names are former project members, and thus have not been annotated by the human expert. In such cases, stacking with probabilities learned to reject, i.e. classify as "false", some of those erroneous predictions. We did not expect to do much better without encoding other features in the meta-level vectors.

The last observation indicates a limitation of our base-level systems which is not unknown in the literature. The limitation is that the IE systems parse a document as a linear sequence of tokens and thus ignore hierarchical structure available within an HTML or XML document, e.g. through using the document object model (DOM). Therefore, our IE systems only capture regularities in text that concern sequences of tokens within the target content and surrounding content of the relevant instances and thus fail to generalize over hierarchical information. For example, we would like to exploit HTML information to separate laptops from other products that are described within the same page.

There exist approaches in the literature that exploit HTML or XML structure in the context of IE, but they mostly suffer from the need of extensive manual effort and/or the lack of adaptability to different structure formats. For example, STALKER (Muslea et al., 2001) learns patterns that match certain sequences of tokens/wildcards, separated by irrelevant intermediate text. This however requires the manual construction and use of an "embedded-catalog" (EC) tree for Web pages that share the same structure in their content, and thus separate trees have to be manually constructed for different structure formats.

Davulcu et al. (2002) exploit DOM-based information in conjunction with a hand-crafted ontology for IE from Web pages with different structure. Incorporating combination methods for IE with their techniques is an issue to be investigated. For example, the hand-crafted patterns for the item-identifiers within the ontology could be replaced by more complex patterns induced by combining a set of IE systems, and thus reduce the effort required for engineering the ontology. Initial efforts towards incorporating machine learning into ontology engineering already exist (Valarakos et al., 2004).

7.2 Analyzing Cases of Partial Agreement in the Output of the Base-level Systems

Figure 7 compares all combination methods based on the number of correctly classified meta-level instances, when some base-level system(s) agree on the same field, while the remaining one(s) abstain from prediction. These instances correspond to the middle column for each domain in Figure 4. In order to analyze the results into greater depth, Figure 7 presents separately the results, based on whether a single base-level system predicts a field, or exactly two out of the three systems agree on the same field.

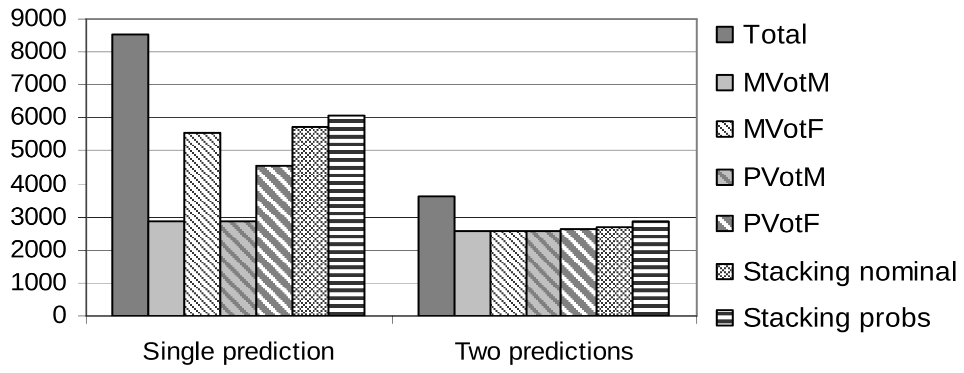

Figure 7. Comparing all combination methods, when either a single or exactly two base-level systems agree on the same field (set of columns on the left and right respectively). Sum of correctly classified meta-level instances over all five domains.

Figure 7 shows the superiority of stacking with probabilities in both situations. Handling missing values does not significantly influence the results. The left part in Figure 7 also confirms the complementary behaviour of *MVotM*/*PVotM* with *MVotF*. When only one system predicts a field for a text fragment, then the column size of *MVotM/PVotM* equals the number of cases where the predicted field is correct. The size of *MVotF* equals the number of cases where the predicted field is incorrect, and thus “false” is returned. The large size of the *MVotF* column in the left part of Figure 7 indicates that single-field predictions are more probably incorrect. The left part of Figure 7 shows that stacking with probabilities learns more than simple stacking with nominal values does and goes beyond what *MVotM*/*PVotM* and *MVotF* straightforwardly deduce in a contradicting manner, by obtaining a higher accuracy than all settings.

When two predictions agree on the same field and the third one is missing, all voting settings, except *PVotF*, obviously return the same field, as also shown in the right part of Figure 7. Stacking with nominal values and *PVotF* perform slightly better than the other voting settings. On the other hand, stacking with probabilities again learns something more than simple stacking does and goes beyond what all voting settings straightforwardly deduce.

7.3 Analyzing Cases of Disagreement in the Output of the Base-level Systems

Figure 8 compares all combination methods, based on the number of correctly classified meta-level instances, when at least two base-level systems contradict in their field predictions. These instances correspond to the right column in Figure 4 for each domain.

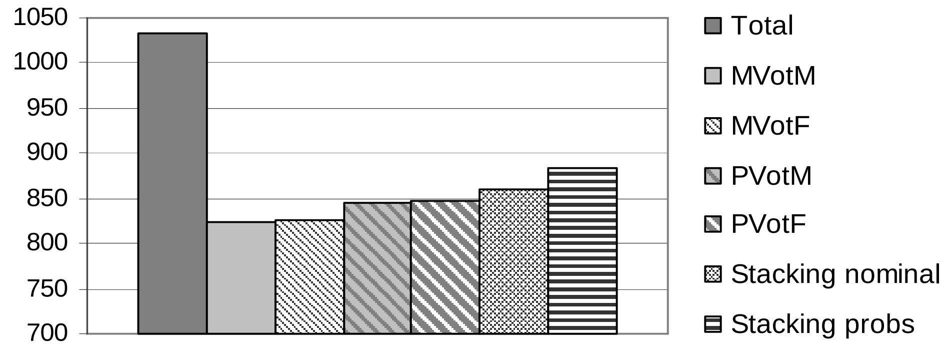

Figure 8. Comparing all combination methods when the base-level systems disagree. Sum of correctly classified meta-level instances over all five domains.

Figure 8 confirms the superiority of stacking with probabilities. Figures 6 to 8 indicate that there is enough room for further improving the results. On the other hand, the presented results are very positive, considering the exploitation at meta-level of simple nominal values and probabilities of correctness in the output of the base-level systems.

7.4 Comparing by Meta-level Classification Accuracy

Figures 6 to 8 compare voting and stacking using the number of correctly classified meta-level instances, summed over all domains, with respect to the different degree of agreement in the output of the base-level systems. *Classification accuracy* can then be defined as the fraction of meta-level instances that have been correctly classified. Table 15 compares again all combination methods, using classification accuracy as a measure of performance when estimating the statistically significant *wins* against *losses*.

	MVotM	MVotF	PVotM	PVotF	Stacking Nominal	Stacking Probs
MVotM		0\4	0\1	0\5	0\5	0\5
MVotF	4\0		4\0	2\0	0\3	0\5
PVotM	1\0	0\4		0\5	0\5	0\5
PVotF	5\0	0\2	5\0		0\3	0\5
Stacking Nominal	5\0	3\0	5\0	3\0		0\4
Stacking Probs	5\0	5\0	5\0	5\0	4\0	

Table 15. Statistically significant *wins* against *losses*, based on classification accuracy, in the five domains, of the row IE system against the column one.

Table 15 shows the clear superiority of stacking with probabilities over all voting settings. This is in contrast to Table 12, which shows that *PVotF* performs comparably to stacking with probabilities in two of the five examined domains. Moreover, *MVotF* seems to be the best voting setting, which also contradicts Table 12, where *PVotF* is the best setting. Those contradicting conclusions are due to the different metrics that are employed in the two tables. Table 12 shows statistically significant *wins* against *losses*, based on the micro-average $F1$ that is measured over the relevant fields $\{f^1 ... f^Q\}$ for each domain. On the other hand, the classification accuracy that is used in Table 15 is defined over all possible values $\{f^1 ... f^Q, \mathit{false}\}$ that a meta-level instance can be classified into, including the *false* value. Actually, classification accuracy is identical to micro-average precision over all classes $\{f^1 ... f^Q, \mathit{false}\}$ for a domain of interest.

Classification accuracy is typically used for comparing different classifiers over all class values. In IE, however, the class value *false* has a special interpretation, since no such field is annotated by the human expert. Similarly none of the employed systems at the base-level predict the value *false* for a text fragment. The value *false* is, however, an option for the meta-level classifier, indicating that at least one of the base-level systems has incorrectly predicted a field for an irrelevant text fragment. However, the evaluation takes place by comparing the relevant instances in the hand-filled templates and the corresponding templates filled by the IE systems, at either base or meta-level. Therefore, the evaluation metrics *precision*, *recall* and $F1$, are naturally defined over the relevant domain fields $\{f^1 ... f^Q\}$ ignoring *false*.

7.5 Stacking Pairs of Information Extraction Systems

Experiments were also conducted on stacking and voting of pairs of base-level systems, trying to investigate whether combining all three systems provides added value over combining pairs of systems. Table 16 compares stacking with probabilities (again using the same classifier, *LogitBoost*, for fair comparisons) for all possible combinations of base-level systems. The comparison is based on statistically significant *wins* against *losses*, based on $F1$, in the five examined domains. Voting on pairs of base-level systems did not provide any statistically significant added value over stacking in the domains of interest. Therefore, the results of voting on pairs are omitted here.

	BWI +HMM	BWI+(LP)^2	HMM+(LP)^2	BWI+HMM+(LP)^2
BWI +HMM		0\4	0\5	0\5
BWI+(LP)^2	4\0		0\4	0\5
HMM+(LP)^2	5\0	4\0		0\4
BWI+HMM+(LP)^2	5\0	5\0	4\0	

Table 16. Statistically significant *wins* against *losses*, based on $F1$, in the five examined domains, of stacking with probabilities the row base-level systems, against stacking the column ones.

In one domain (CS courses), stacking HMMs with (LP)^2 results in a statistically insignificant difference in $F1$ against stacking all three systems. This concludes that the contribution of BWI to the performance of stacking is not significant for this domain. Moreover, stacking HMMs with (LP)^2 proves better than stacking BWI with (LP)^2, which is not always justified by the performance of the individual base-level systems. For example, BWI obtains a higher $F1$ than HMMs in jobs and seminars. Table 8 explains this behaviour, by observing a higher degree of correlation among (LP)^2, which is the best system for jobs and seminars, and BWI, than the corresponding one among (LP)^2 and HMMs.

The results of Appendix B.1 show that BWI suffers from lower recall in most domains, as compared to HMMs and (LP)^2. A general guideline for choosing which base-level systems to stack would be therefore to prefer systems biased towards recall, rather than precision, since stacking always obtains more precise results at meta-level. Higher recall suggests higher chance of covering instances that have not been identified by other systems, and thus leading to a higher degree of disagreement, in favour of stacking. For example, HMMs obtain higher recall, yet lower precision, than BWI in all five domains. Nevertheless, stacking HMMs with (LP)^2 is the best pair-wise combination scheme, according to Table 16. In Section 4.2, the choice of systems biased towards recall is also suggested, since higher recall leads to higher chance of minimizing the number of cases where relevant text fragments have not been identified by any base-level system and thus cannot be also identified at the meta-level.

The only exception to the above rule of thumb for high-recall base-level systems is in the domain of research projects, where (LP)^2 obtains a significantly lower recall than HMMs, but stacking BWI with (LP)^2 performs better than stacking BWI with HMMs. Table 8 explains again this behaviour, by observing a higher degree of correlation among HMMs and BWI, than the corresponding one among (LP)^2 and BWI.

8 Concluding Remarks

Though effective in improving the performance of multiple learning algorithms, typically voting and stacking restrict their applicability to common classification. This article extended the applicability of voting and stacking to information extraction (IE), and demonstrated their effectiveness using a variety of different algorithms and domains. The disagreement in the output of the IE systems that were employed at base-level has been successfully exploited by voting and stacking, leading to higher extraction performance at meta-level.

Experimental results have also shown that voting and stacking work best when using the confidence scores by the individual base-level systems that have been converted into probabilistic estimates of correctness. Voting using probabilities and setting a threshold, below which meta-level instances are rejected, proved particularly effective in most domains by outperforming the best base-level systems. Stacking using probabilities, on the other hand, proved consistently effective over all domains of interest, doing comparably or significantly better than voting. Precision was always improved by stacking at meta-level, as compared to the best base-level systems, while recall was improved in most domains. Whenever voting and stacking were doing comparably, stacking still obtained more precise results.

Since IE has been transformed into a common classification task at the meta-level, there are many opportunities for further improving the extraction performance. The experimental results that were presented for stacking in this article are encouraging, considering also the simplicity of the features in the meta-level vectors that represent only the output of the base-level systems. Additional information could be exploited by stacking towards better results that further justify the additional computational cost over voting. In the domain of laptop products, for example, instances of "processor speed" appear typically after "processor name" instances, while instances of "ram" usually follow. Exploring such dependencies among extraction fields, or possibly other sources of information, could lead to useful extra features within the meta-level vectors to be exploited by the classifiers. The combination of different classifiers at a higher meta-level could also be examined.

A different stacking strategy could be applied by considering each field in isolation during combination, as proposed in (Freitag, 2000). In that case, a separate cross-validation process would take place in the base-level data set for each relevant field, and the problem would be transformed into a binary-learning task at the meta-level. Such a strategy would also deal with a limitation of cross-validation procedures over text documents that concerns *stratification*. In common classification over feature vectors, a similar distribution of classes is maintained in each fold. In IE, however, typically there is a different distribution of fields in each document and thus it is hard to approximate the same distribution of the fields in each fold. The penalty of stacking separately each field is that we cannot take advantage of the cases of contradictory field predictions in the output of the base-level systems.

Creating feature vectors is just one method of handling the meta-level data. Alternative methods can be investigated. An interesting extension is to appropriately encode the information available at the meta-level as special tags, which can be either embedded within the text or used as additional token features. This would allow the training of common IE systems also at the meta-level, since the meta-level data set would again consist of the same set of annotated text documents, including the additional meta-level information embodied within the text. This would also be aligned with Wolpert's two major features of stacking: that data sets at both base-level and meta-level are of equal size, while a learning algorithm which is applied at the base-level can also be applied at the meta-level.

This article contributes to the direction of realizing the high potential of combination methods in the context of accurately identifying relevant information within the abundant of online text, aiming at a framework that can be easily adapted to new domains.

Acknowledgements

The first author was partially supported by a research scholarship from the Institute of Informatics and Telecommunications of NCSR "Demokritos", Athens, Greece. The authors are grateful to Fabio Ciravegna for offering $(LP)^2$ and to Dayne Freitag for offering the two annotated WebKB data sets and seminar announcements. Many thanks to the editor and the anonymous reviewers for their useful comments that helped improving the article.

Appendix A: Summary of the Combination Methods for Information Extraction

Combination method	Short description	Described in section
MVotM	Majority voting. Missing values are ignored	3.2
MVotF	Majority voting. Missing values are encoded as special "false" values, indicating rejection of prediction	3.2
PVotM	Voting with probabilities. Missing values are ignored	3.3
PVotF	Voting with probabilities. A threshold is set (typically 0.5), for accepting/rejecting predictions	3.3
Stacking Nominal	Stacking using simple nominal values	4.2, 4.3
Stacking Probs	Stacking using probability values	4.4

Appendix B: Results of Base-level and Meta-level for the Five Domains of Interest

B.1 Results of Base-level

%	CS courses			Research projects			Laptop products		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
BWI	74.55	39.10	51.30	60.05	61.47	60.75	74.99	53.23	62.26
HMM	60.50	58.29	59.39	56.24	68.18	61.64	62.29	65.42	63.81
$(LP)^2$	71.39	60.90	65.73	63.31	54.92	58.82	63.24	59.41	61.26

%	Job announcements			Seminar announcements		
	Precision	Recall	F1	Precision	Recall	F1
BWI	89.42	72.39	80.01	93.26	74.92	83.09
HMM	72.42	79.31	75.71	78.34	80.09	79.20
$(LP)^2$	87.70	79.18	83.22	91.39	81.63	86.23

B.2 Results of Majority Voting

%	Precision			Recall			F1		
	Best Base	MVotM	MVotF	Best Base	MVotM	MVotF	Best Base	MVotM	MVotF
Courses	71.39	58.68	82.05	60.90	74.35	47.65	65.73	65.59	60.29
Projects	56.24	49.17	68.88	68.18	79.32	65.96	61.64	60.71	67.39
Laptops	62.29	52.89	80.41	65.42	76.00	59.05	63.81	62.37	67.60
Jobs	87.70	71.29	93.06	79.18	90.88	76.31	83.22	79.90	83.85
Seminars	91.39	86.93	97.55	81.63	86.82	78.72	86.23	86.87	87.13

B.3 Results of Voting Using Probabilities

%	Precision			Recall			F1		
	Best Base	PVotM	PVotF	Best Base	PVotM	PVotF	Best Base	PVotM	PVotF
Courses	71.39	58.78	70.16	60.90	74.35	71.12	65.73	65.65	70.64
Projects	56.24	49.20	63.31	68.18	79.37	68.38	61.64	60.75	65.75
Laptops	62.29	53.21	72.86	65.42	76.47	69.30	63.81	62.76	71.03
Jobs	87.70	71.37	80.08	79.18	90.98	86.45	83.22	79.99	83.15
Seminars	91.39	86.99	90.69	81.63	86.82	85.50	86.23	86.90	88.02

B.4 Results of Stacking Using a Single Classifier

%	Precision			Recall			F1		
	Best Base	Stacking Nominal	Stacking Probs	Best Base	Stacking Nominal	Stacking Probs	Best Base	Stacking Nominal	Stacking Probs
Courses	71.39	81.32	79.03	60.90	52.66	66.01	65.73	63.92	71.93
Projects	56.24	68.84	78.21	68.18	63.59	64.45	61.64	66.05	70.66
Laptops	62.29	79.52	84.49	65.42	60.10	62.04	63.81	68.46	71.55
Jobs	87.70	89.89	90.27	79.18	81.82	82.00	83.22	85.67	85.94
Seminars	91.39	92.56	94.69	81.63	84.74	85.80	86.23	88.48	90.03

B.5 Results of All Classifiers in Stacking

Comparison of classifiers is based on $F1$. For readability, *LB*=LogitBoost, *NB*=NaiveBayes

%	Stacking with nominal values						Stacking with probabilities					
	IB1	*j48*	*LB*	*MLR*	*NB*	*SMO*	*IB1*	*j48*	*LB*	*MLR*	*NB*	*SMO*
Courses	66.03	50.79	63.92	54.62	60.11	56.76	70.23	70.24	71.93	70.38	65.16	70.41
Projects	61.18	55.36	66.05	60.89	60.99	55.98	66.77	71.41	70.66	62.17	65.63	62.65
Laptops	64.43	62.56	68.46	66.23	67.29	66.65	69.49	70.66	71.55	70.00	61.36	71.02
Jobs	80.58	83.93	85.67	84.61	81.77	84.52	83.88	85.22	85.94	84.95	77.23	84.75
Seminars	87.34	87.63	88.48	88.22	87.06	88.09	88.45	89.66	90.03	88.78	88.49	88.82

References

Ali, K.M., Pazzani, M.J., Error reduction through learning multiple descriptions. *Machine Learning*, 24, 173-202, 1996.

Breiman, L., Bagging Predictors, *Machine Learning*, 24(2), 123-140, 1996.

Breiman, L., Stacked Regressions, *Machine Learning*, 24, 41-48, 1996a.

Califf, M.E., Mooney, R.J., Bottom-up Relational Learning of Pattern Matching Rules for Information Extraction, *Journal of Machine Learning Research (JMLR)*, 4, 177-210, 2003.

Chan, P., An Extensive Meta-Learning Approach for Scalable and Accurate Inductive Learning, *PhD Thesis*, Columbia University, 1996.

Chang, C.H., Lui, S.C., IEPAD : Information Extraction based on Pattern Discovery, *In Proceedings of the Tenth International WWW conference*, 509-516, New York, USA, 2001.

Chawathe, S., Molina, H-C., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J., Widom, J., The TSIMMIS Project: Integration of Heterogeneous Information Sources, *In Proceedings of the Tenth Meeting of Information Processing Society of Japan (IPSJ)*, 7-18, 1994.

Ciravegna, F., Adaptive Information Extraction from Text by Rule Induction and Generalization, *In Proceedings of the Seventeenth IJCAI Conference.* Seattle, 1251-1256, 2001.

Ciravegna, F., Lavelli, A., LearningPinochio: Adaptive Information Extraction for Real World Applications, *Natural Language Engineering*, 1(1), 1-21, 2003.

Cohen, W., Hurst, M., Jensen, L.S., A Flexible Learning System for Wrapping Tables and Lists in HTML Documents, *In Proceedings of the Eleventh International WWW conference*, Hawaii, USA, 2002.

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, *In Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, 509-516, 1998.

Crescenzi, V., Mecca, G., Merialdo, P., RoadRunner: Towards automatic data extraction from large Web sites, *In Proceedings of the Twenty-seventh International Conference on Very Large Data Bases (VLDB)*, 109-118, Rome, Italy, 2001.

Davulcu, H., Mukherjee, S., Ramakrishman, I.V., Extraction Techniques for Mining Services from Web Sources, *IEEE International Conference on Data Mining (ICDM)*, 601-604, 2002.

Defense Advanced Research Projects Agency (DARPA), *Proceedings of the Sixth Message Understanding Conferences (MUC-6)*, Morgan Kaufmann, 1995.

Defense Advanced Research Projects Agency (DARPA), *Proceedings of the Seventh Message Understanding Conferences (MUC-7)*, Morgan Kaufmann, 1996.

Dietterich, T.G., Machine Learning research: Four current directions. *AI Magazine*, 18(4), 97-136, 1997.

Dietterich, T.G., Approximate Statistical Tests for Comparing Supervised Machine Learning Algorithms, *Neural Computing*, 10(7), 1895-1924, 1998.

Domingos, P., Unifying instance-based and rule-based induction, *Machine Learning*, 24(2), 141-168, 1996.

Džeroski, S., Ženko, B., Is Combining Classifiers Better than Selecting the Best One? *Machine Learning*, 54(3), 255-273, 2004.

Florian, R., Cucerzan, S., Schafer, C., Yarowsky, D., Combining Classifiers for Word Sense Disambiguation, *Natural Language Engineering*, 1(1), 1-14, 2002.

Freitag, D., Machine Learning for Information Extraction in Informal Domains, *Machine Learning*, 39, 169-202, 2000.

Freitag, D., Kushmerick, N., Boosted Wrapper Induction, *In Proceedings of the Sixteenth National conference on Artificial Intelligence (AAAI-99)*, 59-66, 1999.

Freitag, D., McCallum, A.K., Information extraction with HMMs and shrinkage, *AAAI-99 workshop on machine learning for information extraction*, 1999.

Freitag, D., McCallum, A.K., Information extraction with HMM structures learned by stochastic optimization, *In Proceedings of the Seventeenth National conference on Artificial Intelligence (AAAI-00)*, 584-589, 2000.

Freund, Y., Schapire, R., Experiments with a new boosting algorithm, *In Proceedings of the Thirteenth International Conference on Machine Learning (ICML)*, 148-156, Bari, Italy, 1996.

Friedman, J., Hastie, T., Tibshirani, R., Additive Logistic Regression: a Statistical View Of Boosting, *Technical Report*, Stanford University, 1999.

Halteren, H., Zavrel J., Daelemans, W., Improving Accuracy in Word Class Tagging through Combination of Machine Learning Systems, *Computational Linguistics*, 27(2), 199-230, 2001.

John, G.H., Langley, P., Estimating Continuous Distributions in Bayesian Classifiers, *In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (UAI)*, 338-345, Morgan Kaufmann, 1995.

Kauchak, D., Smarr, J., Elkan, C., Sources of Success for Boosted Wrapper Induction, *Journal of Machine Learning Research (JMLR)*, 5, 499-527, 2004.

Kuncheva, L.I., Whitaker, C.J., Measures of Diversity in Classifier Ensembles and their Relationship with the Ensemble Accuracy, *Machine Learning*, 51, 181-207, 2003.

Kushmerick, N., Wrapper Induction for Information Extraction, *PhD Thesis*, University of Washington, 1997.

Lafferty, J., McCallum, A.K., Pereira, F., Conditional random fields: Probabilistic models for segmenting and labeling sequence data, *In Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*, 282-289, Williamstown, MA, USA, 2001.

McCallum, A.K., Freitag, D., Pereira, F., Maximum entropy markov models for information extraction and segmentation, *In Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, 591-598, Stanford University, CA, USA, 2000.

Michalski, R., Tecuci, G., Machine learning: A multistrategy approach, *Morgan Kaufmann*, 1994.

Muslea, I., Minton, S., Knoblock, C., Hierarchical Wrapper Induction for Semistructured Information Sources, *Journal Of Autonomous Agents and Multi-Agent Systems*, 4, 93-114, 2001.

Platt, J., Fast Training of Support Vector Machines using Sequential Minimal Optimization, *Advances in Kernel Methods - Support Vector Learning*, 185-208, MIT Press, 1999.

Quinlan, R.J., C4.5: Programs for Machine Learning, *Morgan Kaufmann*, 1993.

Rabiner, L., A tutorial on hidden Markov models and selected applications in speech recognition, *In Proceedings of the IEEE* 77-2, 257-286, 1989.

RISE, A repository of online information sources used in information extraction tasks, *URL: http://www.isi.edu/info-agents/RISE*, 1998.

Sebastiani, F., Machine Learning for Automated Text Categorization, *ACM Computing Surveys (CSUR)*, 34 (1), 1-47, 2002.

Seewald, A., Towards understanding stacking, *PhD Thesis*, Department of Informatics, Technical University of Wien, Austria, 2003.

Seymore, K., McCallum, A.K., Rosenfeld, R., Learning hidden Markov model structure for Information Extraction, *Journal of Intelligent Information Systems (JIIS)*, 8(1), 5-28, 1999.

Sigletos, G., Voting and stacking for information extraction: Extended results, *Technical Report DEMO 2005/3, NCSR Demokritos*, 2005.

Sonderland, S., Learning Information Extraction Rules for Semi-structured and Free Text, *Machine Learning*, 34-(1/3), 233-272, 1999.

Ting, K., Witten, M., Issues in stacked generalization, *Journal of Artificial Intelligence Research (JAIR)*, 10, 271-289, 1999.

Thompson, C.A., Califf, M.E., Mooney, R.J., Active Learning for Natural Language Parsing and Information Extraction, *In Proceedings of the Sixteenth International Machine Learning Conference (ICML)*, 406-414, Bled, Slovenia, 1999.

Valarakos, A., Paliouras, G., Karkaletsis, V., Vouros G., Enhancing Ontological Knowledge through Ontology Population and Enrichment, *In Proceedings of the Fourteenth International Conference on Knowledge Engineering and Knowledge Management (EKAW)*, LNAI-3257, 144-156, Springer, 2004.

Witten, I., Frank, E., Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, *Morgan Kaufmann*, 2000.

Wolpert, D., Stacked Generalization, *Neural Networks*, 5(2), 241-260, 1992.