Web mining for information extraction from the Web using voting and stacked generalization

Georgios Sigletos[1]

Department of Informatics and Telecommunications, University of Athens,
TYPA Buildings, Panepistimiopolis, Athens, Greece
sigletos@di.uoa.gr

Abstract. This dissertation investigates the effectiveness of voting and stacking techniques in the context of information extraction (IE) from the Web. The motivation derives from the opportunity of obtaining higher performance at meta-level, by exploiting the disagreement in the predictions of the IE systems that are employed at base-level. Existing combination techniques primarily focus on classification. However, IE is not naturally a classification problem. A new methodology is proposed for combining IE systems through voting and stacking. The proposed methodology facilitates the combination of a wide range of systems, since only their output is combined, without taking into account how each system is implemented or models the extraction task. IE is transformed to a common classification problem at meta-level, allowing the applicability of voting and stacking. Voting proved to be effective in most domains in the experiments. Stacking, 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 analyzing the results obtained by voting and stacking, aiming to investigate the sources of their success in IE tasks.

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* [1] 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.

This dissertation investigates 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

[1] Supervisor: Prof. Michael Hatzopoulos

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 [2, 3, 4, 5, 6]. 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 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.

This dissertation 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 base-level IE systems.

A new stacking framework is then introduced that combines a wide range of 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. The remaining of this article is structured as follows: Section 2 presents voting for information extraction. Section 3 presents stacking for information extraction. Section 4 evaluates voting and stacking. Section 5 explains the results, based on the diversity of the base-level systems. Finally, Section 6 presents the conclusions.

2 Voting for Information Extraction

Section 2.1 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. Majority voting and voting with probabilities for IE are presented in Section 2.2

2.1 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. Define $E^1 ... E^N$ the corresponding set of IE systems that exploit the acquired knowledge, to identify relevant instances in new documents. 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 $T^1 ... T^N$, such as the one showed in Table 1.

Table 1. 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.

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

Examining Table 1, 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.

2.2 Majority Voting and Voting Using Probabilities

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 or the highest probability. In the case of a tie, a random selection is typically performed among even fields. Table 2 summarizes all voting settings for information extraction that are defined in this thesis, along with a short description for each setting.

Table 2. Summary of all voting settings, along with a short description.

Combination method	Short description
MVotM	Majority voting. Missing values are ignored
MVotF	Majority voting. Missing values are encoded as special "false" values, indicating rejection of prediction
PVotM	Voting with probabilities. Missing values are ignored
PVotF	Voting with probabilities. A threshold is set (typically 0.5), for accepting/rejecting predictions

3 Stacking for Information Extraction

Section 3.1 presents simple stacking with nominal values, while Section 3.2 presents stacking with probabilities.

3.1 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 jth 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 1 shows the new methodology at runtime.

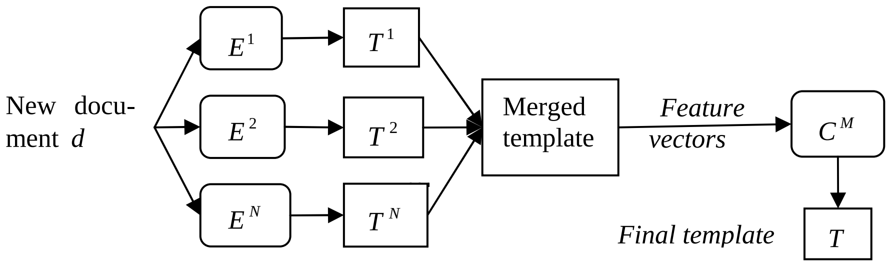

Fig. 1. The stacking framework at runtime

3.1 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 kth 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.

Table 3 shows an illustrative example of the new feature vectors at the meta-level, using probabilistic estimates of correctness.

Table 3. Summary of all voting settings, along with a short description.

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

4 Evaluation

Extensive experiments were performed in five real-world domains, using well-known algorithms at both base-level and meta-level and using five collections of text documents from five different domains. 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. At meta-level, we used six classifiers from the well-known WEKA platform [7]. *Recall*, *precision* and $F1$ (particularly), were used as evaluation metrics. Section 4.1 presents the experimental results by all combination methods (voting and stacking), while Section 4.2 discusses the obtained results.

4.1 Experimental Results

Table 4 shows the best $F1$ scores obtained by all voting settings, stacking with nominal values, stacking with probabilities, and by the best base-level systems. Table 5 compares all combination methods and the best base-level system, based on statistically significant wins against losses, using $F1$, in the five examined domains. Section 4.2 discusses the experimental results.

Table 4. Comparing all combination methods using F1.

	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 5. Comparing all combination methods using statistically significant *wins* against losses, based on F1.

	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	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	■

4.2 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 all combination methods.

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.

5 Explaining the Results

Figures 2 to 4 compare all combination methods, based on the number of correctly classified instances, when all three base-level IE systems, agree, partially agree or disagree on their output, respectively.

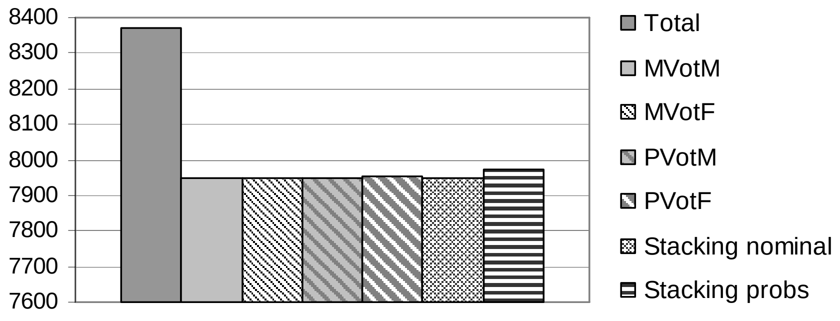

Fig. 2. Comparing all combination methods, when all three base-level systems agree on the same field. Sum of correctly classified meta-level instances over all five domains.

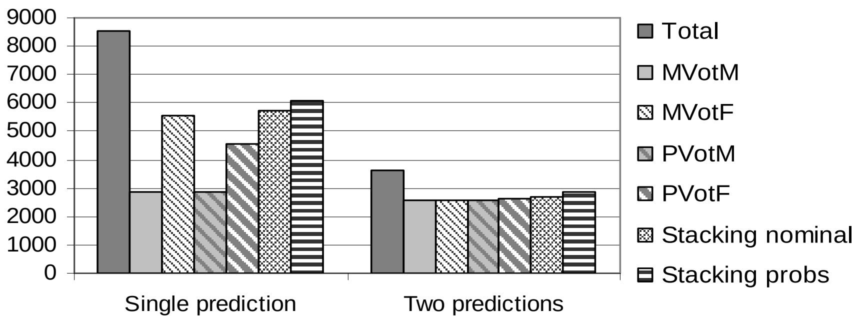

Fig. 3. Comparing all combination methods, when either a single or exactly two base-level systems agree on the same field (set of columns on the left and right respectively). Sum of correctly classified meta-level instances over all five domains.

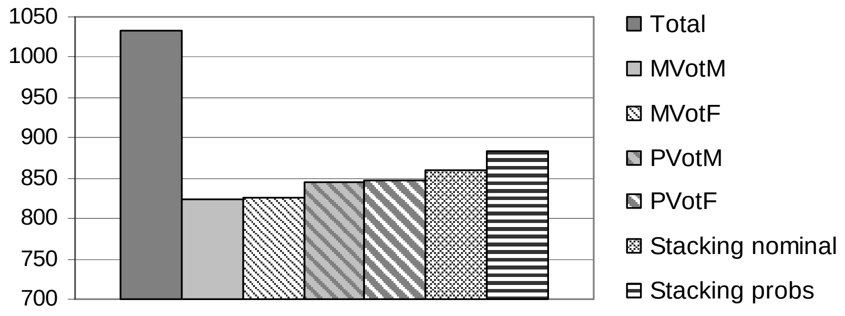

Fig. 4. Comparing all combination methods when the base-level systems disagree. Sum of correctly classified meta-level instances over all five domains.

Figures 2 to 4 demonstrate the superiority of stacking with probabilities in each case, even when all base-level IE systems agree in their output.

6 Conclusions

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 [4]. 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 dissertation 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.

References

1. Wolpert, D. Stacked generalization, Neural Networks, 5(2), 241-260, 1992.
2. Freitag, D., Kushmerick, N., Boosted Wrapper Induction, In Proceedings of the 17th National conference on Artificial Intelligence. (AAAI-1999), 59-66, 1998.
3. Sonderland, S., Learning Information Extraction Rules for Semi-structured and Free Text, Machine Learning, 34-(1/3), 233-272, 1999
4. Freitag, D., Machine Learning for Information Extraction in Informal Domains, Machine Learning, 39, 169-202, 2000.
5. Ciravegna, F., Adaptive Information Extraction from Text by Rule Induction and Generalization. In Proceedings of the 17th IJCAI Conference. Seattle, 2001.
6. Califf M.E., Mooney R.J., Bottom-up Relational Learning of Pattern Matching Rules for Information Extraction, Journal of Machine Learning Research, 4, 177-210, 2003.
7. Witten, I., Frank, E., Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann Publishers, 2000.