Mining Web sites using wrapper induction, named entities and post-processing

Georgios Sigletos[1,2], Georgios Paliouras[1],
Constantine D. Spyropoulos[1], Michalis Hatzopoulos[2]

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

Abstract. This paper presents a novel method for extracting information from collections of Web pages across different sites. Our method uses a standard wrapper induction algorithm and exploits named entity information. We introduce the idea of post-processing the extraction results for resolving ambiguous facts and improve the overall extraction performance. Post-processing involves the exploitation of two additional sources of information: fact transition probabilities, based on a trained bigram model, and confidence probabilities, estimated for each fact by the wrapper induction system. A multiplicative model that is based on the product of those two probabilities is also considered for post-processing. Experiments were conducted on pages describing laptop products, collected from many different sites and in four different languages. The results highlight the effectiveness of our approach.

1 Introduction

Wrapper induction (WI) [7] aims to generate extraction rules, called *wrappers*, by mining highly structured collections of Web pages that are labeled with domain-specific information. At run-time, wrappers extract information from unseen collections and fill the slots of a predefined template. These collections are typically built by querying an appropriate search form in a Web site and collecting the response pages, which commonly share the same content format.

A central challenge to the WI community is *Information Extraction* (IE) from pages across multiple sites, including unseen sites, by a single trained system. Pages collected from different sites usually exhibit multiple hypertext markup structures, including tables, nested tables, lists, etc. Current WI research relies on learning separate wrappers for different structures. Training an effective site-independent IE system is an attractive solution in terms of *scalability*, since any domain-specific page could be processed, without relying heavily on the hypertext structure.

In this paper we present a novel approach to IE from Web pages across different sites. The proposed method relies on domain specific *named entities*, identified within

Web pages. Those entities are embedded within the Web pages as XML tags and can serve as a page-independent common markup structure among pages from different sites. A standard WI system can be applied and exploit the additional textual information. Thus, the new system relies more on page-independent named-entity markup tags for inducing delimiter-based rules for IE and less on the hypertext markup tags, which vary among pages from multiple sites.

We experimented with STALKER [10], which performs extraction from a wide range of Web pages, by employing a special formalism that allows the specification of the output multi-place schema for the extraction task. However, information extraction from pages across different sites is a very hard problem, due to the multiple markup structures that cannot be described by a single formalism. In this paper we suggest the use of STALKER for *single-slot* extraction, i.e. extraction of isolated *facts* (i.e. *extraction fields*), from pages across different sites.

A further contribution of this paper is a method for post-processing the system's extraction results in order to disambiguate facts. When applying a set of single-slot extraction rules to a Web page, one cannot exclude the possibility of identical or overlapping textual matches within the page, among different rules. For instance, rules for extracting instances of the facts *cd-rom* and *dvd-rom* in pages describing laptop products may overlap or exactly match in certain text fragments, resulting in ambiguous facts. Among these facts, the correct choice must be made.

To deal with the issue of ambiguous facts, two sources of information are explored: *transitions* between facts, incorporated in a bigram model, and *prediction confidence* values, generated by the WI system. Deciding upon the correct fact can be based on information from either the trained bigram model and/or the confidence assigned to each predicted fact. A multiplicative model that combines these two sources of information is also presented and compared to each of the two components.

The rest of this paper is structured as follows: In Section 2 we outline the architecture of our approach. Section 2.1 briefly describes the named entity recognition task. Section 2.2 reviews STALKER and in Section 2.3 we discuss how STALKER can be used under the proposed approach to perform IE from pages across different sites. In Section 3 we discuss the issue of post-processing the output of the STALKER system in order to resolve ambiguous facts. Section 4 presents experimental results on datasets that will soon be publicly released. Related work is presented in Section 5. Finally we conclude in Section 6, discussing potential improvements of our approach.

2 Information Extraction from multiple Web sites

Our methodology for IE from multiple Web sites is graphically depicted in Figure 1. Three component modules process each Web page. First, a *named-entity recognizer* (NER) that identifies domain-specific named entities across pages from multiple sites. A trained WI system is then applied to perform extraction. Finally, the extraction results are post-processed to improve the extraction performance.

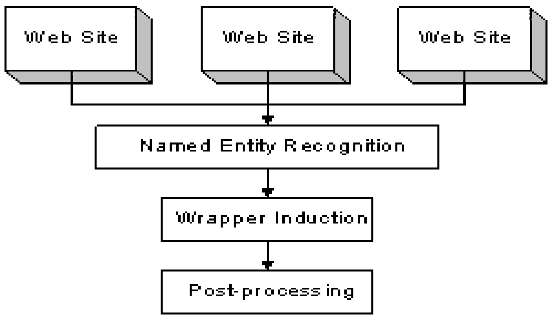


Fig. 1. Generic architecture for information extraction from multiple Web sites

2.1 Named entity recognition

Named entity recognition (NER) is an important subtask in most language engineering applications and has been included as such in all MUC competitions, e.g. [8]. NER is best known as the first step in the IE task, and involves the identification of a set of basic *entity names*, *numerical expressions*, *temporal expressions*, etc. The overall IE task aims to extract *facts* in the form of multi-place relations and NER provides the entities that fill the argument slots. NER has not received much attention in Web IE tasks.

We use NER in order to identify basic named entities relevant to our task and thus reduce the complexity of fact extraction. The identified entities are identified within Web pages as XML tags and serve as a valuable source of information for the WI system that follows and extracts the facts. Some of the entity names (*ne*), numerical (*numex*) and temporal (*timex*) expressions, used in the laptop domain are shown in Table 1, along with the corresponding examples of XML tags.

Table 1. Subset of named entities for the laptop domain

Entity	*Entity Type*	*Examples of XML tags*
Ne	Model, Processor	<*ne type=Model*>Presario</*ne*> <*ne type=Processor*>Intel Pentium </*ne*>
Numex	Capacity, Speed	<*numex type=Speed*>300 MHz </*numex*> <*numex type=Capacity*>20 GB </*numex*>
Timex	Duration	<*timex type=Duration*>1 year </timex>

2.2 The STALKER wrapper induction system

STALKER [10] is a sequential covering rule learning system that performs *single-slot* extraction from highly-structured Web pages. *Multi-slot* extraction –i.e. linking of the isolated facts- is feasible through an *Embedded-Catalog (EC) Tree* formalism, which may describe the common structure of a range of Web pages. The EC tree is constructed manually, usually for each site, and its leaves represent the individual facts. STALKER is capable of extracting information from pages with tabular organization of their content, as well as pages with hierarchically organized content.

Each extraction rule in STALKER consists of two ordered lists of *linear landmark automata* (LA's – also called *disjuncts*), which are a subclass of nondeterministic finite state automata. The first list constitutes the *start* rule, while the second list constitutes the *end* rule. Each rule consists of a group of consecutive tokens or *wildcards* that enable the location of a fact instance x, within a page p. A wildcard represents a class of tokens, e.g. *Number* or *HtmlTag*. A rule for extracting a fact instance *x* within page *p*, consists of a *start* rule that consumes the prefix of *p* with respect to *x*, and an *end* rule, which consumes the suffix of *p* with respect to *x*. Thus the training data for inducing a start rule consists of sequences of tokens (and wildcards) that represent the *prefixes* (or *suffixes* for end rule) that must be consumed by the induced rule.

Using the EC tree as a guide, the extraction in a given page is performed by applying –for each fact- the LA's that constitute the start rule in the order in which they appear in the list. As soon as a LA is found that matches within the page, the matching process terminates. The process is symmetric for the end rule. Figure 2 More details on the algorithm can be found in [10].

2.3 Adapting STALKER to multi-site information extraction

The EC tree formalism used in STALKER is generally not applicable for describing pages with variable markup structure. Different EC trees need to be manually built for different markup structures and thus different extraction rules to be induced. In this paper, we are seeking for a single domain-specific trainable system, without having to deal with each page structure separately. The paper focuses on the widely-used approach of single-slot extraction. Our motivation is that if isolated facts could be accurately identified, then it is possible to link those facts separately on a second step. We therefore specify our task as follows:

For each fact, try to induce a *list iteration* rule as depicted in Figure 2.

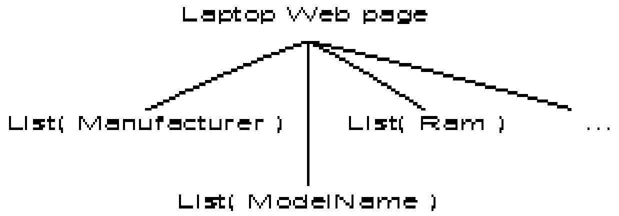


Fig. 2. Simplification of the EC tree. A list iteration rule is learned for each fact and applies to the whole content of a page, at run-time

The EC tree depicted in Figure 2 has the following interpretation: a Web page that describes laptop products consists of a list of instances of the fact *Manufacturer* (e.g.

"Compaq"), a list of instances of the fact *ModelName* (e.g. "Presario"), a list of *Ram* instances (e.g. "256MB"), etc. The system, during runtime, exhaustively applies each rule to the content of the whole page. This simplified EC tree is independent of any particular page structure. The proposed approach relies on the page-independent named entities to lead to efficient extraction rules.

Since each extraction rule applies exhaustively within the complete Web page, rather than being constrained by the EC tree, we expect an extraction bias towards recall, i.e., overgeneration of extracts for each fact. The penalty is a potential loss in precision, since each rule applies to text regions that do not contain relevant information and may return erroneous instances. Therefore we seek a post-processing mechanism capable of discarding the erroneous instances and thus improving the overall precision.

3 Post-processing the extraction results

In single-slot IE systems, each rule is applied independently of the others. This may naturally cause identical or overlapping matches among different rules resulting in multiple ambiguous facts for those matches. We would like to resolve such ambiguities and choose the correct fact. Choosing the correct fact and removing all the others shall improve the extraction precision.

3.1 Problem specification

In this paper we adopt a post-processing approach in order to resolve ambiguities in the extraction results of the IE system. More formally, the task can be described as follows:

1. Let D be the sequence of a document's tokens and T_j (s_j, e_j) a fragment of that sequence, where s_j and e_j are the start and end token bounds respectively.
2. Let $I = \{i_j \mid i_j\text{: } T_j \rightarrow \text{fact}_j\}$ be the set of instances extracted by all the rules, where $fact_j$ is the predicted fact associated with instance T_j.
3. Let DT be the list of all *distinct* text fragments T_j, appearing in the extracted instances in I. Note that $T_1(s_1, e_1)$ and $T_2(s_2, e_2)$ are different, if either $s_1 \neq s_2$ or $e_1 \neq e_2$. The elements of DT are sorted in ascending order of s_j.
4. If for a distinct fragment T_i in DT, there exist at least two instances i_k and i_l so that i_k: $T_i \rightarrow \text{fact}_k$ and i_l : $T_i \rightarrow \text{fact}_l$, $k \neq l$, then $fact_k$ and $fact_l$ are ambiguous facts for T_i.
5. The goal is to associate a single fact to each element of the list DT.

To illustrate the problem, if for the fragment $T_j(24, 25)$="16 x" in a page describing laptops, there are two extracted instances i_k and i_l, where fact_k = *dvdSpeed* and fact_l = *cdromSpeed*, then there are two ambiguous facts for T_i. One of them must be chosen and associated with T_j.

3.2 Formulate the task as a hill-climbing search

Resolving ambiguous facts can be viewed as a *hill-climbing* search in the space of all possible sequences of facts that can be associated with the sequence DT of distinct text fragments.

This hill-climbing search can be formulated as follows:

1. Start from a hypothetical empty node, and transition at each step *j* to the next distinct text fragment T_j of the sorted sequence DT.
2. At each step apply a set of operations *Choose (fact*$_k$*)*. Each operation associates T_j with the $fact_k$ predicted by an instance $i_k = \{T_j \rightarrow fact_k\}$. A *weight* is assigned to each operation, based on some predefined metric. The operation with the highest weight is selected at each step.
3. The goal of the search is to associate a single fact to the last distinct fragment of the sorted list DT, and thus return the final unambiguous sequence of facts for DT.

To illustrate the procedure, consider the fictitious token table in Table 2(a), which is part of a page describing laptop products.

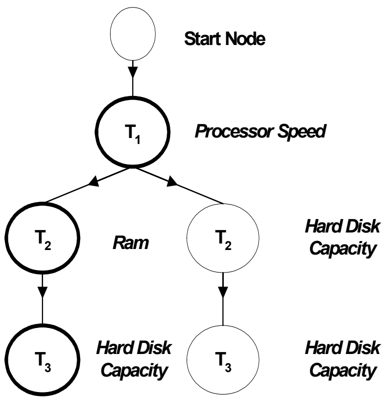
Table 2. (a) Part of a token table of a page dscribing laptops. (b) Instances extracted by STALKER. (c) The tree of all possible fact paths (d) The extracted instances after the disambiguation process

…	*33*	*34*	*35*	*36*	*37*	*38*	*39*	*40*	…
…	1,6	GHz	/	1	GB	/	80	GB	…

(a)

T_j (s_j, e_j)	*fact*$_k$
T_1 (33, 34)	Processor Speed
T_2 (36, 37)	Ram
T_2 (36, 37)	Hard Disk capacity
T_3 (39, 40)	Hard Disk Capacity

(b)



(c)

T_j (s_j, e_j)	*fact*$_k$
T_1 (33, 34)	Processor Speed
T_2 (36, 37)	Ram
T_3 (39, 40)	Hard Disk Capacity

(d)

Table 2(b) lists the instances extracted by STALKER for the token table part of Table 2(a). The DT list consists of the three distinct fragments T_1, T_2, T_3. Table 2(c) shows the two possible fact sequences that can be associated with DT. After the *processor speed* fact prediction for T_1, two operations apply for predicting a fact for T_2: The *choose (Ram)* and *choose (hard disk capacity)* operations, each associated with a *weight*, according to a predefined metric. We assume that the former operation returns a higher weight value and therefore *Ram* is the chosen fact for T_2. The bold circles in the tree show the chosen sequence of facts {*Processor speed*, *Ram*, *Hard disk capacity*} that is attached to the sequence T_1 T_2 T_3. Table 2(d) illustrates the final extracted instances, after the disambiguation process.

In this paper we explore three metrics for assigning weights to the choice operations:

1. *Confidence* values, estimated for each fact by the WI algorithm.
2. *Fact-transition* probabilities, learned by a bigram model.
3. The *product* of the above probabilities, based on a simple multiplicative model.

Selecting the correct instance, and thus the correct fact, at each step and discarding the others, results in improving the overall precision. However, an incorrect choice harms both the recall and the precision of a certain fact. The overall goal of the disambiguation process is to improve the overall precision while keeping recall unaffected.

3.3 Estimating fact confidence

The original STALKER algorithm does not assign confidence values to the extracted instances. In this paper we estimate confidence values by calculating a value for each extraction rule, i.e. for each fact. That value is calculated as the average *precision* obtained by a three-fold cross-validation methodology on the training set. According to this methodology, the training data is split into three equally-sized subsets and the learning algorithm is run three times. Each time two of the three pieces are used for training and the third is kept as unseen data for the evaluation of the induced extraction rules. Each of the three pieces acts as the evaluation set in one of the three runs and the final result is the average over the three runs.

At runtime, each instance extracted by a single-slot rule will be assigned the *precision* value of that rule. For example, if the text fragment "300 Mhz" was matched by the *processor speed* rule, then this fragment will be assigned the confidence associated with *processor speed*. The key insight into using confidence values is that among ambiguous facts, we can choose the one with the highest estimated confidence.

3.4 Learning fact transition probabilities

In many extraction domains, some facts appear in an almost fixed order within each page. For instance, a page describing laptop products may contain instances of the *processor speed* fact, appearing almost immediately after instances of the *processor name* fact. Training a simple bigram model is a natural way of modeling such dependencies and can be easily implemented by calculating ratios of counts (maximum likelihood estimation) in the labeled data as follows:

$$P(i \to j) = \frac{c(i \to j)}{\sum_{j \in K} c(i \to j)}, \tag{1}$$

where the nominator counts the transitions from fact *i* to fact *j*, according to the labeled training instances. The denominator counts the total number of transitions from fact *i* to all facts (including self-transitions). We also calculate a *starting probability* for each fact, i.e. the probability that an instance of a particular fact is the first one appearing in the labeled training pages.

The motivation for using fact transitions is that between ambiguous facts we could choose the one with the highest transition probability given the preceding fact prediction. To illustrate that, consider that the text fragment "16 x" has been identified as both *cdromSpeed* and *dvdSpeed* within a page describing laptops. Assume also that the preceding fact prediction of the system is *ram*. If the transition from *ram* to *dvdSpeed* has a higher probability, according to the learned bigram, than from *ram* to *cdromSpeed*, then we can choose the *dvdSpeed* fact. If ambiguity occurs at the first extracted instance, where there is no preceding fact prediction available, then we can choose the fact with the highest starting probability.

3.5 Employing a multiplicative model

A simple way to combine the two sources of information described above is through a multiplicative model, assigning a confidence value to each extracted instance $i_k : T_i \to fact_k$, based on the *product* of the confidence value estimated for $fact_k$ and the transition probability from the preceding instance to $fact_k$. Using the example of Table 2 with the two ambiguous facts *ram* and *hard disk capacity* for the text fragment T_2, Table 4 depicts the probabilities assigned to each fact by the two methods described in sections 3.3 and 3.4 and the multiplicative model.

Table 3. Probabilities assigned to each of the two ambiguous facts of the text fragment T_2 of Table 2

T_2 (36, 37) = "1 GB"	*WI-Confidence*	*Bigram*	*Multiplicative*
Ram	0,7	0,3	0,21
Hard disk capacity	0,4	0,5	0,20

Using the WI confidence values, the *ram* is selected. However, using bigram probabilities, the *hard disk capacity* is selected. We also experimented with a model that averages the two probabilities, rather than multiplying them. However the experiments led to worse results.

4 Experiments

4.1 Dataset description

Experiments were conducted on four language corpora (Greek, French, English, Italian) describing laptop products. The corpora were collected in the context of CROSSMARC[1].

Approximately 100 pages from each language were hand-tagged using a Web page annotation tool [14]. The corpus for each language was divided into two equally sized data sets for *training* and *testing*. Part of the test corpus was collected from sites not appearing in the training data. The named entities were embedded as XML tags within the pages of the training and test data, as illustrated in Table 1. A separate NER module was developed for each of the four languages of the project.

A total of 19 facts (listed in Table 4) were hand-tagged for the laptop product domain. The pages were collected from multiple vendor sites and demonstrate a rich variety of structure, including tables, lists etc.

Table 4. Hand-tagged facts for the laptop product domain

Fact	**Details**
Manufacturer	The manufacturer of the laptop, e.g. HP
Model	The model name, e.g. ARMADA 110
ProcessorName	The processor name, e.g. Intel Pentium III
ProcessorSpeed	Processor's speed , e.g. 600 MhZ
Ram	RAM, e.g. 512 MB
HdCapacity	The hard disk capacity, e.g. 40 GB
Price	The price of the laptop
Warranty	The laptop's warranty, e.g. 2 years
ScreenSize	The screen size, e.g. 14''
ScreenType	The type of the screen, e.g. TFT
PreinstalledOS	The operating system that is preinstalled, e.g. Windows XP
PreinstalledSoftware	Software that is preinstalled, e.g. Norton AntiVirus
ModemSpeed	The speed of the modem, e.g. 56K
Weight	The weight of the laptop
CdromSpeed	The speed of the cdrom player, e.g. 48x
DvdSpeed	The speed of the dvd player (if any)
ScreenResolution	The resolution of the screen, e.g. 1024x768
BatteryType	The type of the laptop's battery, e.g. Li-Lon
BatteryLife	The duration of the battery, e.g. 3,5 h

[1] http://www.iit.demokritos.gr/skel/crossmarc. Datasets will soon be available on this site.

4.2 Results

Our goal was to evaluate the effect of named entity information to the extraction performance of STALKER and compare the three different methods for resolving ambiguous facts.

We, therefore, conducted two groups of experiments. In the first group we evaluated STALKER on the testing datasets for each language, with the named entities embedded as XML tags within the pages. Table 5 presents the results. The evaluation metrics are *micro-average recall* and *micro-average precision* [12] over all 19 facts. The last row of Table 5 averages the results over all languages.

Table 5. Evaluation results for STALKER in four languages

Language	*Micro Precision (%)*	*Micro Recall (%)*
Greek	60,5	86,8
French	64,1	93,7
English	52,2	85,1
Italian	72,8	91,9
Average	**62,4**	**89,4**

The exhaustive application of each extraction rule to the whole content of a page resulted, as expected, in a high recall, accompanied by a lower precision. However, named-entity information led a pure WI system like STALKER to achieve a bareable level of extraction performance across pages with variable structure. We also trained STALKER on the same data without embedding named entities within the pages. The result was an unacceptably high training time, accompanied by rules with many disjuncts that mostly overfit the training data. Evaluation results on the testing corpora provided recall and precision figures below 30%.

In the second group of experiments, we evaluated the post-processing methodology for resolving ambiguous facts that was described in Section 3. Results are illustrated in Table 6.

Table 6. Evaluation results after resolving ambiguities

Language	*Micro Precision (%)*			*Micro Recall (%)*		
	WI-Conf.	*Bigram*	*Mult.*	*WI-Conf.*	*Bigram*	*Mult.*
Greek	69,3	73,5	73,8	76,9	81,6	81,9
French	77,0	78,9	79,4	82,1	84,1	84,6
English	65,9	67,5	68,9	74,4	76,2	77,5
Italian	84,4	83,8	84,4	87,6	87,0	87,6
Average	**74,2**	**75,9**	**76,6**	**80,3**	**82,2**	**82,9**

Comparing the results of Table 5 to the results of Table 6, we conclude the following:

1. Choosing among ambiguous facts, using any of the three methods, achieves an overall increase in precision, accompanied by a lower decrease in recall. Results are very encouraging, given the difficulty of the task.

2. Using bigram fact transitions for choosing among ambiguous facts achieves better results that using confidence values. However, the simple multiplicative model outperforms slightly the two single methods.

To corroborate the effectiveness of the multiplicative model, we counted the number of correct choices made by the three post-processing methods at each step of the hill-climbing process, as described in section 3.2. Results are illustrated in Table 7.

Table 7. Counting the ambiguous predictions and the correct choices

Language	*Distinct T_i*	*Ambiguous T_i*	*Corrected (WI-Conf.)*	*Corrected (Bigram)*	*Corrected (Mult.)*
Greek	549	490	251	331	336
French	720	574	321	364	374
English	2203	1806	915	996	1062
Italian	727	670	538	458	483
Average	**1050**	**885**	**506**	**537**	**563**

The first column of Table 7 is the number of distinct text fragments T_i, as defined in section 3.1, for all pages in the testing corpus. The second column counts the T_i with more than one –ambiguous- facts (e.g. the T_2 in Table 2). The last three columns count the correct choices made by each of the three methods.

We conclude that by using a simple multiplicative model, based on the product of bigram probabilities and STALKER-assigned confidence probabilities we make more correct choices than by using either of the two methods individually.

5 Related Work

Extracting information from multiple Web sites is a challenging issue for the WI community. Cohen and Fun [3] present a method for learning page-independent heuristics for IE from Web pages. However they require as input a set of existing wrappers along with the pages they correctly wrap. Cohen *et al.* [4], also present one component of a larger system that extracts information from multiple sites. A common characteristic of both the aforementioned approaches is that they need to encounter separately each different markup structure during training. In contrast to this approach, we examine the viability of trainable systems that can generalize over unseen sites, without encountering each page's specific structure.

An IE system that exploits shallow linguistic pre-processing information is presented in [2]. However, they generalize extraction rules relying on lexical units (tokens), each one associated with shallow linguistic information, e.g., lemma, part-of-speech tag, etc. We generalize rules relying on named entities, which involve contiguous lexical units, and thus providing higher flexibility to the WI algorithm.

An ontology-driven IE system from pages across different sites is presented in [5]. However, they rely on hand-crafted (provided by an ontology) regular expressions, along with a set of heuristics, in order to identify single-slot facts within a document. On the other hand, we try to induce such expressions using wrapper induction.

All systems mentioned in this section experiment with different corpora, and thus cannot easily be comparatively evaluated.

6 Conclusions and Future Work

This paper presented a methodology for extracting information from Web pages across different sites, which is based on using a pipeline of three component modules: a named-entity recognizer, a standard wrapper induction system, and a post-processing module for disambiguating extracted facts. Experimental results showed the viability of our approach.

The issue of disambiguating facts is important for single-slot IE systems used on the Web. For instance, *Hidden Markov Models* (HMMs) [11] are a well-known learning method for performing single-slot extraction [6], [13]. According to this approach, a single HMM is trained for each fact. At run-time, each HMM is applied to a page, using the *Viterbi* procedure, to identify relevant matches. Identified matches across different HMMs may be identical or overlapping resulting in ambiguous facts. Our post-processing methodology can thus be particularly useful to HMM extraction tasks.

Bigram modeling is a simplistic approach to the exploitation of dependencies among facts. We plan to explore higher-level interdependencies among facts, using higher order n-gram models, or probabilistic FSA, e.g. as learned by the *Alergia* algorithm [1]. Our aim is to further increase the number of correct choices made for ambiguous facts, thus further improving both recall and precision. Dependencies among facts shall be also investigated in the context of *multi-slot* extraction.

A bottleneck in existing approaches for IE is the labeling process. Despite the use of a user-friendly annotation tool [14], the labeling process is a tedious, time-consuming and error-prone task, especially when moving to a new domain. We plan to investigate *active learning* techniques [9] for reducing the amount of labeled data required. On the other hand, we anticipate that our labeled datasets will be of use as benchmarks for the comparative evaluation of other current and/or future IE systems.

ACKNOWLEDGEMENTS

This work has been partially funded by a research grant -provided by the NCSR "Demokritos"- and CROSSMARC, a EC-funded research project. Many thanks to Georgios Sakis for implementing the STALKER system.

References

1. Carrasco, R., Oncina, J., Learning stochastic regular grammars by means of a state-merging method. *Grammatical Inference and Applications, ICGI'94*, p. 139-150, Spain (1994).

2. Ciravegna, F., Adaptive Information Extraction from Text by Rule Induction and Generalization. *In Proceedings of the 17th IJCAI Conference.* Seattle (2001).
3. Cohen, W., Fan, W., Learning page-independent heuristics for extracting data from Web pages. *In the Proceedings of the 8th international WWW conference (WWW-99).* Toronto, Canada (1999).
4. Cohen, W., Hurst, M., Jensen, L., A Flexible Learning System for Wrapping Tables and Lists in HTML Documents. *Proceedings of the 11th International WWW Conference*. Hawaii, USA (2002).
5. Davulcu, H., Mukherjee, S., Ramakrishman, I.V., Extraction Techniques for Mining Services from Web Sources, *IEEE International Conference on Data Mining*, Maebashi City, Japan (2002).
6. Freitag, D., McCallum, A.K., Information Extraction using HMMs and Shrinkage. *AAAI-99 Workshop on Machine Learning for Information Extraction*, p.31-36 (1999).
7. Kushmerick N., Wrapper induction for Information Extraction, *PhD Thesis, Department Of computer Scienc, Univ. Of Washington* (1997).
8. MUC-7, *http://www.itl.nist.gov/iaui/894.02/related_pr ojects/muc*.
9. Muslea, I., Active Learning with multiple views. *PhD Thesis*, University of Southern California (2002).
10. 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).
11. Rabiner, L., A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77-2 (1989).
12. Sebastiani F., Machine Learning in Automated Text Categorization, *ACM Computing Surveys*, 34(1):1-47 (2002).
13. Seymore, K., McCallum, A.K., Rosenfeld, R., Learning hidden Markov model structure for information extraction. *AAAI Workshop on Machine Learning for Information Extraction*, p. 37-42 (1999).
14. Sigletos, G., Farmakiotou, D., Stamatakis, K., Paliouras, G., Karkaletsis V., Annotating Web pages for the needs of Web Information Extraction Applications. *Poster at WWW 2003*, Budapest Hungary, May 20-24 (2003).