ONDUX: On-Demand Unsupervised Learning for Information Extraction

Eli Cortez[1] Altigran S. da Silva[1] Marcos André Gonçalves [2]

Edleno S. de Moura[1]

[1]Universidade Federal do Amazonas
Departamento de Ciência da Computação
Manaus, AM, Brazil
{eccv,alti,edleno}@dcc.ufam.edu.br

[2]Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
Belo Horizonte - MG - Brazil
mgoncalv@dcc.ufmg.br

ABSTRACT

Information extraction by text segmentation (IETS) applies to cases in which data values of interest are organized in implicit semi-structured records available in textual sources (e.g. postal addresses, bibliographic information, ads). It is an important practical problem that has been frequently addressed in the recent literature. In this paper we introduce ONDUX (On Demand Unsupervised Information Extraction), a new unsupervised probabilistic approach for IETS. As other unsupervised IETS approaches, ONDUX relies on information available on pre-existing data to associate segments in the input string with attributes of a given domain. Unlike other approaches, we rely on very effective matching strategies instead of explicit learning strategies. The effectiveness of this matching strategy is also exploited to disambiguate the extraction of certain attributes through a reinforcement step that explores sequencing and positioning of attribute values directly learned *on-demand* from test data, with no previous human-driven training, a feature unique to ONDUX. This assigns to ONDUX a high degree of flexibility and results in superior effectiveness, as demonstrated by the experimental evaluation we report with textual sources from different domains, in which ONDUX is compared with a state-of-art IETS approach.

Categories and Subject Descriptors

H.2 [**Database Management**]: Miscellaneous ; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Performance, Experimentation

Keywords

Data Management, Information Extraction, Text Segmentation

SIGMOD'10, June 6–11, 2010, Indianapolis, Indiana, USA.

1. INTRODUCTION

The abundance of on-line sources of text documents containing implicit semi-structured data records in the form of continuous text, such as product descriptions, bibliographic citations, postal addresses, classified ads, etc., has attracted a number of research efforts towards automatically extracting their data values by segmenting the text containing them [1, 4, 12, 20]. This interest is motivated by the necessity of having these data stored in some structured format as relational databases or XML, so that it can be further queried, processed and analyzed.

For instance, an article from "The Washington Post" reports that the revenues by Newspapers from classified ads, which was $17 billion in 2006, has been declining since 2000, while the revenues from *on-line* classified ads grew 6 times in the same period, reaching $3.1 billion. Empowering users with services such as sophisticated searching, dissemination, comparison, personalization on top of this content, can have a significant impact on this business. Extracting and structuring these data is a crucial step towards this goal.

As an example of the information extraction task performed by a typical text segmentation system, consider the input ad *"Regent Square $228,900 1028 Mifflin Ave.; 6 Bedrooms; 2 Bathrooms. 412-638-7273"*. A suitable text segmentation over this string would generate a structured record such as:

⟨neighborhood,"*Regent Square*"⟩,
⟨price,"*$228,900*"⟩,
⟨number,"*1028*"⟩,
⟨street,"*Mifflin Ave.;*"⟩,
⟨bedrooms,"*6 Bedrooms;*"⟩,
⟨bathrooms,"*2 Bathrooms.*"⟩,
⟨phone,"*412-638-7273*"⟩

The dominant approach in information extraction by text segmentation (IETS) is the deployment of statistical methods such as as Hidden Markov Models (HMM) [4] or Conditional Random Fields models (CRF) [11] to automatically learn a statistical model for each application domain. These methods usually require training data consisting of a set of representative segmented and labeled input strings. Currently, methods based on CRF are state-of-art, outperforming HMM-based methods in experimental evaluations presented in the literature [19, 20].

Obtaining a large amount of training data may be very expensive or even unfeasible in some situations. Recognizing this problem, recent papers proposed the use of pre-existing datasets to alleviate the need for manually labeled training string segments to associate them with their corresponding attributes [1, 12, 20]. In these methods, the learning process takes advantage of known values of a given attribute to train a model for recognizing values of this attribute occurring in an input textual record.

In this paper we introduce ONDUX (**ON**-**D**emand **U**nsupervised Information E**X**traction), an alternative unsupervised probabilistic IETS approach. Similar to previous unsupervised approaches [1, 12, 20], ONDUX also relies on pre-existing data, more specifically, on sets of attribute values from pre-existing data sources, to associate segments in the input string with a given attribute. Different from previous work, there is not an explicit learning process in this step. Instead, we use simple generic matching functions to compute a score measuring the likelihood of text segments to occur as a typical value of an attribute.

Although this simple greedy matching-based strategy is effective (as shown in our experimental results), it may fail for ambiguous attributes with similar domains. This is the case of attributes such as Title and Keywords, found on bibliographic information extracted from paper headings. To solve this, we rely on positioning and sequencing probabilities of the attribute values. While in traditional methods, such as HMM and CRF, these probabilities are assumed as fixed [1, 20] or are learned through a manual labeling process [4, 18, 12], our method can automatically adapt to variable attribute values positioning and sequencing in an unsupervised way. In other words, it does not rely on the explicit association between unsegmented input strings and the corresponding segmented strings (labeled data) that supervised systems require for training, i.e., the labels "come for free" with the attributes of our pre-existing data source. More importantly, as in some unsupervised learning and transductive methods [9], we take advantage of information about the own records we are trying to extract (the test set) by exploiting the high certainty of the matching step in order to incorporate, on demand, information about the positioning and sequencing of attribute values in these records within the extraction model we generate.

To corroborate our claims regarding the high-quality and flexibility of our approach, we present results of experiments with several textual sources from different domains. In these experiments ONDUX is compared with CRF, the state-of-art method in probabilistic information extraction [11, 19], in its unsupervised version [20]. Results of these experiments reveal that ONDUX was able to correctly identify attribute values in all different datasets, outperforming CRF in most of the cases. Moreover, despite the on-demand, unsupervised nature of ONDUX, in experiments performed to evaluate the time spent on processing instances, our performance was very similar to that of CRF, which applies a previously constructed extraction model generated in an off-line training phase.

In sum, the main contribution of our work is a very effective unsupervised information extraction method that:(1) instead of requiring explicit learning of a model for identifying attributes values on the input texts, uses simple but very effective greedy strategy based on matching; (2) exploits the high accuracy of this matching strategy to learn from the test data the probabilities of positioning and sequencing of attributes in an unsupervised manner, making no rigid assumptions about the order of the attribute values, thus being much more robust and flexible to changes in patterns; (3) despite the fact of operating on-demand, has processing time of test instances similar to that of methods that use explicit learning such as CRF.

This paper is organized as follows. Section 2 discusses the main challenges in IETS and previous approaches in the literature. Section 3 presents an overview of ONDUX and discusses the details on the steps involved in its operation. Section 4 presents experiments for verifying the effectiveness of our approach comparing it with a state-of-art IETS approach. Section 5 presents a comparison of ONDUX with previous related IETS approaches in the literature. Section 6 concludes the paper giving directions for future work.

2. IETS: CHALLENGES AND APPROACHES

Information extraction by text segmentation (IETS) is the problem of segmenting text inputs to extract implicit data values contained in them. Informally, each text input forms an implicit record [19]. A fairly common approach to solve this problem is the use of machine learning techniques, either supervised, i.e., with human-driven training [8, 4, 18], or unsupervised, i.e., with training provided by some form of pre-existing data source [1, 5, 12, 20].

One of the first approaches in the literature addressing this problem was proposed by Freitag and McCallum in [8]. It consisted in generating independent Hidden Markov Models (HMM) for recognizing values of each attribute. This approach was extended in the DATAMOLD tool [4], in which attribute-driven (or *internal*) HMMs are nested as states of an *external* HMM. This external HMM aims at modeling the sequencing of attribute values on the implicit records. Internal and external HMM are trained with user-labeled text segments. Experiments over two real-life datasets yielded very good results in terms of the accuracy of the extraction process.

Later on, *Conditional Random Fields (CRF)* models were proposed as an alternative to HMM for the IETS task [11]. In comparison with HMM, CRF models are suitable for modeling problems in which state transitions and emissions probabilities may vary across hidden states, depending on the input sequence. In [18], a method for extracting bibliographic data from research papers based on CRF is proposed and experimentally evaluated with good results. Currently, CRF constitutes the state-of-art in information extraction due to its flexibility and the quality of the extraction results achieved [18, 12].

Although effective, these supervised IETS approaches based on graphical models such as HMM and CRF usually require users to label a large amount of training input documents. There are cases in which training data is hard to obtain, particularly when a large number of training instances is necessary to cover several features of the test data.

To address this problem, recent approaches presented in the literature propose the use of pre-existing data for easing the training process [1, 12, 20]. According to this strategy, models for recognizing values of an attribute are generated from values of this attribute occurring in a database previously available. These approaches take advantage of large

amounts of existing structured datasets with little or no user effort.

Following this strategy, recent methods in the literature use *reference tables* in combination with graphical models, that is, HMMs [1] or CRFs [12, 20]. For recognizing values of a given attribute among segments of the input string, a model is trained using values available on the reference table for this attribute. No manually labeled training input strings are required for this. Once attribute values are recognized, records can be extracted. The methods proposed in [1, 20] assume that attributes values in the input text follow a single global order. This order is learned from a sample batch of the test instances. On the other hand, the method proposed in [12] can deal with records bearing different attribute value orders. To accomplish this, the CRF model must be learned using additional manually labeled input strings.

A similar strategy is used in [5]. However, when extracting data from a source in a given domain, this approach may take advantage not only from pre-existing datasets, but also from other sources containing data on the same domain, which is extracted simultaneously from all sources using a 2-state HMM for each attribute. Record extraction is addressed in a unsupervised way by aligning records from the sources being extracted.

As these approaches alleviate or even eliminate the need for users to label segments in training input strings; we regard them as *unsupervised* IETS approaches. Despite this, experimental results reported for these methods reveal extraction quality levels similar to those obtained with traditional supervised IETS methods [8, 4, 18].

Our method ONDUX can also be regarded as unsupervised, since it relies on pre-existing data sources to recognize attribute values on input strings. In a first step, it deploys effective generic similarity functions to label text segments based matching scores between these segments and known values of a given attribute. Next, assigned labels are revised based on a reinforcement step that takes into account sequencing and positioning of attribute values directly learned *on-demand* from test data, with no previous human-driven training. As demonstrated by experimental results, in which ONDUX is compared with a state-of-art IETS approach, these features yield highly accurate results which are in most cases superior to the state-of-the-art.

3. THE ONDUX METHOD

In this section, we present the details of ONDUX, our unsupervised probabilistic approach for IETS. Given a text input T containing a set of implicit textual records, ONDUX identifies data values available in these records and associates these values with proper attributes. In the following, we first present an overview of ONDUX and describe the main steps involved in its functioning. Next, each is discussed in turn with details.

3.1 Overview

Consider an input string I representing a real classified ad such as the one presented in Figure 1(a). Informally, the IETS problem consists in segmenting I in a way such that each segment s receives a label ℓ corresponding to an attribute a_ℓ, where s represents a value in the domain of a_ℓ. This is illustrated in Figure 1(d), which is an example of the outcome produced by ONDUX.

Similar to previous approaches [1, 12, 20], in ONDUX, we use attribute values that come from pre-existing data sources from each domain (e.g. addresses, bibliographic data, etc.) to label segments in the input text. These values are used to form domain-specific *Knowledge Bases*(*KBs*).

A Knowledge Base is a set of pairs $K = \{\langle a_1, O_1\rangle, \ldots, \langle a_n, O_n\rangle\}$ in which each a_i is a distinct attribute, and O_i is a set of strings $\{o_{i,1}, \ldots, o_{i,n_i}\}$ called *occurrences*. Intuitively, O_i is a set of strings representing plausible or typical values for attribute a_i.

Given a data source on a certain domain which includes values associated with fields or attributes, building a Knowledge Base is a simple process that consists in creating pairs of attributes and sets of occurrences. Example of possible data sources are: databases, reference tables, ontologies, etc.

In Figure 2 we present a very simple example of a KB which includes only four attributes: *Neighborhood*, *Street*, *Bathrooms*, and *Phone*.

The first step in ONDUX operation is called *Blocking*. In this step, the input string is roughly segmented into units we call *blocks*. Blocks are simply sequences of terms (words) that are likely to form a value of an attribute. Thus, although terms in a block must all belong to a same value, a single attribute value may have terms split among two or more blocks. This concept is illustrated in Figure 1(c). Observe that the blocks containing terms "Mifflin" and "Ave" are parts of the same value of attribute Street.

Next, in the *Matching* step, blocks are matched against known attribute values, which are available in the Knowledge Base, using a small set of specific matching functions. By the end of the matching step, each block is *pre-labeled* with the name of the attribute for which the best match was found.

We notice that Blocking and Matching steps alone are enough to correctly label the large majority of the segments in the input string. Indeed, experiments with different domains, which we have performed and reported here, show that blocks are correctly pre-labeled in more than 80% of the cases. This is illustrated in Figure 1(d) in which the Matching was able to successfully label all blocks except for the ones containing the terms "Regent Square" and "Mifflin".

Problems such as this are likely to occur in two cases. First, *Mismatching*, happens when two distinct attributes have domains with a large intersection. For instance, when extracting from scientific paper headings, values from attributes Title and Keywords have usually several terms (words) in common. In our running example, as shown in Figure 1(c), "Regent Square" was mistakenly labeled with *Street* instead of *Neighborhood*. Second, *Unmatching*, happens when no matching was found for the block in the Knowledge Base, as the case of the block containing the term "Mifflin" in Figure 1(c).

To deal with such problems, our method deploys a third step we call *Reinforcement* in which the pre-labeling resulting from the Matching step is reinforced by taking into consideration the positioning and the sequencing of labeled blocks in the input texts.

To accomplish this, first, a probabilistic HMM-like graph model we call PSM(Positioning and Sequencing Model) is built. This model captures (i) the probability of a block labeled with ℓ appear in position p in the input text, and (ii) the probability of a block labeled with ℓ appear before a block labeled with m in the input text. Next, these probabilities are used to reinforce the pre-labeling resulting from the

Figure 1: Example of an extraction process on a classified ad using ONDUX.

$$K = \{\langle Neighborhood, O_{Neighborhood}\rangle, \langle Street, O_{Street}\rangle, \langle Bathrooms, O_{Bathrooms}, Phone, O_{Phone}\rangle\}$$
$$O_{Neighborhood} = \{\text{“Regent Square”,“Milenight Park”}\}$$
$$O_{Street} = \{\text{“Regent St.”,“Morewood Ave.”,“Square Ave. Park”}\}$$
$$O_{Bathrooms} = \{\text{“Two Bathrooms”,“5 Bathrooms”}\}$$
$$O_{Phone} = \{\text{“(323) 462-6252”,“171 289-7527”}\}$$

Figure 2: A simple example of a Knowledge Base.

Labeling step, assigning labels to previous unmatched blocks and changing labels for blocks found to be mismatched so far.

One important point to highlight regarding ONDUX is that PSM is built without manual training, using the pre-labeling resulting from the Matching step. This implies that the model is learned *on-demand* from test instances, with no *a priori* training, relying on the very effective matching strategies of the Matching step.

In the following we present the details of each step described above.

3.2 Blocking

The first step of ONDUX consists of splitting an input string into substrings we call *blocks*. In our proposed method, we consider blocks as sequences of terms that will compose the same value of a certain attribute. In Figure 1(c) the blocks identified in our example input string are marked with rectangles.

The blocking process is based on the co-occurrence of terms in a same attribute value according to the Knowledge Base. This process is described in Algorithm 1.

Let I be an input string. Initially, terms are extracted from I based on the occurrence of white spaces in the string. Special symbols and punctuation are simply discarded (Line 1).

Next (Lines 7–15), blocks are built as follows: if the current term (say, t_{j-1}) and next term (say, t_j) are known to co-occur in some occurrence in the Knowledge Base, then t_j will compose the same block as t_{j-1}. Otherwise, a new block will be built for t_j. This process is repeated until all terms of I are assigned to a block. Notice that terms that do not occur in the Knowledge Base always form a block alone.

According to the Knowledge Base presented in Figure 2, terms “Regent” and “Square” co-occur as values of the attribute Neighborhood. Thus, as shown in Figure 1(b),these terms are in the same block, i.e, the first block in the figure.

Algorithm 1 Blocking

```
1: I : Input Text
2: K = {⟨a_1, O_1⟩, ..., ⟨a_n, O_n⟩} : Knowledge Base
3: T : ⟨t_0, ..., t_n⟩ ← ExtractTerms(I)
4: B_0 ← ... ← B_n ← ∅ {Initialize blocks}
5: B_0 ← B_0 ∪ ⟨t_0⟩; {Builds the first block}
6: i = 0, j = 1;
7: repeat
8:    C ← {⟨a_k, O_k⟩ ∈ K, o_x ∈ O_k | t_{j-1}, t_j ∈ o_x}
9:    if C = ∅ then
10:      {t_{j-1} and t_j do not co-occur}
11:      i ← i + 1; {Next block}
12:   end if
13:   B_i ← B_i ∪ ⟨t_j⟩; {Adds t_j to the current block}
14:   j++; {Next term}
15: until j = n
```

3.3 Matching

The Matching step consists in associating each block generated on the Blocking step with an attribute represented in the Knowledge Base. For this, we use a small set of specific similarity functions to match each block against the occurrences composing the Knowledge Base and determinate the attribute that the block is more likely to belong to.

The specific function used to match a block is chosen by a simple test over the terms composing this block to define a data type. We consider four distinct types of data with a corresponding matching function: *text*, *numeric*, *urls*, and *email*. These functions are described bellow.

Matching Text values

Values of textual attributes (e.g., names of neighborhoods, streets, authors, etc.) are handled using a function called AF (Attribute Frequency) [14], which estimates the similarity between a given value and the set of values of an attribute. In our case, the function AF is used to estimate the similarity between a block B and the values of attribute a_i available on the occurrences in the Knowledge Base. We define AF as follows.

$$AF(B, a_i) = \frac{\sum_{t \in T(a_i) \cap T(b)} fitness(t, a_i)}{|T(B)|} \quad (1)$$

In Equation 1, $T(a_i)$ is the set of all terms found in the occurrences of attribute a_i in the Knowledge Base and $T(B)$ is the set of terms found in block B. The function $fitness(t, a_i)$ evaluates how typical a term t is among the values of attribute a_i. It is computed as follows.

$$fitness(t, a_i) = \frac{f(t, a_i)}{N(t)} \times \frac{f(t, a_i)}{f_{max}(a_i)} \quad (2)$$

where $f(t, a_i)$ is the number of occurrences of a_i in the Knowledge Base which contains the term t, $f_{max}(a_i)$ is the highest frequency of any term among the occurrences of a_i in the Knowledge Base, and $N(t)$ is the total number of occurrences of the term t in all attributes represented in the Knowledge Base.

The first fraction in Equation 2 expresses the probability of term t to be part of an occurrence of a_i in the knowledge base. Such probability would be suitable for our purposes if all a_i had the same number of occurrences in the Knowledge Base. As this is not true in general, attributes with more occurrences would tend to have higher probability values. Therefore, we add the second fraction, as a normalization factor to avoid this problem. This fraction gives the frequency of t in occurrences of a_i normalized by maximum frequency of a value in occurrences of a_i. Thus, it varies from 0, which means completely infrequent, to 1, which means this is the most frequent. This normalization is also useful for making the frequency terms comparable among all attributes.

Thus, for each block B with textual values in the input string, we calculate $AF(B, a_i)$, for every textual attribute a_i in the Knowledge Base. Finally, B is associated with the attribute which gives the maximum AF value.

We notice that although we could have used some other similarity function, for instance, based on the vector space model, previous results [6, 7, 14] have shown that AF is very effective for dealing with small portions of texts such as the ones typically found in blocks.

Matching Numeric Values

For the case of blocks containing numbers only (e.g. page numbers, year, volume, house number, price, etc.) traditional textual similarity functions do not work properly. Thus, for matching these blocks we assume, as proposed in [2], that the values in numerical attributes follow a gaussian distribution. Based on this assumption, we measure the similarity between a numeric value v_B represented in a block B and the set values $V(a_i)$ of an attribute a_i in the Knowledge Base, by evaluating how close v_B is from the mean value of $V(a_i)$ according to the probability density function of a_i. For this, we use function NM (Numeric Matching), defined in Equation 3, normalized by the maximum probability density of $V(a_i)$, which is reached when a given value is equal to the average[1].

$$NM(B, a_i) = e^{-\frac{v_B - \mu}{2\sigma^2}} \quad (3)$$

where σ and μ are the standard deviation and the average, respectively, of values of $V(a_i)$, and v_B is the numerical value that composes B.

Notice that when v_B is close to the average of values in $V(a_i)$, $NM(B, a_i)$ is close to 1. As v_B assumes values far from the average, the similarity tends to zero.

As for the case of textual values, for each block B with numeric values in the input string, we calculate $NM(B, a_i)$, for every numeric attribute a_i in the Knowledge Baseand B is associated with the attribute which gives the maximum NM value.

[1]The maximum probability density of $V(a_i)$ is $1/\sqrt{2\pi\sigma^2}$

In many cases numeric values in the input strings are formated using special characters. For instance, notice the price and the phone number in the example text input in Figure 1. Thus, prior to the application of the NM function, these characters are removed and the remaining number are concatenated. We call this process *Normalization*. For instance, the string "412-638-7273" is normalized to form a numeric value 4126387273 that can be applied to the function NM. Normalization is also performed over numeric values in the occurrences from the Knowledge Base. This is the case occurrences of attribute Phone illustrated in Figure 2.

Matching URLs and e-mail values

For matching URL and e-mails, considering that values in attributes of these domains follow a specific format, we apply simple binary functions using regular expressions, which identify each specific format and return true or false.

Unmatchings and Mismatchings

Despite its simplicity, the simple matching strategy we adopt to label blocks is by itself a very effective approach for labeling segments in the input text. Indeed, experiments with different domains, which we have performed and reported here, show that blocks are correctly pre-labeled in more than 70% of the cases.

In Figure 1(c) we present the result obtained after the matching phase for our running example. As can be noticed, almost all blocks were assigned to a proper attribute, except for the following cases: (1) the block containing "Mifflin" was left unmatched and (2) the block containing "Regent Square" was mistakenly assigned to Street, instead of being assigned to Neighborhood. These are examples of unmatchings and mismatchings in the context of text attributes, we further discuss here due to its importance.

As defined by Equations 1 and 2, the AF function relies on the intersection between the terms composing a given block B and the set of terms composing the known values of an attribute a_i, i.e., the *vocabulary* of a_i.

Thus, the first case, unmatched blocks, occurs occur when no term from B is found in values of a_i. This may represent a problem if the Knowledge Base. does not contain representative values for the domain of a_i. The second case, mismatched blocks, occurs when a distinct attribute a_j shares a similar vocabulary with a_i, since $AF(B, a_j)$ result in a value greater than $AF(B, a_i)$. This may happen not only due to the misrepresentation of a_i domain but also due to the intrinsic ambiguous nature of both attributes. This is the case for attributes Street and Neighborhood in our example.

To deal with both cases, our method includes a third step, *Reinforcement*, which is discussed in the following section.

3.4 Reinforcement

The Reinforcement step consists in revising the pre-labeling made by the Matching step over the blocks. More specifically, unmatched blocks are labeled and mismatched blocks are expected to be correctly re-labeled. We notice that in our context, the term Reinforcement is used in a sense slightly different from the traditional Reinforcement Learning technique [10]. Indeed, in our case the PSM does not only reinforces the labeling performed by the matching step, but also revises and and possibly corrects it.

Let m and ℓ be the labels which respectively identify attributes a_ℓ and a_m from the Knowledge Base. Consider an

input string $\ldots, B_{i-1}, B_i, \ldots$, so that m is known to label block B_{i-1}. To verify if ℓ can be used to label block B_i, the Reinforcement step takes into account: (1) the probability of the i-th block in the input strings being labeled with ℓ; and (2) the probability of using ℓ to label a block following another block labeled with m (e.g. B_{i-1}).

These probabilities are estimated, based in the knowledge acquired as a result of the Matching step by means of a probabilistic HMM-like graph model we call PSM(Positioning and Sequencing Model). Next, these probabilities are used to reinforce the pre-labeling resulting from the Matching step.

As the pre-labeling of blocks performed in this step has a high accuracy (as demonstrated in our experiments), it can be used to learn features related to sequencing and the positioning of attribute values in input texts. It is important to notice that these features are learned *on-demand* from each set of input text with no need for human training nor assumptions regarding a particular order of attribute values.

Positioning and Sequencing Model

A *Positioning and Sequencing Model* or PSM consists of: (1) a set of states $L = \{begin, l_1, l_2, \ldots, l_n, end\}$ where each state l_i corresponds to a label assigned to a block on the Matching step, except for two special states, *begin* and *end*; (2) A matrix T that stores the probability of observing a transition from state l_i to state l_j; and (3) A matrix P that stores the probability of observing a label l_i in a block in the position k in an input text;

Matrix T, which stores the transition probabilities, is built using the ratio of the number of transitions made from state l_i to state l_j in the output of the Matching step to the total number of transitions made from state l_i. Thus, each element $t_{i,j}$ in T is defined as:

$$t_{i,j} = \frac{\#\ of\ transitions\ from\ l_i\ to\ l_j}{Total\ \#\ of\ transitions\ out\ of\ l_i} \quad (4)$$

Matrix P, which stores the position probabilities, is built using the ratio of the number of times a label l_i is observed in position k in the output of the Matching step to the total number of labels observed in blocks that occupy position k. Thus, each element $p_{i,k}$ in P is defined as:

$$p_{i,k} = \frac{\#\ of\ observations\ of\ l_i\ in\ k}{Total\ \#\ of\ blocks\ in\ k} \quad (5)$$

By using Equations 4 and 4, matrices T and P are built to maximize the probabilities of the sequencing and the positioning observed for the attribute values, according to the labeled blocks in the output of the matching step. This follows the Maximum Likelihood approach, commonly used for training graphical models [4, 19].

In practice, building matrices T and P involve performing a single pass over the output from the Matching phase. Notice that blocks left unmatched are discarded when building these matrices. Obviously, possible mismatched blocks will be used to built the PSM, generating spurious transitions. However, as the number of mismatches resulting from the Matching step is rather small, as demonstrated in our experiments, they do not compromise the overall correctness of the model.

Figure 3 shows an example of the PSM built for a test set of classified ads. As we can see, the graph represents not only information on the sequencing of labels assigned to blocks, but also on the positioning of labels in blocks within text inputs. For instance, in this test set, input texts are more likely to begin with blocks labeled with Neighborhood than with blocks labeled with Street. Also, there is a high probability that blocks labeled with Phone occurring after blocks labeled with Bedrooms.

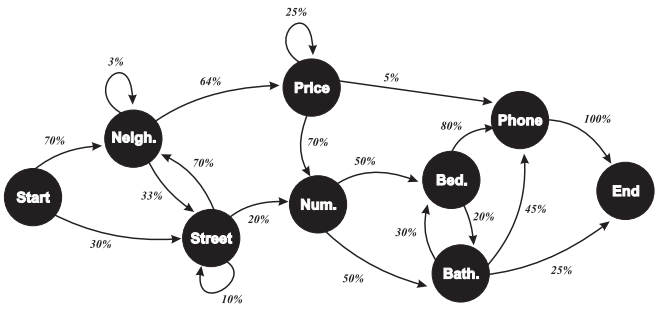

Figure 3: Example of a PSM

After generating the PSM, the estimated probabilities are used to perform label reinforcement, as discussed in the following section.

Label Reinforcement

On the Matching step, the labeling of a block was made based entirely on the matching functions introduced in Section 3.3. However, after building the PSM, the decision on what label to assign to a block can also take into account the probabilities related to positioning and sequencing in text inputs.

To combine these factors, let $M(B, a_i)$ be one of the matching functions presented in Section 3.3 and assume that it represents the probability of a block B to occur in a value of the domain of attribute a_i, according the Knowledge Base.

As $M(B, a_i)$ is estimated based uniquely on the Knowledge Base, it is independent on the particular source of the input strings. On the other hand, the positioning and sequencing probabilities are learned from each particular source during the extraction process, and they are mutually independent.

The independence between the three factors allows us to combine them through the Bayesian disjunctive operator $or(\cdot, \cdot)$, also known as *Noisy-OR-Gate* [17], which is defined as:

$$or(x_1, \ldots, x_n) = 1 - ((1 - x_1) \times \ldots \times (1 - x_n))$$

where each x_i is a probability.

In our case, we use the following:

$$FS(B, a_i) = 1 - ((1 - M(B, a_i)) \times (1 - t_{j,i}) \times (1 - p_{i,k})) \quad (6)$$

where B is a block found in position k in a given input string, preceded by another block known to be assigned to attribute a_j. Factors $t_{j,i}$ and $p_{i,k}$ are the probabilities stored in matrices T and P, respectively.

Informally, by using the disjunctive operator we assume that any of the factors is likely to determine the labeling (i.e., significantly increase its final probability), regardless of other factors [17]. By doing so, we avoid having to fine-tune relative weights for individual factors. As we shall see, this hypothesis will be confirmed in our experiments.

Function $FS(B, a_i)$ is computed for each block B in the input text for all attributes a_i of the same data type (i.e., text,

numeric, URL and e-mail). B is finally labeled with a label representing the attribute which yielded the highest score according to FS. Notice that there will be no unmatched blocks after this process. Once all blocks are labeled, contiguous blocks with a same label are merged. Thus, each block would correspond to a single attribute value.

This is illustrated in our running example in Figure 1(d), in which all blocks are correctly assigned to the attributes. The first block, which was wrongly labeled in the matching phase, has now received a correct assignment to the Neighborhood attribute. Also, the unmatched block containing the term "Miffin" now composes a value of attribute Street.

4. EXPERIMENTAL RESULTS

In this section, we evaluate ONDUX using a variety of real datasets to show that our method is a robust, accurate, and efficient unsupervised approach for IETS. We first describe the experimental setup and metrics used. Then, we report results on extraction quality and performance over all distinct datasets.

4.1 Setup

Baselines

In the experiments, we compare ONDUX with an unsupervised version of CRF, a state-of-art IETS approach. This version was developed by adapting the publicly available implementation of CRF by Sunita Sarawagi [2], according to what is described in [20]. We call this version *U-CRF*. We believe that *U-CRF* represents the most suitable baseline for comparing with ONDUX, as it delivers top performance while at the same time does not require user-provided training. Although the Extended Semi-markov CRF presented in [12] could have been used as baseline, since it relies mostly on features extracted from a KB, it also uses a small portion of manually trained data. Moreover, [20] improves on [12] results. However, since this our first baseline assumes, as we shall see in more details later, that the order of the text sequences to be extracted is fixed, we also included the standard CRF model [11] (called *S-CRF*), that does not have this limitation at all but requires manually labeled training data. Obviously, *S-CRF* is only used as a baseline for cases in which we have the training data. Using the two baselines, also allows us to compare the strengths of each of these models against our approach.

As for the configuration of U-CRF and S-CRF, we deployed the same features described in [20] and in [11]. Overall, these are standard features available on the publicly CRF implementation, e.g., dictionary features, word score functions, transition features, etc., plus, in the case of U-CRF the set of heuristic rules for using negative examples proposed in [20]. Although the basic CRF model is flexible enough to allow features to be tailored for specific extractions tasks, in all experiments we have used the same configurations for U-CRF and S-CRF. This is to ensure a fair comparison since we assume that no specific adjustments were necessary for ONDUX to be used in the experiments.

As required by U-CRF, a batch of the input strings is used to infer the order of the attribute values. Based on the information provided in [20], this batch is composed by 10% of the input strings in all cases.

[2] `http://crf.sourceforge.net/`

Experimental Data

The sources of previous known data, used to generated the KB for ONDUX ,the references tables for U-CRF, the training data for S-CRF, and the test datasets used in the experiments are summarized in Table 1.

We tried to use the same datasets and sources explored by our baselines, when these were publicly available. In the case of restricted sources/datasets, we tried to obtain public versions of similar ones in the same domains.

Indeed, in most cases the data sources and the test datasets we have used came from public available data sources used for the empirical analysis of information extraction methods. This is the case of *Bigbook* and *Restaurants*, from the RISE repository [15], the *CORA* collection [13] and the *PersonalBib* dataset [12]. It is important to notice that in the case of *BigBook* and *CORA*, the KB and the reference tables were build from sets of records already extracted by third-parties and those are completely disjoint (i.e., no common entry) from the strings on the test datasets used in the experiments.

Data on the *Classified Ads* domain were obtained directly from the Web. For building the Knowledge Base, we collected data from a on-line database available from *Folha Online*, a popular Brazilian newspaper site. The test dataset *Web Ads* is formed by unstructured strings containing ads from other five Brazilian newspaper sites. Each website bares a distinct classified ads format, e.g., in terms of attribute values order and positioning. Moreover, the number of distinct attribute occurrences in each instance vary from 5 to 18. These properties result in a high level of heterogeneity in the test instances.

Metrics for Evaluation

In the experiments we evaluated the extraction results obtained after the Matching and Reinforcement steps discussed in Section 3. We aim at verifying how each step contributes to the overall effectiveness of ONDUX. In the evaluation we used the well known precision, recall, and F-measure metrics, but all tables report F-measure values.

Let B_i be a reference set and S_i be a test set to be compared with B_i. We define precision (P_i), recall (R_i) and F-measure (F_i) as:

$$P_i = \frac{|B_i \cap S_i|}{|S_i|} \quad R_i = \frac{|B_i \cap S_i|}{|B_i|} \quad F_i = \frac{2(R_i.P_i)}{(R_i + P_i)} \qquad (7)$$

For all the reported comparisons with U-CRF, we used the Student's T-test [3] for determining if the difference in performance was statistically significant. In all cases, we only drawn conclusions from results that were significant in, at least, 5% level for both tests. Non-significant values are omitted.

Also, we run each experiment five times, each time selecting different samples for building the knowledge base and for testing. For all the experiments we performed, we report the average of the results obtained in each of the five runs.

4.2 Extraction Quality

4.2.1 Blocking Results

The first result we report aims at verifying in practice the strategy we have formulated for the Blocking step, that is, whenever our blocking strategy generates blocks in which all

Domain	Source	Attributes	Records	Dataset	Attributes to be extracted	Text Inputs
Addresses	*BigBook*	5	2000	*BigBook*	5	500 to 2000
				Restaurants	4	250
Bibliographic Data	CORA	13	350	*CORA*	13	150
	PersonalBib	7	395			
Classified Ads	*Folha On-line*	5 to 18	125	*Web Ads*	5 to 18	500

Table 1: Domains, data sources and test datasets used in the experiments.

terms belong to a unique attribute. Thus, we measure how homogeneous each generated block is.

Dataset	Source	% Same	% Unknown
BigBook	*BigBook*	94.13%	5.34%
Restaurants	*BigBook*	92.17%	7.42%
CORA	*CORA*	80.91%	18.88%
CORA	*PersonalBib*	78.00%	19.47%
WebAds	*Folha On-Line*	87.13%	12.32%

Table 2: Results of Experiments on the Blocking Step.

Table 2, column "% Same" shows that in all test datasets a large percentage of blocks contain terms found in the values of the same attribute according to the Knowledge Base. Column "% Unknown" shows the percentage of blocks with terms not represented in the Knowledge Base. As pointed out in Section 3.2, such blocks always contain a single term. We notice that in all cases the percentage of heterogeneous blocks, that is, those that are not homogeneous nor unknown is rather small, less than 3%. Thus, we conclude that our blocking strategy behaves as expected.

It is worth mentioning that the high percentage of unknown blocks in the *CORA* dataset is caused by the diversity of terms that is normally found in the scientific paper metadata, specially in the Title attribute. As we shall see latter, despite this, ONDUX shows an excellent performance in this dataset.

4.2.2 Attribute-Level Results

To demonstrate the effectiveness of the whole extraction process with our method, we evaluate its extraction quality by analyzing, for each attribute, if the (complete) values assigned by our method to this attribute are correct.

Addresses Domain

Table 3 shows the results for the attribute level extraction over the *BigBook* dataset using the *BigBook* data source. Recall that, although the same collection has been used, the dataset and the data source are disjoint. This the same experiment reported in [20], and we include it here for completeness and to validate our baseline implementation. The *BigBook* dataset follows the assumption made by [20], according to which "a batch of text sequences to be segmented share the same total attribute order". We call this *single total attribute order assumption*.

In this table, values in boldface indicate a statistically superior result with at least 95% confidence. Starting by the comparison between the unsupervised methods, we can see that the results of both the U-CRF and ONDUX after the reinforcement are extremely high for all attributes (higher than 0.988 for all attributes). However, the results of our method are statistically superior than those of U-CRF in at least two attributes (i.e., City and Phone and are statistically

			ONDUX	
Attribute	*S-CRF*	*U-CRF*	Matching	Reinforc.
Name	0.997	0.995	0.928	0.996
Street	0.995	0.993	0.893	0.995
City	0.986	0.990	0.924	**0.995**
State	0.999	0.999	0.944	1.000
Phone	0.992	0.988	0.996	**1.000**
Average	0.994	0.993	0.937	**0.997**

Table 3: Extraction over the *BigBook* dataset using data from the *BigBook* source.

tied in the other three attributes. Another important aspect is the importance of the reinforcement step which produced gains of more than 5% over already very strong results. A closer look at this gain, reveals that it is mostly due to the recall, which improved more that 9%, while the precision improved only 2%, in average. This in accordance with our hypothesis regarding the high precision of matching step. The reinforcement step plays the role of "filing the gaps" improving the recall. Notice that the U-CRF results are very similar to those reported in [20], thus further validating our baseline implementation.

Since in this case, we have manually labeled data in the *BigBook* dataset, we were also able to compare the unsupervised methods with S-CRF. In this case, the results of both CRF-based methods are very close, and the conclusions are similar to the ones described before. This also shows that the supervised method, in this particular dataset, could not take much advantage of the training data besides what U-CRF was able to learn from the references tables.

This experiment was repeated using the *Restaurants* collection as the test dataset. Our motivation is to show that IETS approaches based on previously known data such as ONDUX and U-CRF are capable of learning and using source independent properties from these data. In this case, as well as in our others in which the source is different form the test dataset, the comparison with the S-CRF does not make sense, since, for this method to work, the learning data has to come from a similar distribution as the test data. The *Restaurants* collection has the same attributes as the *BigBook* collection, except for the State attribute. The single total attribute order assumption also applies here. The results are reported in Table 4.

Again, both U-CRF and ONDUX achieved high results for all attributes, higher than 0.942 for all attributes. ONDUX had a statistically significant advantage on attributes Name and Phone, while statistical ties were observed for attributes Street and City.

Bibliographic Data Domain

The next set of experiment was performed using the *CORA* test dataset. This dataset includes bibliographic citations

Attribute	*U-CRF*	*ONDUX*	
		Matching	Reinforcement
Name	0.942	0.892	**0.975**
Street	0.967	0.911	0.982
City	0.984	0.956	0.987
Phone	0.972	0.982	**0.992**
Average	0.966	0.935	**0.984**

Table 4: Extraction over the *Restaurants* dataset using data from the *BigBook* source.

in a variety of styles, including citations for journal papers, conference papers, books, technical reports, etc. Thus, it does not follow the single total attribute order assumption made by [20]. The availability of manually labeled data allowed us to include the S-CRF method in this comparison. A similar experiment is reported in [18]. Because of this, we have to generate our KBand the reference tables for U-CRF using the same data available on the unstructured labeled records we use to train the standard CRF, also from the *CORA* collection. As always, this training data is disjoint from the test dataset. The results for this experiment are presented in Table 5.

Attribute	*S-CRF*	*U-CRF*	*ONDUX*	
			Matching	Reinforc.
Author	93.602	90.633	0.911	**0.960**
Booktitle	91.539	76.847	0.900	0.922
Date	90.056	62.694	0.934	**0.935**
Editor	87.005	17.127	0.779	**0.899**
Institution	**93.317**	35.000	0.821	0.884
Journal	90.603	70.916	0.918	**0.939**
Location	88.704	33.333	0.902	0.915
Note	83.243	54.166	0.908	**0.921**
Pages	**98.552**	82.287	0.934	0.949
Publisher	78.508	39.805	0.892	**0.913**
Tech	83.265	16.666	0.753	0.827
Title	**96.215**	77.533	0.900	0.914
Volume	97.290	70.676	0.983	0.993
Average	90.146	55.976	0.887	**0.921**

Table 5: Extraction over the *CORA* dataset using data from the *CORA* source.

First, notice that the high results obtained with the supervised CRF (S-CRF) are similar to those reported in the original experiment [18]. In the case of ONDUX, even though it is an unsupervised method, even superior results were achieved. Statistically superior results were obtained in 6 out of 13 attributes (results in boldface) and statistical ties were observed in other 4 attributes. The results with U-CRF were rather low, what is explained by heterogeneity of the citations in the collections. While the manual training performed for S-CRF was able to capture this heterogeneity, U-CRF assumed a fixed attribute order. On the other hand, ONDUX was able to capture this heterogeneity through the PSM model, without any manual training.

Still on the Bibliographic data domain, we repeated the extraction task over the *CORA* test dataset, but this time, the previously known data came from the *PersonalBib* dataset. This dataset was used in a similar experiment reported in [12]. Again, our aim is demonstrate the source independent nature of unsupervised IETS methods. Notice that not all attributes from *CORA* were present in *PersonalBib* entries. Thus, we only extracted attribute available on both of them. The results for this experiment are presented in Table 6. Notice that in this case we could not perform manual training , since the previously known data came directly from a structured source. Thus, we do not experiment with the S-CRF here.

Attribute	*U-CRF*	*ONDUX*	
		Matching	Reinforcement
Author	0.876	0.733	**0.922**
Booktitle	0.560	0.850	**0.892**
Date	0.488	0.775	**0.895**
Journal	0.553	0.898	**0.908**
Pages	0.503	0.754	**0.849**
Title	0.694	0.682	**0.792**
Volume	0.430	0.914	**0.958**
Average	0.587	0.801	**0.888**

Table 6: Extraction over the *CORA* dataset using data from the *PersonalBib* source.

The results for ONDUX and U-CRF are quite similar to those obtained in the previous experiments, with a large advantage for ONDUX, for the reasons we have already discussed.

Classified Ads Domain

Finally, Table 7 presents the results for the experiments with test dataset *Web Ads*. The Knowledge Base and the reference tables were built using structured data from the *Folha On-Line* collection. In this table, the attribute *Others* corresponds to an amalgamation of a series of attributes present only in few adds such as Neighborhood, Backyard, Garden, etc. For this dataset, ONDUX outperforms U-CRF in about 5% even before the Reinforcement step. After this step, our method significantly outperforms the baseline in all attributes with an overall gain of more than 10% in average. Recall that this is a very heterogeneous dataset bearing several distinct formats. Our good results in this dataset highlights the robustness and the flexibility of our solution, even when compared to our closest competitor.

Attribute	*U-CRF*	*ONDUX*	
		Matching	Reinforcement
Bedroom	0.791	0.738	**0.861**
Living	0.724	0.852	**0.905**
Phone	0.754	0.884	**0.926**
Price	0.786	0.907	**0.936**
Kitchen	0.788	0.776	**0.849**
Bathroom	0.810	0.760	0.792
Suite	**0.900**	0.853	0.881
Pantry	0.687	0.741	**0.796**
Garage	0.714	0.784	**0.816**
Pool	0.683	0.711	**0.780**
Others	0.719	0.777	**0.796**
Average	0.760	0.798	**0.849**

Table 7: Extraction over the *Web Ads* dataset using data from the *Folha On-Line* source.

4.3 Dependency on Previously Known Data

An important question to address is to determine how dependent the quality of results provided by the unsupervised IETS methods studied is from the overlap between the previously known data and the text input. To study such dependency, we performed experiments to compare the behavior of ONDUX and U-CRF when varying the amount of terms given in the Knowledge Base or reference tables that overlap with the terms found in the input text. Recall that the entries in which these terms occur are used to form attribute occurrences in the Knowledge Base for ONDUX, and the reference tables for training U-CRF.

The experiments were performed using the *BigBook* dataset, which contains about 4000 entries. As mentioned earlier, this dataset came from the RISE repository [15]. Thus, the KB and the reference tables were build from sets of records already extracted, which are disjoint from the strings on the test datasets used from the same collections.

In the experiments, we vary the number of know terms that are shared between the previously known data and the input test sequence. We have also varied the number of input strings in the test sequence to check whether the amount of overlap necessary to obtain good results increase as the number of text inputs found in the test sequence also increases.

Figure 4 shows the results for four different sizes of test set, varying the number of text inputs present in the test set from (a) 500, to (d) 2000. The number of shared terms between the Knowledge Baseand the test input sequence varies in all cases from 50 to 1000 terms, and the extraction quality is evaluated by means of F-measure.

An important information obtained from these four graphs is that the quality of results provided by the methods does not vary with the size of the test input for fixed amounts of shared terms. For instance, with a overlap of 250 terms, ONDUX achieved 0.73 of F-measure for the test of size 500 and 0.74 for the test of size 1500. When taking an overlap of 100 terms, values are 0.66, 0.67. 0.68 and 0.64 for the test sizes 500, 1000, 1500 and 2000, respectively. These results indicate that, at least for this dataset domain, both ONDUX and U-CRF could keep good performance with small amount of previously known data even for larger test sets. This behavior was expected, since both methods use the overlap to obtain statistics about the structure of the test input sequence. Once the number of term overlaps is large enough to allow the methods to compute such statistics, both methods are able to learn how to extract data from the test input sequence, no matter what is its size.

We can also see from the graphs that the total number of shared terms necessary to achieve good performance is also not prohibitive, since both methods were able to achieve high quality performance (more than 95% in case of ONDUX) when taking only 750 terms of overlap for all the four size of test set studied. When looking to the smaller test sets, this overlap seems to be high when compared to the size of the test, but it does not need to increase as the test set increases. The number of records from *BigBook* source required to obtain such overlap in the KB was 162 in the results presented in Figure 4(d), about 8% of the size of the test set (remembering that these are disjoint sets). This overlap also represents about 14% of vocabulary overlap between the KB and the test set. These percentages are obviously higher for the smaller tests, since still we need 750 term overlaps to achieve about the same performance, but would tend to zero for larger test sets.

A good question at this point is to know how practical is to have hundred of terms in common between a reference set and a real data source for a system to extract information. To give a better idea about practical scenarios, let us consider all the combinations of data sources and datasets we tested in our experiments, where most collections were taken from previous experiments presented in literature.

The term overlap results found in the experiments with these combinations are depicted in Table 8. As it can be seem, except for the combination of PersonalBib as data source and CORA as dataset, in all the experiments performed the number of shared terms is higher than the amounts of shared terms found in Figure 4, which allowed both ONDUX and U-CRF to achieve high level quality of results in the experiments. For instance, when using *BigBook* as data source and *Restaurants* as the test dataset, the number of shared terms is 2504. Of course, the overlap is not the unique factor to determine the performance of the methods and the amount of overlap required may vary according to other factors presented in our experiments. However, still the amount of overlap required by the two experimented methods is not a prohibitive aspect for their practical application.

Source	Dataset	# of shared terms
BigBook	*BigBook*	3667
BigBook	*LA Restaurants*	2504
PersonalBib	*CORA*	549
CORA	*CORA*	1089
Folha On-line	*Web Ads*	1184

Table 8: Term overlap in the experiments performed with all combinations of data sources and test datasets adopted in the experiments.

4.4 Performance Issues

We move now to discuss performance issues related to ONDUX. This is an interesting aspect to analyze since ONDUX works *on-demand*, in the sense that positioning and sequencing information is learned from test instances, with no *a priori* training. Although this feature gives our method a high level of flexibility, it is important to measure its impact on the performance of the whole extraction process carried out by ONDUX.

Also in this aspect, we compare ONDUX with our baseline U-CRF. For this, we take into account training and test times. This is justified by the fact that every new extraction process carried out by U-CRF requires a new model to be learned from test instances.

The time figures we report here were collected for each one of the quality experiments presented earlier. For each specific task we measure the time in seconds spent by each unsupervised extraction method. These results are presented Table 9.

In spite of the on-demand process performed by ONDUX, the time spent on processing test instances is shorter than the time spent by U-CRF. In all experiments, we notice that ONDUX was faster than U-CRF, i.e., it needed less than *27 seconds* to execute the whole process in all extraction tasks, while U-CRF needed at least *194 seconds*.

To explain that, we notice that in ONDUX the Matching step potentially demands the largest amount of time. How-

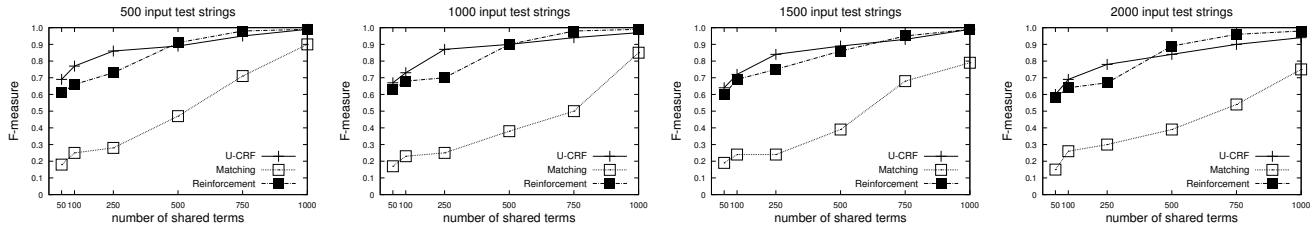

Figure 4: F-Measure values obtained when varying the number of shared terms for four different sizes of datasets built from *BigBook*.

ever, for the (frequent) case of textual attributes, the AF function is implemented using efficient inverted lists, often used in IR systems. All other steps are linear on the number of terms in input strings. On the other hand, the extraction process performed by U-CRF is slower since the generation of the model for each new extraction task requires verifying several state and transition features for each attribute prior to the proper extraction step.

Source	Dataset	*U-CRF*	*ONDUX*
BigBook	*BigBook*	316	23
BigBook	*LA Restaurants*	604	27
PersonalBib	*CORA*	317	21
CORA	*CORA*	194	17
Folha On-line	*Web Ads*	2746	19

Table 9: Time in seconds spent in each extraction task.

5. COMPARISON WITH PREVIOUS APPROACHES

ONDUX falls in the category of methods that apply learning techniques to extract information from data rich input strings. As so, it has several points in common with previous methods that have been successfully applied to such a task, such as HMM [4] and CRF [11]. However, it also has unique characteristics that are worth discussing. As CRF is the current state-of-art method for this problem, we here compare our method to it. More specifically, we compare ONDUX with CRF-based methods in the literature that, like ONDUX, rely on previously known data to generate the extraction model. These are the methods presented in [12] and [20], which we refer to as Extended Semi-CRF (ES-CRF) and Unsupervised CRF (U-CRF, as in the previous section), respectively.

The first distinction between ONDUX and the other two approaches is the matching step. This step relies on a handful of generic matching functions and does not need to be trained for a specific target source, since it relies only on the known data available on the KB. In the case of text attributes, the matching function is based on the vocabulary of the attribute domain, as represented by terms available in the Knowledge Base, while for the numeric attributes the distribution probability of the known values is used. In CRF models, several distinct state features, i.e., those based only on the properties of each attribute [19], are used for learning the extraction model. In ES-CRF and U-CRF some of these features depend on the previously available data, but other features depend on the specific target source. This is the case of segment length and counting of (previously defined) regular expressions that fire in ES-CRF, and negative examples formed from token sequences taken from the input text in U-CRF.

The main difference between ONDUX and the two similar approaches, ES-CRF and U-CRF, is the way features related to positioning and sequencing, of attributed values (transition features [19]) are learned. In ONDUX these features are captured by the PSM model, which, as demonstrated in our experiments, is flexible enough to assimilate and represent variations in the order of attributes on the input texts and can be learned without user-provided training. U-CRF is also capable of automatically learning the order of attributes, but it cannot handle distinct orderings on the input, since it assumes a single total order for the input texts. This difficult the application of the method to a range of practical situations. For instance, in bibliographic data, it is common to have more than one order in a single dataset. Further, the order may vary when taking information from distinct text input sequences, according to the bibliographic style adopted on each input. The order is even more critical in classified ads, where each announcer adopts its own way of describing the object he/she is trying to sell. Another quite common application is to extract data from online shopping sites to store them in a database. The attributes of the offer, such as price, product, discount and so on, seldom appear in a fixed order. In practical applications like these, ONDUX is the best alternative method. Further, it is as good as the baselines for any other practical application.

In ES-CRF, distinct orderings are handled, but user-provided training is needed to learn the transition features, similarly to what happens with the standard CRF model, thus increasing the user dependency and the cost to apply the method in several practical situations.

Finally, ONDUX is largely influenced by FLUX-CiM [6, 7] a unsupervised approach for extracting metadata from bibliographic citations. While FLUX-CiM also relies on a matching step in which the AF function is also used, it does not include a generic reinforcement step. Instead, it uses a set of domain-specific heuristics based on assumptions regarding bibliographic metadata. This includes the use of punctuation as attribute value delimiters, the occurrence of single values for attributes other than author names, etc. As a consequence, FLUX-CiM could not be adopted as a baseline, since it was not designed for most of the datasets we have in our experiments. ONDUX can thus be seem as a significant improvement over FLUX-CiM, which instead of being applied only to bibliographic metadata, is a general IETS approach whose algorithms do not rely on domain-

specific assumptions such as these. Specially, it doest not explicitly relies on the use of punctuation.

6. CONCLUSIONS AND FUTURE WORK

In this paper we presented ONDUX (**ON-D**emand **U**nsupervised Information E**X**traction), an alternative unsupervised probabilistic approach for IETS. ONDUX also relies on pre-existing data, more specifically, on sets of attributes values from pre-existing data sources to associate segments in the input string with a given attribute. Differently from previous work, there is not an explicit learning process in this step. Instead, we use simple generic matching functions to compute a score measuring the likelihood of text segments to occur as a typical value of an attribute.

To corroborate our claims regarding the high-quality, flexibility and effort-saving features of our approach, we tested our method with several textual sources from different domains and found that it achieved similar or better results than CRF, a state-of-art data extraction model. Our experiments also demonstrate that our approach is able to properly deal with different domains in heterogeneous applications.

We believe that the main contributions of our work are: (1) a very effective unsupervised information extraction method that (2) instead of requiring explicit learning of a model for identifying attributes values in the input texts, uses a simple but very effective greedy strategy based on matching, (3) exploits the high accuracy of this matching strategy to learn from the test data the probabilities of positioning and sequencing of attributes in an unsupervised manner, making no rigid assumptions about the order of the attribute values, thus being much more flexible and robust to changes in patterns, and finally (4) despite the fact it operates on-demand, it has processing time of test instances similar to that of methods that use explicit learning such as CRF.

The work we carried out with ONDUX opens opportunities for several future developments. We intend to investigate the use of alternative matching functions that might better distinguish attribute values. One of the functions we consider is the one proposed in [16], which is based on the commonality of features.

In addition, currently ONDUX does not handle nested structures such as lists of values of a same attribute in a record. We also plan to address this issue as future work.

Acknowledgements

This work was partially supported by grants from projects InfoWeb (550874/2007-0 CNPq), INCTWeb (573871/2008-6 CNPq), SIRIAA (55.3126/2005-9 CNPq); MinGroup (575553/2008-1 CNPq) ; by individual CNPq fellowship grants to Edleno S. de Moura, Altigran S. Silva and Marcos André Gonçalves; and by a CAPES scholarship to Eli Cortez. This research was also sponsored by UOL (www.uol.com.br), through its UOL Bolsa Pesquisa program, process number 20090213165000.

7. REFERENCES

[1] E. Agichtein and V. Ganti. Mining reference tables for automatic text segmentation. *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 20–29, Seattle, Washington,USA, 2004.

[2] S. Agrawal, S. Chaudhuri, G. Das, and A. Gionis. Automated ranking of database query results. *Proc. of CIDR 2003, Biennial Conference on Innovative Data Systems Research*, 2003.

[3] T. Anderson and J. Finn. *The New Statistical Analysis of Data*. Springer, 1996.

[4] V. R. Borkar, K. Deshmukh, and S. Sarawagi. Automatic segmentation of text into structured records. *Proc. of the ACM SIGMOD International Conference on Management of Data*, pages 175–186, 2001.

[5] S. Chuang, K. Chang, and C. Zhai. Context-aware wrapping: synchronized data extraction. *Proc. of the 33rd Intl. Conf. on Very Large Databases*, pages 699–710, Viena, Austria, 2007.

[6] E. Cortez, A. da Silva, M. Gonçalves, F. Mesquita, and E. de Moura. FLUX-CIM: flexible unsupervised extraction of citation metadata. *Proc. of the 2007 conference on Digital libraries*, pages 215–224, 2007.

[7] E. Cortez, A. da Silva, M. Gonçalves, F. Mesquita, and E. de Moura. A flexible approach for extracting metadata from bibliographic citations. *Journal of the American Society for Information Science and Technology*, Online version, 2009.

[8] D. Freitag and A. McCallum. Information extraction with hmm structures learned by stochastic optimization. In *Proc. of the 17th National Conf. on Artificial Intelligence and 12th Conf. on Innovative Applications of Artificial Intelligence*, pages 584–589, Austin, Texas, USA, 2000.

[9] T. Joachims. Transductive inference for text classification using support vector machines. In *Proc. of the International Conference on Machine Learning*, pages 200–209, Bled, Slovenia, 1999.

[10] L. P. Kaelbling, M. L. Littman, and A. P. Moore. Reinforcement learning: A survey. *J. Artif. Intell. Res. (JAIR)*, 4:237–285, 1996.

[11] J. Lafferty, A. McCallum, and F. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proc. of the Eighteenth International Conference on Machine Learning*, pages 282–289, 2001.

[12] I. R. Mansuri and S. Sarawagi. Integrating unstructured data into relational databases. In *Proc. of the International Conference on Data Engineering*, page 29. IEEE Computer Society, 2006.

[13] A. McCallum. Cora Information Extraction Collection.

[14] F. Mesquita, A. da Silva, E. de Moura, P. Calado, and A. Laender. LABRADOR: Efficiently publishing relational databases on the web by using keyword-based query interfaces. *Information Processing and Management*, 43(4):983–1004, 2007.

[15] I. Muslea. Rise - A Repository of Online Information Sources used in Information Extraction Tasks.

[16] U. Nambiar and S. Kambhampati. Answering imprecise queries over autonomous web databases. In *Proc. of the International Conference on Data Engineering*, page 45, Washington, DC, USA, 2006.

[17] J. Pearl and G. Shafer. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann San Mateo, CA, 1988.

[18] F. Peng and A. McCallum. Information extraction from research papers using conditional random fields. *Information Processing Management*, 42(4):963–979, 2006.

[19] S. Sarawagi. Information extraction. *Foundations and Trends in Databases*, 1(3):261–377, 2008.

[20] C. Zhao, J. Mahmud, and I. V. Ramakrishnan. Exploiting structured reference data for unsupervised text segmentation with conditional random fields. In *Proc. of the SIAM International Conference on Data Mining*, pages 420–431, Atlanta, Georgia, USA, 2008.