

Approximate Schemas, Query Answering and Learning

Michel de Rougemont, mdr@lri.fr
University Paris-II, CNRS-LRI
France

Adrien Vieilleribiere, vieille@lri.fr
CNRS-LRI, University Paris-Sud
France

Abstract

We use the Edit distance with Moves on words and trees and say that two regular (tree) languages are ε -close if every word (tree) of one language is ε -close to the other. A transducer model is introduced to compare tree languages (schemas) with different alphabets and attributes. Using a statistical embedding of [15], we give a tester for Approximate Query Answering on regular schemas, i.e. approximately decide a query within ε by only looking at $f(\varepsilon)$ samples, i.e. a constant fraction of the input. We also show that regular properties on words can be learnt from samples for the PAC model, using the same distance and embedding.

Contents

1	Introduction	2
2	Preliminaries	3
2.1	Distances	4
2.2	Transductions	5
2.3	Property Testing and Approximation	8
2.3.1	Approximate Schemas	8
2.3.2	Additive approximation of the distance.	9
2.3.3	A statistical embedding on strings.	10
2.4	Learning	12
3	Query answering on words	12
3.1	Tester for Approximate Query Answering	13
3.1.1	Transducer with one state	13
3.1.2	General Transducer	14
3.2	Corrector	15
4	Query answering on trees	15
5	Learning	17
6	Conclusion	18

1 Introduction

For a class \mathbf{K} of finite structures, a Schema is a subset $\mathbf{K}_0 \subseteq \mathbf{K}$, i.e. a subclass. We consider classes of strings, ordered and unordered trees and introduce the notion of *Approximate Schemas*, i.e. a notion to compare languages of strings or trees. We use the *Edit distance with Moves* on words and trees and say that two subclasses (languages) are ε -close if every word (tree) of one subclass is ε -close to the other. When the subclasses use the same alphabet, [18, 15] show, based on Property Testing, how to approximate the distance between words (trees) by sampling a number of subword independent of the size n , but only dependent on some ε parameter, and how to test regular properties in a similar way.

We wish to generalize this approach to languages which do not use the same language, i.e. to compare languages such as $\mathbf{K}_1 = 0^*1^*$ and $\mathbf{K}_2 = c(ab)^*ca^*$. In the classical sense $\mathbf{K}_2 = c(ab)^*ca^*$ is close to $\mathbf{K}_3 = (ab)^*a^*$ as every word of length n of \mathbf{K}_2 is at relative distance $\frac{2}{n}$ to \mathbf{K}_3 , by deleting the two c 's, and conversely every word of \mathbf{K}_3 is at relative distance $\frac{2}{n}$ to \mathbf{K}_2 , by inserting two c 's. We say \mathbf{K}_2 and \mathbf{K}_3 are ε -close for any ε , as $\frac{2}{n} \leq \varepsilon$ for large enough n . There is a simple transducer (finite state machine with one state) \mathcal{T} which translates 0 into ab , 1 into a , such that the image of $\mathbf{K}_1 = 0^*1^*$, by \mathcal{T} , i.e. $\mathcal{T}(0^*1^*) = (ab)^*a^*$. In this case, we say that \mathbf{K}_1 and \mathbf{K}_3 are ε -similar for any ε . We introduce a model of transducer which transforms labelled trees with attributes and is well suited for XML schemas. It allows to compare schemas, XML files with foreign schemas and to construct schemas from positive and negative examples.

We first present the approach for words, and regular expressions as schemas, but it generalizes naturally to trees and other classes of structures. Given a word w in some alphabet Σ , a regular expression r on a different alphabet Σ' and a transducer \mathcal{T} , we say that w is ε -close to r via \mathcal{T} if there exists w' ε -close to w such that $\mathcal{T}(w')$ is ε -close to r . Approximate Query answering is the decision problem which given some $\varepsilon, w, \mathcal{T}, r$ decides if $\mathcal{T}(w) \in r$ or if w is ε -far to r via \mathcal{T} . In the case of XML files, we wish to decide if a given file is close or far from DTDs using different languages of tags and attributes. This problem is related to *supervised and unsupervised learning*. In supervised learning, a search engine tries to classify semi-structured data. Given a set of DTDs and a document F from a source, it decides which DTD the file F is closer to. The search engine has many schemas and it classifies source documents. In the PAC (Probably Approximately Correct) unsupervised learning model introduced by Valiant [28], the input is a sequence of positive and negative XML files F from an unknown DTD M , taken along a distribution \mathcal{D} , and we find a DTD M' such that $error(M') = \Pr_{F \in \mathcal{D}}[M \varepsilon - far M'] \leq \delta$ for two parameters ε, δ .

Property Testing is a framework to approximate decision problems, which is well suited on classes of finite structures with a distance between structures. Given a parameter $0 \leq \varepsilon \leq 1$, an ε -tester [23, 16] for a property P decides if a structure satisfies the property P or if it is ε -far from satisfying the property P in time independent of the size of the structure and only dependent on ε . When the structure is ε -close to the property, a *corrector* finds in linear time a structure which satisfies the property and which is ε -close to the initial structure.

The main results of the paper use the *Edit Distance with Moves* on words and trees, and a transducer model to transform structures from one language into structures of another language. We show:

- Approximate Query Answering on words and trees is testable, i.e. for all ε there is an ε -tester,
- When w is ε -close to r via \mathcal{T} , we can find w' ε -close to w such that $\mathcal{T}(w')$ is ε -close to r in linear time,
- Regular properties of words and trees can be learnt for the PAC-criterion and in some cases for a

distance criterion, using similar techniques.

These results are based on the testers and correctors for words (resp. trees) introduced in [18] for regular words (resp. regular trees) and depend on this specific *Edit Distance with Moves*. Other distances do not provide such results. We also use the embedding of words into statistical vectors introduced in [15] which yields natural testers to decide Equality, Membership of a regular language and a polynomial time algorithm to decide if two non-deterministic automata are ε -equivalent. A corrector for XML along this theory presented in [10], is also used. The paper shows that the problems of learning regular properties on words and trees, are closely connected to the approximate Query Answering.

In section 2, we review the basic data exchange models, the approximation of decision problems provided by the ε -testers and the PAC learning model. We study approximate data exchange on words in section 3, and on trees in section 4.

2 Preliminaries

A k -ranked ordered tree with n nodes is a structure $T = (D_n, \text{Firstchild}, \text{Nextsibling}, \text{root})$ where the domain $D_n = \{1, \dots, n\}$ is the set of nodes with at most k successors. The relations Firstchild , Nextsibling are binary relations such that $\text{Firstchild}(u, v)$ if v is the first child of u , $\text{Nextsibling}(v, v')$ if v' is the next sibling of v , and root is a distinguished element of D_n with no predecessors. There are at most k siblings and the graph with Firstchild and Nextsibling edges is a tree. A word w is a 1-ranked tree and the Nextsibling relation is empty. An unranked ordered tree is a structure $T = (D_n, \text{Firstchild}, \text{Nextsibling}, \text{root})$ without a fixed limitation on the number of siblings, which remains less than $n - 1$. An unranked unordered tree is a structure $T = (D_n, \text{Child}, \text{root})$ with no order on the siblings.

We consider labelled trees with attribute values, i.e. nodes have a label (tag) and attributes with values in a set Str . Given a finite alphabet Σ and a finite set A of attributes, we consider a class $\mathbf{K}_{\Sigma, \mathbf{A}}$ of (Σ, A) labeled tree structures I which can be ordered or unordered. They have two domains: D is the set of nodes, and Str the set of attribute values.

- On ordered trees, $I = (D, \text{Str}, \text{Firstchild}, \text{Nextsibling}, \text{root}, L, \lambda)$
- On unordered trees, $I = (D, \text{Str}, \text{Child}, \text{root}, L, \lambda)$

where root is the root of the tree, $L : D \rightarrow \Sigma$ defines the node label, and $\lambda : D \times A \rightarrow \text{Str}$ is a partial function which defines the attributes values of a node, when they exist. On unordered trees Child is the edge relation of an unranked tree, whereas on ordered trees Firstchild defines the first child of a node and Nextsibling defines the successor along the siblings.

Example 1: Strings, Relations and Trees as $\mathbf{K}_{\Sigma, \mathbf{A}}$ classes.

- (a) Let $\Sigma = \{0, 1\}$, $A = \emptyset$, $D = \{1, \dots, n\}$, $\text{Str} = \emptyset$, $\mathbf{K}_1 = \{I = (D, \text{Str}, \text{Child}, r, L, \lambda)\}$ where Child in the natural successor over D and $L : D \rightarrow \{0, 1\}$. This class represents binary words of length n . If $A' = \{A_1, A_2\}$, $\text{Str} = \{a, c\}$, we have binary words with two attributes, and values in Str . For example $1.0_{[A_1=a]}.1_{[A_1=c, A_2=c]}.0_{[A_1=a]}.1.0_{[A_1=a]}$ is a word where certain letters have attribute values. For example $L(2) = 0$ and $\lambda(2, A_1) = a$.

- (b) Let $\Sigma = \emptyset$, $A = \{A_1, A_2, A_3\}$, $D = \{1, \dots, n\}$, and Str an arbitrary set of string values, $\mathbf{K}_2 = \{I = (D, Str, Child, r, L, \lambda)\}$, such that $Child$ is the edge relation of an unranked tree of depth 1 whose leaves have attributes A_1, A_2, A_3 and values in Str . This class represents ternary relations with $n - 1$ tuples having values in Str .
- (c) Let $\Sigma = \{0, 1\}$, $D = \{d_1, \dots, d_n\}$, $A = \emptyset, Str = \emptyset$, $\mathbf{K}_3 = \{I = (D, Str, Firstchild, Nextsibling, L, \lambda)\}$. This class represents unranked ordered trees with n nodes without attributes.

We consider regular schemas, i.e. classes defined by automata. For unranked ordered trees, the automata have been defined by [27] and DTDs are special cases. XML schemas define regular properties of ordered and unordered trees.

2.1 Distances

The Edit distance has been introduced in [29] for comparing strings and generalized in [26] for trees. The Edit distance between two words is the minimal number of insertions, deletions and substitutions of a letter required to transform one word into the other. The best known algorithm for computing the edit distance between two words is in $O(n^2/\ln n)$ [20]), for the time complexity. Several approximation algorithms have been proposed but no algorithm achieves an approximation with a constant error for this distance. More information on approximation algorithms is given in section 2.3.

Consider an extension of the distance: the *edit distance with moves*, where arbitrary substrings can be moved in one step. Let the *Edit distance with moves* between two strings w and w' , written $\text{dist}(w, w')$, the minimal number of elementary operations (insertions, deletions, substitutions and moves) on w to obtain w' , divided by $\max\{|w|, |w'|\}$.

The Tree Edit distance consider basic operations on ordered labelled trees: *updating* a label or an attribute, *inserting* a node or an edge, and *deleting* a node or an edge. An attribute value is considered as a label: it can be entirely removed or added, at a unit cost, independent of the size of the attribute value. On ordered binary trees without labels, operations must preserve the binary tree and certain edge deletions are prohibited. The survey [3] describes hardness results for unordered trees and classical polynomial algorithms for ordered trees, for the distance problem.

In figure 2.1, we obtain the tree (b) from the tree (a) with a node insertion and the tree (c) from the tree (b) with an edge insertion.

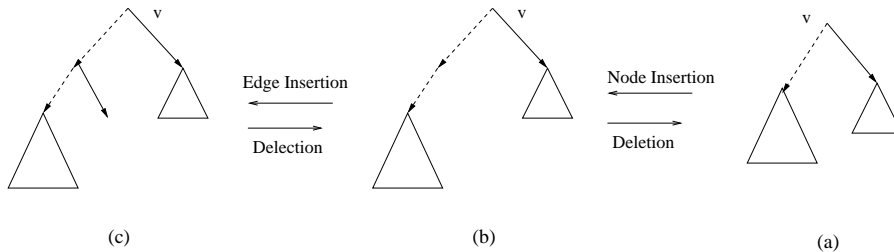


Figure 1: Tree-Edit operations for binary trees.

The *Tree Edit Distance with Moves* generalizes the Edit distance with a new operation, the Move of an entire subtree from one node to another node.

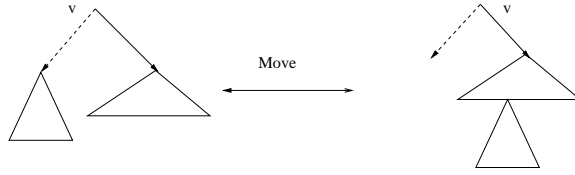


Figure 2: Move operation in a Tree.

Definition 1. . The Edit distance with moves between two ordered unranked trees T and T' , written $\text{dist}(T, T')$, is the minimal number of elementary operations on T to obtain T' , divided by $\max\{|T|, |T'|\}$. The distance between T and a language L , noted $\text{dist}(T, L)$ is the minimum distance between $\text{dist}(T, T')$ for $T' \in L$

We say that two trees T and T' are k -close if their distance is less than k . We naturally extend the Tree Edit distance (with moves) to forests, where the move operation can now either be applied to two subtrees of the forest or take one subtree and generate two new subtrees. Since the Tree Edit distance and the Tree Forest Edit distance are 4-equivalent (see Proposition ??), we do not distinguish them for the sake of simplicity. In other words, the Tree Forest Edit distance allows for some temporarily disconnection of complete subtrees.

2.2 Transductions

Two main approaches are used to specify transformations between a source instance and possible target instances. Let \mathcal{T} be a binary relation defined by some pairs of structures (I, J) where $I \in \mathbf{K}_S$ and $J \in \mathbf{K}_T$ for a target schema \mathbf{K}_T . The transformation \mathcal{T} can be defined by regular transductions or by formulas linking sources to targets.

A transduction transforms a source structure I into a target structure J in the language of the target. It does not change the basic structure of I but transforms the tags from the source language to tags of the target language. There is a large literature on tree transducers and our model is close to the top-down tree transducers of [19], but also handles attributes values.

Let \mathbf{K}_S be a set of (Σ_S, A_S) trees and \mathbf{K}_T be a set of (Σ_T, A_T) trees. A transducer associates in a top-down manner with a node v with label in Σ_S and attribute values along attributes in $A_S = \{A^1, \dots, A^k\}$, a local new finite (Σ_T, A_T) subtree and new attribute values for each node of that tree. In particular, a node v with attribute values $A_1 = a$ can generate a child node with label A_1 and data-value a , and conversely. This setting is motivated by the XSLT language where this feature is standard.

Let H_{Σ_T, A_T} be the set of finite sequences of *finite trees* (hedges) with attributes in A_T and values in $Str \cup Var$. Let $H_{\Sigma_T, A_T}[Q]$ be the set of finite sequences of finite trees where one leaf of each tree is a distinguished element labelled by a sequence of states in Q , which is possibly empty.

The transducer is defined by three functions. The function δ defines the local tree transformation at each node, the function h defines the transformation of attribute values (into possibly null values) and the *partial* function μ defines the positions of the new attribute values in the new finite tree t introduced by δ .

Definition 2. A tree transducer \mathcal{T} between (Σ_S, A_S) trees and (Σ_T, A_T) trees is defined by (Q, q_0, δ, h, μ) where:

- $\delta : \Sigma_S \times Q \rightarrow H_{\Sigma_T, A_T}[Q]$

- $h : \Sigma_S \times Q \times A_S \rightarrow \{1\} \cup Var$,
- $\mu : \Sigma_S \times Q \times A_T \times D_T \rightarrow \{1, 2, \dots, k\}$, where D_T is the set of nodes of the sequence of trees (hedge) defined by δ .

The function h extends to a function $h' : \Sigma_S \times Q \times Str \rightarrow Str \cup Var$ as follows. For label $l \in L_S$, state $q \in Q$, if $h(l, q, A_S^i) = 1$ then $h'(l, q, x_i) = x_i$. If $h(l, q, A_i) = V \in Var$ then $h'(l, q, x_i) = V$. Notice that this model is precisely what XSLT allows, but some attribute values may be kept in some state, i.e. when $h(l, q, A_S^i) = 1$, and assigned Null values in some other states.

A top-down run starts with the root node in state q_0 and transforms each node in a top-down manner. A node v with label $l \in L_S$, state q , attributes in A_S and attribute values in Str is replaced by a finite subtree with labels in L_T , attributes in A_T and attribute values in $Str \cup Var$, through the transformation $\mathcal{T}(l, q)$ defined by:

- $\delta(l, q) = (t_1, t_2, \dots, t_s)$ a set of finite trees with a distinguished leaf element labelled by a sequence of states. The trees t_i are inserted below the node v as siblings, as defined in [19], where duplications and deletions are allowed.
- Let v a node with attribute values $x_1, \dots, x_k \in Str$. The function h extends to a function h' which determines if the value is kept or assigned a Null value.
- If $\mu(l, q, A_T^i, w) = i$ then the value of the first attribute of node $w \in D_T$ is the value $h'(x_i)$. The function sets the value of the attribute of w as the image through h' , defined by h of the i -th value of the node v .

Notice the \mathcal{T} is a finite object as it only depends on finite trees and finitely many attributes. The set Str is not finite as it depends on arbitrary trees but the set Var of Null values is finite and determined by the labels and states.

At each node, we apply the transformation $\mathcal{T}(l, q)$ for label l and state q and we obtain a tree T' with labels in Σ_T , attributes in A_T , and attribute values in $Str \cup Var$. In the case of strings, if each $\delta(a, p) = u[q]$ where u is a finite word, we obtain the classical transducer which replaces in state p a letter a with the word u and goes to state q . The transducer is *linear* when no duplication is allowed.

Example 2 : transductions on strings and trees, with attributes.

- (a) Let $\Sigma_S = \{0, 1\}$, $A_S = \{N, M\}$, $D = \{1, \dots, n\}$, $\mathbf{K}_S = \{I = (D, Str, Child, r, L, \lambda)\}$ where Child is the natural successor over D and $L : D \rightarrow \{0, 1\}$, as in example 1 of binary words. Let $\Sigma_T = \{a, b, n\}$, $A_T = \{P\}$, and the corresponding \mathbf{K}_T , defined by the transducer (Q, q, δ, h, μ) :

- $Q = \{q\}$, $\delta(0, q) = n.d[q]$, $\delta(1, q) = b.d[q]$, i.e. words with only one successor,
- for all l, q , $h(l, q, M) = 1$, $h(l, q, N) = V_1$. Hence $h'(l, q, a) = a$, $h'(l, q, c) = V_1 \in Var$,
- μ sets the value of the attribute M on the node n of the word $n.d$ with the value $@M$ of the attribute M .

The image of the word on the label set $\{0, 1\}$, with attributes N, M defined by

$$1.0_{[N=a]} \cdot 1_{[N=c, M=c]} \cdot 0_{[M=c]} \cdot 1.0_{[N=a]}$$

is a word on the label set $\{a, b, n\}$ with attribute P and attribute values in $Str \cup Var$, i.e.

$$b.d.n_{[P=a]} \cdot d.b.d.n_{[P=V_1]} \cdot b.d.n.d$$

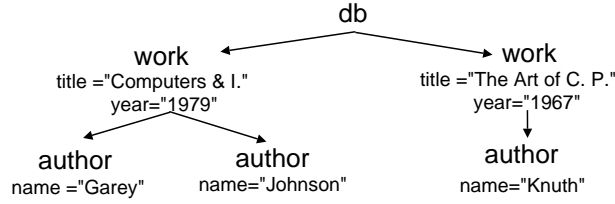


Figure 3: An ordered tree with attributes.

(b) Let I be Source Instance described by:

Let \mathcal{T} be the transducer given by the XSLT program :

```

<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
  <bib>
    <xsl:apply-templates/>
  </bib>
</xsl:template>
<xsl:template match="work">
  <livre>
    <xsl:apply-templates/>
    <titre>
      <xsl:value-of select="@title" />
    </titre>
  </livre>
</xsl:template>
<xsl:template match="author">
  <auteur>
    <xsl:value-of select="@name" />
  </auteur>
</xsl:template>

```

The transducer \mathcal{T} is defined by: $\Sigma_S = \{db, work, author\}$, $A_S = \{title, name\}$, $\Sigma_{ST} = \{bib, livre, auteur\}$, $A_T = \{name\}$.

$Q = \{q\}$, $\delta(db, q) = bib[q]$, $\delta(work, q) = (t_1, t_2)$, where $t_1 = livre[q]$ and $t_2 = titre$, $\delta(author, q) = auteur$, $h(work, q, title) = 1$, $h(author, q, title) = 1$. The function μ sets the attribute value of *title* as default attribute (PCDATA) to the node *titre* of the subtree t_2 , and the attribute value of *author* as default attribute (PCDATA) to the node *auteur* of $\delta(author, q)$.

After the XSLT transduction the structure is:

In practice, the transducer is deterministic and defined by an XSLT program π . It tries to translate tags, modify attributes names and values to fit a target schema. In data exchange [14, 4], there are other models to express relationships between source and target schemas, and there is a close connection between Approximate Query Answering and Approximate Data Exchange.

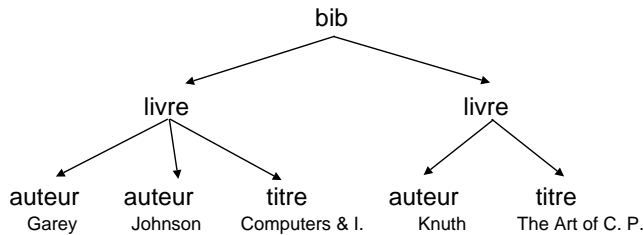


Figure 4: The result of the transducer.

2.3 Property Testing and Approximation

Inspired by the notion of Self-Testing [7, 8], Property Testing has been initially defined in [23] and studied for graph properties [16] in conjunction with learning methods. It has been successfully extended to various classes of finite structures, such as words where regular languages are proved testable [2, 1] for the Hamming distance, and trees where regular tree languages are proved testable [18] for the Edit Distance with Moves. A tester approximates a property by looking at a constant fraction of the input, independent of the global size of the input.

We say that two structures $U_n, V_m \in \mathbf{K}$, whose domains are respectively of size n and m , are ε -close if their distance $\text{dist}(U_n, V_m)$ is less than $\varepsilon \times \max(n, m)$. They are ε -far if they are not ε -close. The distance of a structure U_n to a class \mathbf{K} is $\text{dist}(U_n, \mathbf{K}) = \text{Min}_{V \in \mathbf{K}} \{\text{dist}(U_n, V)\}$. In this paper, we consider this notion of closeness for words and trees since the representation of their structure is of linear size. For other classes, such as graphs, one may define the closeness relatively to the representation size (e.g. εn^2 for graphs) instead of the domain size.

Definition 3. Let $\varepsilon \geq 0$ be a real. An ε -tester for a class $\mathbf{K}_0 \subseteq \mathbf{K}$ is a randomized algorithm A such that:

- (1) If $U \in \mathbf{K}_0$, A always accepts;
- (2) If U is ε -far from \mathbf{K}_0 , then $\text{Pr}[A \text{ rejects}] \geq 2/3$.

The *query complexity* is the number of boolean queries to the structure U of \mathbf{K} . The *time complexity* is the usual time complexity where the complexity of a query is one and the time complexity of an arithmetic operation is also one. A class $\mathbf{K}_0 \subseteq \mathbf{K}$ is *testable* if for all $\varepsilon > 0$, there exists an ε -tester whose time complexity depends only on ε .

Definition 4. An ε -corrector for a class $\mathbf{K}_0 \subseteq \mathbf{K}$ is a (randomized) algorithm A which takes as input a structure I which is ε -close to \mathbf{K}_0 and outputs (with high probability) a structure $I' \in \mathbf{K}_0$, such that I' is ε -close to I .

When an XML file is given by its DOM representation, the operations associated with the Edit distance with moves take unit costs. The move requires only the modifications of a few pointers. Testers for regular properties of words and trees have been presented in [18, 15] for this distance and a corrector for regular trees presented in [10].

2.3.1 Approximate Schemas

We first consider classes of structures on the same alphabet Σ and attributes A .

Definition 5. Let $\varepsilon \geq 0$. Let $\mathbf{K}_1, \mathbf{K}_2$ be two classes of structures. We say that \mathbf{K}_1 is ε -contained in \mathbf{K}_2 , if all but finitely many words of \mathbf{K}_1 are ε -close to \mathbf{K}_2 . \mathbf{K}_1 is ε -equivalent, written \equiv_ε , to \mathbf{K}_2 , if both \mathbf{K}_1 is ε -contained in \mathbf{K}_2 and \mathbf{K}_2 is ε -contained in \mathbf{K}_1 .

Notice that the image of regular properties on strings and trees are also regular, so we need to efficiently distinguish approximately equivalent schemas.

Example 3 :

(a) Let $\mathbf{K}_1 = O^*1^*$ and $\mathbf{K}_2 = c(ab)^*ca^*$ be two regular expressions. There is a transducer with one state which replaces the letter 0 by ab and the letter 1 by a . The transducer \mathcal{T} is specified by $0|ab$ and $1|a$. The image of O^*1^* by \mathcal{T} is $\mathcal{T}(O^*1^*) = (ab)^*a^*$, which is ε -close to $c(ab)^*ca^*$ for any ε . Any word $w \in (ab)^*a^*$ of length n is at distance $2/n$ from a word of $c(ab)^*ca^*$, as two insertions of c are required.

(b) Let $\mathbf{K}_S^1 =$

```
<!ELEMENT db (work*)>
<!ELEMENT work (author*)>
  <!ATTLIST work
    title CDATA #REQUIRED
    year CDATA>
<!ELEMENT author (EMPTY)>
  <!ATTLIST author
    name CDATA #REQUIRED>
```

and $\mathbf{K}_T^1 =$

```
<!ELEMENT bib (livre*)>
<!ELEMENT livre (auteur+, titre , annee)>
<!ELEMENT auteur #PCDATA>
<!ELEMENT titre #PCDATA>
<!ELEMENT annee #PCDATA>
```

be two DTDs, special regular schemas on trees. If we use the the transducer of example 2, then $\mathcal{T}(\mathbf{K}_S^1)$ is also ε -close to \mathbf{K}_T^1 .

A more general definition of closeness, called *similarity* considers structures in different languages and allows for a transducer to compare the schemas.

Definition 6. Let $\varepsilon \geq 0$ and $\mathbf{K}_1, \mathbf{K}_2$ be two classes of structures. We say that \mathbf{K}_1 is ε -included in \mathbf{K}_2 , if there exists a transducer \mathcal{T} such that all but finitely many words of $\mathcal{T}(\mathbf{K}_1)$ are ε -close to \mathbf{K}_2 . \mathbf{K}_1 is ε -similar, written \simeq_ε , to \mathbf{K}_2 , if both \mathbf{K}_1 is ε -included in \mathbf{K}_2 and \mathbf{K}_2 is ε -included in \mathbf{K}_1 .

This definition is closer to the situation where documents involve tags in different languages. Notice that the image of regular properties on strings and trees are also regular, so we need to efficiently distinguish approximately equivalent schemas.

2.3.2 Additive approximation of the distance.

There is a rich connection between approximation algorithms and property testing. For the classical Edit distance, [6] constructs a sublinear tester such that it accepts with high probability pairs of words of size n that are at edit distance $O(n^\alpha)$, for some $0 < \alpha < 1$, and rejects with high probability pairs of words

at distance $\Omega(n)$. The running time is in $\tilde{O}(n^{\max(\alpha/2, 2\alpha-1)})$. These testers can be understood as a *weak approximation* of the edit distance, since it leads to efficient approximation algorithms whenever the distance is large [22]. Nonetheless no sublinear time testers are known for the case $\alpha = 1$. Moreover, there is no hope to get a tester whose running time is independent of the input size (even for $0 < \alpha < 1$) since a lower bound of $\Omega(n^{\alpha/2})$ on the query complexity has been proven [6]. Results of the same kind were proven in the sketching model [5], where a sketch (or a fingerprint) is associated to each string, which is a succinct yet rich enough description to approximate the edit distance. Nonetheless the complexity of computing a sketch is usually not sublinear. For instance, a near linear time algorithm is constructed to distinguish between strings at distance $O(k)$ and strings at distance $\Omega(k^2)$, for any $k \geq 1$.

The edit distance with moves has been considered and approximated efficiently in [12], where a near linear time algorithm has been given for approximating it within a $\tilde{O}(\ln n)$ factor. Whereas computing the Edit distance with moves on strings is NP-hard [24], it can be approximated within an $\tilde{O}(\ln n)$ factor only in near linear time [12]. If one allows other operations in the distance such as copying subwords, then there is a linear time and constant approximation algorithm [13, 25]. In the context of property testing, the Edit distance with moves has been used in [18] for testing regular languages, where the tester is more efficient and simpler than the one of [2], and can be generalized to tree regular languages. To compute the Edit distance with moves on ordered trees is also NP-hard, yet easy to approximate.

The connection between Property Testing and Approximation is captured by these two definitions.

Definition 7 (Tolerant tester [22]). *Let $0 < \varepsilon_1 < \varepsilon_2$ be reals. A tolerant $(\varepsilon_1, \varepsilon_2)$ -tester for a language L is a randomized algorithm A such that, for any word w as input:*

- (1) *If w is ε_1 -close to L , then A accepts with probability at least $2/3$;*
- (2) *If w is ε_2 -far from L , then A rejects with probability at least $2/3$.*

Definition 8 (Approximation). *Let $\alpha, \beta : \mathbb{R} \rightarrow \mathbb{R}$. An (α, β) -approximation of a real function f is a randomized algorithm that, for any word w as input, outputs a value z such that $\Pr[\alpha(f(w)) \leq z \leq \beta(f(w))] \geq 2/3$.*

In this case, a tolerant $(\varepsilon_1, \varepsilon_2)$ -tester implies an $(\varepsilon_1, \varepsilon_2)$ -approximation of the distance.

2.3.3 A statistical embedding on strings.

For a finite alphabet Σ and a given ε , let $k = \frac{1}{\varepsilon}$. Let the $\text{dist}(w, w')$ be the *Edit distance with moves* as defined in Example 3 and the embedding of a word in a vector of k -statistics, describing the number of occurrences of all subwords of length k in w . For all words u of length k , let $\#u$ be the number of occurrences of u in w and:

$$\mathbf{u}\text{-stat}(w)[u] \stackrel{\text{def}}{=} \frac{\#u}{n-k+1}$$

The vector $\mathbf{u}\text{-stat}(w)$ is of dimension $|\Sigma|^k$ is also the *probability distribution* that a uniform random subword of size k of w be a specific u , i.e.

$$\mathbf{u}\text{-stat}(w)[u] = \Pr_{j=1, \dots, n-k+1} [w[j]w[j+1] \dots w[j+k-1] = u]$$

This embedding is a generalized Parikh mapping [21] and also related to the previous work of [11], where the subwords of length k were called *shingles*.

The statistical embedding [15] associates a statistics vector $\mathbf{u}\text{-stat}(w)$ with a string w and a union of polytopes H in the same space to a regular expression r , such that the distance between two vectors (for the L_1 norm)

is approximately $\text{dist}(w, w')$ and the distance between a vector and a union of polytopes is approximately $\text{dist}(w, L(r))$.

Let $w = 000111$ of length $n = 6$, $k = 2$, $n - k + 1 = 5$. For a lexicographic enumeration of the length 2 binary words, the $\mathbf{u}\text{-stat}$ vector associated with the word w is:

$$\mathbf{u}\text{-stat}(w) = \begin{pmatrix} 2/5 \\ 1/5 \\ 0 \\ 2/5 \end{pmatrix}$$

The number of occurrences of $u = 00$, $u = 01$, $u = 10$ and $u = 11$ are respectively 2, 1, 0, 2.

$$\text{Let } s_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, s_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, s_2 = \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \\ 0 \end{pmatrix}$$

Consider the regular expression $r_1 = 0^*1^*$, $r_2 = (001)^*1^*$, $r_3 = 0^*(001)^*1^*$, $k = 2$. The convex hull associated with r_1 is $H_1 = \text{Convex - Hull}(s_0, s_1)$, the one associated with r_2 is $H_2 = \text{Convex - Hull}(s_1, s_2)$ and the one associated with r_3 is $H_3 = \text{Convex - Hull}(s_0, s_1, s_2)$.

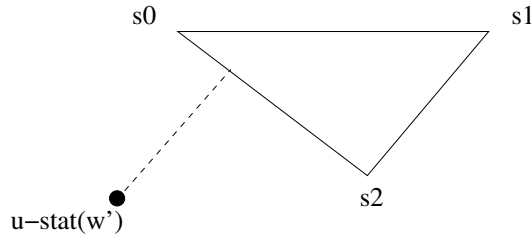


Figure 5: The representation of the polytope H_3 in the space of vectors of size $2^k = 2^2$.

These techniques yield:

- Equality tester between two words w and w' of approximately the same length. Sample the words with at least $N \in O\left(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^3}\right)$ samples, define $\widehat{\mathbf{u}\text{-stat}}(w)$ and $\widehat{\mathbf{u}\text{-stat}}(w')$ as the $\mathbf{u}\text{-stat}$ of the samples. Reject if $|\widehat{\mathbf{u}\text{-stat}}(w) - \widehat{\mathbf{u}\text{-stat}}(w')|_1 \geq \varepsilon$.
- Membership tester between a word and a regular expressions r . Compute $\widehat{\mathbf{u}\text{-stat}}(w)$ as before and the polytope H associated with r in the same space. Reject if the geometrical distance from the point $\widehat{\mathbf{u}\text{-stat}}(w)$ to the polytope H is greater than ε .
- Equivalence tester between two regular expressions r_1 and r_2 . Associate the polytopes H_1 and H_2 in the same space as $\mathbf{u}\text{-stat}(w)$, represented by the nodes $H_{1,\varepsilon}$ and $H_{2,\varepsilon}$ on a grid of step ε . If $H_{1,\varepsilon} \neq H_{2,\varepsilon}$ then r_1 and r_2 are ε far.

The membership tester is polynomial in the size of the regular expression (or non-deterministic automaton) whereas it was exponential in this parameter in [18]. In this paper, we use the Membership tester and the Equivalence tester.

2.4 Learning

Consider the following problems for strings:

- *Distance problem.* Given a regular expression r and a word w in the same language, compute the distance $\text{dist}(w, r)$,
- *Supervised learning.* Given m regular expressions r_1, \dots, r_k and a word w in the same language, decide which regular expression w is closer to.
- *Unsupervised learning.* Given words w_1, \dots, w_k taken from a distribution \mathcal{D} labelled by $+$ or $-$, i.e. positive and negative examples, build a regular expression which minimizes an error criterion.

The statistical embedding gives an approximate solution to the *Distance problem* and to the *Supervised learning* for the case where w, r_1, \dots, r_k use the same language, i.e. \mathcal{T} is the identity. Simply plot the input in the statistics space, of dimension $|\Sigma|^{1/\varepsilon}$. Estimate $\text{u-stat}(w)$ with N samples which only depends on ε , compute the polytopes $H_{i,\varepsilon}$ associated with the regular expression r_i and precision ε . The distances are the geometrical distances, which are easy to compute. We show in section 5, how to approximate the *Unsupervised learning* for the PAC-model [28] and in certain cases for the distance-model.

The PAC-learning model, introduced by Valiant [28] is a framework to approximately learn an unknown boolean function f in a class \mathcal{F} . The model supposes positive and negative samples along a distribution \mathcal{D} , i.e. values $x_i, f(x_i)$ for $i = 1, 2, \dots, N$. The learning algorithm proposes a function h and the error between f and h along the distribution \mathcal{D} is

$$\text{error}(h) = \Pr_{x \in \mathcal{D}} [f(x) \neq h(x)]$$

A class \mathcal{F} of functions f is PAC-learnable if there is a randomized algorithm such that for all $f \in \mathcal{F}, \varepsilon, \delta, \mathcal{D}$, which produces with probability greater than $1 - \delta$ an estimate h for f such that $\text{error}(h) \leq \varepsilon$. It is efficiently PAC learnable if the algorithm is polynomial in $N, \frac{1}{\varepsilon}, \frac{1}{\delta}, \text{size}(f)$, where $\text{size}(f)$ is the size of the finite representation of f . The Vapnik-Chernovenkis (VC) dimension of a class \mathcal{F} , denoted $VC(\mathcal{F})$ is the largest cardinality d of a sample set that is shattered by \mathcal{F} , such that every subset can be separated from its complement by $f \in \mathcal{F}$. A classical result of [9, 17] is that if d is finite then the class is PAC learnable. If $N \geq O(\frac{1}{\varepsilon} \cdot \log \frac{1}{\delta} + \frac{d}{\varepsilon} \cdot \log \frac{1}{\varepsilon})$, then any h which is consistent with the samples is a good estimate. We apply these techniques show that the VC dimension of a finite union of polytopes is finite and conclude that a regular expression is PAC-learnable with just the statistics of the samples.

We can introduce a more general distance problem, adapted to the situation when the word w and the regular expressions do not use the same language. Say that $\text{dist}'(w, r) \leq \varepsilon$ if there is a transducer \mathcal{T} such that $\text{dist}(\mathcal{T}(w), r) \leq \varepsilon$. The unsupervised learning can be generalized in a similar way. For each regular expression r_i , we allow a transducer \mathcal{T}^i , estimate $\text{u-stat}(\mathcal{T}^i(w))$ and compare it with the polytope associated with r_i . In this sense, we extend the learning problems, which generalize to regular trees defined by DTDs.

3 Query answering on words

Approximate Query answering is the decision problem which takes ε , a word w , a regular expression r , and a transducer \mathcal{T} as input and decides:

- (1) If $\mathcal{T}(w) \in r$ or if
- (2) If w is ε -far from any w' such that $\mathcal{T}(w')$ is ε -close to r .

An ε -tester for Approximate Query answering is a randomized algorithm which decides with high probability the conditions 1 and 2. For simplicity, we consider a transducer as a finite state machine.

Example 4: Let \mathcal{T} , \mathbf{K}_2 be defined as in example 3(a). For a given instance $I = 0001111$, $\mathcal{T}^1(I) = ababab.aaaa$ is at distance $\frac{1}{5}$ from the target schema $c(ab)^*ca^*$. A corrector for \mathbf{K}_2 will transform $ababab.aaaa$ into $c.ababab.c.aaaa$ in linear time.

3.1 Tester for Approximate Query Answering

We present an ε -Tester for Approximate Query Answering, first for the case of a transducer with one state, and generalize it in a second step. Let \mathbf{K} be the regular expression r on the alphabet Σ , for which we build its equivalent polytope $H(\mathbf{K})$.

The Tester follows the following approach. It samples $I = w$ to obtain a random subword u as if it wanted to estimate $\text{u-stat}(I)$. Instead we look at $\mathcal{T}(u)$ and obtain a subword v of $\mathcal{T}(I)$, on which we will construct an approximate statistics. As the transducer produces words of different lengths, we have to adjust the sampling probabilities to guarantee the approximation.

3.1.1 Transducer with one state

Let $\alpha = \min_{a \in \Sigma_s} |\mathcal{T}(a)| > 0$, $\beta = LCM_{a \in \Sigma_s} |\mathcal{T}(a)|$, i.e. the Least Common Multiplier of the lengths $\mathcal{T}(a)$ and $k = 1/\varepsilon$.

Tester₁(w, k, N) :

1. Repeat until N outputs are generated.
 - { Choose $i \in_r \{1, \dots, n\}$ uniformly,
 - let $a = w[i]$ and $\gamma = |\mathcal{T}(a)|$,
 - choose $b \in_r \{0, 1\}$ with $Prob(b = 1) = \frac{\gamma}{\beta}$,
 - If (b=1) { Choose $j \in_r \{0, \dots, \gamma - 1\}$ uniformly and
 - output the subword of length k
 - beginning at position j in $\mathcal{T}(a)$ and
 - continuing with k letters on the right side }
 - }
2. If the geometrical distance between $\widehat{\text{u-stat}}$ and $H(\mathbf{K})$ is greater than ε then reject else accept.

Let $N_0 = O(\beta \ln(1/\varepsilon) \ln(|\Sigma|)/(\alpha \varepsilon^3))$.

Lemma 1. For any w, \mathcal{T} and regular \mathbf{K}_T , $\varepsilon > 0$ and $N \geq N_0$, $\text{Tester}_1(w, k, N)$ is an ε -tester for Approximate Query Answering.

Proof. If w is consistent, the tester accepts with high probability.

Let's focus on the second condition in the definition of an ε -tester. Let $|w| = n$ and X be a successful output with indices i and j i.e. a subword $X_{i,j}$ of length k . For $u \in |\Sigma_T|^k$ we define

$$\widehat{\text{u-stat}}(w, k, N)[u] = \sum_{i=1}^N \frac{\#u}{N}, \quad u \in \{X_{i,j}\}$$

Any couple (i, j) is chosen with probability $\frac{1}{n\beta}$, hence all $X_{i,j}$ are obtained with equal probability, hence the expectation of \widehat{ustat} is $\mathbf{u-stat}(\mathcal{T}(w))$. We can apply a Chernoff bound to each component u :

$$Pr(|\widehat{ustat}(w, k, N)[u] - \mathbf{u-stat}(\mathcal{T}(w))[u]| > \varepsilon) \leq 2 \exp(-2\varepsilon^2 N)$$

We apply a union bound to all components and replace N by its lower bound:

$$Pr(|\widehat{ustat}(w, k, N) - \mathbf{u-stat}(\mathcal{T}(w))| > \varepsilon) \leq 1/6$$

If we repeat $\Theta(N_0 \cdot \beta \ln(1/\delta)/\alpha)$ times the outer loop (1), we obtain N samples with probability greater than $1 - \delta$. For a single choice of i , the probability to output a sample is greater than α/β and the repetition gives the expected bound. The tester fails with a probability less than δ . \square

3.1.2 General Transducer

Let \mathcal{A} be a deterministic transducer with m states. We generalize the previous tester, and sample subwords u of w which yield subwords of $\mathcal{T}(w)$. We do not know however the state q in which the transducer is, on sample u . We will consider all the possible states $S_u \subseteq Q$ but need to make sure that the possible states of the samples, i.e. S_{u_1}, \dots, S_{u_N} are compatible, i.e. can be obtained by paths which meet only a set Π of connected components of the automaton. We will enumerate all possible such Π in the acyclic graph defined by the connected components, and apply a Tester which tests if w is close to a word w' such that a run on w' starting in some state of Π meets only the connected components of Π .

A connected component of \mathcal{A} is a set S of states such that for each pair (s, s') of states of S there is a path from s to s' in \mathcal{A} . Let \mathcal{G} be the directed acyclic graph (DAG) associated with the connected components of \mathcal{A} , i.e. a node is a connected component C_i and there is an edge from C_i to C_j if there is a path in \mathcal{A} from a state $s \in C_i$ to a state $s' \in C_j$.

Definition 9. A set Π of connected components is admissible for \mathcal{A} if there exists a word x and a state q in one of the connected components of Π such that the run from q on x meet exactly the connected components of Π . Such a word x is called Π -compatible from q .

A word w is ε -source consistent along Π , if there is an ε -close w' which is Π -compatible from the initial state q_0 such that $\mathcal{T}(w')$ is ε -close to \mathbf{K} . Let H_Π be the polytope associated to the automaton reduced to Π .

Tester₂ (w, k, N, Π) :

Generate $u_1 \dots u_N$ words of length k in w .

Estimate $\widehat{\mathbf{u-stat}}(w)$: if it is ε -far from H_Π , reject.

If it is ε -close to H_Π , associate a set of states S_{u_i} with each u_i from which u_i is Π -compatible.

Apply **Tester₁** (w, k, N) for each possible states of S_{u_i} , as $\mathcal{T}(u_i)$ is well defined.

Accept if there is choice of states such that **Tester₁** (w, k, N) accepts, else reject.

Let $N_0 = O(\beta \ln(1/\varepsilon) \ln(|\Sigma_T|)/(\alpha\varepsilon^3))$.

Lemma 2. For $N \geq N_0$, $\varepsilon > 0$,

Tester₂ (w, k, N, Π) is an ε -tester which decides if a source word w is ε -source consistent along Π .

Proof. If w is such that $\mathcal{T}(w) \in H_{\mathbf{K}}$, for a Π -admissible run, then $\widehat{\mathbf{u-stat}}(w) \in H_\Pi$, any random u in w has a state in Π and its image is a word of $\mathcal{T}(w)$. Hence the \widehat{ustat} computed by tester 1 is in $H_{\mathbf{K}}$.

If w is ε -far from all Π -compatible w' such that $\mathcal{T}(w')$ is ε -close to $H_{\mathbf{K}}$, then the Tester rejects with high probability. If it accepted, there would be states in the connected components of Π such that \widehat{ustat} is ε -close to $H_{\mathbf{K}}$. In this case there would be a Π -compatible w' whose statistics is ε -close to w , contradicting the hypothesis. \square

Tester₃(w, k, N) :

Generate all Π and apply **Tester₂**(w, k, N, Π).

If there is a Π such that **Tester₂**(w, k, N, Π) accepts, then accept, else reject.

Theorem 1. *If $N \geq \beta \ln(1/\varepsilon) \ln(|\Sigma_{\mathcal{T}}|)/(\alpha\varepsilon^3)$, $\varepsilon > 0$, **Tester₃**(w, k, N) is an ε -tester which decides Approximate Query Answering.*

Proof. If w is such that $\mathcal{T}(w) \in H_{\mathbf{K}}$, a run on w is admissible for some Π . Any random u has a state in Π and its image is a word of $\mathcal{T}(w)$. Hence the $\widehat{ustat} \in H_{\mathbf{K}}$.

If w is ε -far from all w' such that $\mathcal{T}(w')$ is ε -close to $H_{\mathbf{K}}$, then the Tester rejects with high probability. If it accepted, there would be a Π and states in the connected components of Π such that \widehat{ustat} is ε -close to $H_{\mathbf{K}}$. In this case there would be a w' whose statistics is ε -close to w which is Π -compatible and it would contradict the previous lemma. \square

3.2 Corrector

The decision of Approximate Query Answering leaves the possibility that w is ε -close to a w' such that $\mathcal{T}(w')$ is ε -close to r . In this case, a corrector introduced in [10], finds such a w' . It is a valuable tool which generalizes error-correction, relative to a schema.

In a typical practical case, the input w belongs to a source schema $\mathbf{K}_{\mathbf{S}}$ which is different from the target schema $\mathbf{K}_{\mathbf{T}}$. If we fix a transducer \mathcal{T} , the image of a regular schema $\mathbf{K}_{\mathbf{S}}$, i.e. $\mathcal{T}(\mathbf{K}_{\mathbf{S}})$ is also regular. We can use the polynomial time algorithm given in [15] to decide if $\mathcal{T}(\mathbf{K}_{\mathbf{S}})$ and $\mathbf{K}_{\mathbf{T}}$ are ε close because they use the same symbols. If these two schemas are close, we know that $w \in \mathbf{K}_{\mathbf{S}}$ has always an image close to $\mathbf{K}_{\mathbf{T}}$.

4 Query answering on trees

Approximate Query answering generalizes to trees, and the input is ε , a tree T , a regular schema \mathbf{K} , and a transducer \mathcal{T} as input. We present an ε -Tester for Approximate Query Answering on trees in the case of a linear transducer with one state.

The Tester follows an approach, similar to the one presented for strings and uses the Membership ε -Tester presented in [18]. It samples the instance $I = T$ to obtain a random subtree t with a uniform distribution. We look at $\mathcal{T}(t)$ and obtain a subtree t' of $\mathcal{T}(I)$. As the transducer produces trees of different sizes, we have to adjust the sampling probabilities to guarantee the near-uniform distribution of t' . A random subtree of size k in an unranked tree is defined through the encoding of an unranked tree as an extended 2-ranked tree.

Every unranked tree T can be coded as a binary tree $e(T)$. Consider the classical encoding where each node v of the unranked tree is a node v in the binary encoding, the left successor of v in the binary tree is its first successor in the unranked tree, the right successor of v in the binary tree is its first sibling in the unranked tree. New nodes with labels \perp are added to complete the binary tree when there are no successor or no sibling in the unranked tree. If we remove the leaves labelled with \perp , we consider trees with degree at most 2, where nodes may only have a left or a right successor and call these structures *extended 2-ranked trees*.

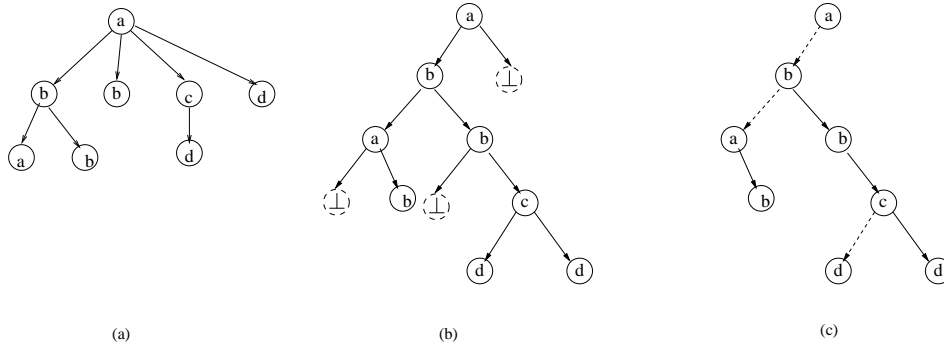


Figure 6: Encoding of an unranked tree in (a) as a binary tree with \perp in (b) and as an extended 2-ranked tree in (c).

The advantage of this representation is that the extended 2-ranked tree has the *same set of nodes* as the unranked tree. It allows to define the notion of a *random subtree of size k* in an unranked tree.

Definition 10. A *random subtree of size k* in an extended 2-ranked tree is the subtree obtained from a uniformly selected random node v , taking a breadth-first subtree below v of size k . A *random subtree of size k* in an unranked tree T is the subgraph of T restricted to the nodes of $e(T)$ of a random subtree of size k of $e(T)$.

Notice that a random subtree in an unranked tree is not a tree, but a hedge. A random subtree of size 3 of the node b second successor of the root in figure 6 consists in the node b with its next sibling c and its successor d .

Let n be the number of nodes of the input tree T , and (Q, q, δ, h, μ) be a transducer with one state q . Let $\alpha = \min_{a \in \Sigma_s} |\delta(a, q)| > 0$, $\beta = LCM_{a \in \Sigma_s} |\delta(a, q)|$, i.e. the Least Common Multiplier of the lengths $\delta(a, q)$ and $k = 1/\varepsilon$. Let $\text{Tester}_{\text{regular}}(t_1, \dots, t_N)$ be the Tester for the regular tree language \mathbf{K} with random samples t_1, \dots, t_N , introduced in [18].

Tester (T, k, N) :

1. Repeat until N outputs are generated.
 - { Choose $i \in_r \{1, \dots, n\}$ uniformly,
 - let $v = T[i]$ a random node with label a and $\gamma = |\delta(a, q)|$,
 - choose $b \in_r \{0, 1\}$ which $\text{Prob}(b = 1) = \frac{\gamma}{\beta}$,
 - If $(b=1)$ { Choose $j \in_r \{0, \dots, \gamma - 1\}$ uniformly and
 - output the subtree t of size k
 - with root j in $\delta(a, q)$ and
 - continuing with k nodes below in $e(T)$ }
 - }
2. If $\text{Tester}_{\text{regular}}(t_1, \dots, t_N)$ rejects then reject else accept.

If m is the size of the DTD of $\mathbf{K}_{\mathbf{T}}$, $\text{Tester}_{\text{regular}}$ has query complexity $O(mr^{4m+3}2^{(2^{2m+1})/\varepsilon}/\varepsilon^2)$, and time complexity in $O(2^m m \cdot 2^{4m+3} 2^{(2^{2m+1})/\varepsilon}/\varepsilon^2)$.

Let $N_0 = O(mr^{4m+3}2^{(2^{2m+1})/\varepsilon}/\varepsilon^2)$.

Lemma 3. For any T, \mathcal{T} , regular \mathbf{K} , and $\varepsilon > 0$, if $N \geq N_0$, $\text{Tester}(T, k, N)$ is an ε -tester for Approximate Query Answering.

Proof. If T is consistent, the tester accepts with high probability because the samples t_1, \dots, t_N are taken with a distribution close to the uniform distribution. If T is ε -far then $\text{Tester}_{\text{regular}}(t_1, \dots, t_N)$ rejects with high probability for the same reason. If we repeat $\Theta(N_0 \cdot \beta \ln(1/\delta)/\alpha)$ times the outer loop (1), we obtain N samples with probability greater than $1 - \delta$. For a single choice of i , the probability to output a sample is greater than α/β and the repetition gives the expected bound. The tester fails with a probability less than δ . \square

If \mathcal{T} is a deterministic transducer with m states, we can generalize the previous tester, using sets Π of connected components of the automaton, as in the case of strings, and obtain a general $\text{Tester}_{\text{tree}}(T, k, N)$.

Theorem 2. *For $N \geq N_0$, $\varepsilon > 0$, $\text{Tester}_{\text{tree}}(T, k, N)$ is an ε -tester which decides Approximate Query Answering.*

5 Learning

We consider the *Unsupervised learning* problem on words, using similar techniques. Given a non-deterministic finite automaton, the statistical embedding [15] associates a union of polytopes in the space of the *u.stat* vectors, i.e. in dimension $|\Sigma|^{\frac{1}{\varepsilon}}$. It has M nodes, as many as small loops of size $\frac{1}{\varepsilon}$, i.e. at most $|\Sigma|^{\frac{1}{\varepsilon}}$ in the worst case. The VC dimension of a convex polygon in dimension 2 is $2d + 1$. In dimension n , it is bounded by $w \cdot \log w$ if w is the number of faces of the polytope, which is less than $n \cdot M$. Hence the VC dimension is less than $n \cdot M \log(n \cdot M)$.

Learning Algorithm A:

Input: m, N samples $(x_1, f(x_1)), \dots, (x_N, f(x_N))$ where $f(x_i) = 1$ if $x_i \in L$ the unknown regular language, and $f(x_i) = 0$ if $x_i \notin L$.

Output: H , a union of at most m polytopes.

 Compute the *u-stat* of each x_i .

 Take the largest convex hull (in the *u-stat* space) which is consistent with the samples, starting from a random point *u-stat*(x_i).

 Create a new convex hull for the point not covered, while the number of convex hulls is less than m .

This algorithm outputs a union of at most m disjoint polytopes, and can be refined to allow the polytopes to intersect. It could equivalently produce an automaton whose set of polytopes is H .

Let \mathcal{A} be an automaton with m states and let its statistical embedding has M nodes. We know that $M \leq |\Sigma|^{\frac{1}{\varepsilon}} = n$.

Theorem 3. *Given $\mathcal{A}, \varepsilon, \delta$, if N samples $(x_i, f(x_i))$ are provided according to a distribution \mathcal{D} and if $N \geq c(\frac{1}{\varepsilon} \cdot \log \frac{1}{\delta} + \frac{2^m(n \cdot M \log(n \cdot M))}{\varepsilon} \cdot \log \frac{1}{\varepsilon})$ then the algorithm A provides with probability greater than $1 - \delta$ an approximate H which is PAC ε -close to f .*

Proof. The number of distinct polytopes generated by the statistical embedding is the number of compatible small loops from given states and is less than 2^m . The VC dimension is less than $2^m(n \cdot M \log(n \cdot M))$. The algorithm A is consistent with the samples. We apply the general result on PAC learning and conclude that A provides with probability greater than $1 - \delta$ an approximate h which is PAC ε -close to f . \square

The estimated set of convex hulls h is PAC ε -close to f , i.e.

$$\text{error}_A(h) = \Pr_{x \in \mathcal{D}} [f(x) \neq H(x)] \leq \varepsilon$$

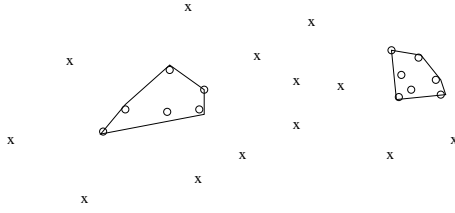


Figure 7: Positive points (o) and Negative points (x) in the statistical space. The learning algorithm computes the largest consistent convex-hulls.

This learning algorithm may not be entirely satisfactory, if we take the ε distance between automata taken in this paper, which is independent of a sampling distribution. It is possible that a distribution misses an entire polytope, i.e. never gives positive samples in a specific polytope, in which case the estimated h is PAC-close to f but far for the Edit distance with Moves. The PAC-error is the volume of the difference whereas the Distance-error is the largest value between two separated points.

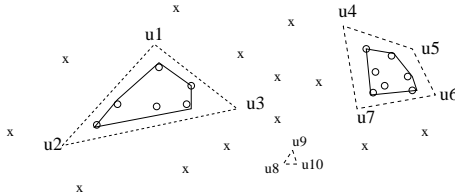


Figure 8: The PAC and the distance error

In the example of figure 8, the distance error is large as the small polytope has not been sampled. If the polytopes are large, then the uniform distribution will sample in every polytope with high probability and the learning algorithm will guarantee a small distance between the estimate H and the unknown polytope H' . We can also take the expectation of the distance, i.e.

$$distance - error_A(h) = E_{x \in \mathcal{D}^+} [Min_{x' \in H} \text{dist}(x, x')]$$

where \mathcal{D}^+ is the distribution of positive examples.

6 Conclusion

We introduced a framework to compare schemas for the *Edit distance with Moves* and consider two different problems: Approximate Query Answering and Unsupervised Learning. We gave efficient solutions for these two problems on words using the same statistical embedding which allows to represent a string w with a finite statistical representation, which only samples w .

This embedding gives a compressed representation of a word, which is enough to approximately decides any regular property. To learn a property, it is enough to look at this statistical representation of the positive and negative samples. Similarly, we gave a tester for Approximate Query Answering on trees.

References

- [1] N. Alon, E. Fischer, M. Krivelevich, and M. Szegedy. Efficient testing of large graphs. *Combinatorica*, 20:451–476, 2000.
- [2] N. Alon, M. Krivelevich, I. Newman, and M. Szegedy. Regular languages are testable with a constant number of queries. *SIAM Journal on Computing*, 30(6), 2000.
- [3] A Apostolico and Z. Galil. Pattern matching algorithms, chapter 14: Approximate tree pattern matching. *Oxford University Press*, 1997.
- [4] Marcelo Arenas and Leonid Libkin. Xml data exchange : Consistency and query answering. In *24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'05)*, To appear 2005.
- [5] Z. Bar-Yossef, T.S. Jayram, R. Krauthgamer, and R. Kumar. Approximating edit distance efficiently. In *Proceedings of the ACM Symposium on Theory of Computing*, 2004.
- [6] T. Batu, F. Ergun, J. Kilian, A. Magen, S. Raskhodnikova, R. Rubinfeld, and R. Sami. A sublinear algorithm for weakly approximating edit distance. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 316–324, 2003.
- [7] M. Blum and S. Kannan. Designing programs that check their work. *Journal of the ACM*, 42(1):269–291, 1995.
- [8] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47(3):549–595, 1993.
- [9] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, 1989.
- [10] U. Boobna and M. de Rougemont. Correctors for XML data. In *XSym*, pages 97–111, 2004.
- [11] A. Broder. On the resemblance and containment of documents. In *SEQUENCES '97: Proceedings of the Compression and Complexity of Sequences 1997*, page 21, Washington, DC, USA, 1997. IEEE Computer Society.
- [12] G. Cormode and S. Muthukrishnan. The string edit distance matching problem with moves. In *SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 667–676, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics.
- [13] F. Ergün, S. Muthukrishnan, and S. Sahinalp. Comparing sequences with segment rearrangements. In *Proceedings of Foundations of Software Technology and Theoretical Computer Science*, pages 183–194, 2003.
- [14] Ronald Fagin, Phokion G. Kolaitis, Renee J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. In *ICDT '03: Proceedings of the 9th International Conference on Database Theory*, pages 207–224, London, UK, 2002. Springer-Verlag.
- [15] E. Fischer, F. Magniez, and M. de Rougemont. Property and equivalence testing on strings. Technical Report TR04-096, ECCC, 2004.

- [16] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, 1998.
- [17] M. Kearns and U. Vazirani. *An introduction to computational learning theory*. MIT Press, Cambridge, MA, USA, 1994.
- [18] F. Magniez and M. de Rougemont. Property testing of regular tree languages. In *International Conference on Automata Languages and Programming (ICALP)*, pages 932–944, 2004.
- [19] W. Martens and F. Neven. Frontiers of tractability for typechecking simple xml transformations. In *PODS '04: Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 23–34, New York, NY, USA, 2004. ACM Press.
- [20] W. Masek and M. Paterson. A faster algorithm for computing string edit distance. *Journal of Computer and System Sciences*, 20(1):18–31, 1980.
- [21] Rohit J. Parikh. On context-free languages. *Journal of the ACM (JACM)*, 13(4):570–581, 1966.
- [22] M. Parnas, D. Ron, and R. Rubinfeld. Tolerant property testing and distance approximation. Technical Report TR04-010, ECCC, 2004. <http://eccc.uni-trier.de/eccc-reports/2004/TR04-010/index.html>.
- [23] R. Rubinfeld and M. Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):23–32, 1996.
- [24] D. Shapira and J. Storer. Edit distance with move operations. In *Proceedings of Symposium on Combinatorial Pattern Matching*, volume 2373 of *Lecture Notes in Computer Science*, pages 85–98. Verlag, 2002.
- [25] D. Shapira and J. Storer. Large edit distance with multiple block operations. In *Proceedings of Symposium on String Processing and Information Retrieval*, pages 369–377, 2003.
- [26] K. C. Tai. The tree-to-tree correction problem. *Journal of the Association for Computing Machinery*, 26:422–433, 1979.
- [27] J. W. Thatcher. Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. *JCSS*, 1:317–322, 1967.
- [28] L. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal of Computing*, 8(3), 1979.
- [29] R. Wagner and M. Fisher. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21:168–173, 1974.