A Randomized Exhaustive Propositionalization Approach for Molecule Classification

Michele Samorani, Manuel Laguna
Leeds School of Business, University of Colorado at Boulder, UCB 419, Boulder, Colorado 80309-0419,
{Michael.Samorani@colorado.edu, laguna@colorado.edu}

Robert Kirk DeLisle and Daniel C. Weaver
Array BioPharma, 3200 Walnut St, Boulder, Colorado 80301
{Kirk.DeLisle@arraybiopharma.com, dweaver@arraybiopharma.com}

Drug discovery is the process of designing compounds that have desirable properties, such as activity and non-toxicity. Molecule classification techniques are used along this process to predict the properties of the compounds in order to expedite their testing. Ideally, the classification rules found should be accurate and reveal novel chemical properties, but current molecule representation techniques lead to less than adequate accuracy and knowledge discovery. This work extends the propositionalization approach recently proposed for multi-relational data mining in two ways: it generates expressive attributes exhaustively and it uses randomization to sample a limited set of complex (“deep”) attributes. Our experimental tests show that the procedure is able to generate meaningful and interpretable attributes from molecular structural data, and that these features are effective for classification purposes.

Key words: Relational learning, propositionalization, molecule classification, drug discovery
Version: May 26, 2010

1. Introduction

The molecule classification problem is a data mining problem (Jiawei and Kamber 2000) that consists of classifying 2 groups of molecules that exhibit different behavior in order to predict the group to which a new molecule belongs. This problem arises in a variety of situations, especially during the different phases of a drug design process, where it is needed to distinguish toxic from non toxic molecules, active from non active molecules, and so on. For a fairly comprehensive overview of the role played by molecule classification within a drug design process, see Weaver (2004). Molecule classification may be addressed in two ways: with a traditional classification technique or with a relational learning technique.

Traditional classification techniques, such as decision trees (Rokach and Maimon 2005) and support vector machines (Cristianini and Shawe-Taylor 2000), operate on data sets represented by a single table — where the rows are molecules and the columns are predefined attributes that describe them. While these methods have enjoyed some success (Svetnik et al. 2003), they require a molecule representation that preserves as much information about the molecule as possible and such representation may not be immediately obvious even for someone with expert knowledge.
A great variety of numerical representations of chemical structures exist within the field of Cheminformatics (Todeschini and Consonni 2000), and they can be classified based upon the degree of structural information required to compute them. At the simplest level, 0-Dimensional descriptors consist of counts of atoms or summations of the properties of atoms present within a compound. These include such values as molecular weight, number of atoms and number of bonds. 1-Dimensional descriptors require a certain degree of connectivity information and typically consist of counts of particular types of fragments (e.g., the number of amide groups or the number of carboxylic acids), or counts of single atoms with particular bonding patterns. 2-Dimensional descriptors have access to the complete connectivity graph of each molecule. A great many descriptors can be computed from this topological representation and include values such as walk and path counts, graph theoretic values such as minimum or maximum eigenvalues, topological fragment indicators or frequencies, and topological separation indices (e.g., the presence/frequency of carbon and chlorine atoms separated by N bonds), among many others. 3-Dimensional descriptors require not only the topological connectivity matrix but additionally the geometric arrangement of atoms within 3-D space. This leads to additional matrix-based calculations as well as geometrically centered mass, charge, and other property indices, as well as geometric separation indices. Considering that many compounds can adopt multiple 3-dimensional conformations, 4-dimensional descriptors attempt to capture the details of multiple, equally viable arrangements of a molecule's atoms. Interestingly, it has been shown that on average, 2-dimensional descriptors perform as well or better than higher dimensional representations calling into question the need to expend computational effort to predict low-energy 3-dimensional structures (Matter and Potter 1999, Dixon and Merz 2001).

One of the most commonly and successfully used molecular representations is the binary fingerprint. Molecular fingerprints consist of a vector of binary digits which represent the presence or absence of a particular molecular fragment within the molecule. A surprisingly diverse collection of fingerprints exist, two of which are used here. The MACCS Keys are a collection of pre-existing molecular substructures (that have presumably been deemed ‘interesting’ or ‘useful’), each on-bit identifies that fragment as existing within the structure in question (Durant et al. 2002). While being used very commonly and showing reasonable success across applications, a fundamental drawback of this fingerprint type is that the set of fragments is unchanging and may over time lose its useful or interesting character or may fail to include newly identified fragments emerging from the chemistry space. We also tested an alternative approach, which we call “Daylight”, consisting of generating a compendium of molecular fragments from the tested datasets (independently). The simplest method of fragment generation consists of enumeration of unique paths (unique in atom type and connectivity) up to a maximum length within the set of molecules followed by a hash function to assign bits within a fingerprint of desired length\(^1\). An example of an alternative fingerprint representation is to generate all fragments consisting of atoms and bonds that extend radially from a central atom the same distance up to a defined maximum (Rogers et al. 2005). However, many variations of molecular fingerprinting exist (Hert at al. 2004).

All these representation techniques suffer from the same fundamental drawback: they consider only predefined characteristics or methods of computation and ignore others outside a predefined scope.

\(^1\) Daylight Chemical Information Systems, Inc. (http://www.daylight.com)
This leads to two major limitations, one on the accuracy and one on the knowledge discovery. First, King et al. (2001) noted that any predefined attribute representation causes some loss of information because “all the information about a particular example is forced into a single row of a table”, which reduces the upper bound of the classification accuracy. Second, the classification rules cannot involve characteristics other than the ones included in the chosen representation. In other words, a traditional data mining process is incapable of generating attributes that were not included in the data set, limiting the knowledge discovery to expressing the classification rule in terms of a logical expression that involves the predefined attributes.

Relational learning techniques do not have these shortcomings. Instead of starting from a single table, they consider a database and search for patterns that involve more tables. A compounds database, for example, consists of a table containing the molecules, a table containing the atoms, and a table containing the bonds, connected to each other through foreign key (FK) relationships. The patterns found are represented by queries performed on a subset of tables. In practice, these techniques search for classification patterns in the space of the queries, by finding the ones that yield to the best classification accuracy. In this way, they potentially use all information contained in the database and find new patterns that have not been explicitly considered.

There are two families of relational learning techniques: multi relational decision trees (MRDTs) and propositionalization. In the former, the data set is iteratively split by successively adding refinements, creating a decision tree — as in Atramentov et al. (2003) — where each leaf corresponds to the attribute needed to classify a subset of molecules. In the latter, the database navigation leads to the generation of a set of attributes used to create a tabular data set on which the data mining task is eventually executed. In practice, the goal of propositionalization techniques is only the generation of new features, after which the classification task is executed by a given classifier.

MRDTs tend to find patterns corresponding to local optima because the refinements are built in a greedy fashion, though variations have been recently proposed by Serrurier and Prade (2008). On the other hand, propositionalization techniques overcome this problem by generating as many attributes as possible without evaluating their utility for the classification, until they have all been generated. The separation between feature generation and classification highlights a practical advantage of propositionalization over MRDTs, namely, it allows for the application of any classification technique. This makes it possible to exploit all the existing work that has been done in this field. Using well studied classifiers accelerates the implementation and increases the classification performance.

On the other hand, MRDTs have an important advantage over propositionalization: the feature space of MRDT techniques is larger. In other words, there are classification rules that can be found by MRDTs and not by propositionalization. The features generated by propositionalization have two limitations compared to MRDTs: they are less “deep” and less “expressive”. Both “depth” and “expressivity” of a feature depend on the complexity of the query used to generate the feature. As it will be explained, this complexity can be arbitrarily large, making the feature space infinite. Obviously, the propositionalization approach needs the definition of a limit in order to stop, while MRDT approaches do not, because the tree will continue growing — generating deeper and more expressive features — as long as the growth increases the accuracy. These differences, together with examples, will be explored below.
Our work extends the current approaches for propositionalization in two ways. First, we generate “expressive” attributes that are not generated by existing approaches, making our approach an “exhaustive” propositionalization approach, in the sense that it is capable of generating all the attributes that can be found using a SQL query. Although the attribute search space is exhaustive for obvious practical reasons the approach does not generate all attributes. Second, we randomly choose and generate a few “deep” attributes, which are ignored by existing approaches, making our approach a “randomized” propositionalization approach. The focus is on the feature generation and not on the classification; therefore we use a set of publicly available classifiers and compare the accuracy obtained with different propositionalization approaches. A statistical analysis shows that our extensions significantly improve the performance of the classification. Interestingly, ours is a general purpose method, and can be used to tackle any classification problem, not only molecule classification.

2. Definitions and Terminology

The concepts presented in this section are similar to the ones originally introduced by Knobbe et al. (1999), but we prefer to present a simplified terminology that also allows us to generalize the previous approaches.

A Types Graph (TG) is a directed graph that describes the types of attributes and associations in the database (DB). The vertices of the TG, called elements, correspond to a physical table in the DB, therefore we alternatively use “element” and table (e.g. rows of an element to indicate rows of the table corresponding to that element); the edges, called associations, correspond to the relationship between any two tables.

An element is characterized by the attributes contained in the table (i.e. columns), each of which has a type and a dimension. We consider 3 different types of attribute: id, categorical, and numeric; the type of an attribute is important because it determines the aggregate functions that can be applied to that attribute. The dimension, on the other hand, is a string representing the unit of measurement of an attribute, and it determines if two attributes can be compared. For example, if the dimension of attribute $a_1$ is “#atoms”, i.e. number of atoms, and the dimension of attribute $a_2$ is “#bonds”, i.e. number of bonds, any relation between $a_1$ and $a_2$, such as $a_1 > a_2$, is meaningless and therefore prohibited.

We consider two types of associations, $0 - 1$ and $0 - N$ associations. An association connecting element $E_1$ to element $E_2$, which corresponds to a foreign key relationship between $E_1$ and $E_2$, is:

- $0 - 1$ if every row in $E_1$ is associated to at most one row in $E_2$;
- $0 - N$ if every row in $E_1$ can be associated to any number of rows in $E_2$.

Note that, even if associations $N - N$ may be present in the DB, our method forbids them. Therefore, it is necessary to add an extra table substituting the $N - N$ association. Since the details of the DB design phase are outside the scope of our current development, they will not be treated; instead, they will be shown through examples.
A TG always contains a target table, which has one row per observation and only two attributes, the id of the observation and the class to which it belongs. The target table is connected with all the other tables through outgoing $0-N$ associations. The specific TG that we consider for our molecule classification problem is depicted in Figure 1.

In Figure 1, the Target table contains the compounds, the Atom table the atoms of all the compounds, the Bond table the bonds of all the compounds. Since $N-N$ associations are not allowed, a table AtomBond is needed that works as a bridge between the Atom and Bond tables. The fields forming the primary key of each table are underlined.

The feature generation procedure considers all paths up to a determined length and, for each of them, generates all possible attribute descriptors. An attribute descriptor (AD) is the description of a new attribute that is added to the target table. It can be viewed as the SQL query that is used to compute the value of the new attribute for all the rows of the target table. In the molecule classification, an example of an AD is:

```
select a.idTarget, count distinct a.idAtom
from Target t, Atom a
where a.element = "C"
and a.idTarget = t.id
group by a.idTarget
```

This AD counts the number of atoms of carbon contained in every compound. The expression “computing an AD” means computing the value of this attribute for all rows of the target table. In our implementation, the tables are in data structures contained in memory; it would be even possible to store them in a database if the amount of data could not be accommodated in memory. Note also that the phases of generating and computing an AD are separated, but we describe them together for presentation purposes. Similarly, the terms AD and attribute are often interchanged. Nevertheless, the distinction between their generation and their computation is important and we will show that the generation is much faster than the computation.

3. Generation of Attribute Descriptors

Our propositionalization algorithm consists of generating and computing a set of ADs, and finally adding them to the target table, so that the number of attributes increases. The generation of ADs is performed in two phases. First, paths through the tables are generated; second, a set of ADs is generated given the current path.
3.1 Finding the paths

Starting from the Target table, the procedure navigates the TG following the existing associations, up to a certain depth, selecting in this way a sequence of elements. A path of depth \( d = 2 \) is, for instance, \( \text{Target} \rightarrow \text{Atom} \rightarrow \text{AtomBond} \). Since each element can be encountered more than once, every element in the path is a copy of the original element. Also, for reasons that will be clear in the next section, an association can be navigated only if it does not generate a subpath \( E_1 \rightarrow E_2 \rightarrow E_1 \), where the first association is \( 0 - N \) and the second \( 0 - 1 \). Therefore, a path cannot contain either the subpath \( \text{Atom} \rightarrow \text{AtomBond} \rightarrow \text{Atom} \) or the subpath \( \text{Bond} \rightarrow \text{AtomBond} \rightarrow \text{Bond} \). Note that there is no theoretical limit in the depth of the path under consideration. The longer the path, the richer the information that can be expressed by an attribute, but also the longer the time required to compute each attribute.

3.2 Aggregations and Refinements

Given a path, we apply the Roll-Up algorithm, which consists of summarizing information and adding it to the target element. Starting from the element preceding the last one in the path and going back to the Target element, a new aggregate attribute is virtually added to each element, using information contained in the following ones. This procedure results in adding a new aggregate attribute to the target element, as shown in the pseudo-code in Figure 2.

--- Figure 2 ---

Suppose that we want to generate all possible attributes for the path \( \text{Target} \rightarrow \text{Atom} \rightarrow \text{AtomBond} \rightarrow \text{Bond} \). In step 1 of Figure 2, \( \text{currentEle} \) is set to Bond. Then, in the first do-loop iteration, the element to which we add an attribute (\( \text{currentEle} \)) is AtomBond, at the second iteration is Atom, and at the third is Target. The \( \text{GenerateNextDerivedAttribute} \) function (Figure 3) completes the definition of the entire procedure.

--- Figure 3 ---

The input to this function is two subsequent elements, \( \text{currentEle} \) and \( \text{followingEle} \), and returns the derived attribute to add to \( \text{currentEle} \).

The derived attribute has to summarize, for each row in \( \text{currentEle} \), the content of \( \text{followingEle} \). The derived attribute can be constructed in two ways, depending on the association connecting \( \text{currentEle} \) to \( \text{followingEle} \):

1. by attaching an attribute of \( \text{followingEle} \), if the association is \( 0 - 1 \) (attachment)
2. by aggregating and possibly refining an attribute of \( \text{followingEle} \), if the association is \( 0 - N \) (aggregate-and-refine)

We now show how attachment works using our example. In the first step of our example, \( \text{currentEle} \) is AtomBond and \( \text{followingEle} \) is Bond. Since the association is \( 0 - 1 \), we choose an attribute of
followingEle and attach it to currentEle. Suppose that the attribute type is chosen. Through a simple join on the composite key [idTarget, idBond] it is possible to retrieve, for each row in AtomBond, the value of type. Note that it would make no sense to choose either idTarget or idBond instead of type, because they are involved in the join and obviously they are already present in the currentEle. The attribute added to currentEle maintains the original dimension and type; then, AtomBond is modified as depicted in Figure 4.

— Figure 4 —

If followingEle contains a derived attribute $X$, then $X$ must be added to currentEle, otherwise an attribute that could be generated by a shorter path would be generated. Enforcing this condition guarantees that no attribute is generated more than once.

If the association is $0 \rightarrow N$, the derived attribute is generated through an aggregate-and-refine process. For each row $r$ in currentEle, there may be many corresponding rows (i.e., the $S(r)$ set) in followingEle. Hence, it is necessary to summarize the $S(r)$ set into one single value that will be the derived attribute for row $r$. Two definitions are necessary to accomplish this:

1. An aggregation of an attribute $A_g$ of followingEle
2. A refinement of an attribute $A_r$ of followingEle

To define an aggregation, we need to choose an attribute $A_g$ of followingEle and a suitable aggregating function. $A_g$ can be any attribute except any used for the join and the aggregating function must be chosen according to the type of $A_g$, as shown in Table 1.

<table>
<thead>
<tr>
<th>Aggregating Function</th>
<th>Input Attribute Type</th>
<th>Output Attribute Type</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>Numeric</td>
<td>Numeric</td>
<td>D</td>
</tr>
<tr>
<td>Max</td>
<td>Numeric</td>
<td>Numeric</td>
<td>D</td>
</tr>
<tr>
<td>Avg</td>
<td>Numeric</td>
<td>Numeric</td>
<td>D</td>
</tr>
<tr>
<td>Sum</td>
<td>Numeric</td>
<td>Numeric</td>
<td>D</td>
</tr>
<tr>
<td>MostFrequent</td>
<td>Categorical</td>
<td>Categorical</td>
<td>D</td>
</tr>
<tr>
<td>CountDistinct</td>
<td>Categorical or ID</td>
<td>Numeric</td>
<td>#D</td>
</tr>
</tbody>
</table>

Table 1. For each aggregating function, the type and dimension of the input attributes and output attributes

Alternative aggregate functions may be implemented (e.g., median) which would generate different attributes, however, we chose the ones shown in Table 1 to be consistent with existing propositionalization approaches (Knobbe et al. 2001). Given the set of rows $S(r)$ in followingEle that corresponds to row $r$ in currentEle, the aggregating functions compute a single value, as follows:

Min returns the minimum value of $A_g$ in $S(r)$
**Max** returns the maximum value of $A_g$ in $S(r)$
**Avg** returns the average value of $A_g$ in $S(r)$
**Sum** returns the summation of the $A_g$ values in $S(r)$
**MostFrequent** returns the most frequent $A_g$ value in $S(r)$
**CountDistinct** returns the number of distinct values in $S(r)$

Table 1 also reports the dimension of the output attribute. For example, consider the second step of the algorithm in Figure 2 relative to our example, for which $currentEle$ is Atom and $followingEle$ is the modified element AtomBond. The attributes that can be chosen as $A_g$ are $idBond$ and $type$. Suppose that we choose $idBond$, whose type is ID, then the countDistinct function must be used, because it is the only aggregating function that allows an input attribute of type ID. Therefore, the derived attribute for Atom is $countDistinct(idBond)$, i.e. the number of bonds with which each atom participates. This new attribute is numeric and its dimension is “#bondID” (see Table 1). Note that the dimension of the new attribute may be compared to other existing attributes in order to create refinements. Nevertheless, in the molecule classification application, this feature is not used because there are no attributes whose type is “#bondID” or “#atomID”.

A refinement is a condition that results in the selection of only a subset of rows in $followingEle$. When no refinements are present then the derived attribute is built using the entire $S(r)$ set. In this work, we consider two types of refinements: value refinements and comparison refinements. A value refinement has the form $A_r \rho v$, where:

- $A_r$ is an attribute belonging to $followingEle$
- $\rho$ is a compatible refinement operator
- $v$ is a value with the same dimension as $A_r$ and that is used for the comparison

Table 2 shows that refinement operators that may be used depending on the type of $A_r$.

<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
<th>Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric</td>
<td>Not considered</td>
<td>$&lt;, &gt;$</td>
</tr>
<tr>
<td>Categorical</td>
<td>$=!, !=$</td>
<td>$=!, !=$</td>
</tr>
<tr>
<td>ID</td>
<td>Meaningless</td>
<td>$=!, !=$</td>
</tr>
</tbody>
</table>

Suppose that after selecting the aggregation $countDistinct(idBond)$ in the second step of the example, we build a value refinement where $A_r$ is type (i.e., the categorical attribute attached to AtomBond in the first step), $\rho$ is “$=$”, and $v$ is “2” (double bond), then the derived attribute may be represented in a SQL-like notation as $countDistinct(idBond) \text{ where type } = \text{"2"}$. If we denote as $C2$ this new aggregate attribute, which is added to the element Atom, then its value is the number of double bonds to which the atom participates. Here too, in order to guarantee that no attribute is generated more than once, if $followingEle$ contains a derived attribute $X$, we must select $X$ to be either $A_g$ or $A_r$. 
Unlike MRDT approaches, we do not consider value refinements on numeric attributes. In the third step of our example, where currentEle is Target and followingEle is the modified Atom element, we could count the number of atoms (the aggregating function would be countDistinct(idAtom)) with $C2 < 2$. In other words, we could count the number of atoms participating in less than 2 double bonds. This refinement on a numeric attribute is not allowed in our method; if a numeric attribute is aggregated, no value refinement is possible. Allowing this would lead to the generation of a large number of attributes that are similar to one another, with the only difference being the chosen numeric threshold (“2” in our case). Furthermore, similar information (although not identical) contained in these attributes is expressed by other attributes that are also generated (e.g. the average value of $C2$ among all atoms). Note also that a value refinement is meaningless if $A_r$ is an ID. For example, consider the following derived attribute: most frequent (type) where idBond = “126”. For any possible $\rho$ (=, !=, > or <), the attribute would be clearly useless, because the id’s have no semantic meaning.

The first difference between our approach and existing propositionalization approaches is that existing propositionalization approaches use refinements only when the aggregate function is countDistinct and not when it is a numeric aggregate function (min, max, etc.). Consider again the third step of our example, where currentEle is Target and followingEle is the modified element Atom. Existing methods generate attributes such as “Maximum $C2$ among the atoms” or "Number of atoms of Oxygen", but not attributes such as “Maximum $C2$ of atoms of Oxygen”. The first example uses, for each row $r$ in currentEle (i.e. for each molecule), all rows in $S(r)$ and aggregates them with the max function, with no refinement. The second example has a refinement (ele = “O”) and counts the rows satisfying this condition. The third example, which is not supported by existing approaches, has both a refinement condition (ele = “O”) and an aggregate function (max) different from count. Our method produces these refinements.

Our approach also considers comparison refinements of the form $A_r \, \rho \, C$, where:

- $A_r$ is an attribute belonging to followingEle
- $\rho$ is a compatible refinement operator
- $C$ is an attribute with the same type and dimension as $A_r$ and either belonging to an element preceding currentEle or to currentEle (join key excluded)

Unlike a value refinement, a comparison refinement compares $A_r$ to another attribute $B$ instead of to a fixed numerical value. In this refinement, $B$ may be chosen only among the attributes of the elements preceding followingEle. In fact, for every row in followingEle there is exactly one associated row in a previous element of the path. This is a direct consequence of limiting the navigation of the TG to 0−1 and 0−N associations, and avoiding N−N associations.

Neither existing MRDT nor propositionalization approaches consider comparison refinements, making it a novel feature of our approach. These refinements generate features that embed important information on sub-paths, such as “number of carbon atoms connected to at least one oxygen” or the presence of a particular ring, even when this may require great depths. Hence, the generation of features with comparison refinements differentiates our approach from traditional propositionalization methods. Additional insight on the implementation of aggregation and refinements may be gained by the detailed example included in the appendix. Table 3 shows, for depths 1 to 7, the cumulative number
of attributes generated by our approach (Exhaustive) and the traditional propositionalization approach, when applied to two datasets that represent two significantly different biological processes, both of which are highly relevant to drug discovery. Table 3 also shows the time (measured in milliseconds on an Intel® Xeon® CPU X5355 at 2.66 GHz equipped with 32 GB of RAM and Microsoft Windows Server 2003 R2 Enterprise x64 Edition) required to compute one attribute per compound at each depth. It is important to point out that the time taken to compute an attribute does not depend on whether the attribute belongs to the traditional or the exhaustive space.

Table 3. Cumulative number of attributes and time to compute one attribute per compound

<table>
<thead>
<tr>
<th>Depth</th>
<th>Estrogen Traditional</th>
<th>Exhastive</th>
<th>Mutagenesis Traditional</th>
<th>Exhaustive</th>
<th>Comp. Time (ms/compound)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>56</td>
<td>80</td>
<td>52</td>
<td>74</td>
<td>0.06</td>
</tr>
<tr>
<td>2</td>
<td>118</td>
<td>238</td>
<td>110</td>
<td>220</td>
<td>0.93</td>
</tr>
<tr>
<td>3</td>
<td>346</td>
<td>1,570</td>
<td>324</td>
<td>1,422</td>
<td>1.71</td>
</tr>
<tr>
<td>4</td>
<td>814</td>
<td>4,934</td>
<td>780</td>
<td>4,534</td>
<td>3.35</td>
</tr>
<tr>
<td>5</td>
<td>2,386</td>
<td>34,962</td>
<td>2,306</td>
<td>30,604</td>
<td>4.48</td>
</tr>
<tr>
<td>6</td>
<td>5,598</td>
<td>197,638</td>
<td>5,474</td>
<td>181,164</td>
<td>6.83</td>
</tr>
<tr>
<td>7</td>
<td>14,514</td>
<td>1,027,570</td>
<td>14,216</td>
<td>948,730</td>
<td>9.02</td>
</tr>
</tbody>
</table>

The estrogen receptor binding data set (Fang, Tong et al. 2001) consists of 232 compounds that have been tested for their ability to bind to the estrogen receptor, expressed as the binding affinity of the compound relative to the natural ligand for the estrogen receptor, 17-β estradiol. This endpoint represents a single biochemical event – small molecule interaction with a protein target – which is ubiquitous in early drug discovery. Early evaluation of binding affinity of compounds that are available or could be synthesized assists in establishing priorities for purchasing or synthesis, thus reducing the resources necessary to identify novel and useful chemical matter.

The mutagenesis dataset (Votano, Parham et al. 2004) consists of 400 compounds that have been evaluated for Ames mutagenicity, a measure of the mutagenic (and thus carcinogenic) potential. While the mutagenicity assay is only a surrogate for the true carcinogenic potential, it is recognized as the de facto standard for evaluating the carcinogenicity of chemical compounds at relative early stages of drug discovery. Relative to interaction with a protein target, Ames mutagenicity is a more complex phenomenon and there exist multiple mechanisms by which a compound could exert a mutagenic effect. The result is that any modeling procedure must be sophisticated enough to isolate multiple mechanisms of action and the compounds that are active/non-active for each mechanism. Again, early identification of potentially problematic chemical matter reduces the resources necessary to develop new drug compounds. Both datasets are publically available and have been evaluated by various authors in the past, although, to the best of our knowledge, they have never been used to test any propositionalization or relation learning technique.
5. Attribute Bound Experiments

The times reported in Table 3 show the significant increase experienced with the depth. The reason is that the number of joins needed to retrieve the value of the attributes linearly increases with the depth. This indicates that only a subset of attributes at depths greater than 4 can be computed in most practical settings. In order to assess the classification accuracy of our proposed approach, we used a 10-fold cross validation (CV) performed by the following 10 Weka classifiers (all set at their default parameters): BayesNet, PART, RandomForest, Bagging, MultilayerPerceptron (neural network), J48 (C4.5), ADTree (alternating decision tree), REPTree (fast decision tree), NNge (nearest-neighbor-like algorithm), Ridor (RIpple-DOwn Rule learner). Bagging uses a Fast decision tree as its elementary classifier, which is also Weka’s default classifier. Weka is an open source data mining framework available for download at http://www.cs.waikato.ac.nz/ml/weka/ (last accessed on 05/18/2010). The software may be downloaded to access literature references to the classifiers. Witten and Frank (2005) provide additional details.

We partitioned the data sets into 10 pairs $P_i = \{\text{training set } i, \text{ test set } i\}, \ i = 1, \ldots, 10$. The classifiers listed above implement well-known classification techniques, such as neural networks, meta-classifiers, decision trees and rules. They also have the desirable property of accepting input data with missing values, which is of particular importance to us because many of the attributes generated are not defined for all molecules. For example, “the most frequent element different from carbon” is not defined for molecules containing only atoms of carbons, however, the database contains such molecules because hydrogen atoms are not included in the molecule representation even if they are actually present in the molecule. Alternatively, it would be possible to fill the missing values with the average value or the mode, but since many attributes at low depths are missing for most molecules, this strategy would fail to capture key patterns at those depths. Since our experiments often involve thousands of attributes, in all experiments of the paper a supervised feature selection is performed before the execution of the test corresponding to each fold. Considering the current training set, a set of attributes is selected using the default feature selection algorithm in Weka (Evaluator: CfsSubsetEval and Search: BestFirst). Then, the selected attributes are fixed and the classifiers are applied to the current $P_i$. We have observed that the feature selection improves the average classification accuracy, particularly when the number of attributes exceeds 1,000. The proportion of selected attributes varies with the number of initial attributes, but never exceeds 50, even in experiments where the initial number of attributes is about 8,000. In our experiments, we measure overall classification accuracy with the average cross validation accuracy obtained by the 10 classifiers. This choice is consistent with the goal of this work, which is to generate attributes that are valuable in a classification process, and not to design new classification or feature selection procedures.

Existing propositionalization approaches compute all attributes up to a predefined depth. Let us refer to this strategy as up-to-depth-$X$ (UD-$X$), where $X$ is the predefined maximum depth. Figure 5 shows the average cross validation accuracy obtained by the 10 classifiers for strategies UD-1, UD-2, UD-3 and UD-4.
Figure 5 suggests that, for both data sets and for both attribute spaces (traditional and exhaustive), the accuracy increases if attributes constructed at deeper levels are included. Also, at depths 3 and 4, the attributes generated by the exhaustive approach yield a higher accuracy than the ones generated by the traditional methods. Interestingly, this is not true for the Estrogen data set at depths 1 and 2. We believe that, at these depths, the information that can be expressed by any exhaustive attribute is also contained in some “simpler” traditional attribute. For example, consider the exhaustive attribute “average number of bonds connecting an atom of Carbon” and the traditional attribute “number of atoms of Carbon”. Both estimate the molecular mass, but, while the former does it in an involved way, the latter does it in a simple way. Thus, training a classifier on the exhaustive space leads to overfitting and, therefore, the accuracy obtained will be lower. The situation is reversed at lower depths, which contain information that can be expressed by the exhaustive attributes but not by the traditional ones. Table 4 reports the best average cross validation accuracy and the best 2 individual cross validation performances obtained in these UD tests, in terms of accuracy. We report also the second best because its accuracy may have a lower standard deviation ($\sigma$), a desirable property. We use the notation UD-X-Y, where X is the maximum depth and Y is E for the exhaustive attribute space and T for the traditional attribute space. Table 4 shows that, with only one exception, all best performances were obtained with UD-4-E. This indicates the value of using Exhaustive and exploring greater depths.

**Table 4.** Average and best individual CV accuracy on the UD tests

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Average CV accuracy</th>
<th>Two best individual CV accuracies</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>First</td>
</tr>
<tr>
<td>Estrogen</td>
<td>76.10% — UD-4-E</td>
<td>79.24% ($\sigma = 9.23%$)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UD-4-E</td>
</tr>
<tr>
<td>Mutagenesis</td>
<td>75.20% — UD-4-E</td>
<td>79.75% ($\sigma = 6.61%$)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UD-3-E</td>
</tr>
</tbody>
</table>

In order to assess the benefits of including attributes found at deeper levels, we must define a strategy that chooses which ones to compute. The problem of selecting the attributes to compute among a vast set of attributes, such as the ones between depths 4 and 7, is a complex search problem. Generating the ADs representing these attributes is relatively fast, requiring approximately 3 minutes for the approximately 1 million ADs in the Exhaustive attribute space. However, computing just one AD for all compounds can consume seconds.

We ran an experiment where the set of computed attributes includes all up to depth 3 and 1,000 randomly chosen between depths 4 and 7. We chose to compute all attributes up to depth 3 (instead of 4) because, as shown in Figure 5, the accuracy obtained with UD-3 and UD-4 is similar, but computing only the attributes up to depth 3 is much faster. We denote this randomized strategy with RAND and Table 5 reports the average accuracy obtained and the standard deviation (between parentheses) across all classifiers when compared to UD-3. Table 6 reports the best results using RAND in similar format as Table 4.

**Table 5.** Comparison between UD-3 and RAND
The randomized strategy yields a higher accuracy for both data sets and both attribute spaces. Interestingly, the gap between RAND and UD-3 is larger than the gap between UD-4 and UD-3, even though the number of attributes at depth 4 is more than 1,000. This reveals the usefulness of the information embedded in attributes derived at depths 5, 6, and 7. The experiment, therefore, suggests the value of developing a mechanism for including attributes at lower depths. When dealing with a computational budget, there is a tradeoff between generating all attributes at higher depths and sampling attributes from lower depths. In our next experiment, we explore this tradeoff.

### 6. Generating Attributes when Limiting the Computational Time

In practical settings, we must consider that there is a time limitation imposed on the generation of attributes from data sets. Since the experiments showed that the complete generation of attributes at lower depths than 3 is extremely time consuming, we now investigate several mechanisms to sample attributes at depths 4 and beyond. In particular, we compare two different strategies:

1. **Scan** — compute all attributes from depth 1 on until reaching the time limit
2. **Scan and Sample** — compute all attributes from depth 1 on until half of the allotted time is reached, then randomly sample from the un-computed attributes at the current depth and any lower depth until the computational budget is exhausted.

We executed both of these strategies in combination with the traditional and exhaustive approaches and time limits of 5, 30, 60, and 120 minutes. All experiments were conducted on an Intel® Xeon® CPU X5355 at 2.66 GHz equipped with 32 GB of RAM and Microsoft Windows Server 2003 R2 Enterprise x64 Edition. Tables 7 and 8 report the average accuracy and the standard deviation (between parentheses) across the classifiers obtained for each data set, time limit, strategy and approach.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Attribute space</th>
<th>UD-3</th>
<th>RAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estrogen</td>
<td>Traditional</td>
<td>72.62% (2.00%)</td>
<td>77.93% (2.52%)</td>
</tr>
<tr>
<td></td>
<td>Exhaustive</td>
<td>75.88% (1.80%)</td>
<td>77.72% (1.96%)</td>
</tr>
<tr>
<td>Mutagenesis</td>
<td>Traditional</td>
<td>74.25% (2.87%)</td>
<td>74.95% (3.30%)</td>
</tr>
<tr>
<td></td>
<td>Exhaustive</td>
<td>75.00% (3.46%)</td>
<td>76.30% (3.48%)</td>
</tr>
</tbody>
</table>

Table 6. Average and best individual CV accuracy on RAND tests

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Average CV accuracy</th>
<th>Two best individual CV accuracies</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>First</td>
</tr>
<tr>
<td>Estrogen</td>
<td>77.93% — RAND-T</td>
<td>81.00% ($\sigma = 8.77%$)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RAND-T</td>
</tr>
<tr>
<td>Mutagenesis</td>
<td>76.30% — RAND-E</td>
<td>81.25% ($\sigma = 7.00%$)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RAND-T</td>
</tr>
</tbody>
</table>

Table 7. Results on Estrogen
Table 9 reports the CV accuracies in the format used in Tables 4 and 6. The notation X-Y-Z is used to indicate the attribute generation strategy (S = scan and SS = scan and sample), the time limit (5, 30, 60 and 120) and the attribute space (T = traditional and E = exhaustive).

Table 9. Average and best individual CV accuracy on time-limit tests

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Average CV accuracy</th>
<th>Two best individual CV accuracies</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>First</td>
</tr>
<tr>
<td>Estrogen</td>
<td>78.18% — SS-120-E</td>
<td>81.83% (σ = 10.65%)</td>
</tr>
<tr>
<td>Mutagenesis</td>
<td>77.90% — SS-120-E</td>
<td>82.75% (σ = 7.95%)</td>
</tr>
</tbody>
</table>

The results in Tables 7 and 8 show that the Scan strategy should be preferred for relatively short computational times. In the runs with a 5-minute limit, Scan generally outperforms the Scan and Sample strategy. This seems to be another instance of the Occam’s razor effect that suggests that simple classification rules — high level attributes in this case — are preferable to more complex ones (Domingos 1999). The effect does not hold after the procedure is able to generate more attributes at lower levels (as is the case when the time limit is extended). The results show two different patterns, which are more clearly identified when the results in Tables 7 and 8 are shown graphically (see Figure 6). The patterns are such that Scan and Sample tends to be the better strategy, particularly when applied to Estrogen, and Exhaustive tends to produce a better attribute space than Traditional, particularly when applied to Mutagenesis. Table 9 shows that the best results are generally obtained by the Scan and Sample strategy, using the exhaustive attribute space, and with a large time limit.
Figure 6 shows an advantage of Exhaustive over Traditional on the Mutagenesis data set, while the different methodologies seem to achieve similar performances on the Estrogen data set — though the statistical analysis below shows that our enhancements lead in fact to a higher accuracy.

This result may be explained by the complexity of the mutagenesis process. Compounds tested for mutagenicity must first diffuse into bacterial cells, a non-complex process itself related to molecular size, charge, hydrophobic/hydrophilic balance, and molecular flexibility. Once inside the cell, compounds can then interact with cellular components in a number of ways leading to DNA damage. Direct interaction with DNA, DNA modifying enzymes, DNA replicating enzymes or the replication process itself, DNA repair enzymes, and DNA packaging proteins may all lead to mutations in the bacterial genome and a positive test. Unlike estrogen receptor binding which occurs within a single, defined location of a single protein, mutation may occur through interactions with various targets (DNA and protein, at least) and those targets may provide multiple points of interaction. It is thus not surprising that a more complex set of rules are required to accurately classify a compound as mutagenic positive or negative.

The data generated in this experiment are amenable to statistical analysis. In particular, we have recorded the classification accuracy obtained by the 10 classifiers for each combination of the following variables:

- Data set (Estrogen and Mutagenesis)
- Time (0.5, 1 and 2 hours)
- Attribute space (traditional and exhaustive)
- Scanning strategy (Scan and Scan and Sample)

In our first experiment, we contrast the merit of the approaches to generate the attribute space (i.e., traditional and exhaustive). For each combination of data set, time, and scanning strategy, we record the proportion of classifiers that obtain a higher CV accuracy if the exhaustive attribute space is used. A $t$-test showed that the proportion of classifiers performing better when employing the Exhaustive over the Traditional approach is greater than 0.5 with a $p$-value of 0.0014. In our second experiment, we compare the merit of the two scanning strategies (i.e., Scan and Scan and Sample). Similarly to the first experiment, for each combination of data set, time, and attribute space, we record the proportion of classifiers that obtain a higher CV accuracy if the SS strategy is used. A $t$-test showed that the proportion of classifiers performing better with the SS strategy when compared to the Scan strategy was greater than 0.5 with a $p$-value of 0.0041. Both of these tests show that for runs longer than 5 minutes classifiers tend to perform better when using Exhaustive or Scan and Sample. However, these tests do not provide evidence indicating which classification techniques benefit the most from either approach.

In order to provide a better assessment of the quality of the attributes generated by our method, we consider the accuracy obtained when using Daylight and MACCS Keys, two of the most popular fingerprint representations currently used in drug discovery. For both data sets, the fingerprint representations were generated using the cheminformatics toolkit, RDKit, which is freely available at [http://www.RDKit.org](http://www.RDKit.org) (accessed on 05/18/2010). We performed a cross validation test on the 2 data
sets using the same classifiers and the same feature selection technique. Table 10 reports the average CV accuracy and the standard deviation (between parentheses) across the classifiers obtained using Daylight, MACCS Keys, and the Scan-and-Sample-Exhaustive approach (reported in Tables 7 and 8) when imposing a time limit of 120 minutes.

Table 10. Results obtained by the different molecule representations

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Daylight</th>
<th>MACCS Keys</th>
<th>Scan and Sample Exhaustive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estrogen</td>
<td>79.69% (2.68%)</td>
<td>75.69% (10.50%)</td>
<td>78.18% (2.71%)</td>
</tr>
<tr>
<td>Mutagenesis</td>
<td>76.33% (1.37%)</td>
<td>78.60% (2.29%)</td>
<td>77.90% (2.84%)</td>
</tr>
</tbody>
</table>

We have compared the performance obtained by each Fingerprint representation to the one obtained by the Scan-and-Sample-Exhaustive approach employing a two-proportion test. For both data sets we recorded the proportion of classifiers that perform better with the Fingerprint representation and then tested if the proportion was statistically greater than 0.5. For the Estrogen data set, 5 classifiers out of 10 perform better using the Daylight representation ($p$-value 0.5), whereas 7 out of 10 perform better using the MACCS Keys representation ($p$-value 0.103); for the Mutagenesis data set the proportion was 7 out of 10 for both fingerprint representations ($p$-value 0.103 for both). In all cases, the proportion was not significant, indicating that both fingerprint representations achieve similar performances to our best attribute representation.

Over the years, the molecule classification problem has been tackled by several techniques, all of which require experts’ knowledge to define the attribute representation. Our propositionalization method, however, allows new features to emerge. Therefore, when classifiers are applied to the Fingerprints data, the classification rules employ attributes that have been chosen a priori by domain experts, whereas our propositionalization-based attribute representation may lead to classification rules involving molecule characteristics that experts have not taken into account beforehand and in this way can be viewed as complementing experts’ knowledge. While it is expected that some attributes will occur which are consistent with pre-existing knowledge of the modeled domain, it is also anticipated that new knowledge may emerge due to the in-process, unbiased methodology. To determine if new information was discovered, we considered the attributes generated by the experiment “Scan and Sample Exhaustive” with a 2 hour time limit. Attributes appearing in each of the 10-cross validation folds for each dataset were evaluated for their biochemical meaning. Following is a list of some of these attributes and an interpretation of their presumed meanings with respect to the modeled endpoint.

**Mutagenicity**

- **Number of bromine atoms.** As mentioned previously, mutagenesis is a complex biological phenomenon which may require more complex rules to fully model the endpoint. Interestingly, the simple count of bromine atoms occurred in each cross-validation fold. Within the dataset used, we found 12 compounds that contain bromine (many with multiple bromine atoms), 11 of which are mutagenic, and thus the number of bromine atoms represents a simple mutagenicity filter based on
these data. This finding is consistent with experimental findings that bromine-containing chemical structures are more mutagenic than analogues within which chlorine is substituted for bromine (Finkelstein 1994).

- **Label each atom with the number of double bonds it has. Compute the mean of these labels across the atoms different from sulfur.** By distinguishing compounds that have large numbers of double bonds, this attribute effectively identifies structures that contain a high degree of aromatic ring systems. Compounds in which those rings are fused (in which some atoms participate in multiple rings rather than two rings joined by a single bond — Figure 7A) further contribute to an increase in the value of this attribute. These types of fused ring systems are inherently planar and are commonly known to be problematic due to their ability to intercalate DNA leading to mutations (Ferguson 2007). The clause “compute the mean of these labels across the atoms different from sulfur” is interesting as it effectively eliminates double bonds to sulfur atoms from impacting the calculation. These substructures are not planar but rather introduce a three-dimensional quality to the molecules — a characteristic that we speculate may prevent DNA intercalation. The electronic nature of such substructures may also prevent direct interaction with DNA via repulsion form the negatively charged phosphate backbone of DNA.

- **Label each atom different from carbon with the number of atoms of nitrogen to which it is connected. Compute the mean of these labels.** While we are not aware of specific evidence that [non-carbon – nitrogen] atom pairs are generally mutagenic, 49 nitro-containing structures (Figure 7B) are present in the dataset, 46 of which are mutagenic. A further evaluation of the dataset reveals 117 compounds exhibiting the [non-carbon – nitrogen] atom pair, only 19 of which are non-mutagenic, suggesting a relatively robust filter and that atom pairs of this sort would be best avoided in the design of new drug compounds. The example in the appendix shows how this attribute is generated.

**Estrogen**

- **Label each bond with the number of oxygen involved in it. Compute the mean of these labels across all bonds of type different from 3.** This attribute most likely acts as an oxygen indicator. The clause “Compute the mean of these labels across all bonds of type different from 3” prevents triple bonds in the molecule from artificially depressing the index as oxygen is not found participating in triple bonds. Generally, binding to the estrogen receptor is influenced by specifically placed oxygen-containing groups placed at either end of the molecule (Brzozowski, Pike et al. 1997), and a lack of these groups tends to decrease the probability of binding to the receptor. Panels C, D, and E of Figure 7 illustrate the importance of oxygen and the impact of oxygen placement on estrogen receptor binding. While this type of indicator appears useful, it is clear that it must be combined with other attributes in order to establish the necessary and sufficient basis for making a prediction.

- **Label each atom A in the following way. 1) Consider the atoms connected to it and count the bonds to which they participate (excluding the bond connecting A to each of them). 2) Compute the sum of these labels and obtain the label for A. Label the molecule with the minimum of these labels across all atoms of oxygen.** The previous attribute acted as a relatively non-specific oxygen indicator, but this new attribute behaves as an indicator of oxygen placement within the molecular scaffold.
Specifically, a high value would represent an oxygen atom that is connected to other atoms participating in a large number of additional bonds - presumably an oxygen atom that is somewhat buried and interacting with highly branched atoms. Low values would represent low branching, and moreover, terminal oxygen atoms would have particularly small values. As illustrated in Panels C, D, and E of Figure 7, all compounds contain oxygen atoms, however, only a specifically placed terminal oxygen atom can impart estrogen binding activity.

7. Additional Attribute Selection Strategies

In section 5, we introduced the Scan and Sample approach for selecting attributes at increasingly lower depths. In addition to this sampling procedure, we attempted two other strategies for selecting attributes. The first strategy, attempted to identify the best mix of attributes from each depth. For this purpose, we set up a search using OptQuest, a commercially available general-purpose optimizer, that is based on the scatter search metaheuristic (Glover, et al. 2003). The optimization model consisted of 4 continuous variables and one constraint. The variables represented the proportion of attributes from levels 4 to 7. The constraint forced the sum of the variables to be equal to one. For each set of proportions suggested during the OptQuest search, a sample procedure was used to sample attributes from each depth, as dictated by the given proportions, and configure a data set. The data set was then used to perform cross validation with the set of classifiers in order to estimate an average accuracy for the proposed proportions.

The second strategy selected attributes one at a time by considering diversity and expected information gain. The diversity of an AD is the Euclidean distance from its vector representation and the vector representation of other attributes already in the set. The vector representation of an AD contains information such as its depth, the number of refinements and the number of times that aggregating functions appear. The expected information gain is given by a Bayesian Network, which, initially, does not have a way of distinguishing between ADs with high information gain and those with low. However, the Bayesian Network learns, through incremental training, and is then able to discriminate between ADs with potential high information gain from those whose potential is low. Our experiments showed that the estimates given by the Bayesian Network became gradually more accurate.

Unfortunately, neither of these two approaches outperformed the random sampling used within the scan and sample strategy. Our conjecture is that attributes at depth 7 contain discriminant information that is not captured by attributes at higher depths. If this is correct, the sampling strategy has a substantial advantage because most of the attributes that it chooses are from depth 7, due to the large number of attributes generated by both the Exhaustive and Traditional approaches at this depth (see Table 3). Unlike the random sampling, the strategic approaches described above choose a mix of attributes that are not biased toward the lowest depth.
8. Conclusions

We improve the current propositionalization procedures that are typically used in the process of drug discovery by creating a method that generates more expressive and complex (deeper) attributes. Our experimental testing shows that the multi-relational data mining (MRDM) methodology that we employed is competitive with current methods within the field of cheminformatics and quantitative structure activity relationship modeling. MRDM has the advantage that no precalculated molecular representations were required but rather the technique uses the molecular structure itself and derives the necessary descriptors from it during the modeling process. Both datasets utilized are very well characterized and represent two biochemical processes that differ dramatically in their complexity, and in both cases MRDM is competitive with published results and those we have obtained previously using more traditional techniques (not shown). To our knowledge, this represents the first application of the MRDM technology within the field of drug discovery highlighting their potential and opening new avenues of research within this hybrid field. Techniques such as these provide a mechanism to prioritize chemical compounds for purchase or synthesis and increase the probability of bringing a successful drug to market within a shorter time and at reduced costs. Given that current estimates of the cost to bring a single drug to market in the range of $500 million to $2,000 million (DiMasi, Hansen et al. 2003), efforts to reduce costs are necessary to ensure efficient improvements in healthcare.

Future research opportunities based on our work include improving the accuracy and interpretability of the attributes that the procedure identifies. We have identified that embedding a procedure for selecting attributes (that is not based on sampling) is a challenging problem. An improved representation of the ADs may result in more interpretable attributes. We currently represent them by a SQL query, but a graphical representation — where an attribute is represented as the presence or absence of a particular characteristic — may provide additional insights. Finally, we believe that our propositionalization approach can be applied to other classification problems in fields such as marketing and finance.
Appendix: Extended Example of Comparison Refinements

The attribute considered in this example is a strong indicator of mutagenicity. The procedure that created it is a simple way to represent:

*Label each atom different from carbon with the number of atoms of nitrogen to which it is connected.*

*Compute the mean of these labels.*

Consider the path in Figure 8. The elements are denoted by a number to distinguish multiple appearances of the same element (e.g., Atom and AtomBond appear twice). The derived attributes are included in boxes and linked to the elements to which they belong. The following table reports, for each step, the values of `currentEle`, \( A_g \), \( A_r \), the aggregating function and the possible refinement chosen.

Table 11: Steps to generate the attribute.

<table>
<thead>
<tr>
<th>Step</th>
<th>currentEle</th>
<th>Aggr. Function</th>
<th>( A_g )</th>
<th>( A_r )</th>
<th>Ref. Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>-</td>
<td>5.ele</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>MostFrequent</td>
<td>5.ele</td>
<td>5.idAtom</td>
<td>!= 1.idAtom</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>-</td>
<td>3.Derived</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>CountDistinct</td>
<td>3.idBond</td>
<td>2.Derived</td>
<td>= N</td>
</tr>
<tr>
<td>5</td>
<td>Target</td>
<td>Avg</td>
<td>1.Derived</td>
<td>1.ele</td>
<td>!= C</td>
</tr>
</tbody>
</table>

This is a detailed description of the steps in Table 11 and the description of the attribute virtually attached to each element by the Roll-Up algorithm:

1. currentEle is 4. Choose \( A_g \) equal to “5.ele”. This results in attaching 5.ele to element 4.
2. currentEle is 3. Choose \( A_g \) equal to “5.ele” with MostFrequent as aggregating function. Choose \( A_r \) equal to “5.idAtom” and the Comparison refinement “not equal to 1.idAtom”. “1.idAtom” is the atom “on the other side” of the bond. Note that since every bond is connected to exactly 2 atoms, the “most frequent element on the other side” is just the “element on the other side”.
3. currentEle is 2. Choose \( A_g \) equal to the attribute derived at step 2 and attach it to element 2.
4. currentEle is 1. Choose \( A_g \) equal to “3.idBond” with CountDistinct as the aggregating function. Choose \( A_r \) equal to the attribute derived at step 2 and the value refinement “equal to N” (i.e., equal to nitrogen). For each row in element 2 (i.e. for each atom), we derived the number of bonds to which it is connected that have a nitrogen on the other side.
5. currentEle is “Target”. Choose \( A_g \) equal to the attribute derived at the previous step with Avg as the aggregating function. Choose \( A_r \) equal to “1.ele” and the value refinement “not equal to C” (i.e., different from carbon). The final attribute is then obtained.

— Figure 8 —
References


1. currentEle := the last element in the path
do {
    2. currentEle := the element preceding currentEle
    3. followingEle := the element following currentEle
    4. GenerateNextDerivedAttribute(currentEle, followingEle)
    5. add the derived attribute to currentEle
} while (currentEle != targetEle)

Figure 1. Types Graph for the molecule classification problem

Figure 2. Pseudo code of the Roll-Up procedure
Input: currentEle p, followingEle f

If (the association from p to f is 0-1) {
    Choose an attribute from f (if f contains a derived attribute, select it);
    Attach it to p;
}
Else {
    Choose an attribute $A_g$ from f and a compatible aggregating function $agg$;
    Optionally choose a refinement as follows:
    Choose an attribute $A_r$ from f (if f contains a derived attribute, it must be selected either as $A_g$ or as $A_r$);
    If building a toValue refinement {
        Choose a compatible refinement operator $\rho$;
        Choose a value $v$ with the same dimension as $A_r$;
        Attach the following derived attribute to p:
        Select $agg(A_g)$
        From p
        Where $A_r \rho v$;
    }
    Else If building a comparison refinement {
        Choose a compatible refinement operator $\rho$;
        Choose an attribute $C$ with the same type and dimension as $A_r$ and belonging to p or to an element preceding p;
        Attach the following derived attribute to p:
        Select $agg(A_g)$
        From p
        Where $A_r \rho C$;
    }
}

Figure 3. Pseudo code of the $\text{GenerateNextDerivedAttribute}$ procedure
Figure 4. Example of attachment.

Figure 5. Average accuracy for UD-1, UD-2, UD-3 and UD-4
Figure 6. Graphical representations of the results in Tables 7 and 8

Figure 7. A) Mutagenic compound benz(a)anthracene containing four, fused aromatic rings and exclusively double bonds. B) Mutagenic compound 2-amino-dinitro-phenol, exhibiting two nitro groups. C) Flavone, non-estrogen receptor binding. D) 6-hydroxyflavone, an estrogen receptor binding compound. E) 7-hydroxyflavone, non-estrogen receptor binding.
**Figure 8.** Graphical representation of extended example

Avg
(  
  Count distinct (3.idBond)  
  Where  
    (  
      Most Frequent (5.ele)  
      Where 5.idAtom != 1.idAtom  
    ) = N  
  )  
Where 1.ele != C

For each atom different from carbon, the number of atoms of nitrogen to which it is connected

Most Frequent (5.ele)  
Where 5.idAtom != 1.idAtom

Count distinct (3.idBond)  
Where  
  (  
    Most Frequent (5.ele)  
    Where 5.idAtom != 1.idAtom  
  ) = N

For each atom, the number of atoms of nitrogen to which it is connected