Knowledge-Based Learning through Feature Generation

Michal Badian
Knowledge-Based Learning through Feature Generation

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science

Michal Badian

Submitted to the Senate
of the Technion — Israel Institute of Technology
Nisan 5779 Haifa April 2019
This research was carried out under the supervision of Prof. Shaul Markovitch, in the Faculty of Computer Science.

**Acknowledgements**

I would like to thank my advisor Shaul Markovitch for his help during the process. Thanks to my parents, my family and my friends who supported and encouraged me.

The Technion’s funding of this research is hereby acknowledged.
## Contents

### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>1</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>3</td>
</tr>
</tbody>
</table>

### 2 Knowledge-Based Feature Generation Framework | 5
<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Problem Definition</td>
<td>5</td>
</tr>
<tr>
<td>2.2 The Basic Algorithm</td>
<td>6</td>
</tr>
<tr>
<td>2.3 The Secondary Learning Task</td>
<td>6</td>
</tr>
<tr>
<td>2.4 Feature Matching</td>
<td>7</td>
</tr>
<tr>
<td>2.5 Recurrent Feature Generation</td>
<td>8</td>
</tr>
<tr>
<td>2.6 Using Multiple Auxiliary Datasets</td>
<td>9</td>
</tr>
</tbody>
</table>

### 3 Empirical Evaluation | 11
<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Original Datasets</td>
<td>11</td>
</tr>
<tr>
<td>3.2 Creating Feature Generation Tasks</td>
<td>11</td>
</tr>
<tr>
<td>3.3 Testing Protocol</td>
<td>12</td>
</tr>
<tr>
<td>3.4 The Utility of the Generated Features</td>
<td>13</td>
</tr>
<tr>
<td>3.5 Experimenting with a Very Large Auxiliary Dataset</td>
<td>13</td>
</tr>
<tr>
<td>3.6 Experimenting with the recurrent algorithm</td>
<td>14</td>
</tr>
<tr>
<td>3.7 Experimenting with multiple datasets</td>
<td>14</td>
</tr>
</tbody>
</table>

### 4 Related work | 21

### 5 Conclusion | 25

### Hebrew Abstract | i
List of Figures

2.1 The general architecture of the feature generation framework . . . . . . 7

3.1 The effect of the addition of features generated in our algorithm, on the accuracy achieved by decision tree with pruning, on the YELP dataset. 16

3.2 The effect of the addition of features generated in our algorithm, on the accuracy achieved by decision tree with pruning, on the AMAZON dataset. 16
Abstract

Machine learning algorithms have difficulties to generalize over a small set of examples. Humans can perform such a task by exploiting vast amount of background knowledge they possess and applying it on the target task. One method for enhancing learning algorithms with external knowledge is through feature generation. Several feature-generation schemes have been proposed for different types of external knowledge. In this work we propose a method for exploiting external knowledge represented in a standard dataset format. With the explosion of machine learning research, the set of such datasets increases rapidly. In our work, we introduce a new algorithm for generating features based on a collection of auxiliary datasets. We assume that, in addition to the training set, we have access to a set of additional datasets. Unlike the transfer learning setup, we do not assume that the auxiliary datasets represent learning tasks that are similar to our original one. The algorithm finds features that are common to the training set and the auxiliary datasets. Based on these features and on examples from the auxiliary datasets, it induces predictors for new features from the auxiliary datasets. The induced predictors are then added to the original training set as generated features. Our method was tested on a variety of learning tasks, including text classification and medical prediction, and showed a significant improvement over using just the given features.
Chapter 1

Introduction

Machine learning algorithms attempt to learn a predictor by generalizing over a given set of tagged examples. In recent years, we have seen a significant progress in solving complex learning problems using algorithms based on Artificial Neural Networks (ANN). These algorithms perform well when a large set of tagged examples is available, but their performance rapidly declines when the size of the training set decreases.

While machines find the task of inducing over a small set of examples problematic, humans are usually very good at such tasks. A human is capable, in many cases, to induce over an extremely small set of examples using extensive background knowledge. With the increasing availability of large-scale knowledge bases, machine learning would have significantly benefited if it could have utilized a similar knowledge-based approach.

There are three major approaches for using extra knowledge to overcome the lack of examples. In the 80’s, several research groups had persuaded an approach called Explanation Based Learning [MKKC86, DM86], where even a single example can be used for induction. The method works by generating and generalizing a proof that explains the example using background knowledge in the form of a set of logical assertions. This approach is still not practical in most cases due to the lack of such logic-based knowledge bases.

A second approach for solving a learning task with a small training set is transfer learning [PY10]. This method assumes that, in addition to the original training set, there is another, usually larger, set of examples of a related learning task. The above two approaches are limited in the type of background knowledge they can exploit for enhancing the learning process.

A third approach uses external knowledge to generate new features. Two prominent works in this direction are Explicit Semantic Analysis [GM09], where Wikipedia-based conceptual features are generated and added to textual examples, and Word2Vec [MSC+13] where latent concepts, based on a large corpus, are generated and used as features for text classification. These two methods proved to be very effective but they are applicable for text-based learning tasks only.

With the significant growth in the usage of machine learning for various problems
domains, the amount of knowledge encoded in datasets has significantly increased. In
addition, we are seeing a considerable proliferation in the volume of user-contributed
knowledge encoded in relational form. These bodies of external knowledge could have
made a great source for enhancing learning algorithms.

In this work we present a general algorithm that enhances machine learning algo-
rithms with features that are automatically generated using such external knowledge
sources. Given a learning task with a specific training set, our approach takes a set of
external datasets and looks for matching features. For each external dataset, these are
its set of features that are in common with the original features of the training set. We
use these features to learn a classifier or a regressor that predicts the values of other
features in the external dataset. The learned predictors are then applied to the original
dataset to get newly generated features.

To understand this process, let us look at the following illustrative example. Assume
we are given a small training set for predicting the risk of a patient to have Alzheimer
disease. The set of features for this dataset includes, among others, BMI, blood pressure
and HDL cholesterol. We also have access to a much larger dataset for evaluating the
risk of a patient to have diabetes that also includes these three features. Our method
builds a new secondary learning problem, where the examples are those of the diabetes
dataset, the features are the three common features, and the target concept is the risk
of having diabetes. The resulting classifier predicts the risk of diabetes based on these
three features. This classifier will then be applied to the original dataset to yield values
for a newly generated feature – the risk of having diabetes. The generated feature is
likely to enhance the ability of the learning algorithm to induce a predictor for the
original task, predicting the risk of Alzheimer, as diabetes is one of the risk factor for
this disease.

When the features are orthogonal our method might generate wrong features. How-
ever, in most real datasets, the features are not fully orthogonal. Even the Naive Bayes
assumption, that the features are independent given the class, does not hold in most
realistic learning tasks. For labeled auxiliary datasets, we always have at least one
potential generated feature: the class itself. In addition, we perform a wrapper-based
feature selection to eliminate uninformative features. Our experiments show that in all
the datasets used in our experiments, the generated features improved the performance
on the held-out test set.

It is important to note that our method does not compete with alternative feature
generation techniques but complements them. It is perfectly feasible to have two fea-
ture generation processes work in tandem, where one generates features using auxiliary
datasets with our method, while the other generates additional features, for example,
using a text corpus with Word2Vec. Our method can also be integrated with combinat-
national feature generation methods – we first apply our algorithm to generate new
features based on external knowledge, and then apply a combinational method on the
enhanced pool of features.
Chapter 2

Knowledge-Based Feature Generation Framework

In this chapter, we present a feature generation framework that can exploit additional datasets and inject the knowledge embedded in these datasets into the learning process. We start with the basic algorithm and continue with extensions.

2.1 Problem Definition

We assume that we are given a training set to learn from, and an additional set of auxiliary datasets, either labeled or unlabeled. More formally, we define a dataset $D$ as a pair $\langle O, F \rangle$, where $O$ is a set of objects and $F$ is a set of features such that $\forall f \in F : O \subseteq \text{Domain}(f)$. We sometimes denote the set of features of a dataset $D$ by $F(D)$.

Given a dataset $D = \langle O, F \rangle$, a labeled dataset with respect to $D$ and a target function $f^*$, is defined as $\langle E, F \rangle$, where $E = \{ \langle o, f^*(o) \rangle \mid o \in O \}$. We sometimes denote the set of objects of a dataset $D$ by $O(D)$.

A learning algorithm takes a labeled dataset $D_l = \langle E_l, F_l \rangle$ and yields a classifier $f_c : O \rightarrow \text{Range}(f^*)$.

A Feature Generation Task (FGT) is a pair $(D_l, A)$, where $D_l = \langle E_l, F_l \rangle$ is a labeled training set, and $A = \{ D_1, ..., D_k \}$, where $D_i$ is either labeled or unlabeled, is a set of auxiliary datasets.

A knowledge-based feature generation algorithm takes a FGT as an input and generates a set of features $F_g$ over $O(D_l)$. We call the dataset $D_l^g = \langle E_l, F_l \cup F_g \rangle$ the enhanced dataset. We name the main learning algorithm, that will be applied to $D_l^g$, the primary learning algorithm, and denote it by $L_p$.

Our goal is to exploit the auxiliary datasets to improve the quality of the induced predictor. We propose to do so via feature generation as described in the next section.
2.2 The Basic Algorithm

For each auxiliary dataset \( D_a \), our algorithm first finds the set of common features with the training set \( F = F(D_a) \cap F(D_t) \), and the set of features \( F^- = F(D_a) \setminus F(D_t) \) that appears only in the auxiliary dataset.

For each feature \( f \in F^- \), the algorithm constructs a new learning problem, called a secondary learning task, where the objects are \( O(D_a) \), the target function is \( f \), and the features are \( F \). A secondary learning algorithm, denoted by \( L_s \), is then applied to the secondary learning task. Note that \( L_p \) and \( L_s \) are not necessarily related.

The learning process results in a predictor \( f_g \). This is a predictor (a classifier or a regressor) that is based only on the features in \( F \) and therefore can be added to \( F(D_t) \) as a newly generated feature. Before we add it to the growing set of generated features, we first apply a wrapper selector\[KJ97\] and add it only if it has positive utility with respect to the main learner, \( L_p \). The pseudo-code for the algorithm is listed below. \( CV \) stands for cross validation estimation.

\[\begin{aligned}
\text{Algorithm 2.1 Knowledge-Based Feature Generation Algorithm} \\
\text{KBFG(} & D_t, D_a \text{)} \\
\text{Input:} & \quad D_t = \langle E_t, F_t \rangle, D_a = \langle O_a, F_a \rangle \\
1: & \quad F_g \leftarrow \{\} \\
2: & \quad F^{\cap} \leftarrow F_t \cap F_a \\
3: & \quad \text{for } f \in F_a - F^{\cap} \text{ do} \\
4: & \quad \quad \hat{E} \leftarrow \{\langle o, f(o) \rangle \mid o \in O_a\} \\
5: & \quad \quad f_g \leftarrow L_s(\hat{E}, F^{\cap}) \\
6: & \quad \quad \text{if } CV(L_p, F_t \cup F_g \cup \{f_g\}) > CV(L_p, F_t \cup F_g) \text{ then} \\
7: & \quad \quad \quad F_g \leftarrow F_g \cup \{f_g\} \\
8: & \quad \quad \text{end if} \\
9: & \quad \text{end for} \\
10: & \quad \text{return } F_g \\
\end{aligned}\]

2.3 The Secondary Learning Task

The algorithm described in the previous subsection generates new features by creating a new learning problem and solving it. The components of this stage are as follows:

1. The objects of the new problem are the objects of the auxiliary dataset.

2. The target function is a feature in the auxiliary dataset that does not appear in the original dataset. Note that the auxiliary dataset can be either labeled or unlabeled. If it is labeled, we treat its target class as a part of the set of features. Thus, the generated features may include the class.
3. In some datasets, for instance in textual ones, the set of such features is very large. Since each feature requires a full learning session, we may want to prioritize this set. When there exists a distance function between potential features and the original dataset, it can be used for making prioritization. In the experimental section, we describe an experiment where the auxiliary dataset is a very large text corpus (Wikipedia). The distance function we used there is based on Word2Vec embedding. We compute the similarity of the potential feature to the positive training examples, and the similarity to the negative ones. The maximal similarity taken as its utility.

4. The features used for the secondary learning problem are those that are common to the original dataset and the auxiliary dataset. When this set is very large we perform a feature selection.

5. The learning algorithm produces a classifier when the target feature is nominal, and a regressor if it is continuous. In the experiments reported here, we have used a random forest classifier (or regressor) but any other algorithm can be used.

The architecture of our framework is illustrated in Figure 2.1.

2.4 Feature Matching

Our algorithm assumes that \( F^\cap = F_t \cap F_a \) is computable. One way to achieve this is to manually specify a feature matching table by the user. Otherwise, we need an automatic way to do the matching. If the names of the features are given explicitly, such as in the UCI collection [DKT17], and the same names are used in the training dataset and the auxiliary dataset, the matching is straightforward.
Otherwise, we define some similarity measures between features, and match them only if their similarity is above some threshold. If the names are only slightly different, we can use stemming and Levenshtein distance to find matching. If the names differ, we can use embedding such as Word2Vec to estimate the similarity between pairs of feature names. When descriptions for features are given, such as the case in many UCI datasets, we embed their description as well.

If feature names are not given, we can estimate the similarity of two categorical or numerical features by measuring the distance between the distributions of their values, using common distribution distance measures.

2.5 Recurrent Feature Generation

Once algorithm 2.1 generates a feature \( f_g \) and add it to the training set, the intersection set \( F^\cap \) now includes a new member. By the way we defined the process, we now have a state where \( f_g \) is in the enhanced training features, and the feature it approximates, \( f \), exists in the auxiliary set of features. Therefore, we are able to match these two features and expand the intersection set, which may lead to an improvement in the induction of additional generated features. The recurrent version of our algorithm includes one additional line. The pseudo-code of the recurrent version of the algorithm is listed below.

**Algorithm 2.2 Recurrent Knowledge-Based Feature Generation Algorithm**

Input: \( \langle E_t, F_t \rangle, \langle O_a, F_a \rangle \)

1: \( F_g \leftarrow \{\} \)
2: \( F^\cap \leftarrow F_t \cap F_a \)
3: **for** \( f \in F_a - F^\cap \) **do**
4: \( \hat{E} \leftarrow \{\langle o, f(o) \rangle | o \in O_a \} \)
5: \( f_g \leftarrow L_a(\hat{E}, F^\cap) \)
6: **if** \( CV(L_p, F_t \cup F_g \cup \{f_g\}) > CV(L_p, F_t \cup F_g) \) **then**
7: \( F_g \leftarrow F_g \cup \{f_g\} \)
8: \( F^\cap \leftarrow F^\cap \cup \{f_g\} \).
9: **end if**
10: **end for**
11: **return** \( F_g \)

The recurrent version of the algorithm enables to generate new compositions from composed features. It creates hierarchy in the space of constructed features, where the core of this hierarchy is \( F^\cap \).
2.6 Using Multiple Auxiliary Datasets

When we get a set of auxiliary datasets, we can call our basic feature-generation algorithm, \( KBFG \), for each auxiliary dataset sequentially. One drawback of this approach, is that the wrapper feature selector evaluates each generated feature only with respect to the local set of features generated for the particular dataset, and not with respect to the global generated set of features. Therefore, we propose to add an extra feature selection stage that will be performed on the accumulated set of generated features after processing all the auxiliary datasets.

In the basic algorithm, we evaluated each newly generate feature, and added it to the set of generated features, only if it had a positive utility, as judged by cross validation. Now we accumulate a potentially large set of generated features, and try to select only the useful ones. One possible method for doing so is to compute some statistical measure for each feature, such as information gain, and select only the top features. This method, however, evaluates each feature independently of the others.

An alternative that we consider is the wrapper approach, which iteratively evaluates the contribution of each feature with respect to the current set and add the one with the best utility. This, however, has quadratic complexity (with respect to \( n = |\hat{F}_g| \), the number of generated features). With thousands of generated features, it may require millions of cross-validation experiments.

We have therefore designed a hybrid feature selection approach that performs an order of \( n \) cross validation operations. We first sort the features \( \hat{F}_g \) by their information gain, starting with the highest. Then we start to add them to \( F_t \) one by one, evaluating their relative utility using cross validation.

The pseudo-code for the \( KBFG^* \) for feature generation with multiple-auxiliary datasets is listed in Algorithm 2.3.

In the experimental section, we will also consider a variation of \( KBFG^* \), denoted by \( KBFG^*_{ns} \), that disables the internal feature selection and performs only the global one.

There are several other considerations when using multiple datasets:

1. Obviously, there is no point in using an auxiliary dataset if the intersection set is empty.

2. It is possible that a feature \( f \) belongs to multiple datasets and therefore multiple approximations of it, \( F = \{f_g^1, \ldots, f_g^k\} \), will be induced. One option to deal with this case is choosing

\[
  f' = \arg \max_{f \in F} U(f, D_t)
\]

where \( U \) is any feature evaluation method such as wrapper or information gain. Alternatively, we can form a committee \( \{f_g^1, \ldots, f_g^k\} \) that will serve as the new generated feature \( f_g \).
3. We can use the recurrent version of the algorithm with multiple auxiliary datasets. Assume that $f_g$ was generated using an auxiliary dataset $D_a$. We can add this feature to other auxiliary datasets, thus enhancing their expressiveness. This can be done only if the features used to learn it are included there.

**Algorithm 2.3** The $KBFG^*$ For Feature Generation with Multiple Auxiliary Datasets

**Input:** $\langle E_t, F_t \rangle, \mathcal{A}$

```
1: $\tilde{F}_g \leftarrow \{\}$
2: for $(O_a, F_a) \in \mathcal{A}$ do
3:   $F_g = KBFG(\langle E_t, F_t \rangle, \langle O_a, F_a \rangle)$
4:   $\tilde{F}_g \leftarrow \tilde{F}_g \cup F_g$
5: end for
6: Sort $\tilde{F}_g$ by information gain in descending order
7: $G \leftarrow \{\}$
8: for $f_g \in \tilde{F}_g$ do
9:   if $CV(L_p, F_t \cup G \cup \{f_g\}) > CV(L_p, F_t \cup G)$ then
10:      $G \leftarrow G \cup \{f_g\}$
11: end if
12: end for
13: return $G$
```
Chapter 3

Empirical Evaluation

We have tested our framework using an extensive set of training sets with associated auxiliary datasets.

3.1 Original Datasets

We have used 12 original datasets that served as the basis for our feature generation experiments. The first three are textual datasets of reviews tagged with their sentiment. The rest are datasets taken from the UCI [DKT17] and Kaggle repositories. Table 3.1 lists the raw datasets used in our experimentation.

3.2 Creating Feature Generation Tasks

A feature generation task (FGT) is a pair of a training set and a set of auxiliary datasets. For the experiments described here, we always used one auxiliary dataset. One major step in implementing our framework is finding matching between the features of the training set and those of the auxiliary dataset. For the text classification tasks, we

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Origin</th>
<th>#Examples</th>
<th>#Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMDB</td>
<td>UCI</td>
<td>1000</td>
<td>N/A</td>
</tr>
<tr>
<td>AMAZON</td>
<td>UCI</td>
<td>1000</td>
<td>N/A</td>
</tr>
<tr>
<td>YELP</td>
<td>UCI</td>
<td>1000</td>
<td>N/A</td>
</tr>
<tr>
<td>Breast Cancer Coimbra</td>
<td>UCI</td>
<td>116</td>
<td>10</td>
</tr>
<tr>
<td>Pima Indians Diabetes</td>
<td>Kaggle</td>
<td>768</td>
<td>9</td>
</tr>
<tr>
<td>ILPD</td>
<td>UCI</td>
<td>583</td>
<td>10</td>
</tr>
<tr>
<td>Hepatitis</td>
<td>UCI</td>
<td>155</td>
<td>19</td>
</tr>
<tr>
<td>Cylinder</td>
<td>UCI</td>
<td>512</td>
<td>39</td>
</tr>
<tr>
<td>SPECTF</td>
<td>UCI</td>
<td>267</td>
<td>44</td>
</tr>
<tr>
<td>QSAR</td>
<td>UCI</td>
<td>1055</td>
<td>41</td>
</tr>
<tr>
<td>Z-Alizadeh</td>
<td>UCI</td>
<td>303</td>
<td>56</td>
</tr>
<tr>
<td>Cardiotocography</td>
<td>UCI</td>
<td>2126</td>
<td>23</td>
</tr>
</tbody>
</table>

Table 3.1: The details of the original datasets used
have just matched the stemmed words. Thus, we created 6 feature generation tasks out of the 3 sentiment analysis datasets. Pima and Breast Cancer datasets are medical records containing common features such as Glucose and BMI, and therefore could be paired for feature generation. So are the Indian Liver and Hepatitis datasets, with common features such as Bilirubin and Age.

Since we needed more FGTs, we have designed a method of creating an FGT out of one dataset. The examples of the datasets are randomly partitioned to create two disjoint sets of examples, one used for the training and one for the auxiliary dataset. The feature set is also carefully partitioned so that we can control the size of the intersection.

Let $D = \langle E, F \rangle$ be a labeled dataset:

1. Randomly partition $E$ into two nearly equal sized sets $E_1$ and $E_2$.
2. Randomly select a subset from $F$ of size $\mu_1|F|$ ($\mu_1 = \frac{1}{3}$ in our experiments) and mark it as the intersection set, denoted by $F^\cap$.
3. Randomly partition $F - F^\cap$ into two sets: $F_1$ of size $\mu_2|F - F^\cap|$ and $F_2$ ($\mu_2 = \frac{2}{3}$ in our experiments).
4. The training set will be $\langle E_1, F_1 \cup F^\cap \rangle$ and the auxiliary dataset $\langle E_2, F_2 \cup F^\cap \rangle$.

### Table 3.2: The FGTs in our experiments. $\alpha = 0.25$, $F_\alpha = f^\cap \cup f^-$

| Training Set | #Examples | Auxiliary dataset | #Examples | AVG $|F^\cap|$ | AVG $|F^-|$ |
|--------------|-----------|------------------|-----------|---------------|--------------|
| IMDB$^{0.25}$ | 225 | AMAZON | 1000 | 611 | 705 |
| AMAZON$^{0.25}$ | 225 | IMDB | 1000 | 611 | 705 |
| IMDB$^{0.25}$ | 225 | YELP | 1000 | 642 | 873 |
| YELP$^{0.25}$ | 225 | IMDB | 1000 | 642 | 873 |
| AMAZON$^{0.25}$ | 225 | YELP | 1000 | 494 | 822 |
| YELP$^{0.25}$ | 225 | AMAZON | 1000 | 494 | 822 |
| Pima$^{0.25}$ | 172 | Breast Cancer | 116 | 4 | 5 |
| Breast Cancer$^{0.25}$ | 26 | Pima | 768 | 4 | 4 |
| ILPD$^{0.25}$ | 130 | Hepatitis | 80 | 5 | 14 |
| Hepatitis$^{0.25}$ | 18 | ILPD | 579 | 5 | 5 |
| Cylinder$^{0.25}$ | 61 | Cylinder | 265 | 16 | 21 |
| SPECTF$^{0.25}$ | 40 | SPECTF | 137 | 18 | 24 |
| QSAR$^{0.25}$ | 117 | QSAR | 531 | 17 | 22 |
| Z-Alizadeh$^{0.25}$ | 32 | Z-Alizadeh | 159 | 24 | 32 |
| Cardio$^{0.25}$ | 235 | Cardio | 1077 | 9 | 11 |

### 3.3 Testing Protocol

One prominent motivation for our work is to overcome the problem of lack of examples by using external knowledge. Therefore we have followed a special testing protocol that simulates such scenarios by reducing the number of examples in the training set.
Given an FGT \( \langle \langle E_t, F_t \rangle, \langle E_a, F_a \rangle \rangle \):

1. Partition \( E_t \) into \( k \) partitions (10 in our experiments) with testing sets \( T_1, \ldots, T_k \) and corresponding training sets \( \langle E_t - T_i, F_t \rangle \).

2. For each test set \( T_i \) and a set of examples \( B_i = E_t - T_i \):
(a) Reduce the number of examples in \( B_i \) by randomly selecting a subset \( B_i^\alpha \) of size \( \alpha \cdot |B_i| \) (\( \alpha = 0.25 \) in our experiments).
(b) Generate a set \( G_i \) of new features based on the auxiliary dataset.
(c) Sort \( G_i \) by information gain with respect to \( B_i \). For the textual datasets, in which the size of \( F^- \) is large, we select the best 50.
(d) For each \( f \) in the sorted \( G_i \), test if it has a positive utility value by the wrapper method, and add it to \( F_t \) if it does. We denote the enhanced feature set by \( \hat{F}_t \).
(e) Learn from \( \langle B_i^\alpha, \hat{F}_t \rangle \).
(f) Test on \( T_i \).

The list of FGTs used for our experiments is shown in Table 3.2. The superscript next to each dataset stands for the \( \alpha \) parameter used to reduce the number of examples. It is important to note that the target function of the auxiliary was treated as a feature only in the medical FTGs. Otherwise, it was ignored.

### 3.4 The Utility of the Generated Features

We performed a set of experiments to test the utility of our feature generation algorithm. Each experiment involved generating a set of features with a given FGT, and testing the enhanced feature set with various learning algorithms, including Decision Tree, Linear SVM, KNN (\( N = 3 \)) and Multilayer-perceptron. We used the implementation in \texttt{sklearn} with the default parameters.

Tables 3.3, 3.4, 3.5 and 3.6 show the results achieved. For each learning algorithm the tables illustrate the performance on the testing set without and with the generated features. The third column shows the added accuracy. We mark by bold differences that are statistically significant by paired t-test with \( p < 0.05 \).

We can see that the generated features significantly enhance the performance in almost all cases. Note that the improvements are orthogonal to potential other improvements that may be achieved by feature-combination algorithms or by feature generation algorithms that use other external resources.

### 3.5 Experimenting with a Very Large Auxiliary Dataset

We have performed another experiment with textual learning task, where the auxiliary dataset is the whole Wikipedia corpus. We consider each Wikipedia article as one
object. Using such a large dataset is problematic. The Bag-Of-Words matrix is, after filtering, of size $850,000 \times 700,000$ approximately. As the training dataset contains only a few thousands of words (the size of $F_t$), the size of $F^- = F_a - F_t$ is also of size of almost $850,000$.

As we cannot afford initiating $850,000$ learning processes, we applied the similarity-based prioritization as described in subsection 2.3. Specifically, we have used Word2Vec distance to select the 5,000 features closest to the centroid of the positive examples of the training set, and another 5,000 features closest to the negative one.

The results are shown in Table 3.8. We can see that the new features added about 5% in accuracy (statistically significant with $p < 0.05$) with both datasets tested.

### 3.6 Experimenting with the recurrent algorithm

In Section 2.5 we have described a recurrent version of our algorithm, where the generated features are used as the basis for new generated features. We have repeated the decision-tree experiments described in Table 3.3, but this time with the recurrent version of the algorithm. Table 3.7 shows the results obtained. We expected the recurrent version to be superior to the basic version but that was not the case. We can see that while the recurrent version has positive utility, it yields improvements that are less than the basic version. We are not sure what is the reason for that. One possible explanation is a multiplicative noise effect, as each feature learning process can be viewed as a noisy source for the next learner that uses it as an input. One optional remedy for this noise problem is performing cross-validation over the auxiliary dataset to estimate the expected accuracy of each generated feature, and allow using only high-accuracy features as a base for the recurrent process.

### 3.7 Experimenting with multiple datasets

We have designed two feature generation tasks that involve multiple auxiliary datasets:

1. Training set: YELP. Auxiliary datasets: IMBD and AMAZON.
2. Training set: AMAZON. Auxiliary datasets: IMBD and YELP

We have applied our $KBFG^*$ Algorithm (see Algorithm 2.3) and its variant the $KBFG_{nis}^*$ algorithm to the two tasks described. For comparison, we have also tested the basic $KBFG$ algorithm with each of the auxiliary datasets separately.

We ran each of the 4 scenarios until 50 generated features passed the wrapper filter. For the $KBFG^*$ and $KBFG_{nis}^*$ we refer to the global wrapper filter.

The experiment was stopped after each 5 features were accepted and the accuracy of the classifier induced with the added features was measured as described in Section 3.3. That is, the pool of 50 ordered features, was split into 10 groups of 5 items per
each. Each point in the graphs, is an addition of the next ranked group of 5 features, where the zero point represent the original set of features (i.e., without any additional features).

It is important to mention that the accuracy was measured using the same folds. That is, the only different is the features added using the single or multiple auxiliary. The number of the features and training set examples, are preserved.

The graphs in figure 3.1 show the average accuracy of a classifier induced by decision tree with pruning with the added generated features, on the YELP dataset, for the 4 scenarios.

There are two observations to be made. First, the $KBFG^*$ algorithm had significant advantage over the $KBFG$ with a single auxiliary dataset (about 10% difference in accuracy). Second, the $KBFG^*_{nis}$ version was not as successful and in fact, had a performance comparable to the basic $KBFG$ algorithm with the AMAZON auxiliary. It is still advantageous to use $KBFG^*_{nis}$ as we cannot predict which of the single auxiliary datasets will be the best.

The same experiment has been done on the AMAZON dataset. The graphs in figure 3.2 show the results. Here there was no large difference between the two variations of the multiple datasets algorithms. In some parts of the graph they have up to 5% advantage over the best single dataset scenario, but this advantage diminishes when more features are added.
Figure 3.1: The effect of the addition of features generated in our algorithm, on the accuracy achieved by decision tree with pruning, on the YELP dataset.

Figure 3.2: The effect of the addition of features generated in our algorithm, on the accuracy achieved by decision tree with pruning, on the AMAZON dataset.
<table>
<thead>
<tr>
<th>Training Set</th>
<th>Auxiliary dataset</th>
<th>DT</th>
<th>DT + FG</th>
<th>Diff DT</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMDB$^{0.25}$</td>
<td>AMAZON</td>
<td>0.566</td>
<td>0.621</td>
<td>+0.055</td>
</tr>
<tr>
<td>AMAZON$^{0.25}$</td>
<td>IMDB</td>
<td>0.542</td>
<td>0.615</td>
<td>+0.073</td>
</tr>
<tr>
<td>IMDB$^{0.25}$</td>
<td>YELP</td>
<td>0.539</td>
<td>0.610</td>
<td>+0.071</td>
</tr>
<tr>
<td>YELP$^{0.25}$</td>
<td>IMDB</td>
<td>0.554</td>
<td>0.612</td>
<td>+0.058</td>
</tr>
<tr>
<td>AMAZON$^{0.25}$</td>
<td>YELP</td>
<td>0.593</td>
<td>0.672</td>
<td>+0.079</td>
</tr>
<tr>
<td>YELP$^{0.25}$</td>
<td>AMAZON</td>
<td>0.553</td>
<td>0.666</td>
<td>+0.113</td>
</tr>
<tr>
<td>Pima$^{0.25}$</td>
<td>Breast Cancer</td>
<td>0.751</td>
<td>0.828</td>
<td>+0.077</td>
</tr>
<tr>
<td>Breast Cancer$^{0.25}$</td>
<td>Pima</td>
<td>0.721</td>
<td>0.738</td>
<td>+0.017</td>
</tr>
<tr>
<td>ILPD$^{0.25}$</td>
<td>Hepatitis</td>
<td>0.660</td>
<td>0.694</td>
<td>+0.034</td>
</tr>
<tr>
<td>Hepatitis$^{0.25}$</td>
<td>ILPD</td>
<td>0.700</td>
<td>0.774</td>
<td>+0.074</td>
</tr>
<tr>
<td>Cylinder$^{0.25}$</td>
<td>Cylinder</td>
<td>0.633</td>
<td>0.692</td>
<td>+0.059</td>
</tr>
<tr>
<td>SPECTF$^{0.25}$</td>
<td>SPECTF</td>
<td>0.615</td>
<td>0.721</td>
<td>+0.106</td>
</tr>
<tr>
<td>QSAR$^{0.25}$</td>
<td>QSAR</td>
<td>0.686</td>
<td>0.732</td>
<td>+0.046</td>
</tr>
<tr>
<td>Z-Alizadeh$^{0.25}$</td>
<td>Z-Alizadeh</td>
<td>0.621</td>
<td>0.718</td>
<td>+0.097</td>
</tr>
<tr>
<td>Cardio$^{0.25}$</td>
<td>Cardio</td>
<td>0.790</td>
<td>0.816</td>
<td>+0.026</td>
</tr>
</tbody>
</table>

Table 3.3: The effect of feature generation on the performance of Decision Tree With pruning.

<table>
<thead>
<tr>
<th>Training Set</th>
<th>Auxiliary dataset</th>
<th>SVM</th>
<th>SVM + FG</th>
<th>Diff SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMDB$^{0.25}$</td>
<td>AMAZON</td>
<td>0.695</td>
<td>0.709</td>
<td>+0.014</td>
</tr>
<tr>
<td>AMAZON$^{0.25}$</td>
<td>IMDB</td>
<td>0.732</td>
<td>0.735</td>
<td>+0.003</td>
</tr>
<tr>
<td>IMDB$^{0.25}$</td>
<td>YELP</td>
<td>0.691</td>
<td>0.711</td>
<td>+0.020</td>
</tr>
<tr>
<td>YELP$^{0.25}$</td>
<td>IMDB</td>
<td>0.695</td>
<td>0.710</td>
<td>+0.015</td>
</tr>
<tr>
<td>AMAZON$^{0.25}$</td>
<td>YELP</td>
<td>0.720</td>
<td>0.749</td>
<td>+0.029</td>
</tr>
<tr>
<td>YELP$^{0.25}$</td>
<td>AMAZON</td>
<td>0.696</td>
<td>0.717</td>
<td>+0.021</td>
</tr>
<tr>
<td>Pima$^{0.25}$</td>
<td>Breast Cancer</td>
<td>0.576</td>
<td>0.703</td>
<td>+0.127</td>
</tr>
<tr>
<td>Breast Cancer$^{0.25}$</td>
<td>Pima</td>
<td>0.602</td>
<td>0.688</td>
<td>+0.086</td>
</tr>
<tr>
<td>ILPD$^{0.25}$</td>
<td>Hepatitis</td>
<td>0.550</td>
<td>0.706</td>
<td>+0.156</td>
</tr>
<tr>
<td>Hepatitis$^{0.25}$</td>
<td>ILPD</td>
<td>0.790</td>
<td>0.840</td>
<td>+0.050</td>
</tr>
<tr>
<td>Cylinder$^{0.25}$</td>
<td>Cylinder</td>
<td>0.578</td>
<td>0.610</td>
<td>+0.032</td>
</tr>
<tr>
<td>SPECTF$^{0.25}$</td>
<td>SPECTF</td>
<td>0.677</td>
<td>0.812</td>
<td>+0.135</td>
</tr>
<tr>
<td>QSAR$^{0.25}$</td>
<td>QSAR</td>
<td>0.726</td>
<td>0.791</td>
<td>+0.065</td>
</tr>
<tr>
<td>Z-Alizadeh$^{0.25}$</td>
<td>Z-Alizadeh</td>
<td>0.619</td>
<td>0.706</td>
<td>+0.087</td>
</tr>
<tr>
<td>Cardio$^{0.25}$</td>
<td>Cardio</td>
<td>0.757</td>
<td>0.804</td>
<td>+0.047</td>
</tr>
</tbody>
</table>

Table 3.4: The effect of feature generation on the performance of linear SVM.
### Table 3.5: The effect of feature generation on the performance of KNN with K=3.

<table>
<thead>
<tr>
<th>Training Set</th>
<th>Auxiliary dataset</th>
<th>3NN</th>
<th>3NN + FG</th>
<th>Diff 3NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMDB(^{0.25})</td>
<td>AMAZON</td>
<td>0.573</td>
<td>0.610</td>
<td>+0.037</td>
</tr>
<tr>
<td>AMAZON(^{0.25})</td>
<td>IMDB</td>
<td>0.609</td>
<td>0.655</td>
<td>+0.046</td>
</tr>
<tr>
<td>IMDB(^{0.25})</td>
<td>YELP</td>
<td>0.552</td>
<td>0.587</td>
<td>+0.035</td>
</tr>
<tr>
<td>YELP(^{0.25})</td>
<td>IMDB</td>
<td>0.567</td>
<td>0.607</td>
<td>+0.040</td>
</tr>
<tr>
<td>AMAZON(^{0.25})</td>
<td>YELP</td>
<td>0.609</td>
<td>0.659</td>
<td>+0.050</td>
</tr>
<tr>
<td>YELP(^{0.25})</td>
<td>AMAZON</td>
<td>0.593</td>
<td>0.659</td>
<td>+0.066</td>
</tr>
<tr>
<td>Pima(^{0.25})</td>
<td>Breast Cancer</td>
<td>0.673</td>
<td>0.729</td>
<td>+0.056</td>
</tr>
<tr>
<td>Breast Cancer(^{0.25})</td>
<td>Pima</td>
<td>0.531</td>
<td>0.548</td>
<td>+0.017</td>
</tr>
<tr>
<td>ILPD(^{0.25})</td>
<td>Hepatitis</td>
<td>0.646</td>
<td>0.675</td>
<td>+0.029</td>
</tr>
<tr>
<td>Hepatitis(^{0.25})</td>
<td>ILPD</td>
<td>0.814</td>
<td>0.840</td>
<td>+0.026</td>
</tr>
<tr>
<td>Cylinder(^{0.25})</td>
<td>Cylinder</td>
<td>0.547</td>
<td>0.570</td>
<td>+0.023</td>
</tr>
<tr>
<td>SPECTF(^{0.25})</td>
<td>SPECTF</td>
<td>0.635</td>
<td>0.729</td>
<td>+0.094</td>
</tr>
<tr>
<td>QSAR(^{0.25})</td>
<td>QSAR</td>
<td>0.706</td>
<td>0.762</td>
<td>+0.056</td>
</tr>
<tr>
<td>Z-Alizadeh(^{0.25})</td>
<td>Z-Alizadeh</td>
<td>0.575</td>
<td>0.711</td>
<td>+0.136</td>
</tr>
<tr>
<td>Cardio(^{0.25})</td>
<td>Cardio</td>
<td>0.708</td>
<td>0.799</td>
<td>+0.091</td>
</tr>
<tr>
<td>AVG Improvement</td>
<td></td>
<td></td>
<td></td>
<td>+0.054</td>
</tr>
</tbody>
</table>

### Table 3.6: The effect of feature generation on the performance of Multi-layer perceptron.

<table>
<thead>
<tr>
<th>Training Set</th>
<th>Auxiliary dataset</th>
<th>MLP</th>
<th>MLP + FG</th>
<th>Diff MLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMDB(^{0.25})</td>
<td>AMAZON</td>
<td>0.709</td>
<td>0.741</td>
<td>+0.032</td>
</tr>
<tr>
<td>AMAZON(^{0.25})</td>
<td>IMDB</td>
<td>0.752</td>
<td>0.771</td>
<td>+0.019</td>
</tr>
<tr>
<td>IMDB(^{0.25})</td>
<td>YELP</td>
<td>0.715</td>
<td>0.738</td>
<td>+0.023</td>
</tr>
<tr>
<td>YELP(^{0.25})</td>
<td>IMDB</td>
<td>0.733</td>
<td>0.747</td>
<td>+0.014</td>
</tr>
<tr>
<td>AMAZON(^{0.25})</td>
<td>YELP</td>
<td>0.739</td>
<td>0.773</td>
<td>+0.034</td>
</tr>
<tr>
<td>YELP(^{0.25})</td>
<td>AMAZON</td>
<td>0.724</td>
<td>0.749</td>
<td>+0.025</td>
</tr>
<tr>
<td>Pima(^{0.25})</td>
<td>Breast Cancer</td>
<td>0.568</td>
<td>0.664</td>
<td>+0.096</td>
</tr>
<tr>
<td>Breast Cancer(^{0.25})</td>
<td>Pima</td>
<td>0.547</td>
<td>0.608</td>
<td>+0.061</td>
</tr>
<tr>
<td>ILPD(^{0.25})</td>
<td>Hepatitis</td>
<td>0.658</td>
<td>0.727</td>
<td>+0.069</td>
</tr>
<tr>
<td>Hepatitis(^{0.25})</td>
<td>ILPD</td>
<td>0.795</td>
<td>0.837</td>
<td>+0.042</td>
</tr>
<tr>
<td>Cylinder(^{0.25})</td>
<td>Cylinder</td>
<td>0.563</td>
<td>0.582</td>
<td>+0.019</td>
</tr>
<tr>
<td>SPECTF(^{0.25})</td>
<td>SPECTF</td>
<td>0.576</td>
<td>0.710</td>
<td>+0.134</td>
</tr>
<tr>
<td>QSAR(^{0.25})</td>
<td>QSAR</td>
<td>0.714</td>
<td>0.703</td>
<td>-0.011</td>
</tr>
<tr>
<td>Z-Alizadeh(^{0.25})</td>
<td>Z-Alizadeh</td>
<td>0.590</td>
<td>0.728</td>
<td>+0.138</td>
</tr>
<tr>
<td>Cardio(^{0.25})</td>
<td>Cardio</td>
<td>0.710</td>
<td>0.773</td>
<td>+0.063</td>
</tr>
<tr>
<td>AVG Improvement</td>
<td></td>
<td></td>
<td></td>
<td>+0.051</td>
</tr>
</tbody>
</table>
### Table 3.7: The effect of recurrent feature generation algorithm on the performance of Decision Tree With pruning.

<table>
<thead>
<tr>
<th>Training Set</th>
<th>Auxiliary dataset</th>
<th>DT</th>
<th>DT + RFG</th>
<th>Diff DT</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMDB&lt;sup&gt;0.25&lt;/sup&gt;</td>
<td>AMAZON</td>
<td>0.566</td>
<td>0.579</td>
<td>+0.013</td>
</tr>
<tr>
<td>AMAZON&lt;sup&gt;0.25&lt;/sup&gt;</td>
<td>IMDB</td>
<td>0.542</td>
<td>0.572</td>
<td>+0.030</td>
</tr>
<tr>
<td>IMDB&lt;sup&gt;0.25&lt;/sup&gt;</td>
<td>YELP</td>
<td>0.539</td>
<td>0.541</td>
<td>+0.002</td>
</tr>
<tr>
<td>YELP&lt;sup&gt;0.25&lt;/sup&gt;</td>
<td>IMDB</td>
<td>0.554</td>
<td>0.579</td>
<td>+0.025</td>
</tr>
<tr>
<td>AMAZON&lt;sup&gt;0.25&lt;/sup&gt;</td>
<td>YELP</td>
<td>0.593</td>
<td>0.602</td>
<td>+0.009</td>
</tr>
<tr>
<td>YELP&lt;sup&gt;0.25&lt;/sup&gt;</td>
<td>AMAZON</td>
<td>0.553</td>
<td>0.565</td>
<td>+0.012</td>
</tr>
<tr>
<td>Pima&lt;sup&gt;0.25&lt;/sup&gt;</td>
<td>Breast Cancer</td>
<td>0.751</td>
<td>0.761</td>
<td>+0.010</td>
</tr>
<tr>
<td>Breast Cancer&lt;sup&gt;0.25&lt;/sup&gt;</td>
<td>Pima</td>
<td>0.721</td>
<td>0.768</td>
<td>+0.047</td>
</tr>
<tr>
<td>ILPD&lt;sup&gt;0.25&lt;/sup&gt;</td>
<td>Hepatitis</td>
<td>0.660</td>
<td>0.673</td>
<td>+0.013</td>
</tr>
<tr>
<td>Hepatitis&lt;sup&gt;0.25&lt;/sup&gt;</td>
<td>ILPD</td>
<td>0.700</td>
<td>0.704</td>
<td>+0.004</td>
</tr>
<tr>
<td>Cylinder&lt;sup&gt;0.25&lt;/sup&gt;</td>
<td>Cylinder</td>
<td>0.633</td>
<td>0.728</td>
<td>+0.095</td>
</tr>
<tr>
<td>SPECTF&lt;sup&gt;0.25&lt;/sup&gt;</td>
<td>SPECTF</td>
<td>0.615</td>
<td>0.620</td>
<td>+0.005</td>
</tr>
<tr>
<td>QSAR&lt;sup&gt;0.25&lt;/sup&gt;</td>
<td>QSAR</td>
<td>0.686</td>
<td>0.719</td>
<td>+0.033</td>
</tr>
<tr>
<td>Z-Alizadeh&lt;sup&gt;0.25&lt;/sup&gt;</td>
<td>Z-Alizadeh</td>
<td>0.621</td>
<td>0.672</td>
<td>+0.051</td>
</tr>
<tr>
<td>Cardio&lt;sup&gt;0.25&lt;/sup&gt;</td>
<td>Cardio</td>
<td>0.790</td>
<td>0.814</td>
<td>+0.024</td>
</tr>
<tr>
<td>AVG Improvement</td>
<td></td>
<td></td>
<td></td>
<td>+0.025</td>
</tr>
</tbody>
</table>

### Table 3.8: The results of generating word-based features using the Wikipedia corpus. The improvement (almost 5%) is statistically significant with \( p < 0.05 \).

<table>
<thead>
<tr>
<th>Training Set</th>
<th>Auxiliary dataset</th>
<th>DT</th>
<th>DT + FG</th>
<th>Diff DT</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMDB&lt;sup&gt;0.25&lt;/sup&gt;</td>
<td>Wikipedia</td>
<td>0.593</td>
<td>0.640</td>
<td>+0.047</td>
</tr>
<tr>
<td>AMAZON&lt;sup&gt;0.25&lt;/sup&gt;</td>
<td>Wikipedia</td>
<td>0.569</td>
<td>0.742</td>
<td>+0.052</td>
</tr>
<tr>
<td>AVG Improvement</td>
<td></td>
<td></td>
<td></td>
<td>+0.050</td>
</tr>
</tbody>
</table>
Chapter 4

Related work

Many feature generation methods have been developed in an attempt to construct new features that better represent the target concepts.

The most common approach for feature generation (feature engineering) is by constructing new features that are combinations of the given ones. The early GALA algorithm [HK96] utilizes a set of logical operators to combine Boolean attributes. Another example is the LFC [RRST93] algorithm that combines binary features through the use of logical operators such as $\lor, \neg$.

The CITRE algorithm [MR89] constructs new features by combining basic features using decision trees. The FICUS algorithm [MR02] presents a general framework that, given a set of constructing functions and operators, applies them over existing features, included generated ones. The more recent ExploreKit algorithm [KSS16] uses a similar technique to generate a large set of candidate features. A ranking classifier, trained on previously analyzed datasets, assigns a score on each of the candidate features. Based on a threshold, the candidates are filtered and those who remained, proceed to the next iterations.


The work of [TXZ16] uses genetic programming (GP) with an embedded approach. That is, the GP tree used in the algorithm, can be a feature or a classifier. The leaves of the trees are either original feature values or random constant values, while the inner leaves are operators or functions chosen from a predefined set of functions.

Cognito [KTSP16] performs automatic feature engineering by exploring various feature construction choices in a hierarchical and non-exhaustive approach. It builds a tree called transformation graph where the root corresponds to the input dataset and each other node corresponds to a transformed dataset. An edge represents a transform, which when applied to a parent, yields the child dataset. The nodes in the tree are associated with an accuracy measure and the framework uses greedy heuristics to search for an optimal node corresponding to this measure. Another work which performs a
tree traversal on the transformation graph built from the data, is [KST17]. It uses
reinforcement learning (RL) to learn an efficient exploration strategy of the graph. The
AutoLearn algorithm [KMP17] identifies associations between feature pairs and uses
a regularized regression model to construct a new feature from each pair. The LFE
algorithm [NSK+17] learns patterns between feature characteristics, class distributions,
and useful transformations, in order to construct a more effective generated feature.
The learning is based on past transformation experiences and examination of their
influence on the performance.

All these methods are based on the assumption that merely combining existing
features in sophisticated manner is sufficient to allow a learner using the combined
features to yield better classifiers than by using only the original ones.

Another group of feature-generation methods use external knowledge to generate
new features. Our algorithm belongs to this class of approaches. Explicit Semantic
Analysis (ESA) [GM09] uses semantic concepts extracted from knowledge sources such
as Wikipedia as features for text classification tasks. Word2Vec [MSC+13] generates
latent concepts based on a large corpus that can also be used for text classification.

Other Algorithms for specific problem domains use domain background knowledge
to construct special features. The bootstrapping algorithm was applied in the domain
of molecular biology [HJ94] to generate new features by using an initial set of feature
sequences produced by human experts and by using a special set of operators. The
features are represented as nucleotides sequences whose structure is modified by biology-
based operators determined by existing background knowledge.

Propositionalization approaches [KF00] rely on relational data to serve as external
knowledge. They take advantage of several operators to create first-order logic predi-
cates connecting existing data and relational knowledge. [CKG+11] devised a generic
propositionalization framework using linked data via relation-based queries. FeGeLOD
[PF12] also utilizes linked data to automatically enrich existing datasets. FeGeLOD
uses feature values as entities and adds related knowledge to the example, thus creating
additional features. OneBM [LTS+17] works with multiple raw tables in a database. It
joins the tables and applies corresponding predefined transformation functions on the
given features types.

Some learning algorithms employ an internal process that can be viewed as feature
generation. One example is the family of multilayer Neural Network (NN) algorithms
which includes Deep learning algorithms [LBBH98]. The activation functions of nodes
in the hidden layers together with the weights of their inputs can be considered as
features that had been built during the training process. Other ensemble models,
such as Random Forest (RF), use randomization to create data subsets where the
new generated features are the output of decision trees applied on these data subsets.
Dimensional reduction methods, such as Principal Component Analysis (PCA) and
its non linear variants (Kernel PCA), map the input dataset into a lower-dimensional
space with fewer features.
Another approach which can be nominated for feature construction is Stacked Generalization. A common method for using a high-level model to combine lower level models and by that achieving greater predictive accuracy [Wol92]. The HME algorithm [JJ94] is a hierarchical mixture model with stacking architecture, where both the mixture coefficients and the mixture components are generalized linear models. There are two main decision issues in stacked generalization. The first is choosing the type of generalizer that is suitable to derive the higher-level model and the second is the features that are used as its input [TW97]. The generalizers in the different levels of the stacking hierarchy, can be heterogeneous [D04].

Feature generation may yield a large pool of features. This might make the problem too complex with irrelevant and high dimensional data [GE03]. Therefore, feature selection is frequently applied [MBN02]. Machine learning algorithms differ in the amount of emphasis they place on feature selection. Some algorithms, such as the nearest neighbour learner, use all the available features in their distance computations. Other algorithms such as decision trees, try to focus on the relevant features and ignore irrelevant ones. By testing the values of certain features, decision tree algorithms attempt to divide training data into subsets containing a strong majority of one class. Regardless of whether a learner attempts to select features itself or ignores the issue, feature selection prior to learning can be beneficial. In our work, we used it as integral part of the algorithm, and by that remained with potential top useful features from the generation process.

Subset feature selection has two common approaches: The filter approach applies various statistical methods, such as mutual information [PLD05], to select the most promising features (independently of the learning algorithm). The wrapper approach [HS99] searches the space of feature subsets using the intended learning algorithm itself, to evaluate the utility of each candidate feature in the context of the already selected ones. The wrapper approach is generally considered to produce better feature subsets but runs much more slowly than a filter method, which evaluates each feature independently according to heuristics based on general characteristics of the data. Our method used the wrapper approach of Cross-Validation operation as a filter method to evaluate the features.

Our algorithm has been proved to be most useful on problems with lack of examples. Another work that deals with this kind of problem, is using transfer learning approach. It suggested using metaphors in order to transform the feature space (using linear, geometric and polynomial transformations) [LM12]. A metaphor, is a mapping function from the target learning task to the source, which preserves label and probability. The sources were related tasks. The main drawback of this approach, is that it works only when the source and the target tasks are related. Otherwise, the concepts are too distant for a simple metaphor to describe their relation.
Chapter 5

Conclusion

One of the advantages of human learners over machine learning algorithms is their ability to exploit excessive background knowledge during the induction process. This ability allows them to generalize even when a small set of labeled examples is given. In this work, we presented a novel algorithm that allows machine learning procedures to similarly exploit external knowledge via feature generation.

The core of our method is the realization that many datasets deal with similar types of objects, such as people and texts. Thus, many datasets are using common features. Given a dataset to learn from, our method utilizes its common features with other available datasets to learn to predict additional features from these external datasets. The predictors are then added as generated features to the original dataset. We have shown that our method significantly enhances the performance of existing learning algorithms.

The features generated using our method are not replacing the original features but added to them. Therefore, other methods that use external knowledge can still be applied, potentially even better as they now have access to an enhanced feature set. Similarly, feature-combination algorithms, such as Cognito [KTSP16], LFE [NSK+17] and ExploreKit [KSS16], can now use the generated features, in addition to the original set of features, in the combination functions.

One direction we plan to explore is trying to utilize a collection of medical datasets, such as MIMIC [joh16], as auxiliary source for our framework for solving medical learning tasks. To do so we will need to come out with a solid feature mapping scheme. We believe that using our method with this extensive knowledge base may significantly enhance the performance of the learned classifiers in this important domain.
Bibliography


After the treatment, the AI system improved its performance on the new dataset, showing a significant reduction in the error rate compared to the original data. Moreover, the system was able to generalize well to unseen data, maintaining high accuracy across different scenarios.

In summary, the developed approach demonstrated robustness and adaptability, outperforming existing methods in terms of both accuracy and generalization. Further research is warranted to explore the potential of similar approaches in other domains with high-dimensional data.
בשנים האחרונים אנחנורואיםيجيدبولשיטותשלבשנים分辨率יפונהואלגוריתמילמידה(דוגמת רשתות נוירונים) לפתרון בעיות במנון.

בר של החמטים במכונה מתוחכם, רוב השיטות מסייעות לשלב התוכנה Müslüימה את התוויות בзыва לעזרת דמיון כל החתי. אם יכול ריבוסוספק.

הتحديים של הספיפת שניוっぱול הערדה יصعب קוברה וידור התוכנה ודרישות תוכנה קדימה.

דרפה פורמליזםיהזוהיאלייצירתתכונות חדשות אשר ייכלונלואתמושגהמטרה turmoil יותר. השיטה הנפוצה בניהנדסתתכונות היא באמצעות תכניות של התוכנה הקמות לע יד התוכנה פקטור או אופרטורים. בעניין זה, יש להחליש את תוכניינותי,艺术品 ודרישות תוכניינוניות שאלקטיוס.

דרך נפוצה להתמודדות עם בעיה זו היא לייצר תוכנה חדשית אשר יכלה ייצירת תוכניינוניות במטרה לעזרת חיות ונדרשות תוכניינוניות חדשות.

דר הדרכה של התוכנה ת(Constructor לעבר התוכנה VLAN מחירה של קוברים הקמות לע)

דרכה הנפוצה בתוכנה בiards הנושאים שיש להשתלב בתוכנה של התוכנה הקמות על יד התוכנה פקטור או אופרטורים. בעניין זה, יש להחליש את תוכניינוניותי,艺术品 ודרישות תוכניינוניות שאלקטיוס.

דר הדרכה של התוכנה ת(Constructor לעבר התוכנה VLAN מחירה של קוברים הקמות לע)

דר הדרכה של התוכנה ת(Constructor לעבר התוכנה VLAN מחירה של קוברים הקמות לע)
המחקר בוצע בהנחייה של פרופסור שאול מרקוביץ' בפקולטה למדעי המחשב.

תודה

אני רוצה להודות למנהלו של שאול מרקוביץ', על עזרתו הרב במהלך התהליך. תודה גם לאריה, משפחתו והברית על כלしましょう והמודר ב.

הכרת תודה מוסריה المتوך על פימו מחקר זה.
למידת מבוססת ידע
באימון ייצוג תכונות

תובור על מחקר

לשם مليוי חלקי של הדירישות לقبلת התחרור
מוניטור למידועים במדעי המחשב

מיכל بدיאן

הוג של ליסט הטכניון – מרכז טכנולוגיה לישראל
נוף התיישバル חיפה אפריל 2019
למידה מבוססתידע
באמצות ייזור תכונות

ميיקל בדיאן