Recursive Feature Generation for Knowledge-based Learning

Paper ID: 538
Keywords: ML: Feature Construction/Reformulation, AIW: Exploiting Linked Open Data, ML: Classification, NLPML: Text Classification

Abstract

When humans perform inductive learning they often enhance the process with background knowledge. With the increasing availability of well-formed collaborative knowledge bases, the performance of learning algorithms could be significantly enhanced if a way were found to exploit these knowledge bases. In this work we present a novel algorithm for injecting external knowledge into induction algorithms using feature generation. Given a feature, the algorithm defines a new learning task over its set of values, and uses the knowledge base to solve the constructed learning task. The resulting classifier is then used as a new feature for the original problem. We have applied our algorithm to the domain of text classification using large semantic knowledge bases. We have shown that the generated features significantly improve the performance of existing learning algorithms.

Introduction

In recent decades, machine learning techniques have become more prevalent in a wide variety of fields. Most of these methods rely on the inductive approach: they attempt to locate a hypothesis that is heavily supported by given labeled examples. These methods have proven themselves successful when the number of examples is sufficient, and a collection of good, distinguishing features is available. In many real-world applications, however, the given set of features is insufficient for inducing a high quality classifier (**?**; **?**).

One approach for overcoming this difficulty is to generate new features that are combinations of the given ones. For example, the LFC algorithm (**?**) combines binary features through the use of logical operators such as $\wedge, \neg$. Another example is the LDMT algorithm (**?**), which generates linear combinations of existing features. Deep Learning methods combine basic and generated features using various activation functions such as sigmoid or softmax. The FICUS framework (**?**) presents a general method for generating features using any given set of constructor functions.

The above approaches are limited in that they merely combine existing features to make the representation more suitable for the learning algorithm. While this approach often suffices, there are many cases where simply combining existing features is not sufficient. When people perform inductive learning, they usually rely on a vast body of background knowledge to make the process more effective (**?**). For example, assume two positive examples of people suffering from some genetic disorder, where the value of the country-of-origin feature is Poland and Romania. Existing induction algorithms, including those generating features by combination, will not be able to generalize over these two examples. Humans, on the other hand, can easily generalize and generate a new feature, *Eastern Europe*, based on their previously established background knowledge.

In this work, we present a novel algorithm that uses a similar approach for enhancing inductive learning with background knowledge through feature generation. Our method assumes a given body of knowledge represented in relational form. The algorithm treats feature values as objects and constructs new learning problems, using the background relational knowledge as their features. The resulting classifiers are then used as new generated features. For example, in the above very simple example, our algorithm would have used the *continent* and *region* features of a country, inferred from a geographic knowledge base, to create the new feature that enables us to generalize. One significant advantage of using background knowledge in the form of generated features is that it allows us to utilize existing powerful learning algorithms for the induction process.

We have implemented our algorithm in the domain of text classification problems, using Freebase and YAGO2 as our background knowledge bases and performed an extensive set of experiments to test the effectiveness of our method. Results show that the use of background knowledge through our methodology significantly improves the performance of existing learning algorithms.

Motivation

Before we delve into the detailed description of our algorithm, we would like to illustrate its main ideas using an example. Suppose we are attempting to identify people with a high risk of suffering from a genetic disorder. Assume that the target concept to be discovered is that those at risk are women with ancestors originating from desert areas. To identify women at risk, we are given a training sample of sick and healthy people, containing various features, including gender and their full name. We call this learning problem T_1. Assuming we have no additional information, an induction algorithm (a decision tree learner, in this example) would likely produce a result similar to that shown in Figure 1. While such a classifier will achieve a low training error, the hundreds of seemingly unrelated surnames will cause it to generalize poorly.

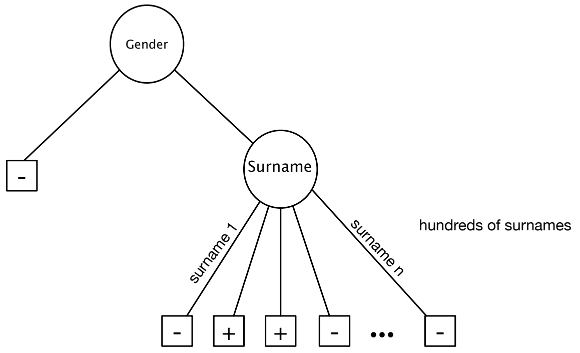


Figure 1: A decision tree for the basic features

The above example illustrates a case where, without additional knowledge, an induction algorithm will yield a very poor result. However, if we assume access to a relational knowledge base connecting surnames to common countries of origin, we can begin to apply our knowledge-based feature generation techniques to the problem, as we can move from the domain of surnames to that of countries. Our algorithm does so by creating a new learning problem T_2. The training objects for learning problem T_2 are surnames; surnames of people at risk are labeled as positive. The features for these new objects are extracted from the knowledge base. In this case, we have a single feature: the country of origin. Solving the above learning problem through an induction algorithm yields a classifier on surnames that distinguishes between surnames of patients with the disease and surnames of healthy individuals. This classifier for T_2 can then be used as a binary feature for the original problem T_1 by applying it to the feature value of surname. For example, it can be used as a feature in the node corresponding a gender of female in Figure 1, yielding the tree seen in Figure 2.

This new feature gives us a better generalization over the baseline solution, as we now abstract the long list of surnames to a short list of countries. This result also allows us to capture previously unseen surnames from those countries. However, this is not a sufficient solution, as we have no way of generalizing on previously unseen countries of origin.

If, however, we would have recursively applied our method for solving T_2, we could have obtained a better generalization. When learning to classify surnames, our method creates a new learning problem, T_3, with countries as its objects. Countries of surnames belonging to people with high risk are labeled as positive. The knowledge base regarding countries is then used to extract features for this new training set. Applying a standard learning algorithm to T_3 will yield a classifier that separates between countries of origin of people at risk and those not at risk. This classifier will do so by looking at the properties of countries, and conclude

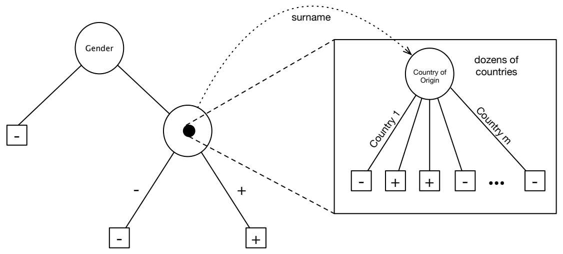


Figure 2: A constructed feature used within a decision tree

that countries with high average temperature and low precipitation, the characteristics of desert areas, are associated with people at high risk.

The result of this process, depicted in Figure 3, is a new feature for T_2, that is, a feature on surnames. This feature is then used to construct a classifier for T_2, which is in turn used as a feature for T_1, yielding a feature on patients. This new feature for patients will check whether their surname corresponds with a country of origin that has desert-like characteristics. We see that this feature allows us to correctly capture the target concept.

Generating features through recursive induction

In the following sections, we formally define the feature generation problem, present a solution in the form of a simple algorithm that generates features by using relational expansions, and then proceed to describe our main recursive feature generation algorithm.

Problem definition

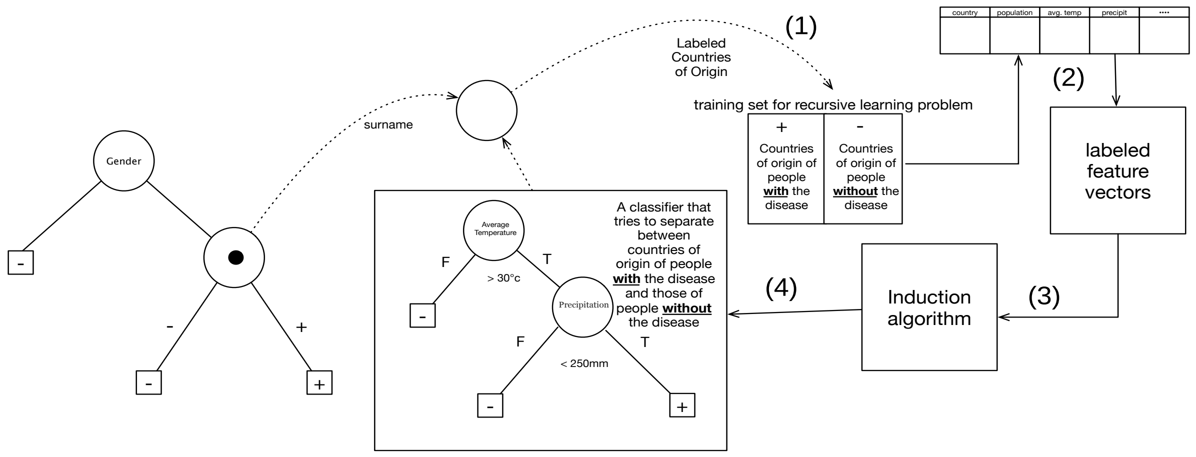
We begin our discussion with a standard definition of an induction problem. Let O be a set of objects. Let $Y = \{0, 1\}$ be a set of labels[1]. Let $C : O \to Y$ be a target concept. Let $S = \{(o_1, y_1), \dots, (o_m, y_m)\}$ be a set of labeled examples such that $o_i \in O, y_i \in Y, C(o_i) = y_i$. Let $F = \{f_1, \dots, f_n\}$ be a *feature map*, a set consisting of *feature functions* $f_i : O \to I_i$ where I_i is the image of f_i. This definition implies a training set represented by feature vectors: $S_F = \{(\langle f_1(o_i), \dots, f_n(o_i)\rangle, y_i) | (o_i, y_i) \in S\}$. A learning algorithm L takes S_F as inputs, and outputs a classifier $h_{S_F} : O \to Y$.

Definition 0.1. Let $L(S, F) = h_{S_F}$ be the classifier given as an output by L given $\langle S, F\rangle$. Assuming $S \sim D$, the generalization error of a learning algorithm L is the probability $Pr(h_{S_F}(x) \neq y)$, where $(x, y) \sim D$.

Definition 0.2. A *feature generation algorithm* A is an algorithm that, given $\langle S, F\rangle$, creates a new feature map $F' = \{f'_1, \dots, f'_k\}, f'_i : O \to I_i$.

In order to evaluate the output of a feature generation algorithm A, we must define its utility. Given $\langle S, F\rangle$, A generates a feature set F'_A. Let S be a training set representative

[1]We assume binary labels for ease of discussion.



(1)
Labeled Countries of Origin
country
population
avg. temp
precipit
....
(2)
training set for recursive learning problem
+
-
Countries of origin of people with the disease
Countries of origin of people without the disease
labeled feature vectors
surname
Gender
-
-
+
-
+
Average Temperature
F
T
> 30°c
Precipitation
F
T
< 250mm
-
-
+
A classifier that tries to separate between countries of origin of people with the disease and those of people without the disease
(4)
Induction algorithm
(3)

Figure 3: Recursive construction of a learning problem on countries of origin. (1) Creating the objects for the new problem. (2) Creating features using the knowledge base. (3) Applying an induction algorithm. (4) The resulting feature.

of the true distribution of the target concept. Given S , a feature set F, a generated feature set F'_A and a learning algorithm L, the utility of A is $U(A(S,F)) = Pr(h_{S_F}(x) \neq y) - Pr(h_{S_{F'_A}}(x) \neq y)$, where $x \in O, y = C(x)$.

Thus, in order for the utility of A to be positive, the generated feature set F'_A must yield a lower generalization error than the original feature map F.

In this work, we assume that, in addition to S_F, A also has access to a set of binary[2] relations $\mathcal{R} = \{R_1, \dots, R_t\}, R_j : D_j \times D_{j'}$ representing our knowledge base. For each individual relation R_j, its set of departure is marked D_j and its co-domain is denoted as $D_{j'}$.

Definition 0.3. A *knowledge-based feature-generation algorithm* A is an algorithm that, given $\langle S, F, \mathcal{R}\rangle$, creates a new feature map $F_{\mathcal{R}} = \{f'_1, \dots, f'_k\}, f'_i : O \to I_i$.

Expansion-based feature generation

In this section, we present our first method for knowledge-based feature generation that ignores the labels of the original learning problem. The algorithm extends each feature value by all of the tuples it appears in. Let f_i be an original feature. Let $R_j : D_j \times D_{j'}$ be a relation such that $Image(f_i) \subseteq D_j$. We can generate a new feature function $f_{i,j} : O \to D_{j'}$ by composing R_j onto f_i, yielding our new feature function $f_{i,j}(x) = R_j \circ f_i$.

In the general case, composing R_j onto F_i yields a set of values, meaning $f_{i,j}(x) = \{v \in D_{j'} | (f_i(x), v) \in R_j\}$. In our work, we preferred to work on singular values rather than set-based features. To do so, we use aggregation functions. For the experiments described in this paper, we use two aggregation function types: Majority and Any, but in general any other reasonable aggregation function can be used instead. The Majority aggregator, for example, is defined as follows. For each value $v \in D_{j'}$, we generate a binary feature function with value 1 only if v is the majority value: $Majority^v(X) = 1 \iff majority(X) = v$. The pseudo-code of this algorithm (called *Expander-FG*) is listed in Algorithm 1.

[2]If our relations are not binary, we can use projection to create multiple binary relations instead.

Algorithm 1 *Expander-FG*

```
$\sigma$ - An aggregation function family
  function GENERATEFEATURES($S$,$F$, $\mathcal{R}$)
    $generated = \emptyset$
    for $f_i \in F$ do
      for $R_j \in \mathcal{R}$ such that $Image(f_i) \subseteq D_j$ do
        if $R_j$ is a function then
          add $\{f_{i,j} = R_j \circ f_i\}$ to $generated$
        else                    $\triangleright$ $R_j$ is a relation $R_j : D_j \times D_{j'}$
          add $F^{\sigma}_{i,j} = \{\sigma^v(f_{i,j}) | v \in D_{j'}\}$ to $generated$
    return $generated$
```

Recursive feature generation algorithm

One way to extend the *Expander-FG* algorithm described in the previous section is to apply it repeatedly to its own output. Extending the algorithm in this fashion, however, would yield an exponential increase in the number of generated features. To that end, we propose an alternative knowledge-based feature generation algorithm. Given an input $\langle S, F, \mathcal{R}\rangle$, for each feature $f_i \in F, f_i : O \to I_i$, our algorithm creates a recursive learning problem whose objects are the values of f_i, where values associated with positive examples in S are labeled positive. Features for this generated problem are created using the relations in $\mathcal{R}$. Once this new learning problem $\langle S'_i, F_{\mathcal{R}}\rangle$ is defined, an learning algorithm is used to induce a classifier $h_i : I_i \to Y$. Finally, our algorithm outputs a single generated feature for f_i, $f'_i(x) = h_i \circ f_i = h_i(f_i(x)), f'_i : O \to Y$. Note that during the induction process of the newly created learning problem, we can apply a feature generation algorithm on $\langle S'_i, F_{\mathcal{R}}, \mathcal{R}\rangle$. In particular, we can apply the above method recursively to create additional features. We call this algorithm *FEAGURE*

(FEAture Generation Using REcursive induction).

Given a feature f_i, we create a recursive learning problem $\langle S'_i, F_{\mathcal{R}} \rangle$. Let $v_i(S) = \{v | (o, y) \in S, f_i(o) = v\}$ be the set of feature values for f_i in the example set S. We use $v_i(S)$ as our set of objects. To label each $v \in v_i(S)$, we examine at the labels in the original problem. If there is a single example $o \in S$ such that $f_i(o) = v$, then the label of v will be the label of o. Otherwise, we take the majority label $label(v) = majority(\{y | (o, y) \in S, f_i(o) = v\})$.

To define our learning problem, we must specify a feature map over $v_i(S)$. Similarly to *Expander-FG*, we use the relations in $\mathcal{R}$ on the elements in the new training set $S'_i = \{(v, label(v)) | v \in v_i(S)\}$. For each $R_j \in \mathcal{R}$, if it is relevant to the problem domain, meaning that $v_i(S) \subseteq D_j$, we utilize it as a feature by applying it to v. If $R_j(v)$ is a set, we use aggregators, as described in the previous section. The result of this process is a generated feature map for S'_i, denoted as $F_{\mathcal{R}}$.

We now have a new induction problem $\langle S'_i, F_{\mathcal{R}} \rangle$. We can further extend $F_{\mathcal{R}}$ by recursively using *FEAGURE*, yielding a new feature map $F'_{\mathcal{R}}$. The depth of recursion is controlled by a parameter d, that will usually be set according to available learning resources. We proceed to use a learning algorithm[3] on $\langle S'_i, F'_{\mathcal{R}} \rangle$ in order to train a classifier, giving us $h_i : I_i \rightarrow Y$. We can then use h_i on objects in S as discussed above, giving us a new feature $f'_i(x) = h_i(f_i(x)), f'_i : O \rightarrow Y$.

The full algorithm is listed in Algorithm 2. While the *FEAGURE* algorithm can be used as described above, we found it more useful to use it in the context of a divide & conquer approach, in a manner similar to the induction of decision trees. In this approach, the set of examples is given as an input to a decision tree induction algorithm. The *FEAGURE* algorithm is applied at each node. This allows us to generate features that are locally useful for a subset of examples. At the end of the process the tree is discarded and the generated features are gathered as the final output.

Algorithm 2 FEAGURE algorithm

```
function GenerateFeatures($F$, $S$, $\mathcal{R}$, $d$)
    for $f_i \in F$ do
        $S'_i, F_{\mathcal{R}}$= CreateNewProblem($f_i$,$S$,$\mathcal{R}$,$d$)
        $h_i$= InductionAlgorithm($S'_i, F_{\mathcal{R}}$)
        add $f'_i(x) = h_i \circ f_i$ to generated features
    return generated features
function CreateNewProblem($f_i$, $S$, $\mathcal{R}$, $d$)
    $v_i(S) = \{v | (o, y) \in S, f_i(o) = v\}$
    Let $s(v) = \{o | (o, y) \in S, f_i(o) = v\}$
    $S'_i = \{(v, \text{majority-label}(s(v))) | v \in v_i(S)\}$
    $F_{\mathcal{R}} = \{R_j(v) | R_j \in \mathcal{R}, v_i(S) \subseteq D_j\}$
    if $d > 0$ then
        $F_{\mathcal{R}} = F_{\mathcal{R}} \cup$GenerateFeatures($F_{\mathcal{R}}$, $S'_i$, $\mathcal{R}$, $d - 1$)
    return $S'_i, F_{\mathcal{R}}$
```

[3]For our experiments, we used a decision tree learner, but any induction algorithm can be used.

Empirical evaluation

We have applied our feature generation algorithm to the domain of text classification.

Application of *FEAGURE* to Text Classification

To use *FEAGURE* for text classification, we use words as binary features and Freebase and YAGO2 as our semantic knowledge bases. YAGO2 (**?**) is a large general knowledge base extracted automatically from Wikipedia, WordNet and GeoNames. YAGO2 contains over 10 million entities and 124 million relational facts, mostly dealing with individuals, countries and events. Freebase (**?**) has been described as "a massive, collaboratively edited database of cross-linked data." Freebase is constructed as a combination of data harvested from databases and data contributed by users. The result is a massive knowledge base containing 1.9 billion facts.

To apply our approach to the domain of text classification, we perform a few minor adjustments to the *FEAGURE* algorithm:

1. To enable linkage between the basic features and the semantic knowledge bases, we use entity linking software (**?**; **?**) to transform these words into semantically meaningful entities.
2. Once we have created a new classifier h_i, we cannot simply compose it on f_i, since every example might contain multiple entities. To that end, we apply h_i on each entity and take the majority vote.
3. Since our features are binary, we use the entities extracted from the text as the set of values $v_i(S)$. We split $v_i(S)$ into several subsets according to relation domains and apply the *FEAGURE* algorithm independently to each domain.

Methodology

We evaluated our performance using a total of 101 datasets from two dataset collections:

TechTC-100 (**?**) is a collection of 100 different binary text classification problems of varying difficulty, extracted from the Open Dictionary project. We used the training and testing sets defined in the original paper. As our knowledge base for this task, we used YAGO2.

OHSUMED (**?**) is a large dataset of medical abstracts from the MeSH categories of the year 1991. First, we took the first 20,000 documents, similarly to **?** (**?**). We limited the texts further to medical documents that contain only a title. Due to the relatively sparse size of most MeSH categories, we only used the two with the most documents, C1 and C20. The result is a dataset of 850 documents of each category, for a total of 1700 documents. We used ten-fold cross-validation to evaluate this dataset. Since the YAGO2 knowledge base does not contain many medical relations, we used Freebase instead. We used the same data dump used by **?** (**?**).

In our experiments, we generated features using the *FEAGURE* algorithm. We then proceeded to use these new features alongside three learning algorithms: SVM (**?**), K-NN (**?**) and CART (**?**).

We compared the performance of a learning algorithm with the generated features to the baseline of the same induction algorithm without the constructed features. In addition, since we could not obtain the code of competitive approaches for relation-based feature generation (such as FeGeLOD, SGLR), we instead compared our algorithm to *Expander-FG*, which we believe to be indicative of the performance of these unsupervised approaches.

Results

Table 1 shows average accuracies across all 10 folds for OHSUMED, as well as the average accuracies for all 100 datasets in techTC-100. When the advantage of our method over the baseline was found to be significant using a pairwise t-test (with $p < 0.05$), we marked the p-value. Best results are marked in bold. For the TechTC-100 dataset, *FEAGURE* shows a significant improvement over the baseline approach. Of particular note are the results for KNN and SVM, where the two-level activation of *FEAGURE* (d=2) shows statistically significant improvement over *Expander-FG* as well as the baseline accuracy ($p < 0.05$). One notable exception to our good results is the poor performance of K-NN for the OHSUMED dataset. This is likely due to the sensitivity of K-NN to the increase in dimension. For SVM as the external classifier, the *FEAGURE* algorithm showed an improvement in accuracy for 87 of 100 datasets for $d = 1$, and 91 datasets for $d = 2$. Using a Friedman test (**?**), we see a significant improvement ($p < 0.001$) over the baseline.

Figure 4 shows the accuracies for datasets in techTC-100 using a SVM classifier. The x axis represents the baseline accuracy without feature generation, and the y axis represents the accuracy using our new feature set generated using *FEAGURE*. Therefore, any dataset that falls above the $y = x$ line marks an improvement in accuracy. The results show a strong trend of improvement, with high ($> 10\%$) improvement being common. We see that for 8 of the datasets, there is a degradation in accuracy. This can be a result of mistakes in the entity extraction and linking process.

In their paper on TechTC-100, (**?**) define a metric called Maximal Achievable Accuracy (MAA). This criterion attempts to assess the difficulty of the induction problem by the maximal ten-fold accuracy over three very different induction algorithms (SVM, K-NN and CART). Table 1 also shows the accuracies for the 25 hardest datasets in TechTC-100, in terms of the MAA criterion. We call this dataset collection "TechTC-25MAA." These results show a much more pronounced accuracy increase, and illustrate that we can, in general, rely on *FEAGURE* to yield positive features for difficult classification problems.

As we have discussed in section , *FEAGURE* creates a generic learning problem as part of its execution. For our main results we learned a decision tree classifier for this new induction problem. We also tested the effects of using K-NN and SVM classifiers instead. This choice is orthogonal to that of the learning algorithm used to evaluate the generated features. Our experiments showed that in general, replacing the internal tree induction algorithm lowers the achievable accuracy of the resulting feature map. The only exception to this trend is the case of an external K-NN classifier for the OHSUMED dataset. In this case, an internal RBF-SVM induction algorithm yields an average accuracy

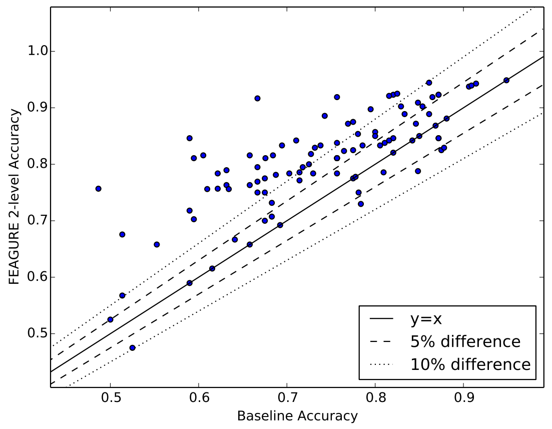


Figure 4: Accuracy of baseline approach compared to two-level activation of *FEAGURE* (SVM). Each point represents a dataset. The dotted lines represent a 5 and 10 percent difference in accuracy

of 0.795 (across ten folds), a significant ($p < 0.05$) improvement over the baseline.

FEAGURE Demonstration

To demonstrate *FEAGURE*, We selected one problem from TechTC-100. In this example, texts refer either to locations in and around Texas, or to locations in and around New York. The extracted entities are locations, with the "Located in" relation as our domain (Figure 5). Applicable relations are used to then create a new induction problem. *FEAGURE* uses the "Located in" and "Happened in" relations as features for this problem, as shown in Figure 6.

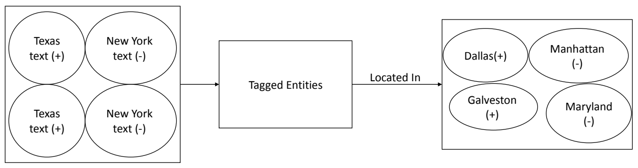
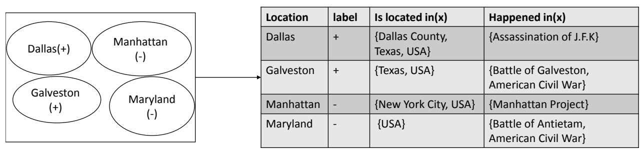


Figure 5: Entities are extracted from the text, and entities in the "Located in" relation are used as labeled objects.



Location	label	Is located in(x)	Happened in(x)
Dallas	+	{Dallas County, Texas, USA}	{Assassination of J.F.K}
Galveston	+	{Texas, USA}	{Battle of Galveston, American Civil War}
Manhattan	-	{New York City, USA}	{Manhattan Project}
Maryland	-	{USA}	{Battle of Antietam, American Civil War}

Figure 6: Construction of a recursive learning problem based on the "Located in" relation. Applicable relations are used to create a feature set for the newly constructed example set.

Since we chose $d = 2$ as the recursion depth parameter,

Table 1: Average accuracy over all datasets. The columns specify feature generation approach, with baseline being no feature generation. The rows specify the induction algorithm used on the generated features for evaluation. Results marked with * are significant with $p < 0.001$.

Dataset	Classifier	Baseline	Expander-FG	FEAGURE(d=1)	FEAGURE(d=2)
OHSUMED	KNN	**0.777**	0.756	0.769	0.75
	SVM	0.797	0.804	0.816 ($p < 0.05$)	**0.819** ($p < 0.05$)
	CART	0.806	0.814	0.809	**0.829** ($p < 0.05$)
TechTC-100	KNN	0.531	0.702*	0.772*	**0.775***
	SVM	0.739	0.782*	0.796*	**0.807***
	CART	0.81	0.815	0.814	**0.825** ($p < 0.05$)
TechTC-25MAA	KNN	0.524	0.723*	**0.803***	0.795*
	SVM	0.751	0.815*	0.817*	**0.829***
	CART	0.82	0.839	0.837	**0.849** ($p < 0.05$)

the algorithm calls *FEAGURE* recursively to try and generate new features for the new induction problem. The values of the feature "Happened in" are events. These events are used as objects for a recursive learning problem (Figure 7). We use the "Type" relation as a feature, relying on hypernyms to classify events (Figure 8). The resulting classifier (a decision tree induction algorithm was used) is shown in Figure 9, and can be interpreted as "is this event a battle or conflict?".

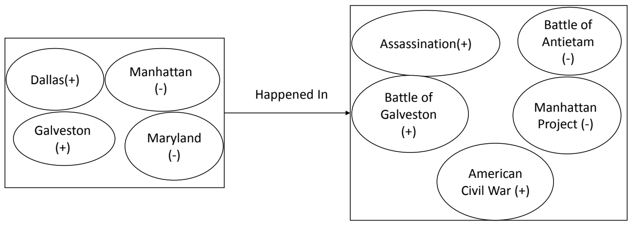
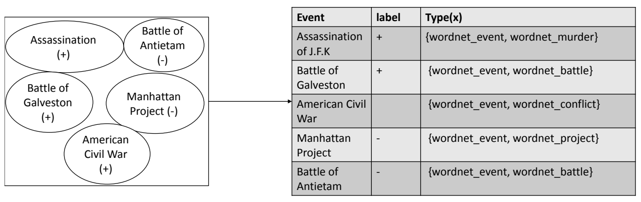


Figure 7: Construction of a second level recursive learning problem based on the "Happened in" relation. Feature values are treated as objects and labeled according to the labels of the problem on locations.



Event	label	Type(x)
Assassination of J.F.K	+	{wordnet_event, wordnet_murder}
Battle of Galveston	+	{wordnet_event, wordnet_battle}
American Civil War		{wordnet_event, wordnet_conflict}
Manhattan Project	-	{wordnet_event, wordnet_project}
Battle of Antietam	-	{wordnet_event, wordnet_battle}

Figure 8: Construction of a recursive learning problem based on the "Happened in" relation. Once the example set has been created, applicable relations are used to create a feature set for the newly constructed induction problem.

Once we have generated this classifier on events, we can use it as a binary feature. *FEAGURE* uses this new feature to expand the constructed induction problem on locations, shown in Figure 6. This feature is applied to a location through a majority vote over events that happened in that location. The result is a feature for locations representing the concept "were most notable events in this location battles/conflicts?" Finally, a decision tree learner is used on the expanded feature set to learn a classifier on locations to be used as a feature for our original learning problem. The new classifier for locations is shown in Figure 10. It can be described as "is this location located in Texas, or the site of battles or conflicts?". Texts mentioning locations in and around Texas are more likely to link to locations that correspond to the output of this classifier. We note that this feature was generated by *FEAGURE*, and was later used by our external induction algorithm due to its high information gain.

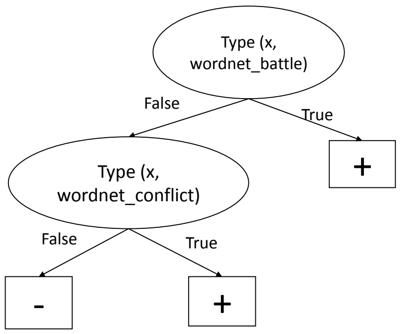


Figure 9: Recursive feature constructed by *FEAGURE* for entities in the "happened in" relation. This feature operates on events, and can be used in a classifier on locations.

Related work

Many feature generation methodologies have been developed to search for new features that better represent the target concepts. There are three major approaches for feature generation: tailored methods, combinational techniques and algorithms utilizing external knowledge.

Tailored approaches (**?**; **?**) are designed for specific problem domains and rely on domain-specific techniques. Such special-purpose algorithms have proven difficult to generalize to other domains and problems.

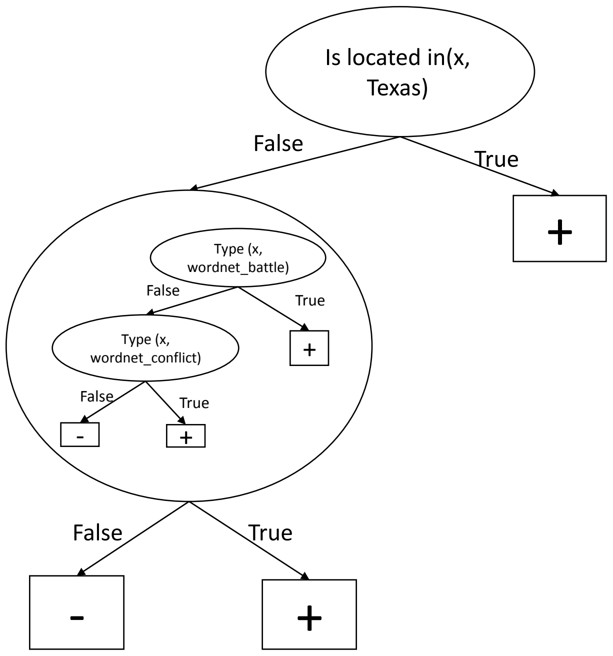


Figure 10: Final generated feature constructed by *FEAGURE* for entities in the "located in" relation. The feature in the left branch is the recursive feature constructed by applying *FEAGURE* to the new learning problem.

Combinational feature generation techniques are domain-independent methods for constructing new features by combining existing features. The LDMT algorithm (?) performs feature construction in the course of building a decision-tree classifier. At each tree node, the algorithm constructs a hyperplane feature through linear combinations of existing features in a way likely to produce concise, relevant hyperplanes. The LFC algorithm (?) combines binary features through the use of logical operators such as $\wedge, \neg$. The FICUS algorithm (?) allows the use of any combinational feature generation techniques, based on an a given set of constructor functions. Recent work by **?** (?) uses a similar approach. *Deep Learning* (?; ?) is another major class of combinational feature generation approaches. Here, the activation functions of the nodes can be viewed as feature schemes, which are instantiated during the learning process by changing the weights.

One limitation of combinational approaches is that they merely combine existing features to make the representation more suitable for the learning algorithm. Our *FEAGURE* algorithm belongs to a third class of approaches that inject additional knowledge into the existing problem through the feature generation process.

Propositionalization approaches (?; ?) rely on relational data to serve as external knowledge. They use several operators to create first-order logic predicates connecting existing data and relational knowledge. **?** (?) devised a generic propositionalization framework using linked data via relation-based queries. FeGeLOD (?) also uses linked data to automatically enrich existing data. FeGeLOD uses feature values as entities and adds related knowledge to the example, thus creating additional features.

Unsupervised approaches allow us to utilize external knowledge, but they have a major issue: Should we try to construct deep connections and relationships within the knowledge base, we would experience an exponential increase in the number of generated features. To that end, *FEAGURE* and other supervised approaches use the presence of labeled examples to better generate deeper features.

The *dynamic feature generation* approach used by the SGLR algorithm (?) can be seen as the supervised equivalent of propositionalization methods. Feature generation is performed during the training phase, allowing for complex features to be considered by performing a best-first search on possible candidates. This process allows SGLR to narrow the exponential size of the feature space to a manageable number of candidates. While this supervised approach overcomes the exponential increase in features that unsupervised approaches suffer from, the space of generated features that it searches is significantly less expressive than that of our approach. Through the use of recursive induction algorithm, our approach automatically locates relationships and combinations that we would not consider otherwise.

Two examples of text-based approaches for feature generation are Explicit Semantic Analysis (ESA) (?) that generates explicit concepts from Wikipedia, and Word2Vec (?) that generates latent concepts based on a large corpus.

Conclusions

When humans use inductive reasoning to draw conclusions from their experience, they use a vast amount of general and specific knowledge. In this paper we introduced a novel methodology for enhancing existing learning algorithms with background knowledge represented by relational knowledge bases. The algorithm works by generating complex features induced using the available knowledge base. It does so through the extraction of recursive learning problems based on existing features and the knowledge base, that are then given as input to induction algorithms. The output of this process is a collection of classifiers that are then turned into features for the original induction problem.

With the recent surge of well-formed relational knowledge bases, and the increase in use of strong learning algorithms for a wide variety of tasks, we believe our approach can take the performance of existing machine learning techniques to the next level.