Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den January 9, 2014

(Alexander Gabriel)

Learning Semantically Coherent Rules

Lernen semantisch kohärenter Regeln
Bachelor-Thesis von Alexander Gabriel
Januar 2014

TECHNISCHE
UNIVERSITÄT
DARMSTADT

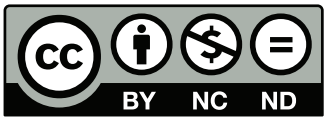
Fachbereich Informatik
Knowledge Engineering Group

Learning Semantically Coherent Rules
Lernen semantisch kohärenter Regeln

Vorgelegte Bachelor-Thesis von Alexander Gabriel

1. Gutachten: Prof. Dr. Johannes Fürnkranz
2. Gutachten: Dr. Frederik Janssen, Dr. Heiko Paulheim

Tag der Einreichung:



Abstract

This work explores the feasibility, limitations and effects of combining classical rule learning heuristics with semantic heuristics based on freely available knowledge sources. The combination is investigated in the context of inductive rule learning using a separate-and-conquer strategy. We found that adding a semantic heuristic to the rule evaluation process can increase the semantic coherence remarkably without sacrificing too much ruleset performance. Different strategies to combine individual similarity scores for condition pairs into a score for the rule have an influence on the rule generating process promoting different kinds of rules. A prerequisite to use semantic heuristics in a meaningful way is a semantic heuristic that fits to the domain of the attribute labels.

Contents

1 Introduction

> *Rule learning is not only one of the oldest but also one of the most intensively investigated, most frequently used, and best developed fields of machine learning. In more than 30 years of intensive research, many rule learning systems have been developed for propositional and relational learning, and have been successfully used in numerous applications.*
> *Fürnkranz et al. [7]*

Despite this tremendous success there is a quality less graced by progress: understandability. The rules we create may be accurate, precise and informative, what they are rarely is understandable.

But what is understandability? Something is understandable if it can be grasped easily. This is the ideal case of a more general notion: interpretability. If something is interpretable one is able to arrive at a meaning through a longer or shorter process of thought. Rules for example can be interpreted as a set of conditions that need to hold true for a certain implication. If A,B,C and D are true E will follow. The interpretability of a rule depends on a number of factors for example its length or its coherence. For a rule to be judged as coherent its conditions have to be related somehow and this relation has to be recognizable by the one judging. Why then are rules rarely understandable or not very interpretable?

The reason for this lies in the algorithms of rule learning themselves. These algorithms generate rulesets from statistical patterns in datasets. A dataset here is a list of examples that are each a realization of a given attribute set. With the aim to predict the realization of a target attribute (often called class attribute) rule learning processes create sets of rules where each rule is a conjunction of attribute realizations (also called conditions).

The so generated rulesets try to capture the statistical patterns underlying the dataset. They have no capacity to capture the meaning inherent in the attributes' labels. Thus the conditions in a rule usually are not semantically coherent, their meanings are not necessarily related.

In scenarios where the recipient of the generated rules is another algorithm this handicap is still of little concern, but as soon as a human being is involved the verdict changes.

For humans every explanation has to have some kind of relation to their prior experience to 'make sense'. If an explanation contains multiple conditions there ought to be a relation between the individual conditions. Consider the following two sentences:

- It is 50cm high and has 3 legs.
- It is 50cm high and green.

While the first sentence gives enough information about the shape of the object to guess that it could be a stool or table the description of the second sentence could also fit a box or a bicycle or any number of flowers, bushes and trees. The coherence between the two conditions in the first sentence (they both relate to the shape of the object) makes the first sentence more interpretable than the second where the coherence between the conditions (height and color) is smaller. Of course the combination of size and color might work better in other cases. Something light blue and vast could be the sky but this is a special case. We know the sky is light blue (at times) and vast and we know few other things that are as well. Thus we remember the relation sky-light blue and sky-vast and this lets us think of the sky when we hear light blue and vast. There is a semantic coherence between light blue and vast because both describe the sky.

Rule learning algorithms perceive a problem much different from a human. They could play an even more significant role in human data analysis, giving insight into big datasets and offering a new perspective on interrelations if they got a bias towards meaningful, more understandable rules.

This work explores the viability of increasing the understandability of modern rule learning results by combining classic rule learning heuristics with semantic coherence heuristics.

The semantic heuristics are employed directly to evaluate the semantic similarity of the attribute label pairs that arise from the conjunctive combination of conditions during the creation of a rule. Multiple

semantic similarity values are then combined to create a semantic coherence score for the rule. This score is then combined with the classic heuristic score in a weighted sum to form the overall rule score.

We are concentrating on the similarity aspect of semantic coherence because semantic coherence in itself is a hard to define problem. It is hard to differentiate 'that which makes sense to humans' from that which does not in an algorithmic fashion because there are many kinds of relations between concepts that allow or disallow the combination of concepts in a way that respects semantic coherence. 'Colourless green ideas dream furiously about dancing trains.' is a sentence that is syntactically correct but semantically senseless. Humans know that colourless contradicts green and that an idea is an abstract concept that has no colour. We also know that trains do not dance and that one can sleep peacefully but hardly furiously. For an algorithm to know all this requires a large amount of knowledge about the world which is at this point not available to us. A few of the relations mentioned in the last paragraph can be found in the semantic structure of our language. A large part of this structure for the English language is captured in WordNet and there are similarity metrics defined on WordNet which allow the computation of similarity scores for pairs of concepts. This is a good starting point for the quest to semantic coherence. We thus derive our semantic similarity from one of these similarity measures and use WordNet as the source of semantic knowledge in our work.

By combining classic and semantic heuristics we can add a bias towards rules with semantically coherent conditions without sacrificing the benefit of using classical rule learning heuristics.

2 Basics

This section will shortly introduce a few topics that should be understood to a certain degree before reading further. It will first explain a few basics about rule learning and then introduce WordNet.

Since the topic of this work is the combination of different heuristics used in rule learning algorithms to evaluate the generated rules with the goal to increase the semantic coherence of said rules, this chapter will explain the rule learning process. It will first give a general idea of the process and then explain rules, learning rules and rulesets as well as how to evaluate them.

2.1 Rule Learning

Rule learning is a process that generates predictive rulesets from lists of examples. An example is a realization of a given attribute set so that every attribute has an assigned value. A group of examples for a common attribute set is referred to as a dataset. A rule learning algorithm generates a ruleset from the patterns it detects in a given dataset. The ruleset can then be used to classify new examples, that means it can be used to predict the outcome of one of the attributes[1] given the other attributes. Rule learning algorithms are employed in a variety of fields including medical diagnosis, stock market analysis, computer games and search engines to name just a few.

2.1.1 Definition of a Rule

A rule consists of two parts: a body and a head. The body of a rule is a conjunction of conditions in the form "attribute=value"[2]. The head defines the rule's prediction which is usually just "example belongs to a specific class". If an example satisfies the conditions in the rule's body it is said to be *covered* by that rule.

$$class = 1 :- planet = true, size >= 4.13 \tag{1}$$

Example 1 shows a rule consisting of two attributes *planet* and *size*. An example is covered by the rule if it is a *planet* and its *size* is bigger than 4.13. The rule assigns the *class* with label 1 to every example it covers.

2.1.2 Rule Evaluation

Since a rule decides if an example belongs to a class or not and each of these decisions can be either right or wrong, the evaluation of a rule is based on the ratios between the sizes of these groups:

- *TP* (True Positives): The number of examples correctly classified as a part of the target class.
- *TN* (True Negatives): The number of examples correctly classified as not a part of the target class.
- *FP* (False Positives): The number of examples falsely classified as a part of the target class.
- *FN* (False Negatives): The number of examples falsely classified as not a part of the target class.

[1] This is usually called the 'class'.

[2] <=, <, >=, > and others are also possible

Derived from these four basic categories is another set of categories that give insight into the rule in relation to the set of examples:

- $TP + FP$ (Coverage): The number of examples that satisfy the conditions of the rule.
- P: The number of positive examples
- N: The number of negative examples
- $P + N$: The total number of examples

The Confusion Matrix in Table 1 offers a more structured view of the categories we just introduced. It shows how the categories from the first list combine to form the categories of the second list. *TP* and *FN* add up to the number of positive examples P while *FP* and *TN* together form the negative examples N. The sum over P and N gives the total number of examples. Lastly $TP + FP$ what is called the coverage denotes the number of example covered by the rule, that means all the examples the rule classifies as positive.

	classified +	classified −	
truly +	*TP*	*FN*	$P = TP + FN$
truly −	*FP*	*TN*	$N = FP + TN$
	Coverage $= TP + FP$	$FN + TN$	$P + N$

Table 1: Confusion Matrix

These measures are combined in a variety of ways to evaluate the quality of a given rule. The heuristics employed in this work are explained here but there are many others, a comparison of which can be found in Fürnkranz and Flach [6].

Accuracy

Accuracy[3] is an evaluation measure that computes the percentage of correctly classified samples. It is calculated by summing up the correctly classified samples (true positives *TP* and true negatives *TN*) and dividing by the total number of samples (P and N).

$$Accuracy = \frac{TP + TN}{P + N} \tag{2}$$

One downside of *Accuracy* as a measure in rule learning is that it does not differentiate between true positives and true negatives. Covering a positive example is just as good as not covering a negative example. In cases where there is a different cost associated with covering a positive example as with not covering a negative example this poses a problem.

Precision

Precision[4] is the probability that a covered example actually belongs to the class the rule assigns it.

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

From Equation 3 it is obvious that *Precision* regards all rules that cover only positive examples as equal. *FP* becomes zero and the score is one regardless of how many positive examples are covered. Similarly all rules that cover only negative examples get a score of zero regardless of the actual number of negative examples covered. An exception occurs where neither positive nor negative examples are covered. For rules with zero coverage *Precision* is undefined.

[3] More precisely: Classification Accuracy

[4] *Precision* is also known as *Confidence*.

Laplace Estimate

The *Laplace Estimate* is a modification of *Precision* that can handle rules without coverage. Like with *Precision* False Positives will decrease the score and True Positives will increase it.

$$L = \frac{TP + 1}{TP + 1 + FP + 1} \tag{4}$$

The *Laplace Estimate* achieves the capability to handle rules without positive and/or negative coverage (*TP* and *FP*) by initialising each of them with one. Another effect of this modification is that rules covering only positive or only negative examples no longer get the same score regardless of how many examples they cover. A rule covering only positive examples gets higher scores for more coverage. A rule covering only negative examples gets lower scores for more coverage.

m-Estimate

The *m-Estimate* is another modification of *Precision*. Here both *TP* and *FP* are increased by adding m times the probability of the appearance of their respective class[5]. Thus the *m-Estimate* adds a prior coverage of m distributed among positive and negative coverage according to the balance of positive to negative examples in the training set.

$$M = \frac{TP + (m\frac{TP+FN}{P+N})}{(TP + (m\frac{TP+FN}{P+N})) + (FP + (m\frac{FP+TN}{P+N}))} = \frac{TP + m\frac{TP+FN}{P+N}}{TP + FP + m} \tag{5}$$

We use a value of 22.466 for m throughout this work as this value was found to produce the best overall result on a variety of datasets in Janssen and Fürnkranz [9] and 3/4 of the datasets used there are also used in this work.

Multi-Class Problems

The measures until now apply to binary classification problems, in other words problems where an example belongs either to one class or not. If the problem allows for the example to belong to more than two classes, learning rules becomes more complex.

To be able to handle a problem with n classes, we split the problem into n sub-problems. We learn for every class one or more rules that classify the example as either belonging to the specific class or not. We then add all the rules to a ruleset. When we later try to classify an example and none of the rules covers the example, we predict the largest class. This is one method of binarization.

2.1.3 Ruleset Evaluation

One way of evaluating multi-class problems is the Macro Average. It allows to apply a binary classification measure to a multi-class problem. Consider a binary classification measure H that makes use of the example categories we introduced above. Then there is a set of TP_λ, TN_λ, FP_λ and FN_λ for each class λ. If there are n classes we can sum up the individual results and divide by the number of classes as shown in Equation 6.

$$H_{macro} = \frac{1}{n}\sum_{\lambda=1}^{n} H(TP_\lambda, TN_\lambda, FP_\lambda, FN_\lambda) \tag{6}$$

This work uses the Macro Average with *Accuracy* as the binary performance measure to evaluate rulesets.

[5] positive($TP + FN$) or negative($TN + FP$)

Training and Validation

But a problem arises if the dataset used to evaluate the ruleset is the same as the dataset used to learn the ruleset because most datasets are not an accurate representation of the underlying truth (or probability distribution). A rule learning algorithm that does not take this into account, tries to fit the rules it generates as close as possible to the dataset it works on. Even algorithms that try to avoid the problem usually can not mitigate it completely. This usually leads to overfitting which means the rules perform much better on the dataset they were generated for than on similar datasets generated from the same data source (or probability distribution). The generated rules are too specialized. To get a more honest evaluation of the generated rules we first split a dataset into n partitions. We then choose one training set and $n-1$ validation sets and generate rules (training) on the one and evaluate them (validation) on the other $n-1$. We do this n times choosing each time a different partition as the training set and using the other $n-1$ for validation. This process is called n-fold cross-validation.

2.1.4 Learning a Rule

Learning a rule is an iterative process that grows a rule into multiple directions concurrently.

Algorithm 2.1 depicts the rule learning algorithm implemented in the SeCo-Framework[10]. First a condition is chosen as an initial rule. This rule is then evaluated by a given heuristic and saved in two locations: (a) in a sorted[6] ruleset and (b) in a 'best rule' slot.

For as long as the ruleset contains rules, some are selected as candidates for refinement. The so selected candidate rules are removed from the ruleset and refined where each refinement consists of one of the candidate rule and an additional condition. A candidate rule can therefore be the source of multiple refinements.

The refinements are then evaluated and tested against a stopping criterion which determines whether they comply to some standard for rules. In our case the stopping criterion ensures that the refinement process is stopped as soon as the rule does not cover negative examples any more.

If a refinement got a better score than the current best rule, the best rule slot is updated to the new best rule.

If it passes the stopping criterion it is added to the sorted ruleset so it can be refined further.

Once all refinements have been either added to the ruleset or discarded, the ruleset is filtered. We use a beam width filter with a setting of one. This means only the best rule in the list is retained, the rest of the rules are removed to reduce complexity.

The algorithm then starts again by selecting new candidate rules to be refined. The algorithms stops if at this point the ruleset is empty, or in other words: when all the rules have been refined to an extent where they no longer pass the stopping criterion and are discarded. What remains then is the best rule, which is returned.

Note that this best rule is usually not the overall best rule but just the best rule the algorithm could find. The reason for this is the greedy nature of the algorithm and the imperfections in the dataset we discussed earlier (see overfitting).

[6] Sorted by rule score

Algorithm 2.1: Learning a rule (based on Fürnkranz [5])

```
input  : A heuristic h
input  : A stopping criterion sc
input  : A set of examples es
input  : An initial rule ir
output: A rule
bestRule =< ir, h.evaluateRule(ir) >;
rules = bestRule;
while rules ≠ ∅ do
    cs =getCandidates(rules); /* use a heuristic to select rules to refine      */
    rules = rules \ cs;
    foreach c ∈ cs do
        refs =refine(c, es); /* generate new rules by adding attributes/conditions */
        foreach ref ∈ refs do
            quality = h.evaluateRule(ref); /* use heuristic to calculate rule quality */
            tuple =< ref, quality >;
            if tuple is better than bestRule then
                bestRule = tuple; /* keep track of best rule                    */
            if sc.isAllowed(tuple, es) then
                rules =InsertionSort(tuple, rules); /* sort rules by quality    */
    rules =filter(rules); /* filter ruleset to manageable size                  */
return bestRule;
```

2.1.5 Separate-and-Conquer

In rule learning, one rule is generally not enough to cover a whole dataset consistently (without covering negative examples). Hence multiple rules have to be learned. One popular group of algorithms doing so are the separate-and-conquer or covering algorithms that can be retraced to the A^q Algorithm[14].

Algorithm 2.2 shows how starting from an empty ruleset, the algorithm iteratively learns one new rule at a time from the available examples and adds it to the ruleset. The rule stopping criterion then checks if the currently learned rule covers more negative examples than positive examples and stops the learning process if so. Otherwise, after the new rule has been added, the examples it covers are removed from the example set.

By implication each round of rule learning has to work on a smaller number of examples until all positive examples have been covered by some rule in the ruleset or the rule learning algorithm can no longer learn a new rule that covers more positive than negative examples.

Algorithm 2.2: Separate-and-Conquer Rule Learning

input : A set of examples *es*
output: A ruleset

```
1 rs = ∅; /* initialize empty ruleset                                    */
2 while positives(es)≠ ∅ do
3     Learn a consistent rule r from es;
4     if ¬rsc.isAllowed(r, rs, es) /* check rule stopping criterion      */
5     then
6         break while;
7     rs = rs ∪ r; /* add rule to ruleset                                */
8     es = es\{ examples covered by r}; /* remove covered examples from example set */
9 return rs;
```

We now leave the topic of rule learning for the moment to introduce WordNet.

2.2 WordNet

WordNet[16] is a dataset containing large parts of the English language and grouping the language into semantically interlinked synsets[7]. Each synset, containing a set of conceptually similar words, represents a distinct concept and is linked through a number of different relations to other synsets.

The relations used include hyponymy[8], hyperonymy[9] and meronymy[10] for nouns; troponymy[11] and entailment for verbs as well as antonymy[12], pertainymy[13] and semantic similarity for adjectives.

Links across word class borders are relatively rare. They are used to link morphosemantic[14] words.

[7] synonym sets
[8] A word that is a type of another word, a specialization. 'conversation' is a hyponym of 'communication'.
[9] A word that is the type of another word, a generalization. 'planet' is a hypernym of 'earth'.
[10] A part or member of something. 'page' is a meronym of 'book'.
[11] A more special kind of doing something. 'to limp' is a troponym of 'to walk'
[12] An opposite.
[13] An adjective pertaining to a noun.
[14] Semantically similar words that share a stem (attention, attentive, to attend).

3 Increasing Semantic Coherence

This chapter will first introduce the general idea that lay behind combining classic and semantic heuristics and continue to provide details about the algorithms and frameworks we combined to explore this idea.

3.1 General Idea

The basic idea behind our approach is to capture some knowledge about the world and leverage it to choose conditions that are semantically more similar to one another, with the goal of creating more meaningful rules.

The amount of knowledge captured in information repositories like for example Wikipedia grows at increasing speeds. Wikipedia, the Semantic Web and WordNet are just some examples of information repositories available online. They contain semantically annotated information, that means information that is saved with markers that identify its meaning. This semantic annotation is initially added by humans and converts information into knowledge[15]. A lot of these repositories are freely available and in machine-readable formats, which make them uniquely suited for our purpose which requires a knowledge source that can be processed algorithmically.

For this initial evaluation we chose WordNet as the source of the semantic influence. It captures a large part of the English language and thus offers knowledge on a fairly broad scope in contrast to a lot of other repositories which offer highly specialized domain knowledge. This allows us to use one source of knowledge for many different datasets. The downside of using WordNet is that special domain vocabulary, as can often be found in attribute labels of scientific datasets, is often not captured in WordNet.

On the side of classical rule learning heuristics we employ a small collection of well known heuristics, namely *Accuracy*, the *Laplace Estimate* and the *m-Estimate*. Although the scores of the two types of heuristics (semantic and classic) are inherently independent of one another, different semantic and classic heuristics may lead to a different ordering of rules during rule creation and thus to different rulesets. The reason for this is that the total score as a weighted sum of them both also depends on both of them.

It is to be expected that an increase in semantic influence will decrease *Precision* and *Accuracy* because the rule learning algorithm focusses less on the quality of the coverage and more on choosing conditions that are semantically coherent. As a result the amount of semantic influence has to be adjustable. Our experiments will show the degree of quality loss with different amounts of semantic influence.

3.2 The SeCo-Framework

The Knowledge Engineering Group of TU Darmstadt describes the SeCo-Framework as a 'modular architecture for learning decision rules on given datasets'[12] It is created at TU Darmstadt by the Knowledge Engineering Group and allows for custom specification of the various parts of a separate-and-conquer rule learning algorithm like for example the heuristic employed, the stopping criterion and rule stopping criterion, the used refinement and filter strategies. It also features highly configurable evaluation and test functionality that enables comparison of different algorithms and/or parameter settings on a group of datasets.

The Seco-Framework allows us to conveniently use the provided reference modules in parts that we do not want to change and to replace the reference implementation where we want to use our own algorithms to achieve a different result. Another feature that let us to chose the SeCo-Framework for this work is the concise summary of the results of our experiment runs.

[15] Knowledge here refers to organized information.

3.3 Arithmetic Weighted Mean Heuristic

To use a combination of heuristics for rule evaluation, a new meta-heuristic was created within the SeCo-Framework. This meta-heuristic allows for the combination of two individual heuristics, a classic and a semantic heuristic, in the form of the weighted arithmetic mean as is shown in Function 3.1.

First the function determines whether or not the rule consists of just one condition. This is done by getting the rule's predecessor, the rule without the last added condition, and checking if it is empty.

If there is just one condition in the rule the next step depends on the level of *influence*. If it is set to 100% the arithmetic weighted mean defaults to a random value since this case is not well defined. The semantic similarity can only be calculated between two conditions.

If the *influence* is not set to 100% the arithmetic weighted mean returns only the weighted classic part of the sum. This has the same overall result as a sum with a semantic heuristic value of 0.

In the other case, where the rule contains more than one condition, the two heuristics are calculated, weighted and summed up normally. The balance of influence between the classic and semantic heuristic is adjustable via the parameter *influence*.

Function 3.1: ArithmeticWeightedMeanHeuristic.evaluateRule(r)

```
input  : A rule r
input  : An influence value influence ∈ [0.0, 1.0]
input  : A classic heuristic hA
input  : A semantic heuristic hB
output: The heuristic value of the rule
rule p = r.getPredecessor();
if p = ∅ then
    if influence = 1 then
        return random();
    else
        return (1.0 − influence) ∗ hA.evaluateRule(r);
else
    vHA = hA.evaluateRule(r);
    vHB = hB.evaluateRule(r);
    return ((1.0 − influence) ∗ vHA) + (influence ∗ vHB);
```

3.4 Semantic Heuristic

This method is called by the Arithmetic Weighted Mean Heuristic and represents the interface between heuristics working on rules and heuristics working on words.

It splits a rule into its constituting conditions and extracts the attribute labels associated with these conditions. It then calls a string pair evaluating heuristic for the attribute labels of each unequal pair of conditions. After accumulating all the results it calls a customizable statistical method to combine the values into a single semantic similarity or semantic coherence score for the rule.

Function 3.2: SemanticHeuristic.evaluateRule(r)

```
input  : A rule r
input  : A heuristic h
input  : A statistic s
output: The heuristic value of the rule
Set results;
foreach condition1 ∈ r do
    foreach condition2 ∈ r do
        if condition1 ≠ condition2 then
            results.add(h.calculateSimilarity(condition1.attributeName, condition2.attributeName);
return s.calculate(results);
```

3.5 WordNet Similarity

The WordNet Similarity heuristic is a heuristic that calculates the similarity of two attribute labels. It is called by the Semantic Heuristic class to get similarity values for attribute label pairs as was described in the last section.

To calculate a similarity metric using WordNet we had to first tie an attribute label to its corresponding synset[16]. This is a complex matter that involves knowledge of the attribute label's POS[17] as well as the handling of ambiguities, both of which are complicated by the relative lack of context information. If the attribute label is a compound of words rather than a single word further possibilities have to be considered.

We will explain this rather complicated algorithm by use of a fictional example. Let's say we wanted to calculated the WordNet Similarity between the two attribute labels 'smartphone vendor' and 'desktop'.

Step 1

The first step is to establish the relationships between the attribute labels to WordNet synsets. To do this we search WordNet for the attribute label. This search returns a possibly empty list of synsets ordered by relevancy.

Listing 1: WordNet search result for 'smartphone vendor'

```
{}
```

Listing 2: WordNet search result for 'desktop'

```
{desktop#n#1, desktop#n#2}
```

The synset *desktop#n#1* describes a tabletop, the synset *desktop#n#2* describes a desktop computer. The 'n' indicates that the synsets are describing nouns. The search for 'smartphone vendor' did not return any synsets.

If the list is not empty, we add it to the attribute label's list of synset lists. If otherwise the list is empty, we check whether the attribute label is a compound of multiple words and restart the search for each of the individual words. The process of splitting a text string into smaller parts (tokens) is called tokenization. We then add all non-empty synset lists we got to the list of synset lists of the attribute label. The result can be seen in Listings 3 and 4.

[16] A set of synonyms, see chapter 2.2.
[17] Part-Of-Speech:lexical class or word class (noun, verb, adjective or adverb).

Listing 3: Result for 'smartphone vendor'

$\{\{smartphone\#n\#1\}, \{vendor\#n\#1\}\}$

Listing 4: Result for 'desktop'

$\{\{desktop\#n\#1, desktop\#n\#2\}\}$

We now generated a list of synset lists for each of the two attribute labels we want to compare. The inner lists are each the result of a WordNet search and contain the synsets similar to a word in the attribute label. We match words to lists of synsets because we can not be sure which of the synsets is the correct one. The outer lists represent attribute labels. They have more than one entry if the attribute label is a compound of more than one word as is the attribute label 'smartphone vendor'. With this work done, we can now calculate the similarity between two attribute labels.

Step 2

In the second step we calculate the distance of two synsets using the LIN[13] metric. We chose this metric as it performs well in comparison with other metrics[4] and has the added benefit of a value range between zero and one. A metric performing better in the aforementioned comparison is the JCN[11] metric but its range of values stretches from infinity to zero giving a score of 12876699.5 for the pair (mars, mars), 0.2007 for the pair (mars, venus) and 0.0617 for the pair (mars, mutant) and is thus hard to scale in a way that allows a combination with a classic heuristic.

LIN metric: The LIN metric is calculated by dividing the Information Content (IC) of the least common synset of the two synsets by their sum and multiplying the result with 2.

This metric limits the similarity calculation to synsets of the same POS and works only with nouns and verbs. Our implementation returns a similarity value of 0 in all other cases.

$$lin(synset1, synset2) = 2 * \frac{IC(lcs)}{IC(synset1) + IC(synset2)} \tag{7}$$

Information Content: Information Content (IC)[17] is a measure for the particularity of a concept. The IC of a concept c is calculated as the negative of the log likelihood, simpler put: the negative of the logarithm of the probability to encounter concept c in a body of text as Philip Resnik defined:

$$IC(c) = -\log(p(c)) \tag{8}$$

Thus higher values denote less abstract, more general concepts while lower values denote more abstract, less specific concepts.

The body of text used for the calculation of the IC values in this work is the SemCor[15] corpus. This is a collection of 100 passages from the Brown corpus which were semantically tagged "based on the WordNet word sense defintion" and thus provides the exact frequency distribution of each synset (instead of the frequency distribution of words in another context) it covers, which is sadly only 25% of the synsets in WordNet[11].

We calculate the LIN metric for each combination of two synsets in each pair of synset lists but only if the two synset lists belong to different attribute labels. The resulting similarity values can be seen in Listings 5 and 6.

Listing 5: Similarity for ('smartphone'; 'desktop')

$lin(smartphone\#n\#1, desktop\#n\#1) = 0.0$
$lin(smartphone\#n\#1, desktop\#n\#2) = 0.5$

Listing 6: Similarity for ('vendor'; 'desktop')

$lin(vendor\#n\#1, desktop\#n\#1) = 0.0$
$lin(vendor\#n\#1, desktop\#n\#2) = 0.0$

We now have a similarity value for each pair of synsets in a pair of synset lists. In other words we have the similarity values for each of the possible synset pairs that correspond to a word pair for every word pair where the words belong to different labels.

Step 3

In the third step we choose the maximum value for each pairing of synset lists so that we end up with the maximum similarity value per pair of words. This assigns each word pair the similarity value of the synset combination that is most similar among all the synset combinations that arise from the two lists of possible synsets for the two words. Listings 7 makes this more concrete.

Listing 7: Similarity values for synset list pairs

```
similarity('smartphone', 'desktop') =
max({smartphone#n#1} × {desktop#n#1, desktop#n#2}) =
max(lin(smartphone#n#1, desktop#n#1), lin(smartphone#n#1, desktop#n#2)) =
0.5

similarity('vendor', 'desktop') =
max({vendor#n#1} × {desktop#n#1, desktop#n#2}) =
max(lin(vendor#n#1, desktop#n#1), lin(vendor#n#1, desktop#n#2)) =
0.0
```

The result up until now is a list of maximum similarity values, where each value is the maximum similarity between two words from two attribute labels.

Step 4

The last step to get the similarity between two attribute labels is to calculate the mean of all the maximum similarities. Calculating the mean makes similarity scores of different attribute label pairs comparable as it mitigates the influence of the number of words in an attribute label.

Listing 8: Similarity value for attribute label pair

```
similarity(smartphone vendor, desktop) =
mean(similarity(smartphone, desktop), similarity(vendor, desktop)) =
mean(0.0, 0.5) =
0.25
```

The similarity score in our simple example is 25%.

Functions 3.3 and 3.4 are the more compressed algorithmic description of this process.

Function 3.3: WordNetSimilarity.calculate(*Attribute1*, *Attribute2*)

```
input  : An attribute Attribute1
input  : An attribute Attribute2
input  : A boolean tokenize
output: The similarity between the two Strings
List lls1 =getSynsets(Attribute1, tokenize);
List lls2 =getSynsets(Attribute2, tokenize);
foreach ls1 ∈ lls1 do
    foreach ls2 ∈ lls2 do
        max = 0;
        foreach s1 ∈ ls1 do
            foreach s2 ∈ ls2 do
                distance = lin(s1, s2);
                if distance > max then
                    max = distance;
        result = result + max;
result = result ÷ (lls1.size()∗lls2.size());
return result;
```

Function 3.4: WordNetSimilarity.getSynsets(*attribute*, *tokenize*)

```
input  : An attribute attribute
input  : A boolean tokenize
output: A list of a list of synsets similar to the attribute
List ls =searchWN(attribute);
if ls = ∅ then
    List lls;
    if tokenize then
        List attrs = tokenize(attribute);
        foreach attribute ∈ attrs do
            List ls =searchWN(attribute);
            if ls ≠ ∅ then
                lls.add(ls);
else
    List lls =List(ls);
return lls
```

3.6 Statistics

We split a rule into its conditions and extracted their attribute labels with the Semantic Heuristic in section 3.4 and calculated a similarity score for each pair of attribute labels with the WordNet Similarity in section 2.2. To combine these similarity values, we implemented three basic statistics that each put the emphasis on a different part of the range of values and return the result to the Semantic Heuristic.

Maximum

The maximum statistic chooses the biggest of the similarity values we got from a pairwise comparison of the conditions in a rule. Thus it gives the similarity of the two conditions that are most similar and ignores the rest.

Function 3.5: Maximum.calculate(l)

input : A list of values l
output: The biggest value in the list (or zero if list is empty)
Value $m = 0$;
foreach $v \in l$ **do**
  if $v > m$ **then**
    $m = v$
return m;

Mean

The mean statistic calculates the mean of the similarity scores associated with the pairs of conditions we get from a rule. Thus it gives a balanced view on the similarity of conditions in a rule, incorporating the scores of both the similar as well as the dissimilar condition pairs equally.

Function 3.6: Mean.calculate(l)

input : A list of values l
output: The mean of the list elements
Value $m = 0$;
foreach $v \in l$ **do**
  $m = m + v$;
return m/l.length();

Minimum

The minimum statistic basically operates as the opposite of the maximum statistic, choosing the score of the most dissimilar condition pair and disregarding all other condition pairs.

Function 3.7: Minimum.calculate(l)

input : A list of values l
output: The smallest value in the list or zero if list is empty
Value $m = 1.0$;
if $l = \emptyset$ **then**
    return 0.0;
else
    foreach $v \in l$ **do**
        if $v < m$ **then**
            $m = v$
    return m;

3.7 Summary of the Evaluation Process

This section will illustrate for the general case how the individual algorithms work together to create the overall rule score. Special cases (like for example rules with only one condition) are covered in the individual sections.

The SeCo-Framework calls the Arithmetic Weighted Mean Heuristic to evaluate a candidate rule. The Arithmetic Weighted Mean Heuristic then calls both a classic heuristic and the Semantic Heuristic to get a classic and a semantic score.

The Semantic Heuristic splits the rule into attribute label pairs and calls the WordNet Similarity for each pair. After collecting all the similarity scores it uses one of the statistics to get a final semantic score which it returns to the Arithmetic Weighted Mean Heuristic.

The Arithmetic Weighted Mean Heuristic combines the semantic score and the classic score using a weighted sum and returns the values as the rule score to the SeCo-Framework.

4 Experimental Set-up

This chapter will describe the setting of our experiments and their constituting parts. It will list the heuristics we use in our experiments as well as the data sources. It will also explain the performance measures and other metrics we use to evaluate the resulting rulesets.

4.1 Heuristics

Heuristics differ both in scale and center of their range of values. When combining two or more heuristics it is generally necessary to normalize them in order to get meaningful results. For a summation of heuristic values as happens in this work, it is preferable to have heuristics that share the same range of values. For sake of simplicity we chose heuristics that match in scale and center and so circumvent the problem of normalization.

Classic Heuristics

The classic heuristics we chose are *Accuracy*, *m-Estimate* and *Laplace Estimate* (see Section 2.1.2). While there are many other (see Janssen and Fürnkranz [9] for an analysis) our choice fell on these because all three have a range of values between zero and one which saves us from using normalization and any losses that might be inherent to a non-perfect normalization. Another reason to choose this group of heuristics was that it has representatives of both *Precision* and *Accuracy*. These are two distinct approaches to evaluating a rule, the one values the coverage of many positive examples few negative examples (*Accuracy*) the other values for a high percentage of positive examples among the examples covered (*Precision*).

Semantic Heuristic

The semantic heuristic we use is described in Chapter 3. It can be used with different statistics that change how the similarity scores of the individual condition pairs in a rule are combined to create a semantic coherence score for the whole rule. It can be used with tokenization to increase the amount of compound attribute names that the underlying WordNet Similarity metrics can handle.

Combining Heuristics

The two heuristics will be combined with the help of the Arithmetic Weighted Mean Heuristic (see Section 3.3). The amount of semantic influence that goes into the final evaluation of the rule can be adjusted here.

4.2 Data Sources

The UCI repository[2] is a widely used source for databases, data-generators and domain-theories. It is curated by the University of Massachusetts Amherst since 1987 and today contains 260 datasets.

Most of these datasets feature technical terms from the various branches of science as attribute labels or use a simple enumeration system without semantic value to label attributes. This complicates the test of our semantic approach as either the label set lacks semantic content or the terms used are so uncommon that either the heuristic or the reader can not make sense of the attribute labels.

Table 2 lists the datasets we use together with their number of attributes and the percentage of those attributes found in WordNet. We chose these datasets because they cover many different domains, have a small to medium sized attribute set and differ strongly in the amount of the attributes that can be found in WordNet.

To show the influence of the semantic heuristic on the generated rules in more detail, we removed an attribute that was distributed about the same as the class in the set of examples to make it harder for the rule learning algorithm and to provoke longer and more different rules. We then changed the remaining 15 attribute labels to three random permutation of the following word list: yellow_bicycle, green_car, blue_train, orange_bus, red_ship, magazine, radio, newspaper, journal, book, television, flower, tree,

Dataset	Attr.	Attr. found	Dataset	Attr.	Attr. found
breast-w	9	0%	heart-statlog	13	31%
iris	4	0%	eucalyptus	19	37%
balance-scale	4	0%	diabetes	8	38%
echocardiogram	8	0%	anneal	38	44%
credit-a	15	0%	horse-colic	22	45%
monk2	6	0%	soybean	35	46%
credit	15	0%	vehicle	18	61%
vowel	10	0%	hepatitis	19	68%
ionosphere	34	0%	primary-tumor	17	71%
sonar	60	0%	bridges2	11	73%
wine	13	0%	zoo	17	94%
audiology	69	4%	flag	27	100%
lymphography	18	6%	auto-mpg	7	100%
vote	16	13%	balloons	4	100%
credit-g	20	30%	glass	9	100%
Heart-c	13	31%			

Table 2: UCI dataset labels found in WordNet

bush, plant. This results in three permutated attribute lists and thus three datasets. We call this the modified vote dataset (permutation 1-3).

This list of words can be split into three conceptual groups: color coded means of transportation, paper media and flora. It can also be split into single word concepts and composite word concepts specifically the color coded means of transportation and the rest.

This labelling enables us to easily spot semantic similarity and dissimilarity between the conditions of a rule which are less than obvious in label sets featuring scientific domain language or indescriptive labels. It thus makes the workings of the semantic heuristic more obvious and facilitates the analysis of the generated rulesets.

4.3 Performance Measures

Macro Average Accuracy

We use the Macro Average Accuracy (see section 2.1.3) to gauge how well the rulesets learned the statistical patterns underlying the example set. We also employ ten-fold cross-validation to reach more honest results.

Semantic Coherence

To get a semantic coherence score for a ruleset we calculate first the semantic coherence of each rule using tokenization and the mean statistic and then the mean of the individual rule scores in a ruleset. This does not take into account how semantically coherent the rules are to one another but it creates a coherence score for a ruleset that is independent of the number of rules and the number of conditions within the rules and makes the semantic coherence of different sets of rules comparable.

4.4 Methodology

The evaluation of the semantic coherence idea happens in three scenarios that cover different scopes. While the first scenario looks in detail at a single dataset the second scenario takes a broad view and employs semantic and classic performance measures to compare the use of the different statistics on the whole of the 31 unmodified datasets. The third and last scenario then splits the group of datasets into smaller groups, looks at the loss of ruleset performance with increasing semantic influence in the rule evaluation and looks at individual rulesets in more detail.

In the first scenario we will choose the best of the three classic heuristics for a single dataset and try to improve the semantic coherence of the generated rules by trying the different statistical methods as well as by trying the use of tokenization. We will analyse the generated rulesets and interpret how they were generated. This should give an insight into the workings of the different statistics in combination with the rule learning process.

In the second scenario we will then generate some semantic and classic performance data for the different statistics and classic heuristics and decide which we will use in the third scenario.

In the third scenario we will measure the impact of increasing semantic influence on rule score in different classes of datasets. The goal in this scenario is to get an idea of the sacrifice in rule quality that is to be expected when adding a semantic heuristic with increasing strength. We will also look at how individual rulesets were changed with the addition of the semantic heuristic.

5 Experiments and Results

5.1 Scenario 1

In the first scenario, we will look at the influence of the semantic heuristic on the semantic coherence of the generated rules. We will use the modified vote dataset for this. We will discuss permutation one here, two other permutations can be found in the appendix for further reference.

Step 1

The first step is to determine which classical heuristic we want to use as a counterpart to the semantic heuristic for each of the three modified datasets. Table 3 shows the Macro Average Accuracy on the

	wine
Accuracy	88.276
Laplace Estimate	87.356
m-Estimate	89.885

Table 3: Scenario 1, Step 1: Comparison of Classic Heuristics

validation set of each of the modified vote datasets[18]. *m-Estimate* clearly outperforms the other two heuristics. Thus we will choose the *m-Estimate* for the following experiments.

Step 2

The next step is to look at the attribute labels of the rules' conditions themselves and evaluate our statistical options.

Listing 9: no semantic influence

```
1 yellow_bicycle, book, journal, plant, blue_train
2 yellow_bicycle, flower, orange_bus
3 newspaper, book, flower
4 yellow_bicycle, plant, magazine
5 flower, blue_train, journal, red_ship
```

Each line in Listing 9 shows the attributes that were chosen for a rule in the ruleset produced by *m-Estimate* without any semantic influence. The attributes appear in the order they were chosen by the algorithm. The underlying rulesets can be found in the appendix. This ruleset consists of 5 rules and 18 conditions. It has an average rule length of 3.6 and a low semantic coherence score of 25.3%. Rule 4 is a prime example of a semantically incoherent rule. Yellow bicycles have nothing to do with plants and both have nothing to do with magazines. Inconsistencies like these make a rule semantically incoherent and we try to reduce these inconsistencies by adding 10% of the semantic coherence score to the heuristic mix:

Listing 10: statistic: min | tokenization: off

```
1 yellow_bicycle, book, journal, plant, blue_train
2 yellow_bicycle, flower, orange_bus
3 newspaper, book, magazine
4 flower, tree, bush
5 yellow_bicycle, plant, magazine
6 red_ship, book, journal, television
```

[18] See chapter 2 on ruleset evaluation for an explanation.

As can be examined in Listings 9 and 10, using the minimum statistic results in a ruleset that is similar to the one without semantic influence. The first two rules start with a compound attribute. Since we do not use tokenization yet the semantic heuristic can not compute proper similarity values between this condition and other conditions, it will always return zero. Since this is the minimal value and we use the minimum statistic the semantic heuristic will return zero for every rule that contains this condition. The rule learning process is continuing to build this rule as if there was no semantic heuristic involved because the semantic heuristic returns a constant zero. The same can be seen in rule two.

The first difference is the choice of magazine over flower in the third rule. In the ruleset without semantic influence the most dissimilar pair in rule 3 is one involving flower. Choosing magazine instead of flower apparently resulted in a difference in semantic score big enough to overcome the difference in classic scores. The rule without semantic influence covered 15 positive and 1 negative example while the rule with semantic influence covered 15 positive and 3 negative examples. This change in coverage results in a different set of remaining examples changing the starting conditions for the following rules.

Listing 11: statistic: max | tokenization: off

```
1  yellow_bicycle, book, journal, plant, blue_train
2  yellow_bicycle, flower, tree
3  yellow_bicycle, flower, plant, magazine
4  flower, book, journal, red_ship
5  newspaper, magazine, book, journal, television
6  yellow_bicycle, orange_bus, flower
7  red_ship, television, newspaper, tree, book
8  flower, tree, blue_train, radio, television, journal
9  red_ship, television, newspaper, plant, tree, bush
10 newspaper, magazine, radio, red_ship, journal
11 flower, plant, book, tree, bush
```

When looking at the results of the maximum statistic in Listing 11, it becomes apparent that the first rule is again the same as without semantic heuristic. After yellow_bicycle was chosen by the classic heuristic[19] the next choice is again not influenced by the semantic heuristic. Still all semantic comparisons to other conditions would result in zero because of the lack of tokenization. The next choice though is influenced by the semantic heuristic but in this case the choice of the classic heuristic (journal) is the same as that of the semantic heuristic. The combination book-journal is the one with the largest similarity score (84.2%) of all the combinations that involve book or journal. The result is that the next choice is again independent of the semantic heuristic as this score can not be topped by any combination with any other condition. With the rule score only influenced by the classic heuristic at this point the algorithm decides to add plant. Plant has combinations with other conditions that have a higher score than the combination book-journal for example plant-tree (92%) or plant-flower (84.5%) but this difference is not enough to raise the total rule score over the score resulting from choosing blue_train.

The second rule is the first to change when using the maximum statistic. The algorithm chooses to add tree instead of adding orange_bus as this raises the semantic score from zero to 81%. Choosing tree over orange_bus results in a change in coverage from 39 to 26 positive examples and 8 to 4 negative examples. But increases the semantic score from zero to 81%. The following rules are again learned on a different set of examples due to the change in coverage.

[19] The first condition is always chosen by the classic heuristic. See Section 3.3.

Listing 12: statistic: mean | tokenization: off

```
1 yellow_bicycle, book, journal, magazine
2 yellow_bicycle, flower, orange_bus
3 newspaper, book, magazine
4 flower, tree, bush
5 yellow_bicycle, plant, flower, television
6 red_ship, book, journal, television
```

In Listing 12 we see that the first rule generated with the help of the mean statistic is different from the ones we looked at before. The mean statistic is the first to make a difference here. Because we use the mean instead of the minimum or maximum the semantic heuristic is highly more likely to return different results for different refinements (adding of a condition). The choice of the second condition lies again in the hands of the classic statistic as explained in the last paragraphs. The mean statistic then agrees with the maximum and minimum statistics and the classic heuristic to add journal to the rule as this is the best combination with book. But then it prefers magazine to be the next choice as this has a high similarity score with both journal (62.4%) and book (81.7%). The combinations book-plant (35.5%) and journal-plant (26.2%) result in a much smaller semantic rule score when using the mean statistic. This results in change in coverage from 89 to 93 positive examples and 0 to 2 negative examples.

Configuration	Coherence Score	Average rule length	Number of rules
Without Semantic Heuristic	25.3	3.60	5
Using the Minimum Statistic	34.2	3.50	6
Using the Mean Statistic	45.0	3.50	6
Using the Maximum Statistic	32.9	4.64	11

Table 4: Scenario 1, Step 1: Comparison of Coherence Scores without Tokenization

The overall changes in semantic coherence can be seen in Table 4. All three statistics resulted in an overall increase in semantic coherence but the mean statistic outperformed the other two by over 10%. All but the maximum statistic also reduced the average number of conditions in a rule. While the minimum and mean statistics increased the number of rules by 1, the maximum statistic more than doubled the number resulting in a ruleset of 11 rules.

The maximum statistic seems to restrict the freedom of the rule learning process when the next condition to add can greatly increase the semantic rule score but it does not restrict adding conditions to a rule as the semantic score of a rule does not decrease by adding many dissimilar conditions. The minimum statistic does restrict this addition of conditions that are even more dissimilar than the conditions already in the rule but can not promote the addition of more similar conditions. If a pair of most dissimilar conditions is present in a rule it ceases to influence the choice of conditions. The mean statistic promotes the addition of conditions that increase the semantic coherence of the rule regardless of the current state of semantic coherence. It also demotes the addition of conditions that would create a pair of dissimilar conditions.

Let's see if enabling tokenization can increase the semantic coherence even more.

Step 3

As a last step, we try to improve the semantic coherence further by introducing a tokenizing preprocessing step in cases where our semantic heuristic does not recognize a compound attribute.

Listing 13: statistic: min | tokenization: on

```
1  yellow_bicycle, book, journal, plant, blue_train
2  yellow_bicycle, flower, orange_bus
3  newspaper, book, magazine
4  flower, tree, bush
5  yellow_bicycle, plant, magazine
6  red_ship, book, journal, television
```

Listing 13 shows the results of using the minimum statistic with tokenization enabled. The result is the same as without tokenization which might come as a surprise because now the compound attribute labels do no longer cause a score of zero for every combination with another attribute label. These combinations result instead in a value which is the mean of all the individual comparisons. The comparison of yellow_bicycle and book for example is the mean of the comparisons of yellow and book as well as bicycle and book. But the result of this combination only gets a small score of 12.8%. Adding journal to the rule results in a new minimum value of 10.3% that is associated with the combination yellow_bicycle-journal but this difference of only 2.5 percentage points is so small that it can not counter the increase on the side of the classic heuristic which is also weighted 9 times heavier in the Arithmetic Weighted Mean Heuristic (10% semantic heuristic vs 90% classic heuristic in this configuration). For a combination including a compound attribute label to have a high semantic score most of the compounds must be similar to the other attribute label (see section 3.5). In this case using tokenization does not change the generated ruleset, this is not always the case though.

Listing 14: statistic: max | tokenization: on

```
1  yellow_bicycle, book, journal, plant, blue_train
2  yellow_bicycle, flower, tree
3  yellow_bicycle, flower, plant, magazine
4  flower, book, journal, red_ship
5  newspaper, magazine, book, television, journal
6  yellow_bicycle, orange_bus, flower
7  red_ship, television, newspaper, tree, book
8  flower, tree, blue_train, radio, television, journal
9  red_ship, television, newspaper, plant, tree, bush
10 newspaper, magazine, radio, red_ship, journal
11 flower, plant, book, tree, bush
```

As a comparison of Listings 14 and 11 shows the differences between these rulesets are not big either. The first change appears in rule 5 where with television is chosen before journal instead of the other way round. The combination newspaper-magazine with a similarity score of 89.8% trumps every other combination in the rule. Thus the semantic coherence score of the rule did not change when television or journal was added regardless in which order. The change in order can thus not be attributed to the semantic heuristic. We attribute it to a tie-breaking mechanism and thus chance. The rest of the ruleset is the same as without tokenization. Since enabling tokenization mostly changes zero values to small values (unless all the parts of a compound label are similar to another attribute label) this should have little effect on the maximum of the comparison scores as long as there are reasonably similar conditions present like in this case. It should have an effect though if most of the attribute labels are dissimilar to each other.

Listing 15: statistic: mean | tokenization: on

```
1 yellow_bicycle, book, journal, magazine
2 yellow_bicycle, flower, orange_bus
3 newspaper, book, magazine
4 flower, tree, bush
5 yellow_bicycle, plant, flower
6 red_ship, book, journal
```

The mean statistic experiences a bigger change when we add tokenization as can be seen in Listing 15. While the first 4 rules remain the same rule 5 is shortened. With tokenization television is not added to the rule. To explain this change we have to look at the similarity scores of the individual condition pairs. yellow_bicycle-plant and yellow_bicycle-flower both had a score of 0 without tokenization. With a score of 84.5% for the combination plant-flower the mean was 28.2% before adding television and 22.1% after adding it. Now with tokenization enabled yellow_bicycle-plant gets a score of 9.6% and yellow_bicycle-flower a score of 15.4% this change results in a larger mean of 36.5%. Adding television lowers the score to 28.3%. Without tokenization adding television resulted in a reduction of the semantic coherence of 6.1% with tokenization enabled the reduction would have been 8.2% enough to counter the classic heuristic. Not adding television to the rule increased the coverage of negative examples from 3 to 4 while the coverage of positive examples remained at 6. It appears that something similar happened to rule 6 although we can not be sure of this since the coverage of the previous rules changed if even so slightly.

Configuration	Coherence Score	Average rule length	Number of rules
Without Semantic Heuristic	25.3%	3.60	5
Using the Minimum Statistic	34.2%	3.50	6
Using the Mean Statistic	46.7%	3.17	6
Using the Maximum Statistic	32.9%	4.64	11

Table 5: Scenario 1, Step 1: Comparison of Coherence Scores with Tokenization

Table 5 shows the semantic coherence scores for the different statistics this time with tokenization enabled. The only change is to the mean statistic where the coherence score improved by 1.7 percentage points and the average rule length decreased from 3.5 to 3.17.

While the influence of enabling tokenization on both maximum and minimum statistic were too small in this case, the difference on the mean statistic were still small but big enough. Getting small but positive scores instead of 0 for the combinations with compound attribute labels increased the semantic coherence score enough to demote the addition of a condition that would have reduced the semantic coherence of the rule.

Summary

In Step 1 our choice fell on the *m-Estimate* because of its superior performance on this dataset. In Step 2 we then looked at the influence of the different statistics and found that when using the maximum statistic the rule learning algorithm searches hungrily for the most similar condition pair but behaves as it normally would without semantic heuristic if there is no condition pair in the current set of refinements that is more similar than the pairs already in the rule. The minimum statistic only tried to avoid adding conditions which are more dissimilar than the most dissimilar pair of conditions already in the rule. The mean statistic exerts a constant pull towards conditions that improve the semantic coherence on the rule learning process. In Step 3 we looked at the influence of tokenization and found that there are cases where this influence is pretty small. There are however cases in which it is sufficient to improve the semantic coherence of the learned rules and there are also other cases in which this influence is larger.

The condition pair train_station-bus_station for example would get a score of 0 without tokenization and a score of 62.8% with tokenization. This is ten times the increase we saw on the modified vote dataset.

For the following experiments, we will enable tokenization. While there are cases (like we have seen with tokenization and the minimum and maximum statistic) where tokenization does not alter the generated ruleset it is still useful to improve the quality of the semantic coherence score.

To decide which statistic to use let's create some statistics in the next scenario.

5.2 Scenario 2

In the second scenario we will decide which statistic to use for the following experiments. To this goal we will learn a ruleset on every of the 31 unmodified datasets for each of the three statistics and each of the three classic heuristics. Table 6 shows the Semantic coherence scores while Table 7 shows the Macro Average Accuracy results.

Statistic	No semantic heuristic	Minimum	Mean	Maximum
m-Estimate	11.827%	16.128%	**16.599%**	16.387%
Laplace Estimate	11.011%	15.006%	13.333%	**15.095%**
Accuracy	11.980%	17.845%	**18.107%**	16.481%
Overall	11.606%	**16.326%**	16.013%	15.988%

Table 6: Scenario 2: Comparison of Semantic Coherence Scores

The overall result visible in the last line of Table 6 shows that the minimum statistic outperforms the other two statistics by just a little bit if all the results from the individual classic heuristics are combined. Looking at the individual results shows the mean statistic two times on place 1 and one time on place 3, the minimum statistics two times on place 2 and one time on place 3 and the maximum statistic one time on place 1, 2 and 3. The runs without semantic heuristic always end up on place 4. The use of a semantic heuristic improves the overall semantic coherence regardless of which statistic is used, but the mean statistic improves the overall semantic coherence for two classic heuristics and is overall just a tiny fraction behind the minimum which is first in none of the individual classic heuristic runs.

Statistic	No semantic heuristic	Minimum	Mean	Maximum
m-Estimate	76.728%	76.671%	76.163%	76.540%
Laplace Estimate	75.064%	74.722%	74.877%	74.691%
Accuracy	74.067%	73.480%	74.248%	73.771%
Overall	75.286%	74.958%	75.096%	75.001%

Table 7: Scenario 2: Comparison of Macro Average Accuracy

Table 7 shows the mean Macro Average Accuracy over all the 31 unmodified datasets. The first column displays the results of the run without semantic heuristic, the following 3 columns display the results for the runs using the different statistics and 10% semantic coherence heuristic. It seems that adding 10% of the semantic score to the rule score mix did not result in a large loss of ruleset performance. About half of the datasets did not have attribute labels that could be found in WordNet though. Nonetheless we registered an increase in semantic coherence of roundabout 5 percentage points as we saw in Table 6.

The third scenario will show how the loss in ruleset performance will develop with increasing amounts of semantic influence. For these experiments we will restrict ourselves to the *m-Estimate* heuristic, which gives the overall best performance on the 31 rulesets and to the mean statistic as this gives the best semantic coherence when the *m-Estimate* is used. We will also keep on using tokenization.

5.3 Scenario 3

The third scenario will cover the semantic heuristic's impact on rule performance. To this end we will perform different experiments on four classes of datasets. First we will introduce each group of datasets listing the datasets involved and giving the mean semantic similarity of the attribute pairs in the group of datasets. We then continue to provide the same data that we gathered in scenario 2 but this time on the group of datasets instead of all the datasets at once. Lastly we will look at how to average ruleset performance develops if we increase the semantic influence even more for the case of the *m-Estimate* and using tokenization and the mean statistic.

Set 1: only badly labelled attributes

The set of datasets used for these experiments has no attribute labels that can be found in WordNet. But tokenization can help to find some compound attribute labels with components that can be found in WordNet. This group incorporates the following datasets: audiology, balance-scale, breast-w, credit, credit-a, echocardiogram, ionosphere, iris, lymphography, monk2, sonar, vote, vowel and wine.

The mean semantic similarity between the pairs of attribute labels in this set is 5.4%. This score can be reached despite the lack of easily recognized attribute labels because tokenization makes some of the compound attribute labels accessible to the semantic heuristic. Nonetheless the semantic heuristic score is mostly low and thus has little influence on the classic performance score of the *m-Estimate* heuristic.

Statistic	No semantic heuristic	Minimum	Mean	Maximum
m-Estimate	11.188%	15.583%	16.682%	**18.200%**
Laplace Estimate	10.641%	17.316%	**17.508%**	16.568%
Accuracy	13.943%	16.604%	**16.701%**	15.547%

Table 8: Scenario 3: Set 1 Comparison of Semantic Coherence for 0% and 10% semantic influence

Table 8 shows the semantic coherence scores of the rulesets that can be reached by adding 10% semantic influence to the rule evaluation. It is remarkable that with the help of tokenization an increase of up to 7 percentage points in semantic coherence can be achieved on this group of datasets. While the mean statistic reaches the best results with the *Laplace Estimate* and *Accuracy* heuristics but does it only with a small distance to the minimum statistic, the max statistic has the lead when using the *m-Estimate* heuristic with a difference of 1.5 percentage points.

Statistic	No semantic heuristic	Minimum	Mean	Maximum
m-Estimate	2.83	2.87	2.89	2.88
Laplace Estimate	2.52	2.54	2.33	2.53
Accuracy	2.60	2.56	2.44	2.60

Table 9: Scenario 3: Set 1 Comparison of Average Rule Lengths for 0% and 10% semantic influence

Table 9 shows the average rule lengths for this group of datasets. Although the semantic heuristic resulted in an increase in semantic coherence the average rule lengths do not differ from the results without semantic heuristic very much.

Heuristic	influence										
	0	10	20	30	40	50	60	70	80	90	100
m-Estimate	95.46	92.81	92.48	92.19	91.84	91.32	91.03	90.79	91.69	91.02	56.49
Accuracy	94.77	91.61	91.13	91.04	90.54	90.57	90.41	90.39	90.42	90.23	50.37
Laplace Est.	98.69	95.08	94.75	94.68	94.71	94.40	95.20	95.17	94.65	93.25	58.80

Table 10: Macro Average Accuracy | Set 1 | mean | tokenization | training

Heuristic	influence										
	0	10	20	30	40	50	60	70	80	90	100
m-Estimate	82.38	80.87	80.64	80.50	80.19	80.16	80.08	79.85	79.62	79.02	53.02
Accuracy	79.56	78.39	78.05	77.80	77.84	77.84	77.16	77.12	76.95	76.82	53.33
Laplace Est.	80.06	79.56	79.44	79.84	79.48	79.29	78.64	78.73	78.70	77.98	53.93

Table 11: Macro Average Accuracy | Set 1 | mean | tokenization | validation

Since most of the attribute pairs just get a 0 on the semantic similarity scale, those attribute pairs that score higher than 0 get a boost, even if they just have a low semantic similarity. The special cases where an attribute is semantically compared with itself and thus receives a score of 1.0 create a bias towards rules with multiple occurrences of conditions in datasets containing numerical attributes. A semantic influence of 90% results in a loss of only 2.08% in ruleset performance for the case of the *Laplace Estimate*. The other classic statistic runs show similar results.

Set 2: mostly badly labelled attributes

The group of datasets used in the second set of experiments has up to 30% attribute labels that can be found in WordNet. It incorporates these datasets: anneal, credit-g, diabetes, eucalyptus, heart-statlog, horse-colic and soybean. In this set the mean semantic similarity between attribute pairs is 9.3%.

Statistic	No semantic heuristic	Minimum	Mean	Maximum
m-Estimate	12.651%	14.368%	**14.559%**	13.374%
Laplace Estimate	9.800%	12.573%	10.592%	**13.510%**
Accuracy	11.696%	19.338	% **21.533%**	17.542%

Table 12: Scenario 3: Set 2 Comparison of Semantic Coherence for 0% and 10% semantic influence

The semantic coherence scores for the 0% and 10% semantic influence are shown in Table 12. Again the leading statistics are the mean and maximum statistic. The differences in semantic coherence gain are large in this group of datasets. When using *Accuracy* the mean statistic grants 10 percentage points more semantic coherence but when using *m-Estimate* the difference is only 2 percentage points. If only a few attribute labels are recognizable by the semantic heuristic the semantic coherence score can vary depending on which classic heuristic is used. The two derivatives of *Precision* profit noticeably less than *Accuracy* from the semantic heuristic on this group of datasets. This holds true regardless of which statistic is used.

Statistic	No semantic heuristic	Minimum	Mean	Maximum
m-Estimate	3.65	3.56	3.63	3.77
Laplace Estimate	2.46	2.50	2.33	2.50
Accuracy	2.46	2.28	2.52	2.78

Table 13: Scenario 3: Set 2 Comparison of Average Rule Lengths for 0% and 10% semantic influence

As Table 13 shows the drastic increase in semantic coherence in the experiment run with *Accuracy* just causes an increase of 0.06 in average rule length. The maximum statistic produces the overall longest rules for this dataset although these rules are only slightly longer than those created by the the use of the other statistics.

Heuristic	influence										
	0	10	20	30	40	50	60	70	80	90	100
m-Estimate	95.62	88.41	88.61	88.01	86.95	85.58	86.69	85.83	84.67	83.63	53.21
Accuracy	96.74	85.88	85.38	85.44	84.38	83.66	83.73	83.69	84.11	83.16	53.21
Laplace Est.	99.88	90.72	88.45	90.92	88.94	90.08	89.94	89.59	87.07	86.25	53.48

Table 14: Macro Average Accuracy | Set 2 | mean | tokenization | training

Heuristic	influence										
	0	10	20	30	40	50	60	70	80	90	100
m-Estimate	77.78	78.64	78.65	78.73	78.96	78.22	77.63	77.44	77.08	76.96	53.91
Accuracy	75.58	77.01	77.13	76.61	75.97	76.29	75.97	75.86	75.49	76.02	54.41
Laplace Est.	75.51	77.19	77.23	77.19	77.24	77.48	77.85	78.49	77.30	76.85	53.68

Table 15: Macro Average Accuracy | Set 2 | mean | tokenization | validation

Surprisingly the ruleset performance on the validation set increases slightly when the semantic heuristic is used. This increase is at its peak 2.91 percentage points for the case of the *Laplace Estimate* at 70% semantic influence. We interpret these values as equal which means that adding up to 90% semantic influence does not alter the performance of the ruleset but as can be seen in Table 12 even 10% semantic influence show an increase in semantic coherence. Nonetheless we do not think increasing the semantic influence further would result in much more semantically coherent rules as most of the attribute labels hold no semantic value.

Set 3: mostly well labelled attributes

The third group of datasets features between 30% and 60% attribute labels from WordNet. It consists of the following datasets: bridges2, heart-c, hepatitis, primary-tumor, vehicle and zoo. In this group of datasets a lot more attribute labels are accessible to the semantic heuristic, but the mean similarity between those attribute labels is still just 13.2%.

Statistic	No semantic heuristic	Minimum	Mean	Maximum
m-Estimate	6.111%	8.513%	**10.493%**	7.853%
Laplace Estimate	4.326%	4.866%	**5.267%**	5.148%
Accuracy	5.306%	10.128%	**12.020%**	8.618%

Table 16: Scenario 3: Set 3 Comparison of Semantic Coherence for 0% and 10% semantic influence

Table 16 details the increase in semantic coherence when adding 10% semantic influence to the rule score. On this group of datasets the mean statistic wins against the minimum and maximum statistics regardless of which classic heuristic was used but the gain is remarkably bigger when using *m-Estimate* or *Accuracy*. The gain is much less when using the *Laplace Estimate*.

Statistic	No semantic heuristic	Minimum	Mean	Maximum
m-Estimate	3.34	2.20	3.11	3.37
Laplace Estimate	2.36	2.36	2.20	2.35
Accuracy	2.34	2.20	2.25	2.50

Table 17: Scenario 3: Set 3 Comparison of Average Rule Lengths for 0% and 10% semantic influence

The gain in semantic coherence we observed in Table 16 is accompanied mostly by a shortening of average rule length as can be observed in Table 17. The maximum heuristic produces again longer rules than the other statistics except for a near equal average rule length between the maximum and minimum statistics in the set of experiments with the *Laplace Estimate*. In the experiments with the *m-Estimate* and *Accuracy* heuristics the minimum statistic produces on average shorter rules than the other two statistics.

Heuristic	influence										
	0	10	20	30	40	50	60	70	80	90	100
m-Estimate	87.65	82.57	81.74	82.21	81.56	82.26	82.08	82.13	82.08	81.70	46.68
Accuracy	90.69	83.24	82.70	83.08	83.93	83.93	83.87	83.65	83.60	82.94	44.64
Laplace Est.	94.17	88.31	88.46	90.95	90.85	89.92	88.98	85.81	85.12	84.78	44.64

Table 18: Macro Average Accuracy | Set 3 | mean | tokenization | training

Heuristic	influence										
	0	10	20	30	40	50	60	70	80	90	100
m-Estimate	66.73	67.20	67.48	67.94	67.86	67.70	67.69	67.44	67.39	67.35	46.41
Accuracy	64.89	65.93	65.71	65.93	65.98	66.03	65.77	65.99	65.99	65.72	45.65
Laplace Est.	66.89	67.03	66.70	66.66	66.61	66.56	67.11	67.50	67.35	67.49	45.78

Table 19: Macro Average Accuracy | Set 3 | mean | tokenization | validation

On this group of datasets the increase in semantic influence again leads to a slight increase in ruleset performance in the *m-Estimate* run as is visible from table 19. It keeps increasing up to 30% semantic influence and then drops steadily. At its peak this increase is a mere 1.2 percentage points which is again not statistically significant. With this group of datasets one can add even up to 90% semantic influence without sacrificing ruleset performance. Since now more attribute labels are interpretable by the semantic heuristic this might indeed lead to a further increase in semantic coherence.

	0%	10%	20%	30%
Average Semantic Coherence	6.111%	10.493%	12.806%	14.346%
Average Rule length	3.34	2.20	3.13	3.03
Number of Rules	17.0	15.0	14.0	15.0

Table 20: Scenario 3: Set 3: 0% to 30% Semantic Influence Using the Mean Statistic and *m-Estimate*

Increasing the semantic influence does improve the average semantic coherence score slightly and comes at little cost regarding the average rule length and number of rules as is visible from Table 20.

Set 4: only well labelled attributes

The last set of datasets features only attributes that are also found in WordNet. The following datasets belong to this set: auto-mpg, balloons, flag and glass. Since the latest test set the mean semantic similarity has more than doubled and reaches a score of 28.8% for this set. This results in a larger influence on the score of the *m-Estimate* heuristic.

Statistic	No semantic heuristic	Minimum	Mean	Maximum
m-Estimate	23.191%	**34.604%**	31.139%	31.465%
Laplace Estimate	27.129%	29.067%	**34.874%**	30.230%
Accuracy	21.991%	32.116%	**36.620%**	31.604%

Table 21: Scenario 3: Set 4 Comparison of Semantic Coherence for 0% and 10% semantic influence

Immediately obvious from Table 21 is the vast increase in semantic coherence of both the experiment run without semantic influence as well as that with semantic influence. Although the semantic coherence is quite good for the classic runs, the semantic heuristic manages to improve on this result by adding another 11 percentage points in the case of the *m-Estimate* with mean statistic configuration and another 15 percentage points for the run with *Accuracy*.

Statistic	No semantic heuristic	Minimum	Mean	Maximum
m-Estimate	2.81	2.61	3.06	3.32
Laplace Estimate	2.37	2.38	2.37	2.42
Accuracy	2.27	2.25	2.36	2.57

Table 22: Scenario 3: Set 4 Comparison of Average Rule Lengths for 0% and 10% semantic influence

The great increase in semantic coherence observable in Table 21 happens again in parallel with a slight increase in average rule length as observable in Table 22. The maximum statistic continues to produce the largest average rule scores while the minimum statistic keeps the minimum average rule length for *m-Estimate* and *Accuracy* as it did in Set 2 and 3.

Heuristic	influence										
	0	10	20	30	40	50	60	70	80	90	100
m-Estimate	91.64	86.69	82.95	83.15	83.02	79.80	80.89	80.18	80.12	80.12	44.71
Accuracy	92.67	87.79	85.84	84.79	84.79	84.79	84.79	84.79	84.79	84.79	44.71
Laplace Est.	95.40	90.42	87.40	90.29	86.86	86.08	83.54	82.49	82.49	82.49	44.71

Table 23: Macro Average Accuracy | Set 4 | mean | tokenization | training

Heuristic	influence										
	0	10	20	30	40	50	60	70	80	90	100
m-Estimate	70.10	68.81	65.76	65.18	65.05	64.07	64.13	64.65	63.81	63.74	45.22
Accuracy	65.95	67.41	65.26	65.58	65.39	65.71	65.32	65.32	65.58	65.51	46.35
Laplace Est.	69.07	66.21	66.27	66.33	65.43	64.97	63.46	63.53	63.20	63.01	46.45

Table 24: Macro Average Accuracy | Set 4 | mean | tokenization | validation

A look at Table 24 shows that in this group of datasets increasing the semantic heuristic influence to high values leads to a slight decline in ruleset performance on all but the *Accuracy* runs. Nonetheless with a semantic influence of 90% there is still only a loss of 7 percentage points.

Summary

In all the four dataset groups our experiments have shown that a semantic influence of even 90% does not result in a drastic decrease of ruleset performance. This could possibly be explained by the small semantic similarity for most condition pairs. Half of the attribute label pairs from the four datasets had a score lower than 10% and a third of them had a score lower than 5%. Looking at just Set 4 the picture changes just a little. Still half of the attribute label pairs got a semantic score lower than 20%, a third of them got a score lower than 10% and a forth of them got a score lower than 5%. At the same time adding just 10% semantic influence can lead to great increases in semantic coherence and improves the semantic coherence in every setting we tested.

6 Conclusion and Future Work

6.1 Conclusion

Scenario 1

As we saw in Scenario 1 adding a small semantic bias to the scoring algorithm by combining semantic and classic heuristics can increase the semantic coherence of the generated rules by a considerable margin. The prerequisite for this gain in semantic coherence is a semantic heuristic that fits to the domain of the attribute labels.

We tested different statistical metrics as a means to combine the results of the pairwise comparison of attribute labels to a semantic coherence score for the whole rule. In this test the mean statistic produced the best results from a semantic coherence perspective. While the maximum focused too much on the single best attribute pair of a rule and the minimum focused too much on the single worst attribute pair, the mean results in a value that values every comparison within a rule equally. This makes the mean heuristic a more steady influence on the rule generating process as its value is changed whenever a condition is added and every new condition is relevant for the rule score.

The tokenization step enhances the ability of the semantic heuristic such that it can handle labels consisting of more than one word as long as some of the individual words can be handled. This leaves less attribute pairs with a score of 0 and therefore distributes the influence of the semantic heuristic more uniformly across the attribute space. Its influence can be small but even if so it can still influence the rule generating process to adopt semantically more coherent rules.

Scenario 2

In the second Scenario we generated data over the bulk of our datasets. We saw that the minimum statistic outperforms the mean statistic slightly when viewed over all the classic heuristics while the mean statistic is better than the minimum statistic on the runs with *m-Estimate* and *Accuracy* while the maximum statistic is better on the runs with *Laplace Estimate*. The semantic coherence could be improved regardless of which statistic was used and the gain in semantic coherence was around 5 percentage points while the decrease in ruleset performance was hardly one percentage point.

Scenario 3

In Scenario 3 we examined the result of using the semantically enhanced heuristic on different classes of datasets. We found that we could add up to 90% semantic influence to the rule score without encountering grave losses in ruleset performance while adding only 10% semantic influence increased the semantic coherence notably. We saw that with growing semantic influence the semantic coherence continues to increase. We saw a mixed but moderate influence on the average number of conditions where the maximum statistic tends to produce longer rules and the minimum statistic tends to produce slightly shorter rules.

Overview

All in all our experiments have shown that adding a semantic heuristic to the rule evaluation process can substantially increase the semantic coherence of the generated rules while coming with only a small penalty to the ruleset performance. As a prerequesite for this the semantic heuristic needs to fit to the domain of the attribute labels.

The mean heuristic provides the most steady influence on the rule generation process while the influence is more focussed to specific situations in the the generation process when the maximum or minimum statistics are used. These situations must provide the possibility to change the minimum or maximum similarity score for these statistics to influence the rule generation process.

The tokenization step increases the amount of attribute labels the semantic heuristic can handle. It is useful in cases where the attributes hold complex data points that can not be put into a single word. In

these cases it spreads the influence of the semantic heuristic more evenly over the attribute space and leads to more semantic coherence in the generated rulesets.

6.2 Future Work

The prototypical character of this work limited the scope of experiments to simple algorithms and scenarios. This leaves much room for improvement and further investigation of the approach presented in this work.

Other Semantic Heuristics

Using WordNet with unmodified attribute names has serious limitations. Nonetheless the results of our experiments appear promising. This makes the application of more potent heuristics, perhaps based on reasoning in ontologies, distance metrics in DBpedia[1] or the folksonomy of Wikipedia as for example in WikiRelate[18] all the more interesting.

Rule Quality Evaluation

An adequate evaluation of the rule quality with the help of human test groups would, given a positive outcome, improve the argumentative basis for further study in this direction.

WordNet Distance Metrics

Another possibility for future work is the evaluation of the performance of the different distance metrics defined on WordNet. The LESK[3] and the HSO[8] metrics both work with adjectives and adverbs in addition to nouns and verbs and they support arbitrary pairing of the POS classes.

Other Classical Heuristics

This work focussed on just a few heuristics that were easy to combine with the chosen semantic heuristic. The investigation of the interaction of different semantic and classic heuristics is a logical next step but requires the use of proper normalization mechanisms.

A More Experiment Results

A.1 Scenario 1

A.1.1 Attribute Permutation 1

Attribute order: orange_bus, bush, yellow_bicycle, flower, television, blue_train, green_car, journal, tree, book, newspaper, magazine, red_ship, plant, radio

Listing 16: permutation: 1 | without semantic heuristic

```
Class = r :- yellow_bicycle = n, book = n, journal = n, plant = n, blue_train = n.  [89|0] Val: 0.
Class = r :- yellow_bicycle = n, flower = y, orange_bus = n.  [39|8] Val: 0.635
Class = r :- newspaper = y, book = n, flower = y.  [15|1] Val: 0.468
Class = r :- yellow_bicycle = n, plant = n, magazine = y.  [7|3] Val: 0.277
Class = r :- flower = y, blue_train = y, journal = n, red_ship = y.  [7|3] Val: 0.261

Class = d.  [252|11]
```

Listing 17: permutation: 1 | statistic: min | tokenization: off

```
Class = r :- yellow_bicycle = n, book = n, journal = n, plant = n, blue_train = n.  [89|0] Val: 0.
Class = r :- yellow_bicycle = n, flower = y, orange_bus = n.  [39|8] Val: 0.572
Class = r :- newspaper = y, book = n, magazine = y.  [15|3] Val: 0.464
Class = r :- flower = y, tree = y, bush = n.  [6|4] Val: 0.294
Class = r :- yellow_bicycle = n, plant = n, magazine = y.  [6|3] Val: 0.217
Class = r :- red_ship = y, book = n, journal = n, television = y.  [4|1] Val: 0.168

Class = d.  [248|9]
```

Listing 18: permutation: 1 | statistic: mean | tokenization: off

```
Class = r :- yellow_bicycle = n, book = n, journal = n, magazine = y.  [93|2] Val: 0.817
Class = r :- yellow_bicycle = n, flower = y, orange_bus = n.  [37|6] Val: 0.577
Class = r :- newspaper = y, book = n, magazine = y.  [12|3] Val: 0.436
Class = r :- flower = y, tree = y, bush = n.  [6|4] Val: 0.303
Class = r :- yellow_bicycle = n, plant = n, flower = y, television = y.  [6|3] Val: 0.241
Class = r :- red_ship = y, book = n, journal = n, television = y.  [4|1] Val: 0.194

Class = d.  [248|10]
```

Listing 19: permutation: 1 | statistic: max | tokenization: off

```
Class = r :- yellow_bicycle = n, book = n, journal = n, plant = n, blue_train = n.  [89|0] Val: 0.
Class = r :- yellow_bicycle = n, flower = y, tree = y.  [26|4] Val: 0.615
Class = r :- yellow_bicycle = n, flower = y, plant = n, magazine = y.  [15|5] Val: 0.482
Class = r :- flower = y, book = n, journal = n, red_ship = y.  [15|4] Val: 0.472
Class = r :- newspaper = y, magazine = y, book = n, journal = y, television = y.  [6|0] Val: 0.338
Class = r :- yellow_bicycle = n, orange_bus = n, flower = y.  [4|1] Val: 0.177
Class = r :- red_ship = y, television = n, newspaper = n, tree = y, book = n.  [5|3] Val: 0.261
Class = r :- flower = y, tree = y, blue_train = y, radio = y, television = y, journal = n.
[3|0] Val: 0.221
Class = r :- red_ship = y, television = n, newspaper = n, plant = n, tree = y, bush = y.  [1|0] Va
Class = r :- newspaper = y, magazine = n, radio = y, red_ship = y, journal = y.  [1|0] Val: 0.142
Class = r :- flower = y, plant = y, book = n, tree = n, bush = y.  [1|0] Val: 0.141

Class = d.  [250|2]
```

Listing 20: permutation: 1 | statistic: min | tokenization: on

```
Class = r :- yellow_bicycle = n, book = n, journal = n, plant = n, blue_train = n.  [89|0] Val: 0.
Class = r :- yellow_bicycle = n, flower = y, orange_bus = n.  [39|8] Val: 0.587
Class = r :- newspaper = y, book = n, magazine = y.  [15|3] Val: 0.464
Class = r :- flower = y, tree = y, bush = n.  [6|4] Val: 0.294
Class = r :- yellow_bicycle = n, plant = n, magazine = y.  [6|3] Val: 0.226
Class = r :- red_ship = y, book = n, journal = n, television = y.  [4|1] Val: 0.18

Class = d.  [248|9]
```

Listing 21: permutation: 1 | statistic: mean | tokenization: on

```
Class = r :- yellow_bicycle = n, book = n, journal = n, plant = n, blue_train = n.  [89|0] Val: 0.
Class = r :- yellow_bicycle = n, flower = y, orange_bus = n.  [39|8] Val: 0.635
Class = r :- newspaper = y, book = n, flower = y.  [15|1] Val: 0.468
Class = r :- yellow_bicycle = n, plant = n, magazine = y.  [7|3] Val: 0.277
Class = r :- flower = y, blue_train = y, journal = n, red_ship = y.  [7|3] Val: 0.261

Class = d.  [252|11]
```

Listing 22: permutation: 1 | statistic: max | tokenization: on

```
Class = r :- yellow_bicycle = n, book = n, journal = n, plant = n, blue_train = n.  [89|0] Val: 0.
Class = r :- yellow_bicycle = n, flower = y, tree = y.  [26|4] Val: 0.615
Class = r :- yellow_bicycle = n, flower = y, plant = n, magazine = y.  [15|5] Val: 0.482
Class = r :- flower = y, book = n, journal = n, red_ship = y.  [15|4] Val: 0.472
Class = r :- newspaper = y, magazine = y, book = n, television = y, journal = y.  [6|0] Val: 0.338
Class = r :- yellow_bicycle = n, orange_bus = n, flower = y.  [4|1] Val: 0.207
Class = r :- red_ship = y, television = n, newspaper = n, tree = y, book = n.  [5|3] Val: 0.261
Class = r :- flower = y, tree = y, blue_train = y, radio = y, television = y, journal = n.
[3|0] Val: 0.221
Class = r :- red_ship = y, television = n, newspaper = n, plant = n, tree = y, bush = y.  [1|0] Va
Class = r :- newspaper = y, magazine = n, radio = y, red_ship = y, journal = y.  [1|0] Val: 0.142
Class = r :- flower = y, plant = y, book = n, tree = n, bush = y.  [1|0] Val: 0.141

Class = d.  [250|2]
```

A.1.2 Attribute Permutation 2

Attribute order: newspaper, flower, journal, bush, tree, orange_bus, yellow_bicycle, red_ship, blue_train, television, plant, book, radio, green_car, magazine

Listing 23: permutation: 2 | without semantic heuristic

```
Class = r :- journal = n, television = n, red_ship = n, green_car = n, orange_bus = n.  [89|0] Val
Class = r :- journal = n, bush = y, newspaper = n.  [39|8] Val: 0.635
Class = r :- plant = y, television = n, bush = y.  [15|1] Val: 0.468
Class = r :- journal = n, green_car = n, book = y.  [7|3] Val: 0.277
Class = r :- bush = y, orange_bus = y, red_ship = n, radio = y.  [7|3] Val: 0.261

Class = d.  [252|11]
```

Listing 24: permutation: 2 | statistic: min | tokenization: off

```
Class = r :- journal = n, book = y.  [123|13] Val: 0.832
Class = r :- plant = y, bush = y.  [25|11] Val: 0.519

Class = d.  [243|20]
```

Listing 25: permutation: 2 | statistic: mean | tokenization: off

```
Class = r :- journal = n, book = y.  [123|13] Val: 0.832
Class = r :- plant = y, bush = y, television = n.  [22|1] Val: 0.545

Class = d.  [253|23]
```

Listing 26: permutation: 2 | statistic: max | tokenization: off

```
Class = r :- journal = n, book = y, television = n, green_car = n.  [95|2] Val: 0.865
Class = r :- journal = n, bush = y, plant = y.  [35|7] Val: 0.638
Class = r :- bush = y, television = n, magazine = y, radio = y.  [19|4] Val: 0.524
Class = r :- journal = n, book = y, bush = y.  [5|3] Val: 0.278

Class = d.  [251|14]
```

Listing 27: permutation: 2 | statistic: min | tokenization: on

```
Class = r :- journal = n, book = y.  [123|13] Val: 0.832
Class = r :- plant = y, bush = y.  [25|11] Val: 0.519
Class = r :- bush = y, orange_bus = y, television = n.  [9|2] Val: 0.304

Class = d.  [241|11]
```

Listing 28: permutation: 2 | statistic: mean | tokenization: on

```
Class = r :- journal = n, television = n, red_ship = n, green_car = n, orange_bus = n.  [89|0] Val
Class = r :- journal = n, bush = y, newspaper = n.  [39|8] Val: 0.635
Class = r :- plant = y, television = n, bush = y.  [15|1] Val: 0.468
Class = r :- journal = n, green_car = n, book = y.  [7|3] Val: 0.277
Class = r :- bush = y, orange_bus = y, red_ship = n, radio = y.  [7|3] Val: 0.261

Class = d.  [252|11]
```

Listing 29: permutation: 2 | statistic: max | tokenization: on

```
Class = r :- journal = n, book = y, television = n, green_car = n.  [95|2] Val: 0.865
Class = r :- journal = n, bush = y, plant = y.  [35|7] Val: 0.638
Class = r :- bush = y, television = n, magazine = y, radio = y.  [19|4] Val: 0.524
Class = r :- journal = n, book = y, bush = y.  [5|3] Val: 0.278

Class = d.  [251|14]
```

A.1.3 Attribute Permutation 3

Attribute order: television, green_car, bush, blue_train, plant, tree, magazine, radio, journal, newspaper, flower, orange_bus, book, red_ship, yellow_bicycle

Listing 30: permutation: 3 | without semantic heuristic

```
Class = r :- bush = n, newspaper = n, radio = n, red_ship = n, tree = n.  [89|0] Val: 0.876
Class = r :- bush = n, blue_train = y, television = n.  [39|8] Val: 0.635
Class = r :- flower = y, newspaper = n, blue_train = y.  [15|1] Val: 0.468
Class = r :- bush = n, red_ship = n, orange_bus = y.  [7|3] Val: 0.277
Class = r :- blue_train = y, tree = y, radio = n, book = y.  [7|3] Val: 0.261

Class = d.  [252|11]
```

Listing 31: permutation: 3 | statistic: min | tokenization: off

```
Class = r :- bush = n, flower = y.  [121|12] Val: 0.823
Class = r :- bush = n, tree = n, plant = y.  [12|3] Val: 0.454
Class = r :- blue_train = y, newspaper = n, yellow_bicycle = y, book = y.  [18|3] Val: 0.429

Class = d.  [249|17]
```

Listing 32: permutation: 3 | statistic: mean | tokenization: off

```
Class = r :- bush = n, flower = y, plant = y, newspaper = n.  [96|3] Val: 0.828
Class = r :- bush = n, tree = n.  [33|12] Val: 0.59
Class = r :- blue_train = y, newspaper = n, tree = y.  [18|2] Val: 0.451
Class = r :- flower = y, plant = y, yellow_bicycle = y, book = y.  [9|6] Val: 0.281

Class = d.  [244|12]
```

Listing 33: permutation: 3 | statistic: max | tokenization: off

```
Class = r :- bush = n, flower = y, newspaper = n, plant = y, radio = n, tree = n, red_ship = n.
[79|0] Val: 0.87
Class = r :- bush = n, flower = y, blue_train = y.  [41|9] Val: 0.652
Class = r :- bush = n, tree = n, blue_train = y, plant = y.  [12|3] Val: 0.465
Class = r :- blue_train = y, newspaper = n, radio = n, book = y.  [14|2] Val: 0.47
Class = r :- flower = y, plant = y, newspaper = n, orange_bus = y, radio = y.  [6|0] Val: 0.331
Class = r :- bush = n, plant = n, red_ship = n.  [4|1] Val: 0.257
Class = r :- blue_train = y, tree = y, bush = y, journal = y, yellow_bicycle = y, book = y.
[4|1] Val: 0.25

Class = d.  [251|8]
```

Listing 34: permutation: 3 | statistic: min | tokenization: on

```
Class = r :- bush = n, flower = y.  [121|12] Val: 0.823
Class = r :- bush = n, tree = n, plant = y.  [12|3] Val: 0.454
Class = r :- blue_train = y, newspaper = n, yellow_bicycle = y, book = y.  [18|3] Val: 0.442

Class = d.  [249|17]
```

Listing 35: permutation: 3 | statistic: mean | tokenization: on

```
Class = r :- bush = n, newspaper = n, radio = n, red_ship = n, tree = n.  [89|0] Val: 0.876
Class = r :- bush = n, blue_train = y, television = n.  [39|8] Val: 0.635
Class = r :- flower = y, newspaper = n, blue_train = y.  [15|1] Val: 0.468
Class = r :- bush = n, red_ship = n, orange_bus = y.  [7|3] Val: 0.277
Class = r :- blue_train = y, tree = y, radio = n, book = y.  [7|3] Val: 0.261

Class = d.  [252|11]
```

Listing 36: permutation: 3 | statistic: max | tokenization: on

```
Class = r :- bush = n, flower = y, newspaper = n, plant = y, radio = n, tree = n, red_ship = n.
[79|0] Val: 0.87
Class = r :- bush = n, flower = y, blue_train = y.  [41|9] Val: 0.652
Class = r :- bush = n, blue_train = y, plant = y, radio = n.  [16|4] Val: 0.496
Class = r :- blue_train = y, newspaper = n, book = y, yellow_bicycle = y, journal = y.  [13|1] Val
Class = r :- flower = y, tree = y, blue_train = y.  [5|1] Val: 0.288
Class = r :- bush = n, plant = n, red_ship = n.  [3|1] Val: 0.224

Class = d.  [251|11]
```

List of Algorithms

List of Tables

List of Listings

References

[1] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A Nucleus for a Web of Open Data. In *The Semantic Web*, pages 722–735. Springer Berlin Heidelberg, 2007.

[2] Kevin Bache and Moshe Lichman. UCI Machine Learning Repository, 2013. URL `http://archive.ics.uci.edu/ml`.

[3] Satanjeev Banerjee and Ted Pedersen. An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet. In *Computational linguistics and intelligent text processing*, pages 136–145. Springer Berlin Heidelberg, 2002.

[4] Alexander Budanitsky and Graeme Hirst. Evaluating WordNet-based Measures of Lexical Semantic Relatedness. *Computational Linguistics*, 32(1):13–47, 2006.

[5] Johannes Fürnkranz. Separate-and-Conquer Rule Learning. *Artificial Intelligence Review*, 13(1): 3–54, 1999.

[6] Johannes Fürnkranz and Peter A. Flach. ROC âĂŹnâĂŹ Rule Learning - Towards a Better Understanding of Covering Algorithms. *Machine Learning*, 58(1):39–77, 2004.

[7] Johannes Fürnkranz, Dragan Gamberger, and Nada Lavrač. *Foundations of Rule Learning*. Springer Berlin Heidelberg, 2012.

[8] Graeme Hirst and David St-Onge. Lexical Chains as Representations of Context for the Detection and Correction of Malapropisms. In Christiane Fellbaum, editor, *WordNet: An Electronic Lexical Database*, pages 305–332. MIT Press, 1995.

[9] Frederik Janssen and Johannes Fürnkranz. On the Quest for Optimal Rule Learning Heuristics. *Machine Learning*, 78(3):343–379, December 2009.

[10] Frederik Janssen and Markus Zopf. The SeCo-Framework for Rule Learning. In *Proceedings of the German Workshop on Lernen, Wissen, Adaptivität - LWA2012*, 2012.

[11] Jay J Jiang and David W Conrath. Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. *In the Proceedings of ROCLING X*, (Rocling X), 1997.

[12] Knowledge Engineering Group TU Darmstadt. The SeCo-Framework for Rule Learning, 2010. URL `http://www.ke.tu-darmstadt.de/resources/SeCo`.

[13] Dekang Lin. An Information-Theoretic Definition of Similarity. In *ICML*, pages 296–304, 1989.

[14] RS Michalski. On the Quasi-Minimal Solution of the General Covering Problem. In *Proceedings of the V International Symposium on Information Processing*, pages 125–128, 1969.

[15] George a. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. A Semantic Concordance. In *Proceedings of the workshop on Human Language Technology*, pages 303–308, Morristown, NJ, USA, 1993. Association for Computational Linguistics.

[16] Princeton University. About WordNet., 2010. URL `http://wordnet.princeton.edu/`.

[17] Philip Resnik. Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, volume 1, 1995.

[18] Michael Strube and Simone Paolo Ponzetto. WikiRelate! Computing semantic relatedness using Wikipedia. In *In Proceedings of the 21st National Conference on Artificial Intelligence*, number February, pages 1419–1424. AAAI Press, 2006.