

GSLT Machine Learning, Assignment 1: Machine Learning for Morphological Analysis

Atelach Alemu Argaw and Eva Forsbom

October 17, 2005

1 Introduction

The purpose of this assignment is to define a learning task as described by Mitchell (1997), choose three algorithms from the book and describe how they could be used/applied to the chosen learning problem.

In many NLP applications, there is a need to look up words in a lexicon where only citation forms of each word are represented. Hence, morphological analysis ('lemmatization') is a crucial step in such applications. Sometimes it is also important to be able to look up parts of a word, for example in the cases where the whole word is not represented in the lexicon (e.g. in MT).

We considered the morphological analysis learning problem as a classification task, i.e. given the class, it should be possible to derive the citation form for a token, since the class contains all the information needed.

The definition of the learning problem is as follows:

Task T Classify tokens (tokens being items that are separated by a space)

Performance measure P Percent of tokens correctly analyzed according to a gold standard

Training experience E Pairs of tokens, their correct analysis and type (Derivation, Inflection, Compounding, Root)

We will consider the following learning algorithms: Bayesian learning, combined inductive and analytical learning (KBANN), and genetic programming.

The training experience will be a set of tuples: tokens, their correct main segmentation and morphology type (Derivation, Inflection, Compounding, Root). An example is shown in Table 1.

A script will be run to extract feature information on prefix, suffix, character changes on the first part of the segmentation, character changes on the second part of the segmentation, and main compound root if any, see Table 2. These features will then be concatenated and each unique concatenation will represent a class.

The target function is a classification function: $c(X) \rightarrow \{\text{Class}_1, \dots, \text{Class}_N\}$.

Root	Segments	Type
ägande	äga-ande	DER
ägare	äga-are	DER
äganderätt	ägande-rätt	CPD
ägde	äga-de	INF
äga	äga	ROOT
okänd	o-känd	DER

Table 1: Training examples before feature extraction.

Token	Segments	Type	Prefix	Suffix	Changes1	Changes2	CmpRoot
ägande	äga-ande	DER		ande	"" → a		
ägare	äga-are	DER		are	"" → a		
äganderätt	ägande-rätt	CMP					rätt
ägde	äga-de	INF		de	"" → a		
äga	äga	ROOT					
okänd	o-känd	DER	o				

Table 2: Feature extraction from script.

The training data has been chosen to be representative of the most common morphological features. Since we only model 2 way splits, the classifier should be run iteratively to find further segmentations.

2 Learning Algorithms

2.1 Naive Bayes Learner

Bayesian learning is an inductive learning method that uses knowledge about the prior probabilities of alternative hypotheses and about the probability of observing various data given the hypothesis. Based on the assumed priors and the observed data, Bayesian methods assign a posterior probability to each candidate hypothesis. The maximum a posteriori (MAP) hypothesis is given by Formulas 1-3.

$$h_{MAP} = \arg \max_{h \in H} \Pr(\text{class}|\text{features}) \quad (1)$$

$$= \arg \max_{h \in H} \frac{\Pr(\text{features}|\text{class}) \cdot \Pr(\text{class})}{\Pr(\text{features})} \quad (2)$$

$$= \arg \max_{h \in H} (\Pr(\text{features}|\text{class}) \cdot \Pr(\text{class})) \quad (3)$$

The training examples and associated feature probabilities used in the Bayesian learner are represented as a prefix trie and a suffix trie. The idea with a trie

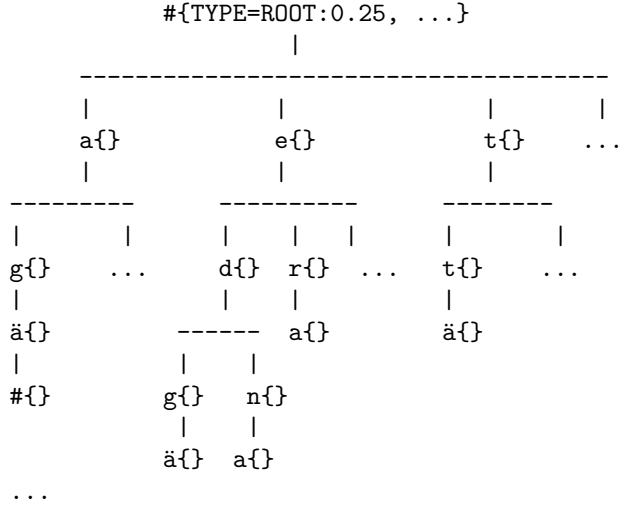


Figure 1: An example of the suffix trie.

representation is an extension of the work of Wicentowski (2002) on inflectional morphology to derivation and compounding.

We adhere to the simplifying assumption of the Naive Bayes classifier that all features are independent, which clearly does not hold in this classification task. Even though that may be the case, Naive Bayes classifiers have proven to give good performance for many problems where this assumption does not hold.

2.2 Combined Inductive and Analytical Learning - KBANN

The KBANN (Knowledge-Based Artificial Neural Network) algorithm uses prior knowledge to initialize the hypotheses to perfectly fit the domain theory, and then inductively refine these initial hypotheses as needed to fit the training data. It assumes a domain theory represented by a set of propositional, non-recursive Horn clauses.

Domain theory: Morphological rules from a grammar book of the language:
e.g.

DER $\leftarrow$ Suff:ande
ROOT $\leftarrow \neg$ Prefix, $\neg$ Suffix

Training examples: Classes, preconditions of the rules in the domain theory (features-value pairs), values for feature-value pairs (binary 0,1), i.e. the feature values that are described in Table 2 are represented as a vector which assigns the values 0 or 1 for each feature-value pair.

Initial Network: An input node for every feature-value, hidden nodes for every consequent of the rules in the domain theory except the class, which is the output node. Link every input node to all hidden nodes in the next layer, for as many layers as there are, and from the hidden nodes in the last hidden layer to the output node

Initialize the weights to some positive constant W for the non-negated preconditions to a consequent node, to $-W$ for the negated ones, and to a positive weight close to 0 for the rest.

Refinement: Adjust the weights by training the network with training examples and with backpropagation as the learning algorithm.

2.3 Genetic programming

Genetic programming is a variant of genetic algorithms, which conduct a randomized, parallel, hill-climbing search for hypotheses that optimize a predefined fitness function in the Darwinian sense. In genetic programming, hypotheses are represented as program trees. A gene pool (with terminal and non terminal functions) is the basis for generating an initial generation of hypotheses by random selection of genes. The programs are then evaluated according to a fitness function; their fitness decides the probability that they would be selected for reproduction, cross-over, or any other genetic operations.

Input: A string of characters (tokens)

Terminals: Functions without arguments in the gene pool

- go right
- read current character

Non terminals: Functions with arguments in the gene pool

- if a then b
- do sth while a
- swap (a, b)
- edit operations

Parameters:

- number of individuals in initial population
- number of generations
- reproduction rate
- mutation rate
- (time threshold)
- (stopping criteria)

Output: A class

Fitness function: Compare similarity between suggested class and the class in the gold standard

3 Discussion

We chose the three learning algorithms because we found them interesting and applicable to our learning problem, although the Naive Bayes approach would probably be the most realistic method.

The advantage of using Bayesian methods is that they can be used to determine the most probable hypothesis given the data - the maximum a posteriori (MAP) hypothesis. This is the optimal hypothesis in the sense that no other hypothesis is more likely. The limitation of optimal Bayesian methods is that they are often computationally intractable. With the simplifying assumption of the Naive Bayes classifier, most problems become tractable and the classifier is often quite effective. Although this assumption doesn't always hold, Naive Bayes classifiers have proven to give good performance for many problems.

The advantage of using KBANN over purely inductive methods is that it typically generalizes more accurately when given an approximately correct domain theory, especially when training data is scarce. It is also advantageous compared to purely analytical methods, since it is more robust to noisy training data. KBANN can learn features (new generalizations) that are not represented in the domain theory. A limitation of using KBANN is that it can accommodate only propositional domain theories. Since we are using feature-value pairs in our domain theory, and since our clauses are equivalent to propositional clauses, this restriction doesn't pose a problem. Another limitation is that KBANN could be misled if the domain theory is inaccurate.

The advantage of using genetic programming is that there is no need to have fixed sized input vectors. The input could be both symbolic and numeric values. They are specially suited to tasks in which hypotheses are complex. The inductive bias here is in the design and implementation of the gene pool and the fitness function. The disadvantage of using genetic programming is that of local optima (inbreeding).

References

- Tom M. Mitchell. *Machine Learning*. The McGraw-Hill Companies, Inc., New York, USA, 1997. ISBN 0-07-042807-7.
- Richard Wicentowski. *Modeling and Learning Multilingual Inflectional Morphology in a Minimally Supervised Framework*. PhD thesis, John Hopkins University, Baltimore, Maryland, USA, 2002.