TEXT CLASSIFICATION

NLTK TUTORIAL
Outline

Algorithms - back to decision trees

Feature Selection

Features

Training Classifiers

The Classifier Interface

Labeled Texts

Using Classifiers

Introduction

Review

Announcements (talk tomorrow, project questions?)
Decision Trees

- consistent, realizable, simple, ockam's razor,

- learn a function f from examples (x, f(x))

Inductive Learning

Learning Agents

Learning from Observations (Chapter 18, AIML)
What kind of learning (from Ch. 18) is this?

What about our class project?

- Choose the most likely category for a given text
- Each text belongs to exactly one category
- Predefined set of categories

Single-Category Text Classification

- Divides texts into topics
Classifying New Texts
Training the Classifier

Training Corpus

Combining the Feature Lists
Sentence Length

The first word
Which words are present

Features

Using Classifiers (Ch. 6)
What do we do first?

Why is this text classification?

Questions ●

Imperatives (commands) ●

Statements ●

Categorize sentences as

Example Task
How do we get ideas for this?

Which aspects of a sentence are relevant to our classification task?
Let’s brainstorm...

Examine a training corpus

How do we get ideas for this?

Which aspects of a sentence are relevant to our classification task?
Example Features (from Tutorial)

Which words are present in the sentence (bag of words)
The length of the sentence
The first word of the sentence
What next?
See code examples
Combine them
Define 3 FDL lists (Feature Detector Lists)
Training Corpus

What next?

What about our project?

See code examples

Load and label
What next?
See code examples

What if we wanted to switch classification algorithms?
Use the ClassifierTrainer to train a new classifier, using the training corpus.
Build a ClassifierTrainer, using the feature detector list.

Training the Classifier
Now, the details.

See code examples

Use the classifier we built to classify new texts.
Relationship to Other Tasks

Multi-category classification

Tagging

Clustering
See code examples

Obvious member functions, type/token distinction

Defined by nltk/classifier module

Labelled text class represents categorized text types
See code examples

labels returns the list of category labels

classify returns a labeled text token

Requires two methods
classifyTextI for single category text classifiers
See code examples

- Returns: new classifier
- Input: list of labeled tokens
- Single method: train

ClassifierTransformer for building classifiers from training corpora

Training Corpus (usually hand-classified, assumed mostly correct)
what are the values returned?

see code example

map labeled texts to feature values

Features are defined by feature detector functions

the label "sports"

example feature: whether a document contains the word "ball" and has

relevant to deciding how likely a label is for that text (classification)

each feature specifies some aspect of a labeled text that is (hopefully)

main/task independence

A way of encoding information used to build classifiers, that provides do-

Feature-Based Classification
### Feature Types

See code examples

- Others (not implemented in NLTK)
- Integer Features
- Boolean (Binary) Features
See code examples

detect can then be used

created from Python functions using the constructor

Function: FeatureDetector is one provided implementation

Returns: feature value

Input: labeled text

Single method: detect

Feature Detectors

Feature Detectors
Efficient implementation for related feature detectors

- unique within a feature detector list, not globally

\[ I - N \]

- feature ids for a feature detector list with \( N \) features are

- associate unique identifiers with each feature detector

- Grouping mechanism for feature detectors

Data structures that represent the feature detector functions for a set of features

Feature Detector Lists
See code examples

- an addition operator
- a delete method
- a length operator
- an indexing operator

Four methods

FeatureDetectorFactory for implementing Feature detector lists
See code example:

- the list contains one feature detector for each (val, l) pair.
- the list contains a function defined over texts, a list of function values, and a list of labels.
- implements feature detector lists consisting of boolean features with detector functions of a specific form (see code).

Implementations of FeatureDetectorList
See code examples

the list contains one feature detector for each (w,l) pair

inputs: a list of relevant words and a list of labels

see code example

whose detector functions have a specific form

again, if implements a feature detector list containing boolean features

BagOfWordsFPLIST checks which words are present in text

Implementations, continued
See reference documentation for other available implementations.

- one feature detector for each (w, l) pair
- a list of relevant words, and a list of labels

```
BagOfWordsPDLList
```

- one feature detector for each (val, l) pair
- function defined over texts, a list of function values, and a list of labels

```
TextFunctionPDLList
```

Comparison
detector list

• the feature ids in the feature value list correspond with those in the feature

• created with detect method of a feature detector list

feature detector functions

• contrast with Feature Detector Lists: data structures that represent the

Data structures that represent the feature values for a set of features

Feature Value Lists
See code examples

assignment returns (feature-id, feature-value) pairs for the rest
default typically returns the majority value

- a default method
- an assignment method
  a length operator (old)
  an indexing operator (old)

Four methods
Another alternative is to use statistical techniques to decide which features are relevant, before constructing the classifier. An alternative is to use all and let the classification algorithm decide, but this has drawbacks (inefficient, over-fitting).

It is often difficult for humans to decide which information is relevant for feature selection.
Instead of trees, we will return to decision trees.

Naive Bayes (as black boxes) for our purposes, we will view NTLK classifier implementations (e.g.)

Let's revisit how we started.

Using Classifiers