Machine Learning: Soft Computing
CSCI 447
Project 1
Design Document

Daniel Church, Dieter Grosswiler, Dylan Wickham, Will Merryman

September 11, 2017

1 Description of the Problem

The Waikato Environment for Knowledge Acquisition (WEKA) Java library doesn't accept the format of the data from the UCI data website. The information is given in .data format for the data and .names for the other information such as attributes, sources, etc. We need to convert this to an Attribute Relationship File Format (ARFF) format for compatibility with WEKA. The ARFF format consists of all the information consolidated in a single file with a header section, consisting of the relation and attributes, followed by a data section listing all the data.

2 Software Architecture

A model of our software is provided as Figure 1. This diagram illustrates the major classes implemented in our software, as well a selection of some of the methods in each class, and the connections between the classes. The Data class stores a string representing a row of a data from the .data file. The Attribute class stores a label describing the attribute label (i.e. Id Number) and attribute type (i.e. numeric). HeaderData is a list consisting of lines from the .name file that has the Title and Sources section. AttributeList and DataList are lists of the Attributes and Data found parsing of the .name and .data files, to be used to with HeaderData to create the final document.

3 Design Decisions

The following subsections will describe key decisions in the design regarding the parsing of files and use of the collected information to create the ARFF. The main idea is to parse the data in the .name and .data files and use that to create a ARFF file. The main difficulties were finding the attribute name and type. Most of the header and data info is fairly standard, but attribute information is not.

3.1 Parsing Files

The first thing that needs to be done for the converter is to ask the user for the directory where the files are stored. In the given directory it is assumed there will be a single .name and a single .data file, if not an error will be given. This subsection will detail the Parse .name file and Parse .data file Processes up to where each stores their data.

Inside the .name file there are 3 relevant pieces of information that may need a regular expression since there is some inconsistencies. A section that is called '1[.|)] Title', a section called '2[.|)] Source', and '7[.|)] Attribute Information'. Title and Source sections are easy to deal with since we are going to take whatever text they have and put them into the comments at the top. The Attribute

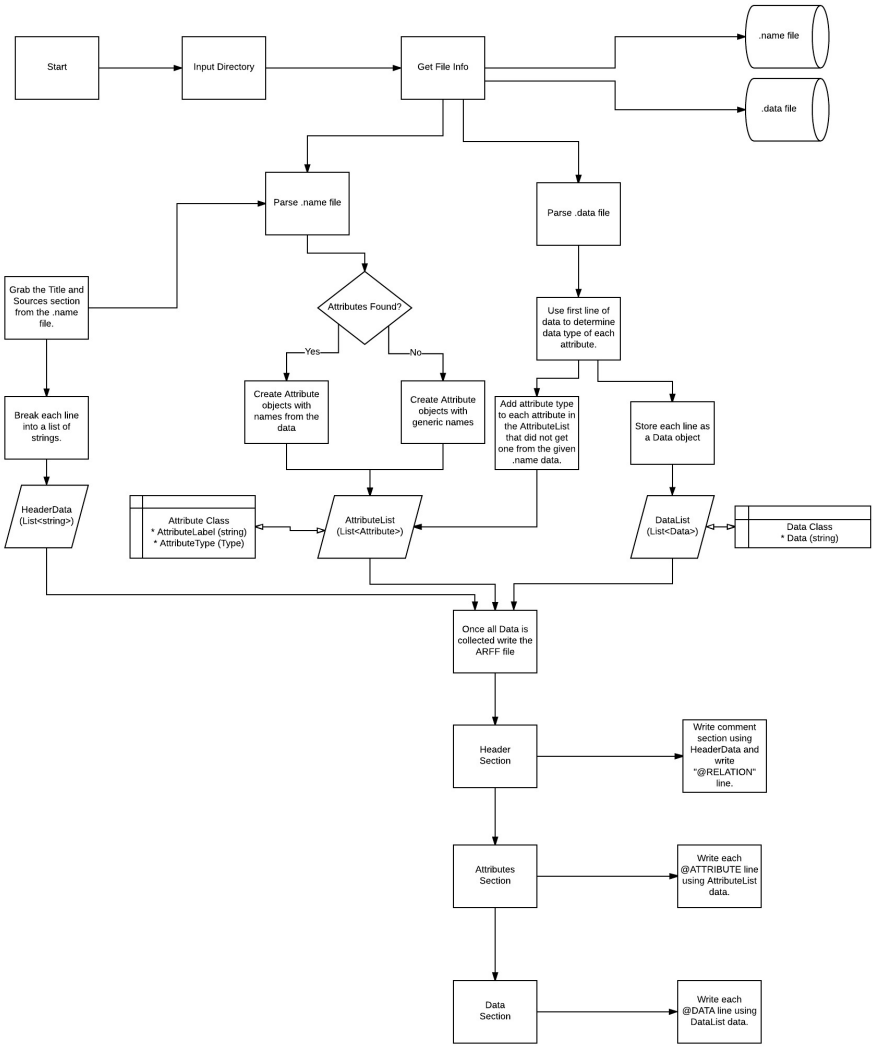

Figure 1: Figure 1

Information section is the problem.

The Attribute Information section is not always present and when it is, it does not always have relevant information. The Attributes will be assumed to be in the format [+1-9][.|)] +[:alpha:]: (i.e. 1. Refractive Index:). If there is no Attribute Information section or it does not have useful information we will create our own attribute names (Attribute 1, Attribute 2, etc.). We will use the .data file to count the number of attributes.

The Attribute Information section even when it does have the name of the attribute does not have consistent Type declarations. While a human can easily determine Type from given information, a computer will be far harder pressed and this is a very hard problem that will be outside the scope of the current problem. Instead we will get whatever data we can from the .name file but rely on the .data file to give us our Type most of the time. We will do this by looking at the first line of data and running regular expressions on each comma delimited item to guess the type. Numbers only will be integer, numbers with '.' will be real, and anything else will be strings. We will try to grab nominal-specification types (Animal: bird, cat, dog, etc) from the .name file when they are easily determined and mark as such, but when this is not possible it will just be marked a string.

We will also store each line of the .data file into a list. Once we have assembled each attribute we will store them into a list as well. Each line of the Title and Source data will also be stored. At this point we are ready to create an ARFF file.

3.2 ARFF Creation

This section is pretty straight forward and simple. The Title and Source will be written into comments at the top of the file along with the @RELATION and a string of the directory name the .name and .data file was pulled from. The attributes will be written one line at a time as '@ATTRIBUTE AttributeLabel AttributeType'. Each data item will be written after the @DATA line is written.

References

[1] http://archive.ics.uci.edu/ml/datasets/Ionosphere

[2] http://archive.ics.uci.edu/ml/datasets/Wine

[3] http://archive.ics.uci.edu/ml/datasets/Waveform+Database+Generator+%28Version+2%29

[4] http://archive.ics.uci.edu/ml/datasets/MAGIC+Gamma+Telescope

[5] http://archive.ics.uci.edu/ml/datasets/Glass+Identification