uClassify

Classification Server Manual

- Getting Started -

About

The uClassify Classification Server is driven by XML calls that allows users to create and train any arbitrary classifier. It can be used to to filter spam, categorize web pages or any other task that requires automatic text classification. It's design to handle huge amounts of data and can process millions of documents daily.

This manual will demonstrate how to install a uClassify classification server on a Windows machine. It will also briefly outline the architecture and explain how the XML API works.

Minimum Requirements

- Runs on Windows 2000, Xp, Vista, Server 2003, Server 2008 (32-bit or 64-bit)
- 1.0GHz processor
- 512 MB internal memory
- 100 MB Hard drive
- Microsoft XML 4.0 library (download from Microsoft)

Setup

This will guide you through installation, testing it and how to run the server as an application or service.

Installation

Installation is really simple, just extract the zip files to a installation directory of choice and make sure that the directory structure is preserved.

1. Download the .zip file
2. Extract the contents of the .zip file to a folder eg. *'c:\program files\uclassify server\'*

Open a command prompt and navigate to the installation directory
1. Open a command line prompt (Start->Run->cmd.exe)
2. Navigate to the installation directory (*c:>cd c:\program files\uclassify server*)

Under this directory you will find a file called uclassifyserver.exe, this is the core and you can do a few things with it by specifying flags. If you type *uclassifyserver* and press enter you will get some information of how it can be used.

Testing it

To make sure that your system has all needed components you can test the server. To do so, start it with the -test flag. You must also specify a configuration file with the -config flag. There is a default configuration file called config.xml in the installation folder which you can use for that purpose.

uclassifyserver -test -config config.xml

This starts the server with the configuration given in config.xml and runs some internal tests. This can take a couple of minutes, but when it's done you should see no errors in the console window.

A common first time error is that Microsoft XML 4.0 is not installed, download and install it from

Microsoft.

Once the tests are working, let's have a quick look at the configuration file.

Configuration File

Open the config.xml file in any text editor (e.g. WordPad), it is located in the same directory as uclassifyserver.exe. This is the configuration that the server reads at startup. Most entries are explained in the configuration file. Make any changes you find necessary and save it.

Running the server as a Windows application

Go back to the command promt and type in:

```
uclassifyserver -config config.xml
```

This will start the server in the command line prompt and you should see “Waiting for connections...” at the end of the console log. This means that the server is ready accept XML requests on the port specified in the configuration file.

You can stop it by pressing CTRL+C in the command line window.

Running the server as a Windows Service

Running the server as an application can be useful for testing purposes, but once you have it setup properly and ready to start using it you should run it as a service. Running it as a service as opposed to an application has the advantage that it will always run in the background even when no user is logged in.

To install the server as a service type the following:

```
uclassifyserver -install -config config.xml
```

Now it's installed as a service, this means that the next time the computer is restarted the uClassify server will start automatically. To start it right away:

```
uclassifyserver -start
```

Now you should have the server running in the background waiting for incoming XML API calls on the port specified in the configuration file.

If you wish to stop the service from running type the following:

```
uclassifyserver -stop
```

Finally, if you wish to uninstall the service:

```
uclassifyserver -uninstall
```

Once you have it setup and running you can start to create classifiers and classifying using the XML API.

Basic Architecture

This section will just give an idea of the different components users will come in contact with.

Communication

The classification service accepts incoming TCP socket connections on the port specified in the configuration file (default is 54441). Each request is validated against an XML schema to make sure it conforms to the API standard.

Classifiers

Classifiers are stored as 'classifiername.dat' files in the 'classifierRootPath' directory given in the configuration file. These files contains all information and data about the classifiers and should never be altered manually however it could be a good point to take a backup of them once in a while.

Logs

Server logs are saved in the file called 'logs/log.txt', this file should not be opened while the server is running. If inspection is needed you should open a copy of it. In the logs directory you may also find dumps. Those occur if something goes wrong and will help us to track down bugs.

API

The API is almost the same as the public web API used on uclassify.com. This makes it easy to port calls that previously has gone via the web API directly to the server.

There are actually only three differences, the XML namespaces, there is no authentication needed and communication is done on sockets instead of post requests.

XML namespaces

The XML request namespace is http://api.uclassify.com/1/server/RequestSchema
The XML response namespace is http://api.uclassify.com/1/server/ResponseSchema

Authentication

The classification server doesn't require any authentication like the web API (no need for read and write keys). So basically just remove all those attributes from the calls and you should be good to go!

Communication

Instead of posting XML requests to http://api.uclassify.com you will send the request directly on a socket to the listening port. It's important to remember to shutdown sockets after usage, see examples below.

Having this in mind, http://www.uclassify.com/XmlApiDocumentation.aspx will have all the information you need.

Example

Example of an XML request to create a classifier

```
<?xml version="1.0" encoding="utf-8" ?>
<uclassify xmlns="http://api.uclassify.com/1/server/RequestSchema" version="1.00">
  <writeCalls classifierName="MySpamClassifier">
    <create id="Create"/>
  </writeCalls>
</uclassify>
```

Note that there is no writeKey attribute and the namespace is different from the ones in the web API.

Socket code samples

These samples are meant to give you an idea of where to start but are far from perfect in terms of performance and robustness.

C# communication example

```
public string sendAndReceive(string xmlRequest)
{
   Byte[] sendBuffer = System.Text.Encoding.UTF8.GetBytes(xmlRequest);
   Socket socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.IP);
   socket.Connect(_host, _port);
   socket.Send(sendBuffer);
   socket.Shutdown(SocketShutdown.Send);

   Byte[] recvBuffer = new Byte[16384];
   Int32 size = socket.Receive(recvBuffer, recvBuffer.Length, 0);
   string response = System.Text.Encoding.UTF8.GetString(recvBuffer, 0, size);
   while (size > 0)
   {
      size = socket.Receive(recvBuffer, recvBuffer.Length, 0);
      response += System.Text.Encoding.UTF8.GetString(recvBuffer, 0, size);
   }
   socket.Shutdown(SocketShutdown.Receive);
   socket.Close();
   return response;
}
```

PHP communication example

```
function sendAndReceive($host, $port, $xmlRequest)
{
   $socket = socket_create(AF_INET, SOCK_STREAM, getprotobyname('tcp'));
   if (!$socket)
       return array(false, "Could not create socket. " + socket_strerror(socket_last_error($socket)));

   if (!socket_connect($socket, $host, $port))
   {
      socket_close($socket);
      return array(false, "Could not connect to the socket. " + socket_strerror(socket_last_error($socket)));;
   }

   $sent = socket_write($socket, $xmlRequest, strlen($xmlRequest));
   if (!$sent)
   {
       socket_close($socket);
       return array(false, "Could not write on the socket. "  + socket_strerror(socket_last_error($socket)));;
   }

   if (!socket_shutdown($socket, 1))
   {
      socket_close($socket);
      return array(false, "Could not shutdown the write socket end. "  + socket_strerror(socket_last_error($socket)));;
   }

   $response = "";
   $len = 1000000;
   socket_recv($socket, $response, $len, MSG_WAITALL);
   socket_close($socket);
   return array(true, $response);
}
```

Final Words

Thanks for using uClassify, if you have any questions, comments or feedback please feel free to contact us at support AT uclassify DOT com.