CASPR Research Report 2007-02

USING MONTYLINGUA 2.1 WITH C# AND MICROSOFT .NET

Michael A. Covington
and
Joe D. McFall

Artificial Intelligence Center
The University of Georgia
Athens, Georgia 30602-7415 U.S.A.
www.ai.uga.edu/caspr

2007

Using MontyLingua 2.1

With C# and Microsoft .NET Framework

Michael A. Covington and Joe D. McFall
Artificial Intelligence Center
The University of Georgia
Athens, GA 30602 U.S.A.

Revised 2007 April 17

Introduction

MontyLingua (Liu 2004) is a freeware natural language processing package written in Python and also supplied as a Java archive (.jar file). This document tells you how to compile MontyLingua into a .NET DLL file and call it from C# programs. This process relies on IKVM, a freeware Java-to-.NET conversion utility.

Acquiring the components

Java 1.4.2 (not 1.5 or 1.6)

You will need the Sun Java compiler, version 1.4.2, installed on your system. Without it, you will have trouble running the MontyLingua Java examples, and we are not sure whether you you can compile MontyLingua into a .NET DLL. Search www.sun.com for version 1.4.2 of the Java SE Development Kit and install it.

You can apparently have several versions of Java installed on your computer, but for purposes of compiling MontyLingua, JDK 1.4.2 must come first on the path (see the next step).

Incompatibility warning

MontyLingua was written in Python and was compiled into Java using a version of Jython (the Python-to-Java converter) that is apparently incompatible with some of the latest releases of Sun Java. The symptom is that when you try to run the demonstration .bat files in MontyLingua's *java* folder, you get the error message

java.lang.IllegalAccessException:
Class org.python.core.PyReflectedFunction can not access a member of class java.lang.AbstractStringBuilder with modifiers "public"

(along with a lot of debugging information). *This problem will not haunt you* after you have compiled MontyLingua into a .NET DLL, because Java will no longer be needed to run it.

Java on the path

Also put the Java 1.4.2 binary directory on your path so that this Java system will actually be used when you type a command beginning with "java" or "javaw". To do that, open Control Panel, System, Advanced, Environment Variables, Path, Edit, and add something like

```
"C:\program files\java\j2re1.4.2_14\bin";
```

(with quotes and semicolon as shown) at the very beginning of the PATH variable. Make sure you get the right path, depending on the exact version of Java that you are using.

This is also a good time to look for other corruption in the path that may have resulted from other software installations misbehaving.

MontyLingua

Download MontyLingua from:

http://web.media.mit.edu/~hugo/montylingua/

No installation is required; just unzip the 12-megabyte ZIP file into a convenient folder. Ultimately, in addition to the .jar file, you will need the .MDF data files.

It's a good idea to try the MontyLingua sample .bat files to make sure MontyLingua works.

IKVM

Download and install the IKVM binaries (ikvmbin) from:

http://sourceforge.net/project/showfiles.php?group_id=69637

or from www.ikvm.net.

Place the binary files (.exe and .dll) in a convenient folder such as C:\Program Files\IKVM, make sure all users have permission to read and execute them, and put this folder on the path the same way you did for Java.

Compiling MontyLingua into .NET

Open a command prompt in the folder where *montylingua.jar* resides and do this:

```
ikvmc -out:montylingua.dll -target:library montylingua.jar
```

This will create a new file, *montylingua.dll*.

Creating a Visual Studio project that uses MontyLingua

Create a new Visual Studio project (a couple of C# examples follow).

Once the project has been created:

- Copy the three files:

montylingua.dll	(which you just created)
IKVM.GNU.Classpath.dll	(from the IKVM folder)
IKVM.Runtime.dll	(also from the IKVM folder)

  into the folder that contains your source code. (This step is optional but recommended.)

- Add references to those three DLLs to your project. (Project, Add Reference, Browse.)

- Copy all the MontyLingua .MDF files into the folder that contains the executable code for your project (typically *bin/Debug*; when you deploy the project, you must keep track of where the .exe file goes and put these files with it). The DLLs get copied here automatically but you must keep up with the data files.

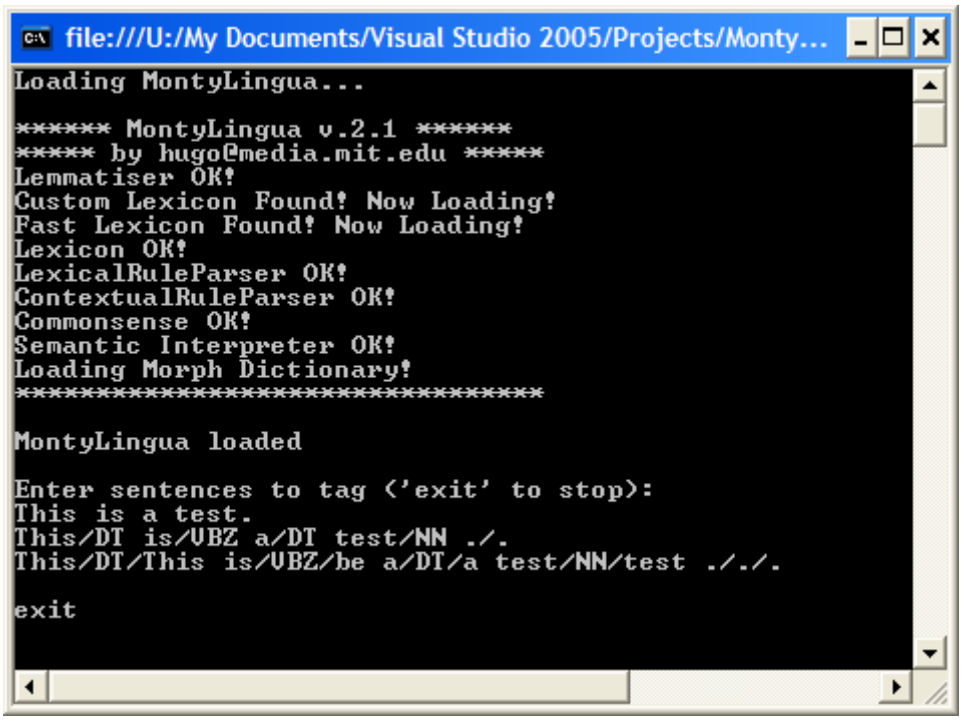
Now write your program, using MontyLingua as if you were working in Java. (There is extensive documentation about how to call MontyLingua from Java.) Here is a simple console application:

```
using System;
using System.Collections.Generic;
using System.Text;
using Monty = montylingua;

namespace MontyTest1
{
  class Program
  {
    public static void Main(string[] args)
    {
      Console.WriteLine("Loading MontyLingua...");
      Monty.JMontyLingua j;
      try
      {
        j = new Monty.JMontyLingua();
      }
      catch
      {
        Console.WriteLine("Couldn't start MontyLingua.");
        Console.WriteLine("MDF files may be missing.");
        return;
      }
      Console.WriteLine("MontyLingua loaded\n");

      Console.WriteLine("Enter sentences to tag ('exit' to stop):");
      string sentence;
      while ((sentence = Console.ReadLine()) != "exit")
      {
        try
        {
          Console.WriteLine(j.tag_text(sentence));
          Console.WriteLine(j.lemmatise_text(sentence));
        }
        catch (Exception e)
        {
          Console.WriteLine(e.Message);
        }
        Console.WriteLine();
      }
    }
  }
}
```

In action, the program looks like this:



Note that loading MontyLingua can take a couple of minutes even on a 2-GHz computer.

MontyLingua messages

Note the lines:

```
****** MontyLingua v.2.1 ******
***** by hugo@media.mit.edu *****
Lemmatiser OK!
Custom Lexicon Found! Now Loading!
Fast Lexicon Found! Now Loading!
Lexicon OK!
LexicalRuleParser OK!
ContextualRuleParser OK!
Commonsense OK!
Semantic Interpreter OK!
Loading Morph Dictionary!
**********
```

If your program is not a console application, they do not appear, but MontyLingua still works as intended. When you run your windowed application in debug mode in Visual Studio, these messages will appear in the Output window along with other debugging output.

Initializing MontyLingua in a separate task

Because it can take a couple of minutes to initialize MontyLingua, you may want to perform the initialization in a separate task so that your main program does not become totally unresponsive. Here's how.

```
using System.Threading;
using Monty = montylingua;

Monty.JMontyLingua j;

// Zero-argument method that will run in the thread
void InitMonty()
{
   try
   {
     j = new Monty.JMontyLingua();
   }
   catch (Exception e)
   {
      throw new Exception(
          "Can't start MontyLingua.  MDF file(s) may be missing.",
           e);
   }
}

// Code to do the initialization in a separate thread
void Initialize()
{
   Application.UseWaitCursor = true;
   Thread init = new Thread(new ThreadStart(InitMonty));
   init.Start();
   while (init.IsAlive)  // process other events
   {
     Application.DoEvents();
   }
   Application.UseWaitCursor = false;
}
```

Here the **`while`** loop keeps the program responding to events while the MontyLingua initialization is going on. Documentation suggests that **`init.Join()`** would do the job, but it doesn't actually update the main window sufficiently.

The MontyLingua license

Pay attention to the MontyLingua license document – it puts some limits on what you can do with your own software that uses MontyLingua. Specifically:

- You can only use MontyLingua for non-commercial purposes, not in a commercial product.
- You must acknowledge MontyLingua in your own program (we suggest putting this acknowledgment in the "About" box).
- MontyLingua is subject to GPL. That means you must tell users of your program where to get MontyLingua, and your own program *probably* needs to be open-source freeware. (If it isn't, read the MontyLingua license documents *very* carefully to make sure your program is sufficiently separate from MontyLingua.)

The IKVM license

IKVM is not subject to GPL, but you should still acknowledge it in the "About" box of your program, or another suitable place. It is freeware subject to the condition that you do not misrepresent its origin.

References

Liu, Hugo (2004) MontyLingua: An end-to-end natural language processor with common sense. http://web.media.mit.edu/~hugo/montylingua.