Introduction to using openNLP in .NET Projects

Introduction

Using the amazing IKVM http://www.ikvm.net/index.html OpenNLP's java files can be converted to a .Net assembly (dll).
Thus allowing you to use the latest releases of OpenNLP from C# (or any other .net language).
So far the .Net assembly has succesfully been used for: Splitting, Tokenising, POS Tagging & Chunking. Full parsing has yet to be fully tested.

Guide

(Don't forget to unblock any downloaded files)

- Download & extract the latest OpenNlp release from http://incubator.apache.org/opennlp/download.cgi . At the time of writing this is **apache-opennlp-1.5.1-incubating-bin.zip**
- The three .jar files (**opennlp-maxent-3.0.1-incubating.jar, jwnl-1.3.3.jar, opennlp-tools-1.5.1-incubating.jar**) in the lib folder can be used to compile a .net assembly as follows.
- Download & extract the latest IKVM from http://sourceforge.net/projects/ikvm/files/ At the time of writing this is **ikvmbin-0.46.0.1.zip**.
- For simplicity, copy the three .jar files above into the **ikvmbin-0.46.0.1/bin folder**
- From a command window, In the **ikvmbin-0.46.0.1/bin** folder use IKVMC & the above three jar files make the **opennlp.dll** as follows:
- ikvmc -target:library -assembly:opennlp opennlp-maxent-3.0.1-incubating.jar jwnl-1.3.3.jar opennlp-tools-1.5.1-incubating.jar
- Copy the following from the **ikvmbin-0.46.0.1/bin** folder to your project folder (or the folder of your choice)
    - opennlp.dll (the assembly you have just created)
    - IKVM.Runtime.dll
    - IKVM.OpenJDK.Core.dll
    - IKVM.OpenJDK.Jdbc.dll
    - IKVM.OpenJDK.Text.dll
    - IKVM.OpenJDK.Util.dll
    - IKVM.OpenJDK.XML.API.dll
      (I found using reflection which IKVM dll's are referenced in the opennlp.dll)

Add references to these assemblies in your project & use at will 😎
The OpenNlp manual is at http://incubator.apache.org/opennlp/documentation/manual/opennlp.html

You will need the models for your language which are currently here http://opennlp.sourceforge.net/models-1.5/

Note: This is still a java in .net clothes, so care has to be taken over some things.
e.g when loading models the inputstreams are java types (referenced from the assemblies above)

sample.cs

```
string modelpath = "C:\models\"; \\Wherever you've stored your downloaded models
java.io.FileInputStream modelInpStream = new java.io.FileInputStream(ModelPath + "en-sent.bin");
opennlp.tools.sentdetect.SentenceModel sentenceModel =new opennlp.tools.sentdetect.SentenceModel
(modelInpStream);
opennlp.tools.sentdetect.SentenceDetectorME SentenceDetectorME=new opennlp.tools.sentdetect.SentenceDetectorME
(sentenceModel);
```

EntityExtractor.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;


namespace NaturalLanguageProcessingCSharp
{

    public class EntityExtractor
    {
        /// <summary>
        /// Extractor for the entity types available in openNLP.
        /// Copyright 2013, Don Krapohl www.augmentedintel.com
        /// This source is free for unlimited distribution and use
        /// TODO:
        ///     try/catch/exception handling
        ///     filestream closure
```

```
        ///     model training if desired
        ///     Regex or dictionary entity extraction
        ///     clean up the setting of the Name Finder model path
        /// </summary>
        /// Call syntax:  myList = ExtractEntities(myInText, EntityType.Person);


        private string sentenceModelPath = "c:\\models\\en-sent.bin";   //path to the model for sentence
detection
        private string nameFinderModelPath;                              //NameFinder model path for English
names
        private string tokenModelPath = "c:\\models\\en-token.bin";     //model path for English tokens
        public enum EntityType
        {
            Date = 0,
            Location,
            Money,
            Organization,
            Person,
            Time
        }

        public List<string> ExtractEntities(string inputData, EntityType targetType)
        {
            /*required steps to detect names are:
             * downloaded sentence, token, and name models from http://opennlp.sourceforge.net/models-1.5/
             * 1. Parse the input into sentences
             * 2. Parse the sentences into tokens
             * 3. Find the entity in the tokens

            */

            //------------------Preparation -- Set Name Finder model path based upon entity
type-----------------
            switch (targetType)
            {
                case EntityType.Date:
                    nameFinderModelPath = "c:\\models\\en-ner-date.bin";
                    break;
                case EntityType.Location:
                    nameFinderModelPath = "c:\\models\\en-ner-location.bin";
                    break;
                case EntityType.Money:
                    nameFinderModelPath = "c:\\models\\en-ner-money.bin";
                    break;
                case EntityType.Organization:
                    nameFinderModelPath = "c:\\models\\en-ner-organization.bin";
                    break;
                case EntityType.Person:
                    nameFinderModelPath = "c:\\models\\en-ner-person.bin";
                    break;
                case EntityType.Time:
                    nameFinderModelPath = "c:\\models\\en-ner-time.bin";
                    break;
                default:
                    break;
            }

            //----------------- Preparation -- load models into objects-----------------
            //initialize the sentence detector
            opennlp.tools.sentdetect.SentenceDetectorME sentenceParser = prepareSentenceDetector();

            //initialize person names model
            opennlp.tools.namefind.NameFinderME nameFinder =  prepareNameFinder();

            //initialize the tokenizer--used to break our sentences into words (tokens)
            opennlp.tools.tokenize.TokenizerME tokenizer = prepareTokenizer();

            //------------------  Make sentences, then tokens, then get names--------------------------------

            String[] sentences = sentenceParser.sentDetect(inputData) ; //detect the sentences and load into
```

```
sentence array of strings
            List<string> results = new List<string>();

            foreach (string sentence in sentences)
            {
                //now tokenize the input.
                //"Don Krapohl enjoys warm sunny weather" would tokenize as
                //"Don", "Krapohl", "enjoys", "warm", "sunny", "weather"
                string[] tokens = tokenizer.tokenize(sentence);

                //do the find
                opennlp.tools.util.Span[] foundNames = nameFinder.find(tokens);

                //important:  clear adaptive data in the feature generators or the detection rate will decrease
over time.
                nameFinder.clearAdaptiveData();

                results.AddRange( opennlp.tools.util.Span.spansToStrings(foundNames, tokens).AsEnumerable());
            }

            return results;
        }

#region private methods
        private opennlp.tools.tokenize.TokenizerME prepareTokenizer()
        {
            java.io.FileInputStream tokenInputStream = new java.io.FileInputStream(tokenModelPath);     //load
the token model into a stream
            opennlp.tools.tokenize.TokenizerModel tokenModel = new opennlp.tools.tokenize.TokenizerModel
(tokenInputStream); //load the token model
            return new opennlp.tools.tokenize.TokenizerME(tokenModel);  //create the tokenizer
        }
        private opennlp.tools.sentdetect.SentenceDetectorME prepareSentenceDetector()
        {
            java.io.FileInputStream sentModelStream = new java.io.FileInputStream(sentenceModelPath);
//load the sentence model into a stream
            opennlp.tools.sentdetect.SentenceModel sentModel = new opennlp.tools.sentdetect.SentenceModel
(sentModelStream);// load the model
            return new opennlp.tools.sentdetect.SentenceDetectorME(sentModel); //create sentence detector
        }
        private opennlp.tools.namefind.NameFinderME prepareNameFinder()
        {
            java.io.FileInputStream modelInputStream = new java.io.FileInputStream(nameFinderModelPath); //load
the name model into a stream
            opennlp.tools.namefind.TokenNameFinderModel model = new opennlp.tools.namefind.TokenNameFinderModel
(modelInputStream); //load the model
            return new opennlp.tools.namefind.NameFinderME(model);                  //create the namefinder
        }
#endregion
    }
}
```

Notes

Workaround if an invalid format exception occurs when reading *en-pos-maxent.bin*
The file **en-pos-maxent.bin** is actually a zip archive.
If you examine the contents of this zip file, it currently has three files (the others seem to only have 2)
manifest.properties, **tags.tagdict**, & **pos.model**
Delete the ***tags.tagdict*** from the zipfile so that it only contains manifest.properties & pos.model
Note: Don't actually unzip **en-pos-maxent.bin** just delete **tags.dagdict**, so that **en-pos-maxent.bin** remains a Zip archive containing the remaining 2 files.