TIKA - LANGUAGE DETECTION

Need for Language Detection

For classification of documents based on the language they are written in a multilingual website, a language detection tool is needed. This tool should accept documents without language annotation *metadata* and add that information in the metadata of the document by detecting the language.

Algorithms for Profiling Corpus

What is Corpus?

To detect the language of a document, a language profile is constructed and compared with the profile of the known languages. The text set of these known languages is known as a **corpus**.

A corpus is a collection of texts of a written language that explains how the language is used in real situations.

The corpus is developed from books, transcripts, and other data resources like the Internet. The accuracy of the corpus depends upon the profiling algorithm we use to frame the corpus.

What are Profiling Algorithms?

The common way of detecting languages is by using dictionaries. The words used in a given piece of text will be matched with those that are in the dictionaries.

A list of common words used in a language will be the most simple and effective corpus for detecting a particular language, for example, articles **a**, **an**, **the** in English.

Using Word Sets as Corpus

Using word sets, a simple algorithm is framed to find the distance between two corpora, which will be equal to the sum of differences between the frequencies of matching words.

Such algorithms suffer from the following problems:

- Since the frequency of matching words is very less, the algorithm cannot efficiently work with small texts having few sentences. It needs a lot of text for accurate match.
- It cannot detect word boundaries for languages having compound sentences, and those having no word dividers like spaces or punctuation marks.

Due to these difficulties in using word sets as corpus, individual characters or character groups are considered.

Using Character Sets as Corpus

Since the characters that are commonly used in a language are finite in number, it is easy to apply an algorithm based on word frequencies rather than characters. This algorithm works even better in case of certain character sets used in one or very few languages.

This algorithm suffers from the following drawbacks:

- It is difficult to differentiate two languages having similar character frequencies.
- There is no specific tool or algorithm to specifically identify a language with the help of *ascorpus* the character set used by multiple languages.

N-gram Algorithm

The drawbacks stated above gave rise to a new approach of using character sequences of a given length for profiling corpus. Such sequence of characters are called as N-grams in general, where N represents the length of the character sequence.

- N-gram algorithm is an effective approach for language detection, especially in case of European languages like English.
- This algorithm works fine with short texts.
- Though there are advanced language profiling algorithms to detect multiple languages in a multilingual document having more attractive features, Tika uses the 3-grams algorithm, as it is suitable in most practical situations.

Language Detection in Tika

Among all the 184 standard languages standardized by ISO 639-1, Tika can detect 18 languages. Language detection in Tika is done using the **getLanguage** method of the **LanguageIdentifier** class. This method returns the code name of the language in String format. Given below is the list of the 18 language-code pairs detected by Tika:

da—Danish	de—German	et—Estonian	el—Greek
en—English	es—Spanish	fi—Finnish	fr—French
hu—Hungarian	is—Icelandic	it—Italian	nl—Dutch
no—Norwegian	pl—Polish	pt—Portuguese	ru—Russian
sv—Swedish	th—Thai		

While instantiating the **LanguageIdentifier** class, you should pass the String format of the content to be extracted, or a **LanguageProfile** class object.

```
LanguageIdentifier object=new LanguageIdentifier(“this is english”);
```

Given below is the example program for Language detection in Tika.

```
import java.io.IOException;

import org.apache.tika.exception.TikaException;
import org.apache.tika.language.LanguageIdentifier;

import org.xml.sax.SAXException;

public class LanguageDetection {

   public static void main(String args[])throws IOException, SAXException, TikaException
{

      LanguageIdentifier identifier = new LanguageIdentifier("this is english ");
      String language = identifier.getLanguage();
      System.out.println("Language of the given content is : " + language);
   }
}
```

Save the above code as **LanguageDetection.java** and run it from the command prompt using the following commands:

```
javac  LanguageDetection.java
java  LanguageDetection
```

It gives the following output:

```
Language of the given content is : en
```

Language Detection of a Document

To detect the language of a given document, you have to parse it using the parse method. The parse method parses the content and stores it in the handler object, which was passed to it as one of the arguments. Pass the String format of the handler object to the constructor of the **LanguageIdentifier** class as shown below:

```
parser.parse(inputstream, handler, metadata, context);
LanguageIdentifier object = new LanguageIdentifier(handler.toString());
```

Given below is the complete program that demonstrates how to detect the language of a given document:

```
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;

import org.apache.tika.exception.TikaException;
import org.apache.tika.metadata.Metadata;
import org.apache.tika.parser.AutoDetectParser;
import org.apache.tika.parser.ParseContext;
import org.apache.tika.parser.Parser;
import org.apache.tika.sax.BodyContentHandler;
import org.apache.tika.language.*;

import org.xml.sax.SAXException;

public class DocumentLanguageDetection {

   public static void main(final String[] args) throws IOException, SAXException,
TikaException {

      //Instantiating a file object
      File file = new File("Example.txt");

      //Parser method parameters
      Parser parser = new AutoDetectParser();
      BodyContentHandler handler = new BodyContentHandler();
      Metadata metadata = new Metadata();
      FileInputStream content = new FileInputStream(file);

      //Parsing the given document
      parser.parse(content, handler, metadata, new ParseContext());

      LanguageIdentifier object = new LanguageIdentifier(handler.toString());
      System.out.println("Language name :" + object.getLanguage());
   }
}
```

Save the above code as SetMetadata.java and run it from the command prompt:

```
javac  SetMetadata.java
java  SetMetadata
```

Note : Assume sample.txt is having the following content:

```
Hi students welcome to tutorialspoint
```

It gives the following output:

```
Language name :en
```

Along with the Tika jar, Tika provides a Graphical User Interface application *GUI* and a Command

Line Interface *CLI* application. You can execute a Tika application from the command prompt too like other Java applications.