

ADVANCED JAVA

ASSIGNMENT 1

LANGUAGE DETECTOR

Oli Dedo

Problem Description

This program is required to take from the user a folder as an input and recognize the language model to which a mystery text file is most similar to by analyzing the files inside the language folders. The program is going to do this by reading the folder contents, where each folder represents a language, tokenize the words into their N-Gram and at the same time counting their frequencies. Based on the frequency of the tokens we calculate the similarity of the document with each language and make an assumption about the language of the mystery text file.

Starting the program

The application is run from the command line and takes as arguments the path of a folder that is in the format specified below and an integer that specifies the n of N-Gram. If n is not specified by the user the program assumes n=2.

Folder Contents

The folder that the user specifies as an argument should have a specific structure. Under the local folder there should be a text file “mystery.txt” and some subfolders that represent a language. In each of these folders there should be multiple text files with files in that language. The goal of the program is to find the language with which the mystery text file is most similar to.

Algorithm

1. Read the language folders concurrently
2. Read each file of each folder concurrently while creating a language model
3. In each file:

* Filter out punctuation
* Standardize to lower case
* Split each word separated by space
* Create the N-Gram of each word
* If the token is already on the language model histogram increment its count, otherwise just put it in the model with frequency 1.

1. Calculate the similarity of the mystery text file with each language model
2. Sort the similarities
3. Find the language that is most similar to the mystery text file
4. Display the results

Non-functional Requirements

Streams and Lambdas

The first non-functional requirement is not to use loops in the program. This is made possible by using streams and lambdas that are features of Java 8.

Streams- A sequence of objects that can be pipelined and modified by multiple methods to produce a desired result.

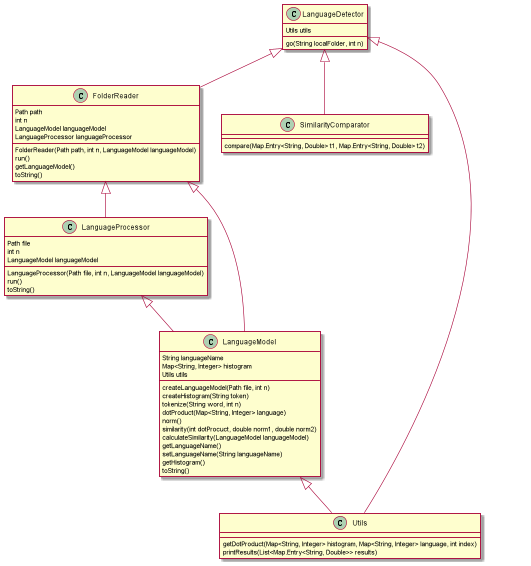
Lambdas- A function that is created without belonging to a class. It is the first step of functional programming, whose main purpose is to specify what the program does rather than how.

Concurrency

Secondly, the files need to be processed concurrently. This is implemented through multithreading capabilities of Java. First the folders are processed concurrently and then the files in each folder. For each folder a new instance of LanguageModel is initialized. This class keeps the name of the language and a histogram which is a Map of Strings and Integers that keeps the tokens of each language and their frequency. This Map is accessed by all files of a folder at the same time. When the files are read, each file is first separated into lines, punctuation is removed, it is standardized into lower case, it is split into words and each word is split into tokens. Each token is analyzed by the method createHistogram, which checks if the token is already present in the histogram. If it is, the counter is incremented. Otherwise it is simply put in the map with frequency = 1.

Each folder and each text file inside the folders is executed by a different thread. Both classes FolderReader and LanguageProcessor implement the interface Runnable, which makes it possible for each folder and file to be processed concurrently. They both override the run method. ExecutorService allows us to pass a task that is going to be executed by a thread. The method newCachedThreadPool creates a pool of threads that creates a new thread if one is needed or it reuses older free threads. Each thread is executed by the executorService, then it is shutdown in order to finish executing the previous tasks, but it does not accept new ones. Method awaitTermination blocks until all tasks have finished their execution after shutdown.

UML Diagram



Program Description

The program starts from the class LanguageDetector. This class contains a main method that gets the input from the user and starts an instance of the application by calling the method go, which takes as an argument the name of the local folder and an integer n. Firstly the method go gets the mystery text file and all the other subdirectories and executes them concurrently through ExecutorService. The class FolderReader takes as an argument a Path, and integer and an instance of the class LanguageModel, because for each folder we create a new language model and also for the mystery text file. FolderReader implements the Runnable interface and implements the run method. Inside the run method we concurrently process each text file inside the folder through the LanguageProcessor class, which is also a Runnable. From the method run of LanguageProcessor we set the name of the language, by getting it from the folder name and we call method createLanguageModel from the LanguageModel class.

LanguageModel is the class that does most of the work. This class holds language name and a histogram to keep the tokens and frequencies. Method createLanguageModel takes as an argument a file Path and an integer n. It reads the contents of the file and calls method tokenize. Method tokenize takes as an argument a String and an integer n. It creates the N-Gram of the String that is passed to it.

Creating N-Gram

Based on the n value that the user provides as argument the program splits the words into tokens.

When n=1, unigrams are created. Unigram is a sequence of individual letters.

When n=2, bigrams are created. The word is split into a sequence that groups the letters two by two.

When n=3, trigrams are created. The word is split into a sequence that groups the letters three by three.

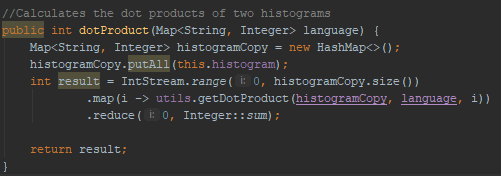
In case a word’s length is less than n the word is treated as a unigram if the length is one or a bigram if the length is two.

After getting the tokens of each word in the file, it calls method createHistogram that takes as an argument a String that represents a token. It constructs the histogram by counting each token that is passed to it as an argument. This class also provides methods that analyze each language model in order to find similarity and a getter for the histogram. The methods dotProduct, norm, similarity and calculateSimilarity are discussed below, since they are used in order to produce the results.

Getting Results

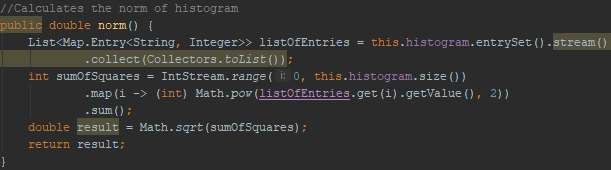
If we treat two language models as two vectors then their similarity is calculated as:

Where A \* B is the dot product of the two vectors. In the case of our program we are going to calculate the similarity of the mystery model with each language model created from the folders.



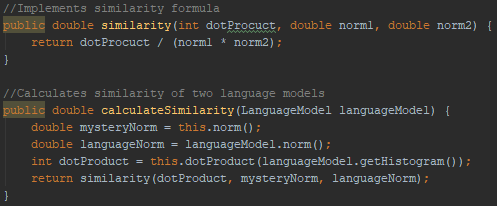
The method defined above finds the sum of products of frequencies of common tokens in the histograms of two languages. So, getDotProduct finds the common tokens and finds their product.

||A|| and ||B|| represent the norms of the vectors.



The method finds the square root of sum of squares of the frequencies of each token in the histogram of a model.

Finally we calculate the similarity as:



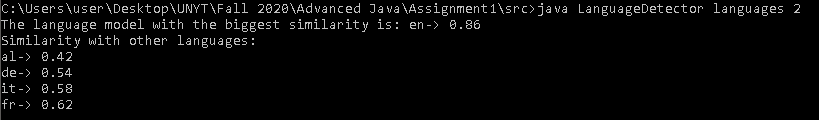
In order to calculate the similarities we calculate the dot product of mystery model with each language model and their respective norms.

To provide abstraction, method calculateSimilarity is the one that calls all the methods mentioned above and is then called in the main class for the calculation of the similarity of the mystery text file with each language model.

After calculating all the similarities, we sort them using a SimilarityComparator. The last element in the list is the language with the biggest similarity, which is also the language that the mystery text is most alike.

The similarity with the other languages is also printed in the results. To print the results and to get the dot product of the two vectors, class Utils is created, since it provides utility methods for the program that are called from LanguageDetector and LanguageModel classes.

Sample Output



To run the application from the console I have used a sample folder called languages that provides text files for English, Albanian, German, Italian and French. The mystery text file holds text in English and as seen from the output the English language model is the one with the biggest similarity. The program also outputs the similarity with the other language models.