Algorithm

Components

Training Text Map (TTMap) : TTMap is hash map of format <String, Integer>, in this the string represents key and integer is the value the key is mapped to. The Training Text is split along the spaces and each different word is a different key in the map, mapped to the occurrence of the word respectively.

Training Text Map (TTMap), are based on statistical model, there are no external frequencies as there are in N-gram rank ordering methods.

Functions and other data structures associated with TTMap :

Name	**Description**
String **LanguageName**	Represents the name of language to which the map belongs.
*Map<String, Integer>**TTMap***	Map containing word as key and their frequency as value.
List<integers> ***nGramCount***	An array containing all the different values of frequency.
Integer ***matchedNGrams***	The no. of n-Grams matched with current input.
String ***getNGram****(integer* freq*)*	Returns the first n-Gram having frequency = *freq.*
Integer **getFrequency**(*String* key)	Returns the frequency of the key.
String **take**(*integer* freq)	Returns and remove the first n-gram with Frequency = freq.
List<String> **getKeys**()	Returns the array with all the keys in map

Input Map: It is a map mapping *<Integer, String List>*, created from the given input text whose language is to be determined. It maps the Frequency of the words with the respective words (n-grams).

Name	Description
String **extractMaxFreq**()	Returns the n-gram with maximum frequency in the Input Map.

ALGORITHM

String **findLanguage**(*InputMap* **input**, *List<TTMap>* **trainingData**, *String* **inputStr**, *integer* **margin** = 3)

findLanguage is a function categorize the input according to its language and gives the language of the input.

Parameters:
input: it is the *InputMap* of the input whose language is needed to be identified.

trainingData: it is array of *TTMap* of all the languages whose training text is available to us.

inputStr: it is the input string whose language is needed to be identified.

margin: it is special input which deals with the possibilities of presence of nouns i.e. names which are same in all languages. It is required as each name may or may not be there in a training text of each and every language. It basically represents the maximum difference b/w the *matchedNGrams* is allowed for given input.
For example: “Taj Mahal” has same spellings and characters in most of the

languages hence if it is present in training text of 1 language and not in other, then it may lead to elimination of that language, which can lead to wrong results. Assume there is a sentence in German which contains “Taj Mahal” and is given as input to the algo.
But the training text of German Language does not have Taj Mahal in its training text, but Training text of English Language consist “Taj Mahal”. The above scenario produces fair possibility that the German is eliminated and English is kept.
Now if we have this margin variable then it will considerably reduce the chances of eliminating German as the algo can tolerate some difference in the no. of matched N-Grams in different languages.
The value of margin is calculated dynamically and changes as the algorithm exceeds.

<u>Steps:</u>

1. **If** only **1** language is present in **training data**
    a. **Return** the **name** of only **language** present
2. **Else if** there are **N-grams** still left **in input**
    a. **Extract** the **most frequent** n-gram from **input** and **eliminate** the **languages** in which the **maxMatchedNGrams < matchedNgrams + margin**
    b. **If** there are **no elimination: reduce** the size of **margin**
    c. Make a **recursive call to function** with **new values of parameters**
3. **If** there are **no N-grams left** in input and **possibility** for more than one **language remains** then **ask user** to choose the **language**

***TTMap* createTTMap(*String* traingText)**

This function is an overhead which is executed before the algorithm and only once. It creates TTMap for the given training text as input.

Steps:

1. **Remove** the **punctuation** marks from the **traingText.**
2. Take out **words** from the **trainingText**
3. **Create map** which contains the word as key and its occurrence as value.
4. Make a **list** of all the **keys** of map
5. Return the **TTMap**

inputMap createInputMap(*String* Input):

This function is overhead and is called only when new input is given to the algo. This function creates the input map for the

Pseudo-Code

```
One time over head
1. For all trainingText :
    a. createTTMap(helpingText)
    b. add it to trainingData
2. createInputMap(inputStr)

Language detection starts

1. startSize = trainingData.size();          //no. of  languages
2. if (startSize == 1)     //one Language is left
    a. return trainingData[0].languageName
3. else
    a. if (startSize != 0)     //if languages are present for elimination
        i. maxMatch = 0;
        ii. String mostFrequent = input.extractMaxfreq();
        iii. For(int  i = 0;  i < startSize;  i++)
            1. If( trainingData[i].getFreqency(mostFrequent) != -1 )
                a. trainingData[i].matchedNGram++;
            2. if(trainingData[i].matchedNGram > maxMatch)
                a. maxMatch = trainingData[i].matchedNGram;
        iv. remove TTMap from trainingData
            where(trainingData[i].matchedNGram + margin < maxMatched)
        v. if(startSize == trainingdata.size())   //if no eliminations
            1. if( margin > 0 )      //margin is there to reduce
                a. margin--;
                b. add mostFrequent to input
        vi. return findLanguage(input,  trainingdata,  inputStr,  margin )
            //recursive call with new values
    b. if( input is empty but still languages remains for  elimination)
        i. ask user to choose from remaining languages
    c. return "Unknown Language"
```