Readability algorithms compability on multiple languages

ROBIN TILLMAN AND LUDVIG HAGBERG

Stockholm 2014

Degree Project
School of Computer Science
Kungliga Tekniska Högskolan

Abstract

This paper aims to test the compatibility of readability algorithms when using text written in different languages as parameters, the languages used is Swedish and English. A readability algorithm aims to approximate the readability of a text. Readability can be defined in many ways but the definition used in this paper is simply in which ease a text can be read and understood. The tests conducted was done on the Swedish and English version of the same text, hence the readability is expected to be fairly alike. Three algorithms was tested, Coleman-Liau index (CLI), Läsbarhetsindex (LIX) and Automated Readability Index (ARI). The texts used was a collection of Wikipedia articles, "On the Origin of Species" by Charles Darwin and the Bible and their respective translations. The main focus was put into the Wikipedia articles because of the amount of text they consisted of and "On the Origin of Species" due to the similar set of variables in both languages and due to their similar sentence structure. The tests showed that both ARI and LIX works for both Swedish and English on texts which by the the definitions of the formulas are less readable. CLI however seem to perform less well on these higher level texts, but works excellent on the Bible which by all where defined as easy to read. This leads to the conclusion that ARI and LIX work on hard and average texts in both English and Swedish and that CLI work only on easy to rad texts in both languages.

Contents

1 Introduction

Readability is the ease in which text can be read and understood. Various factors to define readability have been used, such as:

- Speed of perception
- Perceptibility at a distance
- Perceptibility in peripheral vision
- Visibility
- The reflex blink technique
- Rate of work (e.g., speed of reading)
- Eye movements
- Fatigue in reading, etc ...

Thus there are many ways to look upon readability and various ways in measuring it. The Oxford dictionary defines the word "readable", from which the word "readability" derives, as:

> 1. Able to be read deciphered; legible: a code which is readable by a computer readable copies of very old newspapers
>
>   (a) Easy or enjoyable to read: a marvellously readable book

For our purposes readability will be defined as the ease in which a text can be read and understood. This definition is coming from "Legibility of Print" [6] and is chosen due to its simplicity and its focus on understanding.

How to determine readability varies and this introduces a problem which have to be taken into major consideration in this paper. We will study existing algorithms, which all use the their own definition of readability and their own methods on how to measure it. The most common factors in these existing algorithms are:

- The total amount of words
- The length of sentences
- The amount of words defined as complicated
- The amount syllables, etc ...

1.1 Statement of the problem

Despite using the right factors one must also determine how the chosen factors relate to each other to give the text its readability. The question examined in this report is how readability algorithms perform when processing texts written in different, not too distantly related, human languages. The languages Swedish and English was chosen for the research due to their resemblance.

To do this texts with the same readability in both Swedish and English have to be evaluated with the same algorithms. As algorithms use different variables to determine readability this might pose a problem with different languages.

The problem is as follows:

How do readability algorithms perform when processing texts written in Swedish and English?

2 Background

Different types of research has been done on most of the existing formulas used for determining readability. The major part of the existing research is aiming to evaluate the results given by the formulas and link them to a certain level of readability or to improve the formula it self. Readability formulas are also researched each on its where the studies aim to prove that their correctness is lacking. This is for example seen in different publications made by the creators of the formulas[9, 1, 7]. Readability formulas are used to give a definition of the readability of a text and the result can then be used to draw various conclusions, some examples are "Using the probability of readability to order Swedish texts", "Generating and Rendering Readability Scores for Project Gutenberg Texts" and "Wikipedia's Writing — Tests Show It's Too Sophisticated for Its Audience"[5, 11, 3].

Research on readability of different languages have also been conducted but not focused on the correctness of the formulas in different languages. Läsbarhetsindex (LIX), which is developed and adapted to the Swedish language, have had some limited research in how it preforms on different languages[2]. The readability formulas developed for English have had little research done on their applicability on the Swedish language probably due to the limited use of Swedish in the world.

3 Readability evaluation algorithms

Readability algorithms aims to approximate the readability of a text using different methods and arguments. In this section the readability algorithms used in this thesis shall be introduced. These algorithms were chosen due to the width of their use and their fit for the purposes of the thesis. The parameters most important for the purpose of the thesis was simplicity, usage and language. Simplicity since it is essential when the purpose is to use the algorithm on a various of languages. The simplicity main focus we have are the ease in which to acquire the variables needed to compute the formula and the language independence.

3.1 Läsbarhetsindex (LIX)

LIX was developed by Carl-Hugo Björnsson[3]. It was developed for the Swedish language but existing research has indicated that the formula performs well also when using most of the Western European languages[4]. The score is calculated by a formula using the number of words, periods and long words. A long word is

for this instance defined as a word with more than six characters. The formula is as follows[3, 1]:

$$LIX = W/S + (L * 100)/W \tag{1}$$

W:= The total amount of words
S := The total amount of sentences
L := The total amount of long words

Hence the formula can be defined as the average amount of words per sentence added with the percentage of long words by the total amount of words.

The result of the LIX formula can be translated by the following table:

<25	Children's books, etc
25-30	Simple texts
30-40	Normal texts / Fiction
40-50	Factual texts
50-60	Technical texts
>60	Difficult technical texts / Research / Dissertations

Table 1: LIX result table

3.2 Coleman-Liau (CLI)

The Coleman-Liau formula was developed by Meri Coleman and T. L. Liau and was published in 1975. The Coleman-Liau formula differs from some of the earlier formulas in such way as it does not rely on syllables[9]. Using syllables is said to be more accurate, however ruling syllables out improved simplicity which was crucial since CLI was intended for computer use where simplicity always is an important factor[9].

The Coleman-Liau index calculates readability as follows:

$$CLI = 0.0588L - 0.296S - 15.8 \tag{2}$$

L := Average amount of letters, numbers and punctuation marks per 100 words
S := Average amount of sentences per 100 words

The original CLI formula can be rewritten as such:

$$CLI = 5.88(L/W) - 29.6(S/W) - 15.8 \tag{3}$$

L := Total amount of letters, numbers and punctuation marks
W := Total amount of words
S := Total amount of sentences

A result from the Coleman-Liau formula corresponds to a United States grade level[9].

3.3 Automated Readability Index (ARI)

Automated readability index was developed to monitor electrical typewriters in real time. Like the Coleman-Liau formula, ARI does not use syllables to

compute readability. The result given by ARI corresponds to the same United States grade level needed to read and understand the text[10]. ARI was developed for the United States Air Force as a tool to determine the ease which manuals and text books could be read[10].

ARI calculates readability as follows:

$$ARI = 4.71(L/W) + 0.5(W/S) - 21.43 \tag{4}$$

L := Total amount of letters, numbers and punctuation marks
W := Total amount of words
S := Total amount of sentences

4 Method

4.1 Readability formulas

Some readability formulas suit the thesis more than others. First of all formulas using syllables was excluded due to a problem when parsing texts in different languages. This problem is by the fact that syllables are defined in such a way that there is no easy way to read them with a computer and the definition differs slightly in different languages[9].

Second formulas relevant to both English and Swedish are desirable. Since readability are most often adapted to English, due to the size of the language and the nationality of the scientists developing the formulas, it is not a problem to find formulas relevant for English. However Swedish being a much smaller language, LIX is the only readability formula with a definite reliability to the Swedish language. The readability formulas implemented are:

- Läsbarhetsindex (LIX)
- Coleman-Liau (CLI)
- Automated Readability Index (ARI)

LIX, CLI and ARI all only uses variables that can be calculated by looking on individual characters, words and sentences without having to do any special interpretations when handling multiple languages. Being independent from language restrictions thereby makes these formulas suitable. They are also wildly spread and used which makes them more relevant and interesting to test.

4.2 Texts

The criterias when choosing texts for this thesis is first that we want large enough texts to give a fair result. Second texts translated into both Swedish and English are needed. Third texts in different readability levels are desirable since readability formulas perform differently upon different levels.

The texts chosen to be examined are Wikipedia articles, the Bible and "On the Origin of Species" by Charles Darwin[14, 15, 12, 13]. Wikipedia has the advantage of being easy to access and having a large amount of texts written in both Swedish and English on the same subject. To generate a somehow accurate result with readability formulas larger texts are necessary. Since about

60% of Wikipedia articles does not consist of more than a few sentences, most of them are not suitable for this research[11]. Therefore articles exclusively about countries was used due to the guarantee of getting articles with a large enough content. However the English Wikipedia articles almost always contain more content than the Swedish translation, which would result in a not as well based result. Wikipedia articles are factual texts and should be somewhere in the middle of LIX result table (table 1).

The Bible and "On the Origin of Species" was chosen because of the difference of genre and thereby writing. The Bible is easy to read in the sense that it consists of mainly short sentences and words. "On the Origin of Species" is scientifically written and thereby hard to read. Also both sources have a large content and are easy to find in both languages.

The problem with all sources is that they are not written by the same author for both of the translations, which causes the otherwise assumable likewise readability indexes to differ.

4.3 Parsing

To parse the texts Python3 was used. This programing language was chosen because of it's simplicity and since there are no performance requirements.

A Python library called "Wikipedia" collect the Wikipedia articles and a library called "Translate" which inherit from "Google Translate" is used not to translate the texts, but to translate the country names to Swedish.

To calculate the amount of characters, words and sentences in a text regular expressions (regex) is used. Regex is also used to clean the text from unnecessary characters and spaces to simplify the general parsing of the text. The Python regex library "re" contributes with the regex functionality needed. Having counted the arguments needed for the readability formulas, each formula was calculated with the arguments given by each individual text giving the result in a JSON file for easy management.

5 Result

The results were all calculated by the Python readability program using the appropriate libraries, as described. The main program is defined in the python file "readability.py" which uses an other python file named "nations.py" for the Wikipedia country article parsing. The program as whole can be found in Appendix A, and the data calculated by it can be found in Appendix B in addition to the sections following.

5.1 Average scores of Wikipedia articles

To get an understanding of the large amount of readability scores resulting from the Wikipedia articles, an average score is calculated as an interpretation. To calculate the average scores a function to the Python program was added named "calc_average_wiki". This function first calculates the readability of each individual article to afterwards calculate the average score.

The following data was collected from all of the Wikipedia articles as whole:

Data	EN Value	SV Value
Characters	5510252	2185390
Words	1054382	370377
Long words	337148	135419
Sentences	48479	28498

Table 2: Wikipedia parameter count

Using functions 1, 3, 4 and the average scores, calculated as mentioned above, the following scores was given:

Formula	English	Swedish
CLI	13.6	16.5
ARI	14.0	13.0
LIX	53.7	49.8

Table 3: Wikipedia readability scores

5.2 Scores of "On the Origin of Species"

The method calculating the scores of "On the Origin of Species" is named "calc_darwin". By running the method the following data was collected:

Data	EN Value	SV Value
Characters	723035	861388
Words	150763	164724
Long words	37362	45374
Sentences	4328	5074

Table 4: "On the Origin of Species" parameter count

Using functions 1, 3, 4 and the data above, the following scores was given:

Formula	English	Swedish
CLI	11.5	14.0
ARI	18.6	19.4
LIX	59.6	60.0

Table 5: "On the Origin of Species" readability scores

5.3 Scores of the Bible

The method calculating the scores of the Bible is named "calc_bible". By running the method the following data was collected:

Data	EN Value	SV Value
Characters	3224231	3370368
Words	790051	765993
Long words	95726	113058
Sentences	29813	55123

Table 6: Bible parameter count

Using functions 1, 3, 4 and the data above, the following scores was given:

Formula	English	Swedish
CLI	7.1	7.9
ARI	11.0	6.2
LIX	38.6	28.7

Table 7: Bible readability scores

6 Discussion

Evaluating readability will always come with problems as it is very hard to find a scale which cover all the existing aspects of it. The scales used by the most known formulas are given by matching the score to a large amount of data. Doing this will give a good value for an average human being but since no person is average evaluating text after using humans will present difficulties because readability is subjective and usually only works on large amounts of data. Having problems collecting suitable data to evaluate will pose a problem to get a trustworthy result.

A problem exists with the data that is used from Wikipedia. Wikipedia articles are written by random users whom follow only a few rules when writing and the free text in the articles can vary wildly. This might have influenced the result if the articles was written by different authors using different writing techniques and therefore giving a not trustworthy result of readability. Also the length of the articles plays a part as the ones written in English tend to be more thorough and having a larger content than the respective article written in any other language. This will result in a wider base of data for the English result sets, for better or worse. Actions has been taken in order to prevent this by using articles about nations to minimize the difference of content size, this is because the articles about nations usually have a sufficient amount of words. Nation articles is also easy to find and almost always have an article in both Swedish and English, although the English articles are usually longer. However this will not remove the fact that the articles are not the same and can not be looked upon as such. To work the problem one would have to evaluate all the texts used by hand, which would be very time and resource consuming and neither the time or the resources existed for it to be an option. Even if it would be done there would still be no guarantee that the texts would be the same. This is of course something that regards all used sources, but it would not be unjustified to assume that the Wikipedia articles are more exposed to this problem due to the fact that anyone could have written the articles from Wikipedia, unlike the Bible and "On the Origin of Species".

The data collected from the Wikipedia articles shows that the readability corresponds to the middle of the scales used in each separate algorithm, as expected due to the fact that Wikipedia articles are factual texts. The LIX algorithm showed a difference of less than one percent comparing the result of the English texts to the Swedish ones. The score also represented the level expected from Wikipedia articles. Looking at "On the Origin of Species" shows that the LIX score also corresponds to the expected level of readability and that there is a small difference between the results of the languages. This indicates

that LIX works on texts from the middle and higher readability scales for both English and Swedish. The unexpected result turned when applying LIX to the Bible. Although both scores would indicate that the Bible is fairly easy to read, they were in fact put in different section of the LIX table. The Swedish score would indicate that the Bible is a simple text while the English score would indicate that the bible has the readability of a normal text or fiction. The amount of articles used from Wikipedia would make it likely that the different languages used the same style. Looking at the variables of "On the Origin of Species" show that both the English and Swedish version had almost the same percentual spread, this indicates the translation was written in a similar style to the original. the Bibles variables differed looking at percentage comparing Swedish and English which would indicate the translation was written in a differing style from the original and the assumption would be that the text would also have a differing readability.

The ARI formula differed almost eight percent comparing English to Swedish which would correspond to a year in the American school but is still a neglectable difference on the Wikipedia comparison and gives a similar difference for "On the Origin of Species". The minor difference in score would indicate that ARI and LIX are in fact applicable to both English and Swedish. ARI similar to LIX only gives unexpected results when comparing the Bible in the different languages, which would indicate that the Swedish translation of the Bible varies in format compared to the English one.

The CLI formula differed slightly over seventeen percent between the languages on the Wikipedia articles and this is a relative large difference. CLI is also the only one of the three algorithms that gives a higher result for Swedish on all three sources and the only one not to give a differing result on the readability of the Bible.

A recurring problem was the differing of the Bible readability results. However the CLI gave a differing result than the other two formulas, showing a similar result for both the Swedish and the English versions. This could be an effect of CLI being the only one of the three readability formulas taking in account the amount of sentences per hundred words and the fact that the Bible is using a relatively different sentence structure in comparison to the other sources. The sentences of the Bible tend to be significantly shorter than the other sources. The result of the CLI might indicate that the sentence structure used in the Bible is a better fit for the CLI than for the ARI or for the LIX while the other results would indicate the oposit on the other texts.

6.1 Conclusion

Both LIX and ARI worked well for the texts of medium and high difficulty leading to the conclusion that they both work on texts of medium and high difficulty texts in both languages. However the ARI score on the easy text where not satisfying due firstly to the difference of the Swedish and the English score. Secondly the Swedish score was closer to a real value than the English one, even though the formula is developed for English. Thereby the credibility of this result is small and a estimation of the formulas performance on easy texts in Swedish contra English can not be determined.

CLI in difference to the other readability formulas only worked proper calculating the readability of the Bible, leading to the conclusion that it only works

for both English and Swedish on easy to read texts. The scores of the Wikipedia articles and of "On the Origin of Species" was either to far from an estimated readability value of the text or the difference in between the languages was to large, giving no basis in claiming CLI to work for both Swedish and English on average to hard texts but more so the opposite.

References

[1] Carl-Hugo Björnsson, *Läsbarhet*. Liber, Stockholm, 1968.

[2] Carl-Hugo Björnsson, *Readability of Newspapers in 11 Languages*. Wiley, 1983.

[3] Ronald P. Reck and Ruth A. Reck, *Generating and Rendering Readability Scores for Project Gutenberg Texts*. 2007.

[4] Jonathan C. Brown and Maxine Eskenazi, *Student, Text and Curriculum Modeling For Reader-Specific Document Retrieval*. Carnegie Mellon University, Pittsburgh, PA. 2005.

[5] Johan Falkenjack and Katarina Heimann Muhlenbock, *Using the probability of readability to order Swedish texts*. 2012.

[6] Tinker and Miles A, *Legibility of Print*. Iowa State University Press, Iowa, 1963.

[7] Kincaid, J P ; Fishburne, Jr , Robert P ; Rogers, Richard L ; Chissom, Brad S, *Derivation of New Readability Formulas (Automated Readability Index, Fog Count and Flesch Reading Ease Formula) for Navy Enlisted Personnel*. Naval Technical Training Command Millington TN Research Branch, 1975.

[8] McClure G, *Readability formulas: Useful or useless. (an interview with J. Peter Kincaid.)*. IEEE Transactions on Professional Communications, 1987.

[9] Coleman, M.; and Liau, T. L., *A computer readability formula designed for machine scoring*. Journal of Applied Psychology, vol. 60, 1975.

[10] E. A. Smith and R. J. Senter, *Automated Readability Index*. Aerospace Medical Research Laboratories Aerospace Medical Division Air Force Systems Command Wright-Patterson Air Force Base, Ohio, 1967.

[11] Kent Anderson, *Wikipedia's Writing — Tests Show It's Too Sophisticated for Its Audience*. The Scholarly Kitchen, 2012.

[12] *The King James Version of the Bible*. From Project Gutenberg, 1989.

[13] *Bibeln eller Den Heliga Skrift i överensstämmelse med den av konungen år 1917 gillade och stadfästa översättningen*. From Project Gutenberg, 1999.

[14] Charles Darwin, *On the Origin of Species*. From Project Gutenberg, 1859.

[15] A. M. Shelling, *On the Origin of Species, translation of the fifth edition*. From Project Gutenberg, 1871.

[16] G. Harry Mc Laughlin, *SMOG Grading —– a New Readability Formula*. 1969.

[17] P. Ley and T. Florio, *The use of readability formulas in health care*. Psychology, Health and Medicine, vol. 1, no. 1, 1996.

A Code

A.1 readability.py

```
import sys
import re
import io
import wikipedia
import translate
import json
import nationsArray


#   Coleman-Liau Index
def CLI(chars, words, sents):
    words = float(words)
    res = (5.88*(chars/words))-(29.6*(sents/words))-15.8
    return res

#   Automated Readability Index
def ARI(chars, words, sents):
    words = float(words)
    res = (4.71*(chars/words))+(0.5*(words/sents))-21.43
    return res


#   Lasbarhetsindex
def LIX(words, sents, longs):
    words = float(words)
    res = (words/sents) + (longs*100/words)
    return res


#   Clean text for easy parsing;
#   remove/replace unwanted characters etc.
def clean_text(text):
    # Remove wiki titles
    text = re.sub(r"==+ \w+[ \w+]* ==+", " ", text) # Merge paragraphs to one text
    text = re.sub(r"==+ \w+[\W \w+]* ==+", " ", text) # Merge paragraphs to one text

    # Curly quotes etc
    text = re.sub("\xe2\x80\x98", "'", text)
    text = re.sub("\xe2\x80\x99", "'", text)
    text = re.sub("\xe2\x80\x9c", '"', text)
    text = re.sub("\xe2\x80\x9d", '"', text)
    text = re.sub("\xe2\x80\x93", "-", text)
    text = re.sub("\xe2\x80\x94", "--", text)
    text = re.sub("\xe2\x80\xa6", "...", text)
    text = re.sub(chr(145), "'", text)
    text = re.sub(chr(146), "'", text)
```

```
    text = re.sub(chr(147), '"', text)
    text = re.sub(chr(148), '"', text)
    text = re.sub(chr(150), "-", text)
    text = re.sub(chr(151), "--", text)
    text = re.sub(chr(133), "...", text)
    text = re.sub("'", "", text)

    # Replace commas, hyphens, quotes etc (count as spaces)
    text = re.sub('[",:;()/\-]', " ", text);

    # Remove newlines (count as spaces)
    text = re.sub("\n", " ", text)

    # Unify terminators
    text = re.sub("[\.!?]", ".", text)

    # Check for duplicate terminators
    text = re.sub("\.\.+", ".", text)

    # Remove numeric values
    text = re.sub("[0-9]+.?[0-9]*", "", text)

    # Remove overflow spaces
    text = re.sub("[ ]+", " ", text)
    text = re.sub(" +\.", ".", text)

    # Remove unwanted non-ascii characters
    text = re.sub(" \W+ ", " ", text)


    # Add "." to end if not existing
    if text[len(text)-1] != ".":
        text += "."

    return text


#   After cleaning text the number of words are equal to
#   the amount of spaces + 1.
def word_count(text):
    res = text.count(" ") + 1
    return res


#   After cleaning text the number of sentences are equal to
#   the amount of dots. The result will be tripped by
#   occurences of shortened words such as "U.S" or "Mr. ".
#   However this will not have a very big impact on the result.
def sentence_count(text):
    res = text.count(".")
```

```
    return res


#   Remove all spaces and other non-characters
#   and the length of the remaining string will
#   be equal to the amount of characters
def character_count(text):
    res = len(re.sub("[\. \W]+", "", text))
    return res


#   LIX use the amount of long words in its formula
#   (a long word is defined as a word with more than
#   6 characters)
def long_word_count(text):
    text = re.sub("\.", "", text)
    word_list = text.split(" ")
    res = 0;
    for word in word_list:
        if len(word) > 6:
            res += 1

    return res

# Calculate scores of "On the Origin of Species"
def calc_darwin():
    en_version = open('texts/darwinEN.txt', 'r')
    sv_version = open('texts/darwinSV.txt', 'r')

    # Read files, return string
    en = en_version.read()
    sv = sv_version.read()

    # Clean strings
    en = clean_text(en)
    sv = clean_text(sv)

    # Calc parameters
    en_chars = character_count(en)
    en_words = word_count(en)
    en_sents = sentence_count(en)
    en_longs = long_word_count(en)

    sv_chars = character_count(sv)
    sv_words = word_count(sv)
    sv_sents = sentence_count(sv)
    sv_longs = long_word_count(sv)

    print("EN CHARS: ", en_chars, " EN WORDS: ", en_words, " EN SENTS: ", en_sents)
    print("SV CHARS: ", sv_chars, " SV WORDS: ", sv_words, " SV SENTS: ", sv_sents)
```

```
    # Calc indexes
    EN_CLI = CLI(en_chars, en_words, en_sents)
    EN_ARI = ARI(en_chars, en_words, en_sents)
    EN_LIX = LIX(en_words, en_sents, en_longs)

    SV_CLI = CLI(sv_chars, sv_words, sv_sents)
    SV_ARI = ARI(sv_chars, sv_words, sv_sents)
    SV_LIX = LIX(sv_words, sv_sents, sv_longs)

    print()
    print("EN CLI: ", EN_CLI)
    print("EN ARI: ", EN_ARI)
    print("EN LIX: ", EN_LIX)
    print("SV CLI: ", SV_CLI)
    print("SV ARI: ", SV_ARI)
    print("SV LIX: ", SV_LIX)
    print()

# Calculate scores of the Bible
def calc_bible():
    en_version = open('texts/bibleEN.txt', 'r')
    sv_version = open('texts/bibleSV.txt', 'r')

    # Read files, return string
    en = en_version.read()
    sv = sv_version.read()

    # Clean strings
    en = clean_text(en)
    sv = clean_text(sv)

    # Calc parameters
    en_chars = character_count(en)
    en_words = word_count(en)
    en_sents = sentence_count(en)
    en_longs = long_word_count(en)

    sv_chars = character_count(sv)
    sv_words = word_count(sv)
    sv_sents = sentence_count(sv)
    sv_longs = long_word_count(sv)

    print("EN CHARS: ", en_chars, " EN WORDS: ", en_words, " EN SENTS: ", en_sents, " EN L
    print("SV CHARS: ", sv_chars, " SV WORDS: ", sv_words, " SV SENTS: ", sv_sents, " SV L

    # Calc indexes
    EN_CLI = CLI(en_chars, en_words, en_sents)
    EN_ARI = ARI(en_chars, en_words, en_sents)
    EN_LIX = LIX(en_words, en_sents, en_longs)
```

```
    SV_CLI = CLI(sv_chars, sv_words, sv_sents)
    SV_ARI = ARI(sv_chars, sv_words, sv_sents)
    SV_LIX = LIX(sv_words, sv_sents, sv_longs)

    print()
    print("EN CLI: ", EN_CLI)
    print("EN ARI: ", EN_ARI)
    print("EN LIX: ", EN_LIX)
    print("SV CLI: ", SV_CLI)
    print("SV ARI: ", SV_ARI)
    print("SV LIX: ", SV_LIX)
    print()

# Calculate scores of wikipedia articles and write to JSON file
def write_json_wiki():
    nations = nationsArray.nations

    EN_CLI = {}
    EN_ARI = {}
    EN_LIX = {}

    SV_CLI = {}
    SV_ARI = {}
    SV_LIX = {}

    translator = translate.Translator(to_lang="sv")

    for nation in nations:
        sv_nation = translator.translate(nation) # Gain Swedish country name

        wikipedia.set_lang("en")
        en_text = wikipedia.page(nation).content # Gain English wikipedia text

        wikipedia.set_lang("sv")
        sv_text = wikipedia.page(sv_nation).content

        en_text = clean_text(en_text) # Parse and clean the wikipedia text
        sv_text = clean_text(sv_text) # Parse and clean the wikipedia text


        # Count needed parameters
        en_chars = character_count(en_text)
        en_words = word_count(en_text)
        en_sents = sentence_count(en_text)
        en_longs = long_word_count(en_text)

        sv_chars = character_count(sv_text)
        sv_words = word_count(sv_text)
        sv_sents = sentence_count(sv_text)
```

```
            sv_longs = long_word_count(sv_text)

            # Calculate and store readability scores
            EN_CLI[nation] = CLI(en_chars, en_words, en_sents)
            EN_ARI[nation] = ARI(en_chars, en_words, en_sents)
            EN_LIX[nation] = LIX(en_words, en_sents, en_longs)

            SV_CLI[nation] = CLI(sv_chars, sv_words, sv_sents)
            SV_ARI[nation] = ARI(sv_chars, sv_words, sv_sents)
            SV_LIX[nation] = LIX(sv_words, sv_sents, sv_longs)

            data = [{'country': key, 'score': val} for key, val in EN_CLI.items()]
            json_string = json.dumps(data)
            with open('EN_CLI.json', 'w') as outfile:
                json.dump(json_string, outfile)

            data = [{'country': key, 'score': val} for key, val in EN_ARI.items()]
            json_string = json.dumps(data)
            with open('EN_ARI.json', 'w') as outfile:
                json.dump(json_string, outfile)

            data = [{'country': key, 'score': val} for key, val in EN_LIX.items()]
            json_string = json.dumps(data)
            with open('EN_LIX.json', 'w') as outfile:
                json.dump(json_string, outfile)

            data = [{'country': key, 'score': val} for key, val in SV_CLI.items()]
            json_string = json.dumps(data)
            with open('SV_CLI.json', 'w') as outfile:
                json.dump(json_string, outfile)

            data = [{'country': key, 'score': val} for key, val in SV_ARI.items()]
            json_string = json.dumps(data)
            with open('SV_ARI.json', 'w') as outfile:
                json.dump(json_string, outfile)

            data = [{'country': key, 'score': val} for key, val in SV_LIX.items()]
            json_string = json.dumps(data)
            with open('SV_LIX.json', 'w') as outfile:
                json.dump(json_string, outfile)

    # Calculate average scores of wikipedia articles
    def calc_average_wiki():
        nations = nationsArray.nations

        translator = translate.Translator(to_lang="sv")

        SV_CLI = 0
        SV_ARI = 0
        SV_LIX = 0
```

```
EN_CLI = 0
EN_ARI = 0
EN_LIX = 0

en_chars = 0
en_words = 0
en_sents = 0
en_longs = 0
sv_chars = 0
sv_words = 0
sv_sents = 0
sv_longs = 0

for nation in nations:
    sv_nation = translator.translate(nation) # Gain Swedish country name

    wikipedia.set_lang("en")
    en_text = wikipedia.page(nation).content # Gain English wikipedia text

    wikipedia.set_lang("sv")
    sv_text = wikipedia.page(sv_nation).content

    en_text = clean_text(en_text) # Parse and clean the wikipedia text
    sv_text = clean_text(sv_text) # Parse and clean the wikipedia text


    # Count needed parameters
    en_chars_this = character_count(en_text)
    en_words_this = word_count(en_text)
    en_sents_this = sentence_count(en_text)
    en_longs_this = long_word_count(en_text)

    sv_chars_this = character_count(sv_text)
    sv_words_this = word_count(sv_text)
    sv_sents_this = sentence_count(sv_text)
    sv_longs_this = long_word_count(sv_text)

    EN_CLI += CLI(en_chars_this, en_words_this, en_sents_this)
    EN_ARI += ARI(en_chars_this, en_words_this, en_sents_this)
    EN_LIX += LIX(en_words_this, en_sents_this, en_longs_this)

    SV_CLI += CLI(sv_chars_this, sv_words_this, sv_sents_this)
    SV_ARI += ARI(sv_chars_this, sv_words_this, sv_sents_this)
    SV_LIX += LIX(sv_words_this, sv_sents_this, sv_longs_this)

    en_chars += en_chars_this
    en_words += en_words_this
    en_sents += en_sents_this
    en_longs += en_longs_this
    sv_chars += sv_chars_this
```

```
    sv_words += sv_words_this
    sv_sents += sv_sents_this
    sv_longs += sv_longs_this

SV_CLI_AVG = SV_CLI / len(nations)
SV_ARI_AVG = SV_ARI / len(nations)
SV_LIX_AVG = SV_LIX / len(nations)
EN_CLI_AVG = EN_CLI / len(nations)
EN_ARI_AVG = EN_ARI / len(nations)
EN_LIX_AVG = EN_LIX / len(nations)

print("EN CHARS: ", en_chars, " EN WORDS: ", en_words, " EN SENTS: ", en_sents, " EN L
print("SV CHARS: ", sv_chars, " SV WORDS: ", sv_words, " SV SENTS: ", sv_sents, " SV L

EN_CLI_TOTAL = CLI(en_chars, en_words, en_sents)
EN_ARI_TOTAL = ARI(en_chars, en_words, en_sents)
EN_LIX_TOTAL = LIX(en_words, en_sents, en_longs)

print("EN CLI TOTAL: ", EN_CLI_TOTAL, " EN ARI TOTAL: ", EN_ARI_TOTAL, " EN LIX TOTAL:

SV_CLI_TOTAL = CLI(sv_chars, sv_words, sv_sents)
SV_ARI_TOTAL = ARI(sv_chars, sv_words, sv_sents)
SV_LIX_TOTAL = LIX(sv_words, sv_sents, sv_longs)

print("SV CLI TOTAL: ", SV_CLI_TOTAL, " SV ARI TOTAL: ", SV_ARI_TOTAL, " SV LIX TOTAL:

print()
print("EN CLI AVG: ", EN_CLI_AVG)
print("EN ARI AVG: ", EN_ARI_AVG)
print("EN LIX AVG: ", EN_LIX_AVG)
print("SV CLI AVG: ", SV_CLI_AVG)
print("SV ARI AVG: ", SV_ARI_AVG)
print("SV LIX AVG: ", SV_LIX_AVG)
print()
```

A.2 nations.py

```
nations = ['Afghanistan', 'Albania', 'Algeria', 'Andorra', 'Angola',
 'Argentina', 'Armenia', 'Australia', 'Austria', 'Azerbaijan', 'Bangladesh',
 'Barbados', 'Belarus', 'Belgium', 'Belize', 'Bolivia', 'Botswana', 'Brazil',
 Brunei', 'Bulgaria', 'Burma', 'Cambodia', 'Cameroon', 'Canada', 'Chad',
 'Chile,'China', 'Colombia', 'Comoros', 'Croatia', 'Cuba', 'Cyprus', 'Denmark',
 'Djibouti', 'Dominica', 'Ecuador', 'Egypt', 'Eritrea', 'Estonia', 'Ethiopia',
 'Fiji', 'Finland', 'France', 'Gabon', 'Germany', 'Ghana', 'Greece', 'Grenada',
 'Guatemala', 'Guinea', 'Guinea-Bissau', 'Guyana', 'Haiti', 'Honduras',
 'Hungary', 'Iceland', 'India', 'Indonesia', 'Iran', 'Iraq', 'Ireland', 'Israel
 , 'Italy', 'Jamaica', 'Japan', 'Jordan', 'Kazakhstan', 'Kenya', 'Kiribati',
 'Kuwait', 'Kyrgyzstan', 'Laos', 'Latvia', 'Lebanon', 'Lesotho', 'Liberia',
 'Libya', 'Liechtenstein', 'Lithuania', 'Luxembourg', 'Malawi', 'Malaysia',
 'Maldives', 'Mali', 'Malta', 'Mauritius', 'Mexico', 'Moldova', 'Monaco',
 'Mongolia', 'Montenegro', 'Morocco', 'Mozambique', 'Namibia', 'Nauru', 'Nepal',
 'Netherlands', 'Nicaragua', 'Niger', 'Nigeria', 'Norway', 'Oman', 'Pakistan',
 'Palestine', 'Panama', 'Paraguay', 'Peru', 'Philippines', 'Poland', 'Portugal',
 'Qatar', 'Romania', 'Russia', 'Rwanda', 'Senegal', 'Serbia', 'Singapore',
 'Slovakia', 'Slovenia', 'Somalia', 'Spain', 'Sudan', 'Swaziland', 'Sweden',
 'Switzerland', 'Syria', 'Tajikistan', 'Tanzania', 'Thailand', 'Togo', 'Tonga',
 'Tunisia', 'Turkey', 'Turkmenistan', 'Tuvalu', 'Uganda', 'Ukraine', 'Uruguay',
 'Uzbekistan', 'Vanuatu', 'Venezuela', 'Vietnam', 'Yemen', 'Zambia', 'Zimbabwe',
 'Taiwan']
```

B Data

B.1 ARI

Sheet1

EN ARI		SV ARI	
score	country	score	country
13.96963019	Bolivia	14.17855678	Bolivia
15.82929992	Colombia	13.81670322	Colombia
13.56332821	Morocco	12.73109829	Morocco
15.60922624	Mexico	12.37570186	Mexico
14.21064153	Cambodia	13.06978277	Cambodia
13.67548853	Guinea	12.21684939	Guinea
14.83901243	Cyprus	11.20996779	Cyprus
14.11655735	Monaco	12.11945831	Monaco
11.75441356	Brunei	11.95202142	Brunei
15.61607734	China	10.14216761	China
13.53642494	Uganda	11.93519712	Uganda
14.63414328	Indonesia	14.12813589	Indonesia
12.25727786	Sudan	10.91058543	Sudan
14.22034861	Montenegro	13.23755701	Montenegro
15.61604519	Malawi	13.34314188	Malawi
13.45040855	Philippines	13.43010653	Philippines
14.78661443	Poland	13.18060946	Poland
12.36102214	Maldives	13.03038739	Maldives
12.30594874	Kuwait	11.50300159	Kuwait
12.98130401	Libya	12.86830856	Libya
13.26821291	Eritrea	13.08119121	Eritrea
14.78516567	Tajikistan	14.83561303	Tajikistan
14.36732366	Pakistan	12.50182986	Pakistan
14.35175227	Mozambique	11.94454969	Mozambique
15.37939503	Slovenia	11.87080669	Slovenia
13.82894913	Netherlands	14.5177972	Netherlands
12.50939274	Barbados	12.24614939	Barbados
14.19262813	Russia	17.15270342	Russia
15.34770972	India	11.51196123	India
13.053864	Vanuatu	10.96415199	Vanuatu
14.46243411	Armenia	12.7463719	Armenia
14.06911753	Bangladesh	12.68178683	Bangladesh
13.83752459	Uzbekistan	16.53272659	Uzbekistan
12.61881635	Chad	12.4005029	Chad
15.56051822	Argentina	14.2120217	Argentina
14.40430415	Zimbabwe	16.52885875	Zimbabwe
13.20765582	Qatar	10.15766306	Qatar
14.54643718	Kyrgyzstan	16.01531845	Kyrgyzstan
13.78758731	Laos	14.28832512	Laos
16.23909577	Brazil	13.12404008	Brazil
11.44692083	Grenada	14.44360505	Grenada
15.03008308	Germany	13.13083086	Germany
14.6697405	Niger	12.10616786	Niger
13.8146993	Japan	12.37685142	Japan
14.06312222	Lebanon	12.73255462	Lebanon
13.1465182	Djibouti	10.54261381	Djibouti
13.8393546	Sweden	12.92694477	Sweden
15.00217425	Angola	13.70943994	Angola
13.9703604	Belarus	14.45289978	Belarus
14.04937672	Ireland	12.32755908	Ireland
13.33149137	Jordan	12.36383135	Jordan
14.20016255	Guyana	14.75073354	Guyana
14.72833857	Ecuador	13.3597082	Ecuador
14.4573196	Zambia	11.88687558	Zambia

14.405326 Liechtenstein	13.21973052 Liechtenstein
14.97547684 Belgium	16.46368133 Belgium
15.02966425 Moldova	15.56164847 Moldova
11.96466973 Peru	12.86027576 Peru
12.26495987 Belize	13.40963618 Belize
14.73680594 Rwanda	12.39704349 Rwanda
12.64851936 Andorra	10.61724838 Andorra
14.40311959 Uruguay	14.42734005 Uruguay
12.2591297 Ethiopia	12.60526824 Ethiopia
13.04292904 Slovakia	12.28414514 Slovakia
15.21398787 Romania	12.8883977 Romania
13.72609515 Burma	11.78329898 Burma
14.35870211 Lesotho	11.50791767 Lesotho
13.29486911 Algeria	13.30162281 Algeria
13.87852288 Iceland	13.52652806 Iceland
13.8673692 Paraguay	12.98582384 Paraguay
14.29782116 Estonia	13.15399798 Estonia
14.40069378 Azerbaijan	13.95437385 Azerbaijan
14.247467 Honduras	14.39592093 Honduras
15.64479301 Greece	12.47530664 Greece
12.43105748 Dominica	13.84926471 Dominica
16.51194777 Portugal	13.08587221 Portugal
15.85820006 Australia	11.77962746 Australia
14.67099604 Chile	13.30481516 Chile
15.93144605 Italy	14.6547201 Italy
15.05407077 Turkey	8.388127618 Turkey
15.07740178 Ukraine	12.19885208 Ukraine
14.50081742 Switzerland	13.65311279 Switzerland
14.8394337 Taiwan	15.47337769 Taiwan
13.13981585 Gabon	12.85564207 Gabon
15.62511795 France	15.54089364 France
14.47688042 Serbia	13.08280876 Serbia
12.27724777 Nauru	11.77123805 Nauru
14.70254516 Bulgaria	12.70844603 Bulgaria
14.7464037 Spain	14.25075484 Spain
15.89801964 Canada	15.26431581 Canada
13.27743875 Finland	13.3563179 Finland
14.68660555 Israel	13.7990836 Israel
14.30704265 Botswana	11.53562758 Botswana
14.49849648 Luxembourg	12.50736842 Luxembourg
13.74842843 Kenya	12.80701474 Kenya
14.36886007 Namibia	12.90793021 Namibia
11.44630628 Haiti	13.20514591 Haiti
13.04599711 Liberia	13.83370379 Liberia
13.18826144 Cameroon	16.59740335 Cameroon
13.58584306 Senegal	13.72448436 Senegal
13.66045532 Fiji	10.67179646 Fiji
11.62659362 Yemen	13.45471768 Yemen
13.3028359 Tunisia	13.83692713 Tunisia
15.35904884 Venezuela	15.27856715 Venezuela
14.37530679 Hungary	13.86844096 Hungary
12.42601776 Mali	12.51551192 Mali
13.63633687 Norway	11.10426124 Norway
15.06460987 Vietnam	12.54202703 Vietnam
14.62235991 Afghanistan	11.46825187 Afghanistan
16.86864395 Ghana	12.12375425 Ghana

14.07328643	Mauritius	12.62322315	Mauritius
12.76714454	Nepal	14.34123748	Nepal
14.50707016	Austria	14.02266852	Austria
13.09797857	Mongolia	14.87833056	Mongolia
16.40991712	Tuvalu	11.47019021	Tuvalu
13.73811457	Egypt	11.09588976	Egypt
14.06587203	Thailand	11.05345574	Thailand
14.23254459	Syria	11.270764	Syria
14.19776611	Albania	12.24515296	Albania
12.45275436	Oman	9.543201524	Oman
13.06657478	Swaziland	10.7080237	Swaziland
12.41058808	Tanzania	11.85683599	Tanzania
13.48361174	Malaysia	11.67648604	Malaysia
13.58613237	Guatemala	13.51544716	Guatemala
14.28849455	Singapore	12.64058039	Singapore
14.5635609	Croatia	12.49146458	Croatia
15.19445158	Somalia	14.84271727	Somalia
13.59553567	Iran	12.24399132	Iran
12.89249008	Cuba	12.73363266	Cuba
14.61625531	Nigeria	11.15699235	Nigeria
13.54612921	Jamaica	13.53051802	Jamaica
14.47076524	Kazakhstan	16.71851865	Kazakhstan
12.9711811	Malta	10.86619001	Malta
14.22842689	Kiribati	13.48115618	Kiribati
15.63428404	Palestine	12.40422829	Palestine
14.50061401	Guinea-Bissau	13.73701211	Guinea-Bissau
14.20089946	Turkmenistan	15.09342173	Turkmenistan
13.16820216	Nicaragua	12.99438375	Nicaragua
14.01095858	Latvia	11.80003976	Latvia
14.39198888	Comoros	12.56522643	Comoros
12.8261481	Togo	12.85284705	Togo
14.26148561	Lithuania	12.48641286	Lithuania
12.86557918	Iraq	11.87540326	Iraq
12.12754178	Panama	13.73245509	Panama
13.80339694	Denmark	11.68758793	Denmark
15.12490883	Tonga	11.23264906	Tonga

B.2 CLI

EN CLI		SV CLI	
score	country	score	country
13.7781498	Bolivia	16.66474215	Bolivia
14.45649444	Colombia	17.0663331	Colombia
13.4738963	Morocco	15.97805226	Morocco
13.8453538	Mexico	17.09303473	Mexico
13.81274417	Cambodia	15.95197982	Cambodia
13.49879248	Guinea	16.36568341	Guinea
13.17396276	Cyprus	16.04438319	Cyprus
12.54329845	Monaco	14.81662367	Monaco
12.42645553	Brunei	15.98552333	Brunei
13.92712227	China	14.5781336	China
13.74904366	Uganda	14.80451202	Uganda
15.05210536	Indonesia	17.73790941	Indonesia
12.89523104	Sudan	15.29373239	Sudan
13.62265407	Montenegro	16.82743374	Montenegro
13.95138288	Malawi	16.8252921	Malawi
13.71910229	Philippines	17.13725851	Philippines
13.31942001	Poland	16.2365377	Poland
13.01246167	Maldives	17.05188971	Maldives
13.04262795	Kuwait	15.27900648	Kuwait
12.63991142	Libya	16.02793975	Libya
13.41834326	Eritrea	16.42089402	Eritrea
14.23534641	Tajikistan	19.20234467	Tajikistan
13.76762276	Pakistan	17.56891169	Pakistan
14.33304699	Mozambique	16.0372619	Mozambique
14.47643974	Slovenia	15.61398011	Slovenia
13.20302838	Netherlands	18.07879612	Netherlands
12.89273284	Barbados	15.62588235	Barbados
13.09191385	Russia	18.96642224	Russia
14.0662355	India	15.9980938	India
13.42537064	Vanuatu	15.52610733	Vanuatu
14.09652834	Armenia	16.1650914	Armenia
14.04538865	Bangladesh	16.62210948	Bangladesh
13.90908523	Uzbekistan	18.52636248	Uzbekistan
13.0822311	Chad	15.65799422	Chad
14.15745001	Argentina	17.56438938	Argentina
13.89419457	Zimbabwe	17.47093699	Zimbabwe
12.22118551	Qatar	14.9175419	Qatar
13.68965425	Kyrgyzstan	17.72958636	Kyrgyzstan
13.47359675	Laos	17.74835546	Laos
14.2109261	Brazil	16.79481013	Brazil
12.63583683	Grenada	17.41930348	Grenada
14.20808104	Germany	16.59384024	Germany
13.64852082	Niger	15.88564752	Niger
13.37851791	Japan	14.92530233	Japan
13.74803902	Lebanon	15.11576504	Lebanon
13.0619165	Djibouti	14.86490231	Djibouti
12.97964039	Sweden	17.38118359	Sweden
13.40089863	Angola	16.55701149	Angola
14.0415236	Belarus	17.86331268	Belarus
12.94970605	Ireland	15.82985463	Ireland
13.0211486	Jordan	15.27363709	Jordan
13.76299807	Guyana	17.13033535	Guyana
13.94846021	Ecuador	16.62650817	Ecuador
13.68447159	Zambia	15.38208857	Zambia

14.37048951	Liechtenstein	17.67333333	Liechtenstein
14.0310556	Belgium	18.79215909	Belgium
14.255869	Moldova	17.52213695	Moldova
13.96043548	Peru	15.76326032	Peru
12.68986772	Belize	16.22666667	Belize
14.02344118	Rwanda	16.77065241	Rwanda
12.22072311	Andorra	14.94535714	Andorra
13.73252226	Uruguay	18.12494888	Uruguay
13.32032224	Ethiopia	17.39261067	Ethiopia
13.23459224	Slovakia	16.95021223	Slovakia
13.58646477	Romania	17.68574882	Romania
13.3899492	Burma	15.63301173	Burma
13.41754988	Lesotho	15.80443944	Lesotho
13.28315924	Algeria	16.07513761	Algeria
13.58539062	Iceland	16.77388644	Iceland
14.4767651	Paraguay	17.86065858	Paraguay
13.72094592	Estonia	16.931695	Estonia
14.22578079	Azerbaijan	18.01878201	Azerbaijan
13.76996231	Honduras	17.53449566	Honduras
14.13389182	Greece	16.55549818	Greece
13.11861551	Dominica	16.9931746	Dominica
14.1992567	Portugal	16.41156411	Portugal
14.40137228	Australia	16.68159965	Australia
13.67773183	Chile	16.53279087	Chile
13.62255782	Italy	18.7968384	Italy
13.58954131	Turkey	11.84174455	Turkey
13.81418227	Ukraine	17.54974253	Ukraine
13.78764744	Switzerland	17.40215494	Switzerland
12.94729636	Taiwan	17.58342268	Taiwan
14.06166667	Gabon	16.11772512	Gabon
13.5557569	France	18.70406922	France
13.87973337	Serbia	16.27810446	Serbia
12.77410096	Nauru	14.51329779	Nauru
14.00898424	Bulgaria	17.34901779	Bulgaria
13.64286014	Spain	16.87366712	Spain
14.63607369	Canada	17.2395749	Canada
13.65495231	Finland	17.01676641	Finland
13.3737201	Israel	17.03618863	Israel
14.40931813	Botswana	15.7744277	Botswana
13.86980282	Luxembourg	16.44271762	Luxembourg
13.66042607	Kenya	15.69484705	Kenya
13.89117181	Namibia	16.67616178	Namibia
12.92205545	Haiti	16.12753906	Haiti
13.19003559	Liberia	16.07709526	Liberia
14.29893136	Cameroon	18.22122925	Cameroon
13.1908521	Senegal	17.00169625	Senegal
12.79059634	Fiji	15.23895945	Fiji
12.06339031	Yemen	16.58007308	Yemen
12.92542031	Tunisia	15.91979613	Tunisia
13.91350471	Venezuela	18.6972617	Venezuela
13.96034257	Hungary	17.3699345	Hungary
13.21217932	Mali	15.3454212	Mali
13.39054907	Norway	15.2801232	Norway
13.6345671	Vietnam	15.00604891	Vietnam
13.44349284	Afghanistan	16.22023425	Afghanistan
13.89626538	Ghana	15.66719388	Ghana

13.74148543	Mauritius	15.26940344	Mauritius
12.9819022	Nepal	16.75969231	Nepal
13.6419141	Austria	18.90804325	Austria
13.33654899	Mongolia	17.52754209	Mongolia
13.72455078	Tuvalu	15.25460823	Tuvalu
13.04994239	Egypt	15.3731632	Egypt
13.10036247	Thailand	15.53682988	Thailand
13.223292	Syria	16.36224138	Syria
13.55158228	Albania	16.11547002	Albania
11.99360933	Oman	14.17511543	Oman
13.42481312	Swaziland	14.62240363	Swaziland
13.09188795	Tanzania	15.7491535	Tanzania
13.59234808	Malaysia	16.68946903	Malaysia
14.02537457	Guatemala	16.8596093	Guatemala
14.02569425	Singapore	17.77123682	Singapore
14.24995758	Croatia	15.94304756	Croatia
14.05468805	Somalia	17.43274573	Somalia
12.94581298	Iran	15.92035444	Iran
12.78802738	Cuba	17.19793211	Cuba
13.60211389	Nigeria	15.46969802	Nigeria
13.71552018	Jamaica	16.49409002	Jamaica
14.40320054	Kazakhstan	17.99161446	Kazakhstan
12.55962582	Malta	14.66350997	Malta
13.8963394	Kiribati	16.16581006	Kiribati
13.31162193	Palestine	16.26509167	Palestine
13.45448788	Guinea-Bissau	17.16502415	Guinea-Bissau
14.60820114	Turkmenistan	17.98919476	Turkmenistan
13.91602626	Nicaragua	16.70490013	Nicaragua
13.57536493	Latvia	16.07157258	Latvia
13.78348988	Comoros	16.99152294	Comoros
12.90846369	Togo	14.8288961	Togo
14.63463681	Lithuania	16.6851954	Lithuania
12.4892562	Iraq	15.38426811	Iraq
12.76265578	Panama	16.81413139	Panama
13.34774864	Denmark	15.57836237	Denmark
13.75210562	Tonga	14.6719906	Tonga

B.3 LIX

EN LIX		SV LIX	
score	country	score	country
54.40475152	Bolivia	52.87906536	Bolivia
59.09950763	Colombia	49.95668951	Colombia
54.43365938	Morocco	48.78053388	Morocco
56.87401788	Mexico	49.62605636	Mexico
54.38288753	Cambodia	49.76680387	Cambodia
52.48421976	Guinea	45.91292083	Guinea
54.61661341	Cyprus	45.72741121	Cyprus
49.88368851	Monaco	46.82692559	Monaco
48.57537841	Brunei	46.38815381	Brunei
57.12077964	China	42.32831361	China
52.04836962	Uganda	46.33730939	Uganda
57.00517874	Indonesia	53.19860627	Indonesia
48.79917111	Sudan	45.97367085	Sudan
53.72380113	Montenegro	50.48032411	Montenegro
56.17700101	Malawi	49.71052201	Malawi
52.6230688	Philippines	51.41423331	Philippines
53.36763666	Poland	47.84378941	Poland
50.45087783	Maldives	51.16425831	Maldives
49.74979453	Kuwait	45.12157554	Kuwait
49.95792428	Libya	50.34833439	Libya
54.12943371	Eritrea	50.28431109	Eritrea
56.18262764	Tajikistan	53.41129639	Tajikistan
55.33479269	Pakistan	49.21872018	Pakistan
54.9003878	Mozambique	47.77326605	Mozambique
57.42526006	Slovenia	48.58058232	Slovenia
52.54358468	Netherlands	52.07254585	Netherlands
51.03907372	Barbados	48.9038282	Barbados
51.71809317	Russia	58.05277324	Russia
54.83897188	India	46.31794666	India
52.97039086	Vanuatu	46.840564	Vanuatu
56.49933498	Armenia	51.99684671	Armenia
53.29729563	Bangladesh	49.53273253	Bangladesh
53.27881405	Uzbekistan	58.58964836	Uzbekistan
49.8853689	Chad	47.85960486	Chad
56.96129878	Argentina	52.45675136	Argentina
54.390235	Zimbabwe	57.18188737	Zimbabwe
49.57036264	Qatar	42.13476747	Qatar
54.03801239	Kyrgyzstan	55.88864763	Kyrgyzstan
52.96580216	Laos	52.29287088	Laos
58.60053016	Brazil	50.99156118	Brazil
48.71045622	Grenada	54.76502105	Grenada
55.62851995	Germany	48.6221092	Germany
52.9065785	Niger	45.82526658	Niger
52.7000515	Japan	46.94586563	Japan
54.69112438	Lebanon	48.85810959	Lebanon
51.92688133	Djibouti	46.28979246	Djibouti
51.47439873	Sweden	49.74926502	Sweden
55.20844501	Angola	48.94170235	Angola
54.80443042	Belarus	53.82432083	Belarus
53.4146555	Ireland	48.80843875	Ireland
50.86118	Jordan	46.68080419	Jordan
54.29736942	Guyana	53.22410546	Guyana
56.61751625	Ecuador	51.54054567	Ecuador
54.37334416	Zambia	45.3629977	Zambia

53.18983033	Liechtenstein	48.55518303	Liechtenstein
57.09011286	Belgium	58.22821494	Belgium
57.36583239	Moldova	56.29655502	Moldova
51.01964366	Peru	50.11005959	Peru
48.39226511	Belize	51.68089085	Belize
55.01577364	Rwanda	48.27040783	Rwanda
51.83341401	Andorra	44.11038961	Andorra
56.06415872	Uruguay	53.43973123	Uruguay
50.35255662	Ethiopia	47.02069333	Ethiopia
50.66054563	Slovakia	47.2315612	Slovakia
55.90372087	Romania	48.7902531	Romania
53.52507816	Burma	47.11593303	Burma
55.659579	Lesotho	46.32619027	Lesotho
52.58795478	Algeria	51.0493924	Algeria
54.06640676	Iceland	49.80996563	Iceland
55.80725769	Paraguay	51.46474994	Paraguay
55.3646512	Estonia	50.36574245	Estonia
54.54624578	Azerbaijan	51.84911196	Azerbaijan
55.57078199	Honduras	52.1710477	Honduras
55.52082375	Greece	49.90411212	Greece
50.65837016	Dominica	54.3076564	Dominica
59.29423386	Portugal	50.42302666	Portugal
57.81268199	Australia	48.62067538	Australia
54.86215199	Chile	49.56169877	Chile
57.97948166	Italy	53.64629101	Italy
55.92645138	Turkey	40.35245461	Turkey
57.1002301	Ukraine	46.81002136	Ukraine
54.00902038	Switzerland	51.73939565	Switzerland
52.97606844	Taiwan	53.92502343	Taiwan
52.6229908	Gabon	47.24202212	Gabon
55.85005691	France	55.98248472	France
54.90628152	Serbia	51.49679204	Serbia
48.71494365	Nauru	44.13953626	Nauru
55.75467875	Bulgaria	49.62749349	Bulgaria
54.92527811	Spain	54.79999412	Spain
58.55080598	Canada	54.00772829	Canada
52.45626885	Finland	51.75140874	Finland
53.41296315	Israel	51.20986856	Israel
56.56943452	Botswana	47.41809068	Botswana
55.26261737	Luxembourg	49.12509778	Luxembourg
52.15571652	Kenya	48.3403051	Kenya
55.30327696	Namibia	50.90642347	Namibia
47.74711064	Haiti	49.71453011	Haiti
52.37299066	Liberia	51.2461517	Liberia
52.84541914	Cameroon	59.70332312	Cameroon
52.84695323	Senegal	52.94725275	Senegal
51.71850717	Fiji	45.20169544	Fiji
47.04936996	Yemen	47.97374717	Yemen
52.44864586	Tunisia	54.01878801	Tunisia
57.54755784	Venezuela	58.05460397	Venezuela
54.15896622	Hungary	50.39143433	Hungary
49.58923924	Mali	47.6821184	Mali
51.75450254	Norway	44.37982296	Norway
56.98910068	Vietnam	47.58633205	Vietnam
54.28441801	Afghanistan	46.71037521	Afghanistan
58.15679566	Ghana	47.16530612	Ghana

53.62155029	Mauritius	52.72802364	Mauritius
49.75822358	Nepal	51.60409035	Nepal
54.99800049	Austria	50.81725194	Austria
52.54793386	Mongolia	53.96177896	Mongolia
58.87279322	Tuvalu	47.4705265	Tuvalu
52.10872296	Egypt	47.1037261	Egypt
53.8964833	Thailand	49.95867911	Thailand
52.77459313	Syria	46.14730386	Syria
55.28952686	Albania	49.5521564	Albania
48.00483234	Oman	42.98421593	Oman
52.34821806	Swaziland	46.69477757	Swaziland
51.98075789	Tanzania	47.36826288	Tanzania
53.30457398	Malaysia	47.67235261	Malaysia
53.28272521	Guatemala	50.42445711	Guatemala
55.10419808	Singapore	49.45420008	Singapore
57.11853618	Croatia	48.66737432	Croatia
56.80679901	Somalia	55.54910783	Somalia
52.56793505	Iran	48.48353	Iran
51.02834047	Cuba	48.94547148	Cuba
54.38270316	Nigeria	45.98401921	Nigeria
54.14708502	Jamaica	51.75478685	Jamaica
54.6355166	Kazakhstan	59.44061962	Kazakhstan
50.61897193	Malta	43.25647337	Malta
55.83653836	Kiribati	52.57639058	Kiribati
55.96923336	Palestine	49.44770515	Palestine
53.81445581	Guinea-Bissau	49.92687446	Guinea-Bissau
55.63181965	Turkmenistan	53.81483034	Turkmenistan
52.95626595	Nicaragua	50.72537202	Nicaragua
54.02206836	Latvia	46.73074484	Latvia
55.53862662	Comoros	51.2371134	Comoros
50.37152805	Togo	49.44085644	Togo
55.26897341	Lithuania	48.99813607	Lithuania
50.85938017	Iraq	45.92903224	Iraq
48.77023002	Panama	51.38765471	Panama
52.62938032	Denmark	47.69167412	Denmark
55.41146456	Tonga	44.45454064	Tonga