C# Split a string with mixed language into different language chunks

Asked 5 years, 5 months ago Modified 5 years, 4 months ago Viewed 3k times

I am trying to solve a problem where I have a string with mixed language as input.

> E.g. "Hyundai Motor Company 현대자동차 现代 Some other English words"

And I want to split the string into *different language chunks*.

> E.g. ["Hyundai Motor Company", "현대자동차", "现代", "Some other English words"]

OR (Space/Punctuation marks and order do not matter)

> ["HyundaiMotorCompany", "현대자동차", "现代", "SomeotherEnglishwords"]

Is there an easy way to solve this problem? Or any assembly/nuget package I can use?

Thanks

Edit: I figured that my "language chunks" is ambiguous. What I want by "language chunks" is language character sets.

> For example "Hyundai Motor Company" is in English character set, "현대자동차" in Korean set, "现代" in Chinese set, "Some other English words" in English set.

Additions to clarify the requirement of my problem is:

1: The input can have spaces or any other punctuation marks, but I can always use regular expressions to ignore them.

2: I will pre-process the input to ignore Diacritics. So "å" becomes "a" in my input. So all the English like characters will become English characters.

What I really want is to find a way to parse the inputs into different language-character sets ignoring spaces & punctuation marks.

E.g. From "HyundaiMotorCompany현대자동차现代SomeotherEnglishwords"

To ["HyundaiMotorCompany", "현대자동차", "现代", "SomeotherEnglishwords"]

c# split multilingual

Share Improve this question

Follow

edited Jun 20, 2020 at 9:12

Community Bot

1 1

asked Aug 10, 2017 at 16:55

Kenneth Song

51 1 5

Thanks for the response, I need a solution that works for other languages as well. The input can be in any language and in multiple languages (e.g. 1 or more different languages). – Kenneth Song Aug 10, 2017 at 16:59

2 What do you consider an other language, just non a-Z characters? Or do you really want to split that "Hello, Hallo, Hola, Ciao" into 4 different chunks? – Rand Random Aug 10, 2017 at 17:00

You may try an approach that passes chunks of the string through a language detection library like ntextcat github.com/ivanakcheurov/ntextcat – Kelson Ball Aug 10, 2017 at 17:01

Can `Hyundai` really be interpreted as an english word? – Rand Random Aug 10, 2017 at 17:04

@rand OP means Latin script which is used for writing English. It should match Rand and Nikhil both even if they are not valid words in English. – Nikhil Vartak Aug 10, 2017 at 17:05

3 Answers

Sorted by: Highest score (default)

4

Language chunks can be defined by using UNICODE blocks. The current list of UNICODE blocks is available at ftp://www.unicode.org/Public/UNIDATA/Blocks.txt. Here is an excerpt from the list:

```
0000..007F; Basic Latin
0080..00FF; Latin-1 Supplement
0100..017F; Latin Extended-A
0180..024F; Latin Extended-B
0250..02AF; IPA Extensions
02B0..02FF; Spacing Modifier Letters
0300..036F; Combining Diacritical Marks
0370..03FF; Greek and Coptic
0400..04FF; Cyrillic
0500..052F; Cyrillic Supplement
```

The idea is to classify the characters using the UNICODE block. Consecutive characters belonging to the same UNICODE block define a *language chunk*.

First problem with this definition is that what you might consider a single script (or *language*) spans several blocks like *Cyrillic* and *Cyrillic Supplement*. To handle this you can merge blocks containing the same name so all *Latin* blocks are merged into a single *Latin* script etc.

However, this creates several new problems:

1. Should the blocks *Greek and Coptic*, *Coptic* and *Greek Supplement* be merged into a single script or should you try to make a distinction between Greek and Coptic script?
2. You should probably merge all the *CJK* blocks. However, because these blocks contain both Chinese as well as Kanji (Japanese) and Hanja (Korean) characters you will not be able to distinguish between these scripts when CJK characters are used.

Assuming that you have a plan for how to use UNICODE blocks to classify characters into scripts you then have to decide how to handle spacing and punctuation. The space character and several forms of punctuation belong to the *Basic Latin* block. However, other blocks may also contain non-letter characters.

A strategy for dealing with this is to "ignore" the UNICODE block of non-letter characters but include them in chunks. In your example you have two non-latin chunks that happens to not contain space or punctuation but many scripts will use space as it is used in the latin script, e.g. Cyrillic. Even though a space is classifed as *Latin* you still want a sequence of words in Cyrillic separated by spaces to be considered a single chunk using the Cyrillic script instead of a Cyrillic word followed by a Latin space and then another Cyrillic word etc.

Finally, you need to decide how to handle numbers. You can treat them as space and punctuation or classify them as the block they belong to, e.g. Latin digits are *Latin* while Devanagari digits are *Devanagari* etc.

Here is some code putting all this together. First a class to represent a script (based on UNICODE blocks like "Greek and Coptic": 0x0370 - 0x03FF):

```
public class Script
{
    public Script(int from, int to, string name)
    {
        From = from;
        To = to;
        Name = name;
    }
```

```
    public int From { get; }
    public int To { get; }
    public string Name { get; }

    public bool Contains(char c) => From <= (int) c && (int) c <= To;
}
```

Next a class for downloading and parsing the UNICODE blocks file. This code downloads the text in the constructor which might not be ideal. Instead you can use a local copy of the file or something similar.

```
public class Scripts
{
    readonly List<Script> scripts;

    public Scripts()
    {
        using (var webClient = new WebClient())
        {
            const string url =
"ftp://www.unicode.org/Public/UNIDATA/Blocks.txt";
            var blocks = webClient.DownloadString(url);
            var regex = new Regex(@"^(?<from>[0-9A-F]{4})\.\.(?<to>[0-9A-F]
{4}); (?<name>.+)$");
            scripts = blocks
                .Split(new[] { '\r', '\n' },
StringSplitOptions.RemoveEmptyEntries)
                .Select(line => regex.Match(line))
                .Where(match => match.Success)
                .Select(match => new Script(
                    Convert.ToInt32(match.Groups["from"].Value, 16),
                    Convert.ToInt32(match.Groups["to"].Value, 16),
                    NormalizeName(match.Groups["name"].Value)))
                .ToList();
        }
    }

    public string GetScript(char c)
    {
        if (!char.IsLetterOrDigit(c))
            // Use the empty string to signal space and punctuation.
            return string.Empty;
        // Linear search - can be improved by using binary search.
        foreach (var script in scripts)
            if (script.Contains(c))
                return script.Name;
        return string.Empty;
    }

    // Add more special names if required.
    readonly string[] specialNames = new[] { "Latin", "Cyrillic", "Arabic",
"CJK" };

    string NormalizeName(string name) => specialNames.FirstOrDefault(sn =>
name.Contains(sn)) ?? name;
}
```

Notice that blocks above UNICODE code point 0xFFFF are ignored. If you have to work with these characters you will have to expand a lot on the code I have provided that assumes that a UNICODE character is represented by a 16 bit value.

Next task is to split a string into UNICODE blocks. It will return words consisting of a string of consecutive characters that belong to the same script (the second element of the tuple). The `scripts` variable is an instance of the `Scripts` class defined above.

```
public IEnumerable<(string text, string script)> SplitIntoWords(string text)
{
    if (text.Length == 0)
        yield break;
    var script = scripts.GetScript(text[0]);
    var start = 0;
    for (var i = 1; i < text.Length - 1; i += 1)
    {
        var nextScript = scripts.GetScript(text[i]);
        if (nextScript != script)
        {
            yield return (text.Substring(start, i - start), script);
            start = i;
            script = nextScript;
        }
    }
    yield return (text.Substring(start, text.Length - start), script);
}
```

Executing `SplitIntoWords` on your text will return something like this:

```
Text      | Script
----------+----------------
Hyundai   | Latin
[space]   | [empty string]
Motor     | Latin
[space]   | [empty string]
Company   | Latin
[space]   | [empty string]
현대자동차 | Hangul Syllables
[space]   | [empty string]
现代      | CJK
...
```

Next step is to join consecutive words belonging to the same script ignoring space and punctuation:

```
public IEnumerable<string> JoinWords(IEnumerable<(string text, string
script)> words)
{
    using (var enumerator = words.GetEnumerator())
    {
        if (!enumerator.MoveNext())
            yield break;
```

```
            var (text, script) = enumerator.Current;
            var stringBuilder = new StringBuilder(text);
            while (enumerator.MoveNext())
            {
                var (nextText, nextScript) = enumerator.Current;
                if (script == string.Empty)
                {
                    stringBuilder.Append(nextText);
                    script = nextScript;
                }
                else if (nextScript != string.Empty && nextScript != script)
                {
                    yield return stringBuilder.ToString();
                    stringBuilder = new StringBuilder(nextText);
                    script = nextScript;
                }
                else
                    stringBuilder.Append(nextText);
            }
            yield return stringBuilder.ToString();
        }
    }
```

This code will include any space and punctuation with the preceeding words using the same script.

Putting it all together:

```
var chunks = JoinWords(SplitIntoWords(text));
```

This will result in these chunks:

- Hyundai Motor Company
- 현대자동차
- 现代
- Some other English words

All chunks except the last have a trailing space.

Share Improve this answer Follow

edited Aug 11, 2017 at 7:57

answered Aug 11, 2017 at 6:33

Martin Liversage
103k 22 204 251

4

It's a language identification problem. You need to use the proper library for this. There's C# package which supports 78 languages trained on Wikipedia and Twitter. But in general Python is better for solving such kind of problems. For Python I can recommend this package.

So, you need to split your text into sentences or words and apply text detection algorithm to recognise language. Next, you can group result by languages.

Share Improve this answer Follow

edited Aug 12, 2017 at 15:07

answered Aug 10, 2017 at 17:14

Vasyl Zvarydchuk
3,709 24 35

1

As far as I understand from your question you want to distinguish between English and non-English (Unicode) characters. We can use `[\x00-\x7F]+` regex here for that. Note that `^` is used for non-English chars.

```
string input = "Hyundai Motor Company 현대자동차 现代 Some other English words";

string englishCharsPattern = "[\x00-\x7F]+";
var englishParts = Regex.Matches(input, englishCharsPattern)
                        .OfType<Match>()
                        .Where(m =>
!string.IsNullOrWhiteSpace(m.Groups[0].Value))
                        .Select(m => m.Groups[0].Value.Trim())
                        .ToList();

string nonEnglishCharsPattern = "[^\x00-\x7F]+";
var nonEnglishParts = Regex.Matches(input, nonEnglishCharsPattern)
                           .OfType<Match>()
                           .Select(m => m.Groups[0].Value)
                           .ToList();

var finalParts = englishParts;
finalParts.AddRange(nonEnglishParts);

Console.WriteLine(string.Join(",", finalParts.ToArray()));
```

Which gives us:

```
Hyundai Motor Company,Some other English words,현대자동차,现代
```

Share Improve this answer Follow

answered Aug 10, 2017 at 17:54

Nikhil Vartak
4,942 3 24 32