



ChatGPT와 익명 간의 대화의 사본입니다.
콘텐츠 신고

한국인 이름을 생성하는 오픈소스 함수를 발견했어. 상세 코드는 아래와 같아.

```
python
def generateCustom(trainedDataMatrix):
    """
    랜덤한 한국 이름을 생성한다.

    :param isMale: 남자 또는 여자의 성별 여부.
    """

    def ensure(n):
        return n or 0

    # 요소들의 가중치에 비례한 확률로 랜덤 뽑기
    def pick(count, item):
        sum = 0
        selected = 0

        for i in range(count):
            sum += item(i)

        pivot = random.random() * sum

        for i in range(count):
            pivot -= item(i)
            if pivot <= 0:
                selected = i
                break

        return selected

    # 랜덤으로 음절 생성
    def pickSyllable(set):
        try:
            choseong = pick(19, lambda n: ensure(tr
        except Exception:
            choseong = 0

        try:
            jungseong = pick(21, lambda n: ensure(tr
        except Exception:
            jungseong = 0

        try:
            jongseong = pick(28, lambda n: ensure(tr
```

```

except Exception:
    jongseong = 0

    return constructFromJamoIndex([choseong, jur

def pickLastName():
    lastNameIndex = pick(len(trainedData.lastName

    return chr(trainedData.lastName[lastNameInc

    return pickLastName() + pickSyllable(0) + pickSy

```

이걸 활용해서 반대로 주어진 텍스트가 한국인 이름인지 아닌지 판별하는 함수를 만들순 없을까? 아래는 아마 이름을 생성할 때 샘플링 가중치를 정리한 파일인것 같아.

python



```

lastNameNames = [
    576, 7028, 4117, 8540, 7189, 21, 7280, 6948, 707
    6692, 5404, 5856, 396, 10857, 6472, 5537, 7172,
    6545, 5520, 4144, 4145, 10696, 1192, 5868, 1400,
    7420, 6832, 364, 4092, 3416, 1176, 7620, 7616, 6
    4137, 245, 10756, 10600, 4288, 6652, 6636, 8596,
    5412, 3528, 568, 6640, 6916, 10332, 3717, 560, 4
    6693, 6945, 7032, 3577, 7196, 3752, 1192, 9409,
]

lastNameFrequency = [
    1068, 730, 419, 233, 239, 126, 145, 102, 101, 10
    76, 75, 98, 70, 69, 68, 68, 74, 55, 94, 47, 46,
    27, 31, 23, 20, 19, 19, 23, 19, 19, 17, 16, 16,
    12, 9, 8, 8, 13, 6, 8, 6, 5, 5, 4, 4, 4, 3, 3, 3
    2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1
]

firstNames = [[], []]

# 남자 이름
firstNames[0] = [
    [
        905, -1, 45, 839, -1, 17, 665, 309, -1,
        405, -1, 334, 19, 1382
    ],
    [
        77, -3, 150, -1, 237, 1, 5, 94, -2, 9, 2
        150, -21, 40, 1, -6, 2, -4, 1, -4, 1, -2
        650, -4, 20, -4, 1, -23, 4, 7, -4, 2, -5
        84, -1, 5, -4, 44, -6, 525, 4, 3, -2

```

-4, 4, -6, 1, -21, 401, 5, -2, 786, 111,
500, -1, 92, -21, 3, -1, 8, -1, 7, 1, 41
223, 163, -1, 1, 232, 72, 36, 161, 43, 5
252, -4, 717, -6, 632, -21, 307, 21, -2,
-22, 333, -6, 1, -18, 7, -13, 12, 163, 3
16, 202, 32, -1, 7, 57, 24, -2, 17, 1, 3

],

[

89, 27, -2, 320, -3, 2, 1, -6, 35, 1, -3
1, -12, 1, -19, 9, -34, 37, 130, -2, 260
941, -6, 152, -27, 5, 64, -2, 504, -3, 3
-6, 77, -27, 302, 1, -2, 14, -3, 2, -12,
67, -34, 7, -27, 88, -16, 1, -3, 166, -6
1, -12, 64, -10, 175, -51, 19, -27, 187,
1, -2, 97, -3, 2, -12, 508, -6, 149, -27
34, -7, 6, 1

],

[

285, 11, -2, 225, -3, 13, -7, 1, 1, -3,
1, -6, 278, 18, -2, 3, -11, 1, -4, 539,
4, -6, 24, -3, 558, -4, 1, -11, 82, -6,
1, -7, 83, -4, 170, -34, 345, 113, -2, 1
1432, -34, 415, 28, -2, 574, -3, 17, -7,
1106, -3, 855, -12, 1, -3, 691, -34, 36,
140, -34, 321, 13, -34, 12, -12, 7, -6,
2, 1, -3, 254

]

],

[

[

870, -1, 35, 81, -1, 120, 585, 309, -1,

],

[

7, -3, 45, -1, 69, -1, 19, 45, -2, 12, 6
148, -21, 14, -5, 8, -1, 4, -4, 3, -4, 6
52, -4, 3, -28, 16, 13, 1, -3, 20, -1, 2
32, 6, -5, 25, -1, 47, -4, 40, -6, 467,
27, -4, 3, -6, 144, -21, 81, 4, -2, 918,
54, -1, 96, -21, 19, -1, 6, -1, 29, -1,
113, 938, 355, -1, 1, 167, 32, 10, 155,
62, -4, 754, -6, 348, -21, 193, 6, -2, 1
13, 108, -25, 6, -1, 1, -3, 22, -7, 18,
1180, -1, 594, 330, -1, 11, 11, 471, -2,

],

[

77, 54, -2, 263, -11, 24, -4, 100, -6, 3
7, -34, 40, 414, -2, 223, -3, 109, -7, 1
45, -7, 1, -8, 6, -11, 316, -2, 783, -3,
606, -34, 654, 37, -2, 34, -3, 13, -12,
24, -34, 11, -27, 46, -20, 119, -6, 1139
10, 12, 125, 10, 207, 51, 20, 27, 26

```

10, -12, 133, -10, 307, -31, 20, -27, 20
1, -2, 113, -3, 7, -7, 7, 1, -3, 54, -6,
1024, -3, 105, -7, 20, 4, -5, 5
],
[
454, 23, -2, 278, -3, 30, -7, 13, -4, 72
-7, 14, -4, 6, -6, 26, 17, -2, 6, -11, 4
-2, 10, -3, 20, -7, 25, 4, -3, 12, -6, 5
-6, 26, 18, -2, 139, -3, 4, -7, 110, -4,
-2, 99, -3, 16, -7, 3, 117, -3, 427, -34
129, -7, 2, 82, -3, 569, -6, 405, 3, -2,
-3, 172, -3, 102, -12, 35, -6, 1, -27, 7
-12, 6, -6, 872, 341, -2, 1585, -11, 6,
]
]
]

```



여자 이름

```

firstNames[1] = [
[
[
33, -1, 14, 43, -1, 1, 65, 15, -1, 215,
],
[
20, -5, 1, -1, 2, -8, 6, 3, -1, 1, -21,
1, -8, 1, -4, 2, -6, 62, -6, 1, -1, 14,
-2, 38, -4, 52, -4, 15, -1, 17, -21, 9,
-4, 1, 1, -2, 69, 26, 3, 7, 3, 7, -2, 1
-21, 2, 21, -2, 1, -3, 3, -2, 2, -31, 7
25, 33, 3, -3, 4, -3, 1, -2, 17
],
[
94, -3, 7, -11, 1, -4, 3, -6, 40, -83, 5
27, -6, 21, -27, 2, -3, 40, -16, 7, -6,
62, 2, -27, 4, -27, 86, -3, 3, -27, 1, -
-27, 28, -3, 1, -7, 1, -4, 14, -6, 20, -
],
[
28, -3, 2, -11, 1, -4, 2, -34, 13, -15,
-34, 1, -27, 11, -3, 54, -23, 14, -20, 1
2, -12, 29, -34, 109, -3, 66, -16, 5, -6
-34, 26, -3, 2, -16, 1, -34, 7, -27, 1,
]
],
[
[
↓ -1, 17, 3, -1, 48, 47, 38, -1, 63, -
],
[
-4, 1, -1, 32, -13, 1, -21, 15, -17, 2,
4, -11, 1, -1, 34, -20, 47, -20, 38, -25

```



```

1, -1, 4, -21, 20, 1, 1, -1, 1, -1, 149,
27, 59, -1, 21, -1, 5, -2, 24, -8, 30, -
1, -3, 71, 10, 1, 1, -2, 2, -5, 1, 29
],
[
46, -3, 5, -11, 1, -4, 1, -6, 10, -48, 2
25, -38, 159, -3, 1, -12, 96, -6, 16, -2
1, -83, 2, -27, 48, -3, 4, -27, 50, -79,
59, -3, 3, -7, 1, 1, -10, 29, -27, 53, -
],
[
1, -3, 1, -3, 1, -12, 31, -34, 15, -7, 2
17, -11, 17, -4, 4, -6, 6, -3, 41, -23,
42, 2, -2, 10, -3, 8, -12, 1, -34, 40, -
60, -6, 64, -3, 53, -16, 24, -34, 3, -11
]
]
]


```



18초 동안 생각 ▾

Yes, it's possible to create a function that determines whether a given text is a Korean name by leveraging the provided name generation code and data. Here's how you can approach it:

1. Understand the Name Generation Logic:

- The `generateCustom` function creates a Korean name by selecting a last name and two syllables for the first name based on  weighted probabilities.
- The last names and first name syllables are stored in arrays with associated frequencies, indicating how common each is.

2. Reverse the Process:

- To check if a given text is a plausible Korean name generated by this function, you can reverse-engineer the process:
 - Extract the Last Name:** Check if the beginning of the text matches any of the last names in the `lastNames` array.
 - Validate First Name Syllables:** Verify that the remaining characters correspond to valid syllables generated from the `firstNames` data.

3. Implement Probability Thresholds:

- Calculate the likelihood that the given name could be generated by the function.
- Set a probability threshold to decide whether the name is considered valid.

Below is an example of how you might implement such a function:

```
python
```

 코드 복사

```
import random
```

```
# Assuming lastNames, lastNameFrequency, and firstNames are defined as per your data
```

```
# Helper function to construct a syllable from Jamo indices
```

```
def constructFromJamoIndex(indices):
```

```
    choseong_base = 0x1100
```

```
    jungseong_base = 0x1161
```

```
    jongseong_base = 0x11A7 # Note: 0x11A7 is one less than the actual base to account for the null character
```

```
    choseong, jungseong, jongseong = indices
```

```
    syllable = chr(0xAC00 + (choseong * 21 * 28) + (jungseong * 28) + jongseong)
```

```
    return syllable
```

```
# Function to generate all possible syllables based on firstNames data
```

```
def generate_valid_syllables(firstNamesData):
```

```
    valid_syllables = set()
```

```
    for genderData in firstNamesData:
```

```
        for setData in genderData:
```

```
            choseong_weights, jungseong_weights, jongseong_weights, _ = setData
```

```
            for i, cw in enumerate(choseong_weights):
```

```
                if cw > 0:
```

```
                    for j, jw in enumerate(jungseong_weights):
```

```
                        if jw > 0:
```

```
                            for k, kw in enumerate(jongseong_weights):
```

```
                                if kw > 0:
```

```
                                    syllable = constructFromJamoIndex([i, j, k])
```

```
                                    valid_syllables.add(syllable)
```

```
    return valid_syllables
```

```
# Function to check if the first name is valid
```

```
def is_valid_first_name(first_name, valid_syllables):
```

```
    # Split the first name into syllables
```

```
    syllables = list(first_name)
```

```
    for syllable in syllables:
```

```
        if syllable not in valid_syllables:
```

```
            return False
```

```
    return True
```

```
# Main function to determine if a text is a Korean name
```

```
def is_korean_name(name, lastNames, lastNameFrequency, firstNames):
```

```
    # Normalize the name by removing spaces
```

```
    name = name.strip()
```

```
    # Generate a mapping of last name codes to characters
```

```
    last_name_chars = [chr(code + 0xAC00) for code in lastNames]
```

```
    last_name_set = set(last_name_chars)
```

```
    # Generate valid syllables from the firstNames data
```

```
    valid_syllables = generate_valid_syllables(firstNames)
```


```

# Try possible splits for the last name (considering one or two syllable last n
max_last_name_length = 2 # Adjust based on known last name lengths
for i in range(1, min(max_last_name_length + 1, len(name))):
    last_name_candidate = name[:i]
    if last_name_candidate in last_name_set:
        first_name_candidate = name[i:]
        if len(first_name_candidate) >= 1 and is_valid_first_name(first_name_ca
            return True
    return False

# Example usage
name = "김민수" # Replace with the name you want to check
if is_korean_name(name, lastNames, lastNameFrequency, firstNames):
    print(f"{name} is likely a Korean name.")
else:
    print(f"{name} is unlikely to be a Korean name.")

```

Explanation of the Code:

- **constructFromJamoIndex:** Reconstructs a syllable from its constituent Jamo indices.
- **generate_valid_syllables:** Generates all possible valid syllables based on the `firstNames` data. This is a computationally intensive task but necessary for accurate validation.
- **is_valid_first_name:** Checks if the first  ne consists only of valid syllables.
- **is_korean_name:** Combines the above functions to check if the given name is a valid Korean name.

Considerations:

- **Performance:** Generating all valid syllables might be slow due to the large number of combinations. You may want to optimize this by caching results or narrowing down the possibilities.
- **Accuracy:** This method relies on the data provided and may not account for all Korean names, especially rare or non-standard ones.
- **Normalization:** Ensure that the input name is properly normalized (e.g., Unicode normalization) to match the generated syllables.

Alternative Approaches:

- **Regular Expressions:** Use regular expressions to match the patterns of Korean names, though this might be less accurate.
- **Machine Learning:** Train a classifier using a dataset of Korean names and non-names to distinguish between them.
- **External Libraries:** Utilize libraries like `py-hanspell` for Korean text processing, which might offer functions for name validation.

Example with Regular Expressions:

python

코드 복사

```
import re

def is_korean_name_regex(name):
    # Korean characters range from \uAC00 to \uD7A3
    korean_char = ' [\uAC00-\uD7A3]'
    # Common pattern: one syllable last name and two syllable first name
    pattern = f'^({korean_char}){{1,2}}({korean_char}){{1,2}}$'
    if re.match(pattern, name):
        return True
    return False

# Example usage
name = "박지성"
if is_korean_name_regex(name):
    print(f"{name} matches the Korean name pattern.")
else:
    print(f"{name} does not match the Korean name pattern.")
```

Limitations of Regular Expressions:

- This method checks only the structural pattern and doesn't validate against actual name data.
- Less accurate for names with unusual lengths or characters.

Final Thoughts:

Creating a function to validate Korean names using the provided data is feasible but requires careful handling of the data structures and a good understanding of Korean linguistics. The above code provides a starting point, but you may need to refine it based on your specific requirements and test it thoroughly with various names.

이 이후의 메시지는 사용자에게만 나타납니다

대화를 계속하려면 메시지를 보내세요.



ChatGPT는 실수를 할 수 있습니다. 중요한 정보를 확인하세요.

?