Exploring Linguistics

Understanding & Manipulating Language On iOS

David Fox

iOS Engineer @ Cookpad

My Daily Workflow

Copy/Paste

“English”

Romaji

My Challenge

かんじ　がくしゅう　　よういい
いかにして漢字学習を容易にするか

How can I make learning kanji easier?

news.yahoo.co.jp

Y!ニュース

ログイン | ID新規取得

次期iPhoneにトリプルカメラ? 4/11(水) 15:34

アップル次期iPhoneで「トリプルカメラ」搭載か、5

Main Challenges

1. How did I determine where each kanji begins and ends with no white space to split on?
2. How did I extract each kanji or group of kanji's hiragana?
3. Each kanji can have multiple readings. How did the app know how to interpret each one?

What We'll Cover Today

- **Why are understanding linguistics as a developer important?**
- **Basic usage of iOS's linguistics APIs**
- **Transcribing and transforming tokens**
- **Natural Language Processing possibilities in iOS**

What Exactly Is A “Word”?

“このストリングの中にいくつの言葉がありますか？”

```
func wordCount(in input: String) -> Int {
    return input.split(separator: " ").count
}
```

1 ✗

A More Generic Term Is Required When Referring To The Elements Which Make Up A String…

Tokens

CFStringTokenizer

- A foundation level class for tokenizing strings
- Takes into account locale
- Understands the concept of “context” within strings
- Can also transform tokens and detect their language!

Basic Tokenizer Usage

```
let input = "Let's tokenize a string!"

var tokens = [CFString]()

let allocator = kCFAllocatorDefault
let cfString = input as CFString
let tokenizer = CFStringTokenizerCreate(allocator,
                                        cfString,
                                        CFRangeMake(0, input.count),
                                        kCFStringTokenizerUnitWord,
                                        CFLocaleCopyCurrent())

// Move to the first token in the string
var result = CFStringTokenizerAdvanceToNextToken(tokenizer)

while !result.isEmpty {

    let currentRange = CFStringTokenizerGetCurrentTokenRange(tokenizer)
    let currentToken = CFStringCreateWithSubstring(allocator, cfString,
currentRange)
    tokens.append(currentToken!)

    // Advance to the next token
    result = CFStringTokenizerAdvanceToNextToken(tokenizer)
}

// tokens = ["Let's", "tokenize", "a", "string!"]
```

Basic Tokenizer Usage

```
let input = "Let's tokenize a string!"

var tokens = [CFString]()

let allocator = kCFAllocatorDefault
let cfString = input as CFString
let tokenizer = CFStringTokenizerCreate(allocator,
                                        cfString,
                                        CFRangeMake(0, input.count),
                                        kCFStringTokenizerUnitWord,
                                        CFLocaleCopyCurrent())

// Move to the first token in the string
var result = CFStringTokenizerAdvanceToNextToken(tokenizer)

while !result.isEmpty {

    let currentRange = CFStringTokenizerGetCurrentTokenRange(tokenizer)
    let currentToken = CFStringCreateWithSubstring(allocator, cfString,
currentRange)
    tokens.append(currentToken)

    // Advance to the next token
    result = CFStringTokenizerAdvanceToNextToken(tokenizer)
}

// tokens = ["Let's", "tokenize", "a", "string!"]
```

English: 28 tokens

Here is some sample text!

iOS's tokenizer options really are pretty impressive. There's lots of options and transforms to play with.

Let's have a look at a few…

Japanese: 40 tokens

こんにちはみんなさま。このAPIは言葉し文し段落が分かりやすいです！それからスイフトより早い！

結構多いコードを書かなければなりませんけど便利よ。一緒にCFを使って見よ

We can now identify individual tokens!

But what about manipulating them?

CFStringTransform

A foundation-level API for manipulating strings

CFStringTransform
Basic Usage

```
let toTransform = NSMutableString(string: "Let's transform some text!")

CFStringTransform(toTransform,
                  nil,
                  kCFStringTransformLatinKatakana,
                  false)
```

Transcribing Tokens

Extracting normalised data from strings

Transcribing Tokens

“日本語を話せますか？”

```
CFStringTokenizerCopyCurrentTokenAttribute(tokenizer,
    kCFStringTokenizerAttributeLatinTranscription)
```

```
["nippon", "go", "wo", "hanase", "masu", "ka"]
```

Latin Transcriptions

+

CFStringTransform

🤔

“日本語を話せますか？”

```
CFStringTokenizerCopyCurrentTokenAttribute(tokenizer,
  kCFStringTokenizerAttributeLatinTranscription)
```

["nippon", "go", "wo", "hanase", "masu", "ka"]

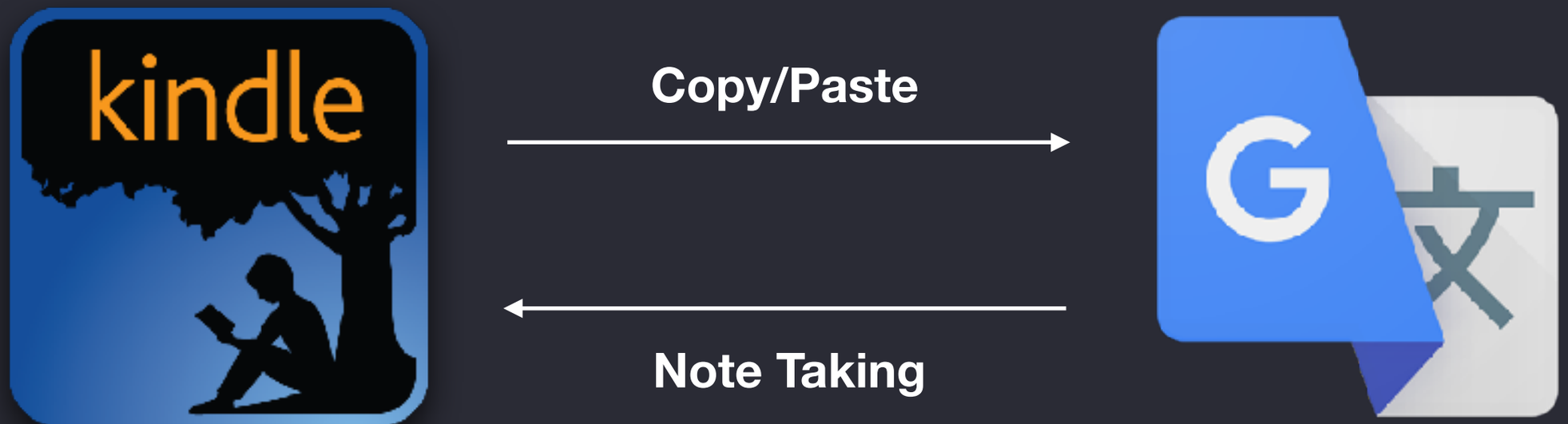
```
CFStringTransform(toTransform,
                  nil,
                  kCFStringTransformLatinKatakana,
                  false)
```

["にっぽん", "ご", "を", "はなせ", "ます", "か"]

Understanding Language

Natural Language Processing Possibilities On iOS

A Common Workflow For Studying English



“Required Vocabulary” Prep Method

Giving a passage’s list of nouns and verbs along with their definition in list form.

Vocabulary / 単語

Mother	母
Pig	豚
Lazy	怠惰な
World	世界
House	家
Dance	踊る

Once upon a time there was an old mother pig who had three little pigs and not enough food to feed them. So when they were old enough, she sent them out into the world to seek their fortunes.

The first little pig was very lazy. He didn't want to work at all and he built his house out of straw. The second little pig worked a little bit harder but he was somewhat lazy too and he built his house out of sticks. Then, they sang and danced and played together the rest of the day.

…

Can NLP be used to automatically generate this lesson format from any given text?

Demo

Main Challenges

- Detecting individual tokens
- Determining their types within context

NSLinguisticTagger

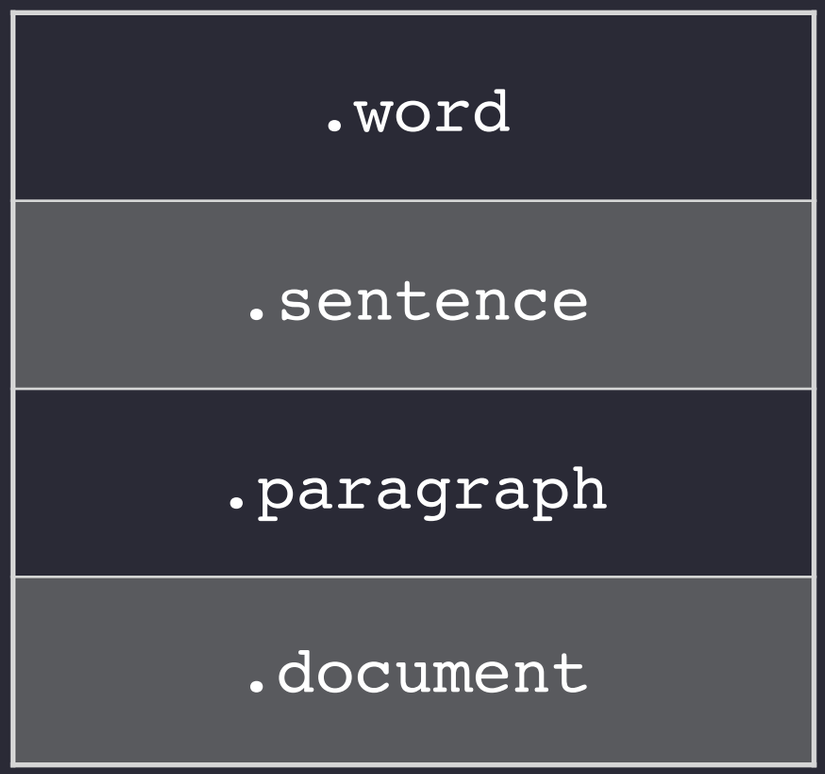
- **A high-level API for NLP on iOS**
- **Can detect, extract and inspect granular string data**

Main Components

```
NSLinguisticTaggerUnit
NSLinguisticTagScheme
NSLinguisticTag
```

NSLinguisticTaggerUnit

Defines the token to enumerate over



NSLinguisticTagScheme

Defines the *type* of data you want to extract from each token

Option	Description	Example
`.tokenType`	Returns each token's base type	Word, sentence, paragraph…
`.nameType`	Detects people, place & landmark names	Tokyo, America, John…
`.lemma`	Returns word stems	"Drinking" -> "Drink"
`.language`	Returns the predominant language	English, Japanese…
`.lexicalClass`	Returns each token's class type	Nouns, adjectives, verbs…

NSLinguisticTag

The result of all calls to NSLinguisticTagger

`.noun`	`.preposition`
`.verb`	`.idiom`
`.adjective`	`.interjection`
`.adverb`	…

Basic Syntax

```
let tagger = NSLinguisticTagger(tagSchemes: [NSLinguisticTagScheme],
                                options: 0)
tagger.string = ""

tagger.enumerateTags(in: range,
                     unit: NSLinguisticTaggerUnit,
                     scheme: NSLinguisticTagScheme,
                     options: options) { tag, tokenRange, stop in

}
```

Retrieving nouns...

```swift
let input = "Hello, my name's David and I'm doing a presentation on NLP!"

let tagger = NSLinguisticTagger(tagSchemes: [.lexicalClass], options: 0)
tagger.string = input

let options: NSLinguisticTagger.Options = [.omitPunctuation, .omitWhitespace]
let range = NSRange(location: 0, length: input.utf16.count)

var tokens = Set<String>()
tagger.enumerateTags(in: range,
                     unit: .word,
                     scheme: .lexicalClass,
                     options: options) { tag, tokenRange, stop in

                        if case .noun = tag! {
                            let token = (input as NSString).substring(with: tokenRange)
                            tokens.insert(token)
                        }
}

// tokens = ["name", "David", "presentation", "NLP"]
```

Thanks!

github.com/davefoxy/linguistics-prez