Natural Language Processing and your Apps

Session 208

Vivek Kumar Rangarajan Sridhar, Software Engineering Manager
Doug Davidson, Senior Engineer
Goal
Goal
Goal

Natural language input
- typed text
- recognized handwriting
- transcribed speech
Goal

Natural language output
- typed text
- recognized handwriting
- transcribed speech
Goal

NLP

Natural language text
- typed
- recognized handwriting
- transcribed speech

Confer intelligence
NLP APIs
NLP APIs
NLP APIs
Natural Language Processing
Natural Language Processing

Natural language text
typed
recognized handwriting
transcribed speech
Natural Language Processing

Natural language text
- typed
- recognized handwriting
- transcribed speech
Natural Language Processing

Natural language text
- typed
- recognized handwriting
- transcribed speech

Processing → Extract information
Confer intelligence
Natural Language Processing

Natural language text
- typed
- recognized handwriting
- transcribed speech

Processing

Extract information
Confer intelligence
Natural Language Processing

Natural language text
- typed
- recognized handwriting
- transcribed speech

Confer intelligence
Natural Language Processing

Language identification

Detect language

Natural language text
- typed
- recognized handwriting
- transcribed speech
Welcome to our talk on Natural Language Processing

Natural language text
typed
recognized handwriting
transcribed speech

Language identification

Detect language

欢迎参加我们关于自然语言处理的讲座
Bienvenidos a nuestra plática sobre Procesamiento de Lenguaje Natural
Willkommen zu unserem Vortrag über Verarbeitung natürlicher Sprache
Natural Language Processing

Language identification

Tokenization
- Word
- Sentence
- Paragraph

Natural language text
- typed
- recognized handwriting
- transcribed speech

Tokenize text
Mr. Tim Cook presided over the earnings report of Apple Inc. on Tuesday.
星期二，蒂姆·库克先生主持了苹果公司的财报会议。
星期二，蒂姆·库克先生主持了苹果公司的财报会议。
Natural Language Processing

Natural language text
- typed
- recognized handwriting
- transcribed speech

Language identification
Tokenization
- Word
- Sentence
- Paragraph

Part of speech

Assign parts of speech
Mr. Tim Cook presided over the earnings report of Apple Inc. on Tuesday.
Mr. Tim Cook presided over the earnings report of Apple. The stock was up 3% after hours.
Mr. Tim Cook presided over the earnings report of Apple. The stock was up 3% after hours.
Mr. Tim Cook presided over the earnings report of Apple Inc. on Tuesday.
Mr. Tim Cook presided over the earnings report of Apple Inc. on Tuesday.
Natural Language Processing

Natural language text
- typed
- recognized handwriting
- transcribed speech

Language identification
Tokenization
- Word
- Sentence
- Paragraph
Part of speech
Lemmatization
Named entity recognition

Confer intelligence
Natural Language Processing

Natural language text
- typed
- recognized handwriting
- transcribed speech

Language identification
Tokenization
- Word
- Sentence
- Paragraph
Part of speech
Lemmatization
Named entity recognition

Confer intelligence

Linguistics
Natural Language Processing

Natural language text
- typed
- recognized handwriting
- transcribed speech

Confer intelligence

Language identification

Tokenization
- Word
- Sentence
- Paragraph

Part of speech

Lemmatization

Named entity recognition

Linguistics

Machine learning
Natural Language Processing

NLP APIs

Language identification
Tokenization
- Word
- Sentence
- Paragraph
Part of speech
Lemmatization
Named entity recognition

Natural language text
- typed
- recognized handwriting
- transcribed speech

Confer intelligence

Linguistics
Machine learning
Enough!
Tell me how to use them
NLP APIs
NLP APIs

NSLinguisticTagger
NSLinguisticTagger

Class in foundation

Segment and tag text

Linguistic tasks: tagSchemes
NSLinguisticTagger

Class in foundation
Segment and tag text
Linguistic tasks : tagSchemes
NSLinguisticTagger
NSLinguisticTagger

Tagging units
public enum NSLinguisticTaggerUnit : Int {
    case word
    case sentence
    case paragraph
    case document
}
Tagging units

```swift
public enum NSLinguisticTaggerUnit : Int {
    case word
    case sentence
    case paragraph
    case document
}
```
Tagging units

```swift
public enum NSLinguisticTaggerUnit : Int {
    case word
    case sentence
    case paragraph
    case document
}
```
Tagging units

```swift
public enum NSLinguisticTaggerUnit : Int {
    case word
    case sentence
    case paragraph
    case document
}
```
public enum NSLinguisticTaggerUnit : Int {
    case word
    case sentence
    case paragraph
    case document
}
NSLinguisticTagger

Tagging units

```swift
public enum NSLinguisticTaggerUnit : Int {
    case word
    case sentence
    case paragraph
    case document
}
```

Units and schemes
public enum NSLinguisticTaggerUnit : Int {
    case word
    case sentence
    case paragraph
    case document
}

class func availableTagSchemes (for unit: NSLinguisticTaggerUnit, language: String) -> [NSLinguisticTagScheme]
NSLinguisticTagger

Tagging units

```swift
public enum NSLinguisticTaggerUnit : Int {
    case word
    case sentence
    case paragraph
    case document
}
```

Units and schemes

```swift
class func availableTagSchemes (for unit: NSLinguisticTaggerUnit, language: String) -> [NSLinguisticTagScheme]
```
NSLinguisticTagger
NSLinguisticTagger

dominantLanguage
NSLinguisticTagger

dominantLanguage

Swift 4: named types for tags and tagSchemes
NSLinguisticTagger

dominantLanguage

Swift 4: named types for tags and tagSchemes

Improved performance

Higher accuracy

Additional language support
Winnow and Whisk
Winnow and Whisk
Winnow and Whisk
Winnow
Tag photos with descriptions
Winnow
Tag photos with descriptions
We hiked to the top and then ran down. Lizzie won!

Kids partied with all the goodies...

Die Kleinen haben friedlich zusammen gespielt.

July 4th, 2016... summer parties at the Holden's begin!
Winnow
Search for pictures
Winnow
Search for pictures

Hike
Winnow
Search for pictures

Hike ➔ No results
Winnow
Improve search experience using NLP
Winnow

Improve search experience using NLP

Hike
Winnow
Improve search experience using NLP

Hike

We hiked to the top and then ran down. Lizzie won!

Great hikes make great pics!

Holden family loves hiking and also posing for pictures

The hiking boys!
Winnow
Winnow
Winnow

NLP

Language identification
Winnow

NLP

<table>
<thead>
<tr>
<th>Language identification</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Tokenization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word</td>
</tr>
</tbody>
</table>

[Diagram showing NLP process with sub-processes like Language identification and Tokenization with Word, Sentence, and Paragraph levels]
Winnow

NLP

<table>
<thead>
<tr>
<th>Language identification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tokenization</td>
</tr>
<tr>
<td>Word</td>
</tr>
<tr>
<td>Sentence</td>
</tr>
<tr>
<td>Paragraph</td>
</tr>
<tr>
<td>Part of speech</td>
</tr>
</tbody>
</table>
Winnow

NLP

- Language identification
- Tokenization
  - Word
  - Sentence
  - Paragraph
- Part of speech
- Lemmatization

Example text: "Nous marchons le long du chemin jusqu'au sommet!"
Language Identification

```swift
import Foundation

let tagger = NSLinguisticTagger(tagSchemes: [.language], options: 0)
tagger.string = "Die Kleinen haben friedlich zusammen gespielt."

let language = tagger.dominantLanguage
```
import Foundation

let tagger = NSLinguisticTagger(tagSchemes: [.language], options: 0)
tagger.string = "Die Kleinen haben friedlich zusammen gespielt."

let language = tagger.dominantLanguage
import Foundation

let tagger = NSLinguisticTagger(tagSchemes: [.language], options: 0)

tagger.string = "Die Kleinen haben friedlich zusammen gespielt."

let language = tagger.dominantLanguage
import Foundation

let tagger = NSLinguisticTagger(tagSchemes: [.language], options: 0)
tagger.string = "Die Kleinen haben friedlich zusammen gespielt."

let language = tagger.dominantLanguage
import Foundation

let tagger = NSLinguisticTagger(tagSchemes: [.language], options: 0)

tagger.string = "Die Kleinen haben friedlich zusammen gespielt."

let language = tagger.dominantLanguage
import Foundation

let tagger = NSLinguisticTagger(tagSchemes: [.tokenType], options: 0)

let text = "NSLinguisticTagger provides text processing APIs.\nNSLinguisticTagger 是苹果的文字处理平台。"

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

tagger.enumerateTags(in: range, unit: .word, scheme: .tokenType, options: options) {
    tag, tokenRange, stop in
    let token = (text as NSString).substring(with: tokenRange)
    // Do something with each token
}
import Foundation

let tagger = NSLinguisticTagger(tagSchemes: [.tokenType], options: 0)

let text = "NSLinguisticTagger provides text processing APIs.\nNSLinguisticTagger 是苹果的文字处理平台。"

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

tagger.enumerateTags(in: range, unit: .word, scheme: .tokenType, options: options) {
    tag, tokenRange, stop in
        let token = (text as NSString).substring(with: tokenRange)
    // Do something with each token
}
import Foundation

let tagger = NSLinguisticTagger(tagSchemes: [.tokenType], options: 0)

let text = "NSLinguisticTagger provides text processing APIs.
NSLinguisticTagger 是苹果的文字处理平台。"

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

tagger.enumerateTags(in: range, unit: .word, scheme: .tokenType, options: options) {
tag, tokenRange, stop in
    let token = (text as NSString).substring(with: tokenRange)
    // Do something with each token
}
import Foundation

let tagger = NSLinguisticTagger(tagSchemes: [.tokenType], options: 0)

let text = "NSLinguisticTagger provides text processing APIs.\nNSLinguisticTagger 是苹果的文字处理平台。"

tagger.string = text

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

tagger.enumerateTags(in: range, unit: .word, scheme: .tokenType, options: options) {
tag, tokenRange, stop in
    let token = (text as NSString).substring(with: tokenRange)
    // Do something with each token
}
import Foundation

let tagger = NSLinguisticTagger(tagSchemes: [.tokenType], options: 0)

let text = "NSLinguisticTagger provides text processing APIs.
NSLinguisticTagger 是苹果的文字处理平台。"

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

tagger.enumerateTags(in: range, unit: .word, scheme: .tokenType, options: options) {
tag, tokenRange, stop in
    let token = (text as NSString).substring(with: tokenRange)
    // Do something with each token
}
Tokenization

```swift
import Foundation

let tagger = NSLinguisticTagger(tagSchemes: [.tokenType], options: 0)

let text = "NSLinguisticTagger provides text processing APIs.\nNSLinguisticTagger 是苹果的文字处理平台。"

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

tagger.enumerateTags(in: range, unit: .word, scheme: .tokenType, options: options) { tag, tokenRange, stop in
    let token = (text as NSString).substring(with: tokenRange)
    // Do something with each token
}
```
import Foundation

let tagger = NSLinguisticTagger(tagSchemes: [.tokenType], options: 0)

let text = "NSLinguisticTagger provides text processing APIs.\nNSLinguisticTagger 是苹果的文字处理平台。"

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

tagger.enumerateTags(in: range, unit: .word, scheme: .tokenType, options: options) {
tag, tokenRange, stop in
    let token = (text as NSString).substring(with: tokenRange)
    // Do something with each token
}
import Foundation

let tagger = NSLinguisticTagger(tagSchemes: [.lemma], options: 0)

let text = "Great hikes make great pics! Wonderful afternoon in Marin County."

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

tagger.enumerateTags(in: range, unit: .word, scheme: .lemma, options: options) {
tag, tokenRange, stop in
    if let lemma = tag?.rawValue {
        // Do something with each lemma
    }
}
import Foundation

let tagger = NSLinguisticTagger(tagSchemes:[.lemma], options: 0)

let text = "Great hikes make great pics! Wonderful afternoon in Marin County."

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

tagger.enumerateTags(in: range, unit: .word, scheme: .lemma, options: options) {
tag, tokenRange, stop in
    if let lemma = tag?.rawValue {
        // Do something with each lemma
    }
}
import Foundation

let tagger = NSLinguisticTagger(tagSchemes: [.lemma], options: 0)

let text = "Great hikes make great pics! Wonderful afternoon in Marin County."

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

tagger.enumerateTags(in: range, unit: .word, scheme: .lemma, options: options) {
tag, tokenRange, stop in
  if let lemma = tag?.rawValue {
    // Do something with each lemma
  }
}
import Foundation

let tagger = NSLinguisticTagger(tagSchemes: [.lemma], options: 0)

let text = "Great hikes make great pics! Wonderful afternoon in Marin County."

tagger.string = text

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

tagger.enumerateTags(in: range, unit: .word, scheme: .lemma, options: options) {
    tag, tokenRange, stop in
        if let lemma = tag?.rawValue {
            // Do something with each lemma
        }
    }
}
import Foundation

let tagger = NSLinguisticTagger(tagSchemes: [.lemma], options: 0)

let text = "Great hikes make great pics! Wonderful afternoon in Marin County."

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

tagger.enumerateTags(in: range, unit: .word, scheme: .lemma, options: options) { tag, tokenRange, stop in
    if let lemma = tag?.rawValue {
        // Do something with each lemma
    }
}
import Foundation

let tagger = NSLinguisticTagger(tagSchemes:[.lemma], options: 0)

let text = "Great hikes make great pics! Wonderful afternoon in Marin County."

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

tagger.enumerateTags(in: range, unit: .word, scheme: .lemma, options: options) {
    tag, tokenRange, stop in
        if let lemma = tag?.rawValue {
            // Do something with each lemma
        }
    }
import Foundation

let tagger = NSLinguisticTagger(tagSchemes: [.lemma], options: 0)

let text = "Great hikes make great pics! Wonderful afternoon in Marin County."

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

tagger.enumerateTags(in: range, unit: .word, scheme: .lemma, options: options) {
tag, tokenRange, stop in
    if let lemma = tag?.rawValue {
        // Do something with each lemma
    }
}
import Foundation

let tagger = NSLinguisticTagger(tagSchemes:[.lemma], options: 0)

let text = "Great hikes make great pics! Wonderful afternoon in Marin County."

tagger.string = text
let range = NSRange(location:0, length: text.utf16.count)
let options: NSLinguisticTagger.Options = [.omitPunctuation, .omitWhitespace]
tagger.enumerateTags(in: range, unit: .word, scheme: .lemma, options: options) {
tag, tokenRange, stop in
    if let lemma = tag?.rawValue {
        // Do something with each lemma
    }
}
Demo
Whisk
Collate social media feeds
Whisk
Collate social media feeds
Whisk
Collate social media feeds
Whisk
Organize feeds by People, Organization, and Location using NLP
Whisk

Organize feeds by People, Organization, and Location using NLP
Whisk
Organize feeds by People, Organization, and Location using NLP
Whisk
Whisk
Whisk
Whisk
Whisk

Language identification

Tokenization

Word  Sentence  Paragraph

Named entity recognition

NLP

Whisk

Twitter
Facebook
LinkedIn
Pinterest

Extract All Entities
import Foundation

let tagger = NSLinguisticTagger(tagSchemes: [.nameType], options: 0)

let text = "Tim Cook is the CEO of Apple Inc. which is located in Cupertino, California"

tagger.string = text

let range = NSRange(location:0, length: text.utf16.count)
let options: NSLinguisticTagger.Options = [.omitPunctuation, .omitWhitespace, .joinNames]
let tags: [NSLinguisticTag] = [.personalName, .placeName, .organizationName]

tagger.enumerateTags(in: range, unit: .word, scheme: .nameType, options: options) {
tag, tokenRange, stop in
    if let tag = tag, tags.contains(tag) {
        let name = (text as NSString).substring(with: tokenRange)
    }
}
Named Entity Recognition

```swift
import Foundation

let tagger = NSLinguisticTagger(tagSchemes: [.nameType], options: 0)

let text = "Tim Cook is the CEO of Apple Inc. which is located in Cupertino, California"

tagger.string = text

let range = NSRange(location:0, length: text.utf16.count)
let options: NSLinguisticTagger.Options = [.omitPunctuation, .omitWhitespace, .joinNames]
let tags: [NSLinguisticTag] = [.personalName, .placeName, .organizationName]

tagger.enumerateTags(in: range, unit: .word, scheme: .nameType, options: options) {
tag, tokenRange, stop in
    if let tag = tag, tags.contains(tag) {
        let name = (text as NSString).substring(with: tokenRange)
        print(name)
    }
}
```
import Foundation

let tagger = NSLinguisticTagger(tagSchemes: [.nameType], options: 0)

let text = "Tim Cook is the CEO of Apple Inc. which is located in Cupertino, California"

tagger.string = text

let range = NSRange(location:0, length: text.utf16.count)
let options: NSLinguisticTagger.Options = [.omitPunctuation, .omitWhitespace, .joinNames]
let tags: [NSLinguisticTag] = [.personalName, .placeName, .organizationName]

tagger.enumerateTags(in: range, unit: .word, scheme: .nameType, options: options) {
tag, tokenRange, stop in
  if let tag = tag, tags.contains(tag) {
    let name = (text as NSString).substring(with: tokenRange)
  }
}
import Foundation

let tagger = NSLinguisticTagger(tagSchemes: [.nameType], options: 0)

let text = "Tim Cook is the CEO of Apple Inc. which is located in Cupertino, California"

tagger.string = text

let range = NSRange(location:0, length: text.utf16.count)
let options: NSLinguisticTagger.Options = [.omitPunctuation, .omitWhitespace, .joinNames]
let tags: [NSLinguisticTag] = [.personalName, .placeName, .organizationName]

tagger.enumerateTags(in: range, unit: .word, scheme: .nameType, options: options) {
tag, tokenRange, stop in
    if let tag = tag, tags.contains(tag) {
        let name = (text as NSString).substring(with: tokenRange)
    }
}
import Foundation

let tagger = NSLinguisticTagger(tagSchemes: [.nameType], options: 0)

let text = "Tim Cook is the CEO of Apple Inc. which is located in Cupertino, California"

tagger.string = text

let range = NSRange(location: 0, length: text.utf16.count)

let options: NSLinguisticTagger.Options = [.omitPunctuation, .omitWhitespace, .joinNames]

let tags: [NSLinguisticTag] = [.personalName, .placeName, .organizationName]

tagger.enumerateTags(in: range, unit: .word, scheme: .nameType, options: options) { tag, tokenRange, stop in
    if let tag = tag, tags.contains(tag) {
        let name = (text as NSString).substring(with: tokenRange)
    }
}
import Foundation

let tagger = NSLinguisticTagger(tagSchemes: [.nameType], options: 0)

let text = "Tim Cook is the CEO of Apple Inc. which is located in Cupertino, California"

tagger.string = text

let range = NSRange(location:0, length: text.utf16.count)
let options: NSLinguisticTagger.Options = [.omitPunctuation, .omitWhitespace, .joinNames]
let tags: [NSLinguisticTag] = [.personalName, .placeName, .organizationName]

tagger.enumerateTags(in: range, unit: .word, scheme: .nameType, options: options) {
tag, tokenRange, stop in
    if let tag = tag, tags.contains(tag) {
        let name = (text as NSString).substring(with: tokenRange)
        
    }
}
import Foundation

let tagger = NSLinguisticTagger(tagSchemes: [.nameType], options: 0)

let text = "Tim Cook is the CEO of Apple Inc. which is located in Cupertino, California"

tagger.string = text

define range = NSRange(location:0, length: text.utf16.count)
define options: NSLinguisticTagger.Options = [.omitPunctuation, .omitWhitespace, .joinNames]
define tags: [NSLinguisticTag] = [.personalName, .placeName, .organizationName]

tagger.enumerateTags(in: range, unit: .word, scheme: .nameType, options: options) { tag, tokenRange, stop in
    if let tag = tag, tags.contains(tag) {
        let name = (text as NSString).substring(with: tokenRange)
    }
}
import Foundation

let tagger = NSLinguisticTagger(tagSchemes: [.nameType], options: 0)

let text = "Tim Cook is the CEO of Apple Inc. which is located in Cupertino, California"

tagger.string = text

let range = NSRange(location:0, length: text.utf16.count)
let options: NSLinguisticTagger.Options = [.omitPunctuation, .omitWhitespace, .joinNames]
let tags: [NSLinguisticTag] = [.personalName, .placeName, .organizationName]

tagger.enumerateTags(in: range, unit: .word, scheme: .nameType, options: options) {
tag, tokenRange, stop in
    if let tag = tag, tags.contains(tag) {
        let name = (text as NSString).substring(with: tokenRange)
    }
}
import Foundation

let tagger = NSLinguisticTagger(tagSchemes: [.nameType], options: 0)

let text = "Tim Cook is the CEO of Apple Inc. which is located in Cupertino, California"

tagger.string = text

let range = NSRange(location:0, length: text.utf16.count)
let options: NSLinguisticTagger.Options = [.omitPunctuation, .omitWhitespace, .joinNames]
let tags: [NSLinguisticTag] = [.personalName, .placeName, .organizationName]

tagger.enumerateTags(in: range, unit: .word, scheme: .nameType, options: options) { tag, tokenRange, stop in
  if let tag = tag, tags.contains(tag) {
    let name = (text as NSString).substring(with: tokenRange)
  }
}
Why should you use the NLP APIs?
Homogeneous Text Processing
Homogeneous Text Processing

NLP APIs

Consistent text processing
Consistent user experience

Natural language text
- typed
- recognized handwriting
- transcribed speech
Privacy
Privacy

On-device machine learning

User data stays on-device
Performance
Performance

Highly optimized on-device

Multi-threaded

Existing clients—significant speedup
Performance

Highly optimized on-device

Multi-threaded

Existing clients—significant speedup

Chinese tokenization is 30% faster on iOS
Performance

Highly optimized on-device

Multi-threaded

Existing clients—significant speedup

Named Entity Recognition 80% faster on iOS
Performance
Performance

Part of Speech Tagging

50000 tokens/sec

80000 tokens/sec

1 thread
Performance

Part of Speech Tagging
- 50,000 tokens/sec
- 80,000 tokens/sec

Named Entity Recognition
- 40,000 tokens/sec
- 65,000 tokens/sec

1 thread
Language Support
## Language Support

<table>
<thead>
<tr>
<th>Language Identification</th>
<th>29 scripts, 52 languages</th>
</tr>
</thead>
</table>

---
## Language Support

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language identification</td>
<td>29 scripts, 52 languages</td>
</tr>
<tr>
<td>Tokenization</td>
<td>All iOS/macOS system languages</td>
</tr>
</tbody>
</table>
## Language Support

<table>
<thead>
<tr>
<th>Language identification</th>
<th>29 scripts, 52 languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tokenization</td>
<td>All iOS/macOS system languages</td>
</tr>
<tr>
<td>Lemmatization</td>
<td></td>
</tr>
<tr>
<td></td>
<td>English</td>
</tr>
<tr>
<td></td>
<td>French</td>
</tr>
<tr>
<td></td>
<td>Italian</td>
</tr>
<tr>
<td></td>
<td>German</td>
</tr>
<tr>
<td></td>
<td>Spanish</td>
</tr>
<tr>
<td></td>
<td>Portuguese</td>
</tr>
<tr>
<td></td>
<td>Russian</td>
</tr>
<tr>
<td></td>
<td>Turkish</td>
</tr>
</tbody>
</table>
## Language Support

**Language identification**  
29 scripts, 52 languages

**Tokenization**  
All iOS/macOS system languages

**Lemmatization**  
- English
- French
- Italian
- German
- Spanish
- Portuguese
- Russian
- Turkish

**Part of speech**
## Language Support

<table>
<thead>
<tr>
<th>Feature</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language identification</td>
<td>29 scripts, 52 languages</td>
</tr>
<tr>
<td>Tokenization</td>
<td>All iOS/macOS system languages</td>
</tr>
<tr>
<td>Lemmatization</td>
<td>English, French, Italian, German, Spanish, Portuguese, Russian, Turkish</td>
</tr>
<tr>
<td>Part of speech</td>
<td></td>
</tr>
<tr>
<td>Named entity recognition</td>
<td></td>
</tr>
</tbody>
</table>
## Language Support

<table>
<thead>
<tr>
<th>Feature</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language identification</td>
<td>29 scripts, 52 languages</td>
</tr>
<tr>
<td>Tokenization</td>
<td>All iOS/macOS system languages</td>
</tr>
<tr>
<td>Lemmatization</td>
<td>English, French, Italian, German, Spanish, Portuguese, Russian, Turkish</td>
</tr>
<tr>
<td>Part of speech</td>
<td>English, French, Italian, German, Spanish, Portuguese, Russian, Turkish</td>
</tr>
<tr>
<td>Named entity recognition</td>
<td>English, French, Italian, German, Spanish, Portuguese, Russian, Turkish</td>
</tr>
</tbody>
</table>
Accuracy
Debugging Hints
Debugging Hints

Tags are NSLinguisticTagOtherWord
• Model not downloaded
Debugging Hints

Tags are NSLinguisticTagOtherWord

• Model not downloaded

Downloaded over-the-air
Debugging Hints

Tags are NSLinguisticTagOtherWord
- Model not downloaded

Explicitly set language if known

Downloaded over-the-air
Summary

NLP APIs—NSLinguisticTagger
• Support for new units
• Faster
• Higher accuracy
• More languages
More Information

https://developer.apple.com/wwdc17/208
# Related Sessions

<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>What's New in Cocoa Touch</td>
<td></td>
<td>WWDC 2017</td>
</tr>
<tr>
<td>Introducing Core ML</td>
<td></td>
<td>WWDC 2017</td>
</tr>
<tr>
<td>Vision Framework: Building on Core ML</td>
<td>Hall 2</td>
<td>Wednesday 3:10PM</td>
</tr>
<tr>
<td>Core ML in Depth</td>
<td>Hall 3</td>
<td>Thursday 9:00 AM</td>
</tr>
<tr>
<td>Accelerate and Sparse Solvers</td>
<td>Grand Ballroom A</td>
<td>Thursday 10:00 AM</td>
</tr>
<tr>
<td>Lab Name</td>
<td>Location</td>
<td>Time</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Core ML &amp; Natural Language Processing Lab</td>
<td>Technology Lab D</td>
<td>Thu 11:00AM-3:30PM</td>
</tr>
<tr>
<td>Vision Lab</td>
<td>Technology Lab A</td>
<td>Fri 1:50PM-4:00PM</td>
</tr>
<tr>
<td>Core ML &amp; Natural Language Processing Lab</td>
<td>Technology Lab D</td>
<td>Fri 1:50PM-4:00PM</td>
</tr>
<tr>
<td>Cocoa Lab</td>
<td>Technology Lab B</td>
<td>Fri 1:50PM-3:20PM</td>
</tr>
</tbody>
</table>