Introducing Natural Language

Session 713

Doug Davidson, Senior Software Engineer
Vivek Kumar Rangarajan Sridhar, Software Engineering Manager
Natural Language Input
Typed text
Transcribed speech
Recognized handwriting

Intelligence
Natural Language Input

NSLinguisticTagger

Language Identification
Tokenization
  Word  Sentence  Paragraph
Part of Speech
Lemmatization
Named Entity Recognition

Intelligence

Linguistics  Machine Learning
What’s new in NLP?
Natural Language Framework
Swift APIs for NLP
Swift APIs for NLP  Custom NLP models
Swift APIs for NLP  Custom NLP models  Performance
Swift APIs for NLP

Custom NLP models

Performance

Privacy
Natural Language
Design is coming together but needs some final tweaks.

It's getting late and I'm tired. Will pick it up tomorrow morning... good night!

John M
Design is coming together but needs some final tweaks.

It's getting late and I'm tired. Will pick it up tomorrow morning...good night!

John M
Design is coming together but needs some final tweaks. It's getting late and I'm tired. Will pick it up tomorrow morning... good night!

困死啦睡觉去了
Design is coming together but needs some final tweaks.

It's getting late and I'm tired. Will pick it up tomorrow morning...good night!

困死啦睡觉去了
Language Identification

```swift
import NaturalLanguage

let recognizer = NLLanguageRecognizer()

recognizer.processString("困死啦睡觉去了")

let lang = recognizer.dominantLanguage

let hypotheses = recognizer.languageHypotheses(withMaximum: 2)
```
import NaturalLanguage

let recognizer = NLLanguageRecognizer()

recognizer.processString("困死啦睡觉去了")

let lang = recognizer.dominantLanguage

let hypotheses  = recognizer.languageHypotheses(withMaximum:2)
Language Identification

```swift
import NaturalLanguage

let recognizer = NLLanguageRecognizer()

recognizer.processString("困死啦睡觉去了")

let lang = recognizer.dominantLanguage

let hypotheses = recognizer.languageHypotheses(withMaximum: 2)
```
import NaturalLanguage

let recognizer = NLLanguageRecognizer()

recognizer.processString("困死啦睡觉去了")

let lang = recognizer.dominantLanguage

let hypotheses = recognizer.languageHypotheses(withMaximum: 2)
Language Identification

```swift
import NaturalLanguage

let recognizer = NLLanguageRecognizer()

recognizer.processString("困死啦睡觉去了")

let lang = recognizer.dominantLanguage

let hypotheses = recognizer.languageHypotheses(withMaximum: 2)
```

zh-Hans: Simplified Chinese
Language Identification

```swift
import NaturalLanguage

let recognizer = NLLanguageRecognizer()

recognizer.processString(“困死啦睡觉去了”)

let lang = recognizer.dominantLanguage

let hypotheses = recognizer.languageHypotheses(withMaximum:2)
```

zh-Hans: Simplified Chinese
Tokenization

```swift
import NaturalLanguage

let tokenizer = NLTokenizer(unit: .word)

let str = "困死啦睡觉去了"

let strRange = str.startIndex..<str.endIndex

tokenizer.string = str

let tokenArray = tokenizer.tokens(for: strRange)
```
import NaturalLanguage

let tokenizer = NLTokenizer(unit: .word)

let str = "困死啦睡觉去了"

let strRange = str.startIndex..<str.endIndex

tokenizer.string = str

let tokenArray = tokenizer.tokens(for: strRange)
import NaturalLanguage

let tokenizer = NLTokenizer(unit: .word)

let str = “困死啦睡觉去了”

let strRange = str.startIndex ..< str.endIndex

tokenizer.string = str

let tokenArray = tokenizer.tokens(for: strRange)
import NaturalLanguage

let tokenizer = NLTokenizer(unit: .word)

let str = "困死啦睡觉去了"

let strRange = str.startIndex..<str.endIndex

tokenizer.string = str

let tokenArray = tokenizer.tokens(for: strRange)
import NaturalLanguage

let tokenizer = NLTokenizer(unit: .word)

let str = "困死啦睡觉去了"

let strRange = str.startIndex ..< str.endIndex

tokenizer.string = str

let tokenArray = tokenizer.tokens(for: strRange)
import NaturalLanguage

let tokenizer = NLTokenizer(unit: .word)

let str = “困死啦睡觉去了”

let strRange = str.startIndex ..< str.endIndex

tokenizer.string = str

let tokenArray = tokenizer.tokens(for: strRange)
import NaturalLanguage

let tokenizer = NLTokenizer(unit: .word)

let str = "困死啦睡觉去了"

let strRange = str.startIndex ..< str.endIndex

tokenizer.string = str

let tokenArray = tokenizer.tokens(for: strRange)
Design is coming together but needs some final tweaks
As Prince Harry and Meghan Markle Wed, a New Era Dawns

By ELLEN BARRY  MAY 19, 2018

WINDSOR, England — A thousand-year-old English castle echoed with the exhortations of an African-American bishop and a gospel choir on Saturday, as Prince Harry wed Meghan Markle, an American actress, nudging the British royal family into a new
As Prince Harry and Meghan Markle Wed, a New Era Dawns

By ELLEN BARRY  MAY 19, 2018

WINDSOR, England — A thousand-year-old English castle echoed with the exhortations of an African-American bishop and a gospel choir on Saturday, as Prince Harry wed Meghan Markle, an American actress, nudging the British royal family into a new
import NaturalLanguage

let tagger = NLLTagger(tagSchemes: [.nameType])

let str = "Prince Harry and Meghan Markle had their wedding ceremony in Windsor"

let strRange = str.startIndex..<str.endIndex

tagger.string = str

tagger.setLanguage(.english, range: strRange)

let tags = tagger.tags(in: strRange, unit: .word,
    scheme: .nameType, options: .omitWhitespace)
import NaturalLanguage

let tagger = NLTagger(tagSchemes: [.nameType])

let str = "Prince Harry and Meghan Markle had their wedding ceremony in Windsor"

let strRange = str.startIndex..<str.endIndex

tagger.string = str

tagger.setLanguage(.english, range: strRange)

let tags = tagger.tags(in: strRange, unit: .word, 
scheme: .nameType, options: .omitWhitespace)
import NaturalLanguage

let tagger = NLTagger(tagSchemes: [.nameType])

let str = "Prince Harry and Meghan Markle had their wedding ceremony in Windsor"

let strRange = str.startIndex..<str.endIndex

tagger.string = str

tagger.setLanguage(.english, range: strRange)

let tags = tagger.tags(in: strRange, unit: .word,
    scheme: .nameType, options: .omitWhitespace)
import NaturalLanguage

let tagger = NLTagger(tagSchemes: [.nameType])

let str = "Prince Harry and Meghan Markle had their wedding ceremony in Windsor"

let strRange = str.startIndex..<str.endIndex

tagger.string = str

tagger.setLanguage(.english, range: strRange)

let tags = tagger.tags(in: strRange, unit: .word,
                        scheme: .nameType, options: .omitWhitespace)
import NaturalLanguage

let tagger = NLTagger(tagSchemes: [.nameType])

let str = "Prince Harry and Meghan Markle had their wedding ceremony in Windsor"

let strRange = str.startIndex..<str.endIndex

tagger.string = str

tagger.setLanguage(.english, range: strRange)

let tags = tagger.tags(in: strRange, unit: .word,
    scheme: .nameType, options: .omitWhitespace)
Named Entity Recognition

```swift
import NaturalLanguage

let tagger = NLTagger(tagSchemes: [.nameType])

let str = "Prince Harry and Meghan Markle had their wedding ceremony in Windsor"

let strRange = str.startIndex..<str.endIndex

tagger.string = str

tagger.setLanguage(.english, range: strRange)

let tags = tagger.tags(in: strRange, unit: .word,
scheme: .nameType, options: .omitWhitespace)

PER: Prince Harry
Meghan Markle

LOC: Windsor
```
Natural Language

Developer documentation

Future support for new NLP APIs
Swift APIs for NLP
Swift APIs for NLP

Custom NLP models
Custom NLP Models

Doug Davidson, Senior Software Engineer
Concept Learning
Concept Learning

Examples
Concept Learning
Concept Learning

Examples → Classify and Understand
Machine Learning

Training Data ➔ ML ➔ Classify and Understand
Types of Custom Model
Types of Custom Model

Text Classification

Label

Label

Label
Types of Custom Model

Text Classification

Word Tagging

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Text Classification
Sentiment classification

I am really excited, would definitely recommend it highly!  POSITIVE
This was terrible, much worse than I expected.  NEGATIVE
It was OK, something I could live with for now.  NEUTRAL
Foles’ late TD pass leads Eagles to 1st Super Bowl title

Apple introduces new 9.7 inch iPad with Apple Pencil support

Apple Reports Second Quarter Results
I’m looking for a place to stay in Barcelona.

Where can I get good Mexican food on a Sunday?

Find me an inexpensive round-trip flight to London.

HOTELS

RESTAURANTS

FLIGHTS
Word Tagging
Part of speech

Το παιδί διάβασε το βιβλίο

Το DETERMINER παιδί NOUN διάβασε VERB το DETERMINER βιβλίο NOUN
The iPad is popular in Singapore
<table>
<thead>
<tr>
<th>Slot Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round trip</td>
<td>ROUND_TRIP</td>
</tr>
<tr>
<td>fares</td>
<td>NONE</td>
</tr>
<tr>
<td>from</td>
<td>NONE</td>
</tr>
<tr>
<td>Pittsburgh</td>
<td>FROM_LOCATION</td>
</tr>
<tr>
<td>to</td>
<td>NONE</td>
</tr>
<tr>
<td>Philadelphia</td>
<td>TO_LOCATION</td>
</tr>
</tbody>
</table>
Supervised Machine Learning
Training and inference
Supervised Machine Learning
Training and inference

Training → Annotated data → Laptop → ML
Supervised Machine Learning

Training and inference

Training
- Annotated data

Inference
- User data

Prediction
Custom NLP Models

Training
Custom NLP Models

Training

Training Data
Custom NLP Models

Training

Training Data → Create ML
Custom NLP Models

Training

Training Data → Create ML → Natural Language
Custom NLP Models
Training

Training Data → Create ML → Natural Language
Custom NLP Models

Training

Training Data → Create ML

Natural Language → ML
Text Classifier
Annotated data

[
  {
    "text": "I am really excited, would definitely recommend it highly!",
    "label": "Positive"
  },
  {
    "text": "It was OK, something I could live with for now.",
    "label": "Neutral"
  },
  {
    "text": "This was terrible, much worse than I expected.",
    "label": "Negative"
  }
]
Annotated data

[  
  
  {
    "text": "I am really excited, would definitely recommend it highly!",
    "label": "Positive"
  },
  
  {
    "text": "It was OK, something I could live with for now.",
    "label": "Neutral"
  },
  
  {
    "text": "This was terrible, much worse than I expected.",
    "label": "Negative"
  }
]
import CreateML
import Foundation

let trainingData = try MLDataTable(contentsOfFile: Bundle.main.url(forResource: "news", withExtension: "json")!)

let model = try MLTextClassifier(trainingWith: trainingData, 
                                 textColumn: "text", labelColumn: "label")

try model.write(to: URL(fileURLWithPath: "/Users/me/Desktop/textclassifier.mlmodel")
import CreateML
import Foundation

let trainingData = try MLDataTable(contentsOfFile: Bundle.main.url(forResource: "news", withExtension: "json")!

let model = try MLTextClassifier(trainingWith: trainingData,
                                 textColumn: "text", labelColumn: "label")

try model.write(to: URL(fileURLWithPath: "/Users/me/Desktop/textclassifier.mlmodel")

import CreateML
import Foundation

let trainingData = try MLDataTable(contentsOfFile: Bundle.main.url(forResource: "news", withExtension: "json")!)

let model = try MLTextClassifier(trainingWith: trainingData, 
                                textColumn: "text", 
                                labelColumn: "label")

try model.write(to: URL(fileURLWithPath: "/Users/me/Desktop/textclassifier.mlmodel"))
import CreateML
import Foundation

let trainingData = try MLDataTable(contentsOfFile: Bundle.main.url(forResource: "news", withExtension: "json")!)!

let model = try MLTextClassifier(trainingWith: trainingData, 
    textColumn: "text", labelColumn: "label")

try model.write(to: URL(fileURLWithPath: "/Users/me/Desktop/textclassifier.mlmodel")"
[{
  "tokens": ["AirPods", "are", "a", "fantastic", "product", "from", "Apple", "."],
  "labels": ["PROD", "NONE", "NONE", "NONE", "NONE", "NONE", "ORG", "NONE"]
},
{
  "tokens": ["Apple", "and", "Tim", "Cook", "have", "another", "hit", "."],
  "labels": ["ORG", "NONE", "PER", "PER", "NONE", "NONE", "NONE", "NONE"]
}]
Word Tagger
Annotated data

```json
[
  {
    "tokens": ["AirPods", "are", "a", "fantastic", "product", "from", "Apple", "."],
    "labels": ["PROD", "NONE", "NONE", "NONE", "NONE", "NONE", "ORG", "NONE"]
  },
  {
    "tokens": ["Apple", "and", "Tim", "Cook", "have", "another", "hit", "."],
    "labels": ["ORG", "NONE", "PER", "PER", "NONE", "NONE", "NONE", "NONE"]
  }
]
```
import CreateML
import Foundation

let trainingData = try MLDataTable(contentsOfFile:
    Bundle.main.url(forResource: "products", withExtension: "json")!
)

let model = try MLWordTagger(trainingWith: trainingData,
    tokenHeader: "tokens", labelColumn: "labels")

try model.write(to: URL(fileURLWithPath: "/Users/me/Desktop/wordtagger.mlmodel"))
import CreateML
import Foundation

let trainingData = try MLDataTable(contentsOfFile:
    Bundle.main.url(forResource: "products", withExtension: "json")!)

let model = try MLWordTagger(trainingWith: trainingData,
    tokenHeader: "tokens", labelColumn: "labels")

try model.write(to: URL(fileURLWithPath: "/Users/me/Desktop/wordtagger.mlmodel")

---

Introducing Create ML

Hall 2

Tuesday 2:00PM
import CreateML

import Foundation

let trainingData = try MLDataTable(contentsOfFile:

    Bundle.main.url(forResource: "products", withExtension: "json")!

let model = try MLWordTagger(trainingWith: trainingData,

    tokenHeader: "tokens", labelColumn: "labels")

try model.write(to: URL(fileURLWithPath: "/Users/me/Desktop/wordtagger.mlmodel"))
import CreateML
import Foundation

let trainingData = try MLDataTable(contentsOfFile: Bundle.main.url(forResource: "products", withExtension: "json")!)

let model = try MLWordTagger(trainingWith: trainingData, tokenHeader: "tokens", labelColumn: "labels")

try model.write(to: URL(fileURLWithPath: "/Users/me/Desktop/wordtagger.mlmodel"))
<table>
<thead>
<tr>
<th>Event</th>
<th>Venue</th>
<th>Date/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>What’s New in Core ML, Part 1</td>
<td>Hall 1</td>
<td>Wednesday 9:00AM</td>
</tr>
<tr>
<td>What’s New in Core ML, Part 2</td>
<td>Hall 1</td>
<td>Wednesday 10:00AM</td>
</tr>
<tr>
<td>Event</td>
<td>Hall</td>
<td>Date</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>-------</td>
<td>---------------</td>
</tr>
<tr>
<td>What’s New in Core ML, Part 1</td>
<td>1</td>
<td>Wednesday 9:00AM</td>
</tr>
<tr>
<td>What’s New in Core ML, Part 2</td>
<td>1</td>
<td>Wednesday 10:00AM</td>
</tr>
</tbody>
</table>
Natural Language

Inference

User data → Natural Language

<table>
<thead>
<tr>
<th>Event Description</th>
<th>Location</th>
<th>Date/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>What’s New in Core ML, Part 1</td>
<td>Hall 1</td>
<td>Wednesday 9:00AM</td>
</tr>
<tr>
<td>What’s New in Core ML, Part 2</td>
<td>Hall 1</td>
<td>Wednesday 10:00AM</td>
</tr>
</tbody>
</table>
Natural Language
Inference

User data → Natural Language

<table>
<thead>
<tr>
<th>Event</th>
<th>Hall</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>What’s New in Core ML, Part 1</td>
<td>Hall 1</td>
<td>Wednesday 9:00AM</td>
</tr>
<tr>
<td>What’s New in Core ML, Part 2</td>
<td>Hall 1</td>
<td>Wednesday 10:00AM</td>
</tr>
</tbody>
</table>
Natural Language
Inference

User data → Natural Language

<table>
<thead>
<tr>
<th>Event</th>
<th>Hall</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>What’s New in Core ML, Part 1</td>
<td>Hall 1</td>
<td>Wednesday 9:00AM</td>
</tr>
<tr>
<td>What’s New in Core ML, Part 2</td>
<td>Hall 1</td>
<td>Wednesday 10:00AM</td>
</tr>
</tbody>
</table>
What’s New in Core ML, Part 1
Hall 1
Wednesday 9:00AM

What’s New in Core ML, Part 2
Hall 1
Wednesday 10:00AM
Using a Custom Model

```swift
import NaturalLanguage

if let modelURL = Bundle.main.url(forResource: "classifier",
                                 withExtension: "mlmodelc") {

    let model = try NLModel(contentsOf: modelURL)

    let label = model.predictedLabel(for: "I really loved it!")

}
```
Using a Custom Model

```swift
import NaturalLanguage

if let modelURL = Bundle.main.url(forResource: "classifier",
                                 withExtension: "mlmodelc") {

    let model = try NLMModel(contentsOf: modelURL)

    let label = model.predictedLabel(for: "I really loved it!"

}
```
Using a Custom Model

```swift
import NaturalLanguage

if let modelURL = Bundle.main.url(forResource: "classifier",
    withExtension: "mlmodelc") {
    let model = try NLModel(contentsOf: modelURL)
    let label = model.predictedLabel(for: "I really loved it!")
}
```
import NaturalLanguage

if let modelURL = Bundle.main.url(forResource: "classifier",
   withExtension: "mlmodelc") {

    let model = try NLModel(contentsOf: modelURL)

    let label = model.predictedLabel(for: "I really loved it!")

}
Using Custom Models with NLTagger

```swift
let myTagScheme = NLTagScheme("MyTagScheme")

let tagger = NLTagger(tagSchemes: [myTagScheme])

tagger.setModels([\(model\)], forTagScheme: myTagScheme)

tagger.string = \(myString)

tagger.enumerateTags(in: range, unit: .sentence,

    scheme: myTagScheme, options: []) { (tag, tokenRange) -> Bool in

    // use the generated tags
    return true
}
```
Using Custom Models with NLTagger

```swift
let myTagScheme = NLTagScheme("MyTagScheme")

let tagger = NLTagger(tagSchemes: [myTagScheme])
tagger.setModels([model], forTagScheme: myTagScheme)
tagger.string = myString

tagger.enumerateTags(in: range, unit: .sentence,
    scheme: myTagScheme, options: []) { (tag, tokenRange) -> Bool in

    // use the generated tags
    return true
}
```
Using Custom Models with NLTagger

```swift
let myTagScheme = NLTagScheme("MyTagScheme")

let tagger = NLTagger(tagSchemes: [myTagScheme])

tagger.setModels([model], forTagScheme: myTagScheme)

tagger.string = myString

tagger.enumerateTags(in: range, unit: .sentence,
    scheme: myTagScheme, options: []) { (tag, tokenRange) -> Bool in
        // use the generated tags
        return true
}
```
Using Custom Models with NLTagger

```swift
let myTagScheme = NLTagScheme("MyTagScheme")

let tagger = NLTagger(tagSchemes: [myTagScheme])

tagger.setModels([model], forTagScheme: myTagScheme)

tagger.string = myString

tagger.enumerateTags(in: range, unit: .sentence,
                      scheme: myTagScheme, options: []) { (tag, tokenRange) -> Bool in

    // use the generated tags
    return true

}
Using Custom Models with NLTagger

```swift
let myTagScheme = NLTagScheme("MyTagScheme")
let tagger = NLTagger(tagSchemes: [myTagScheme])
tagger.setModels([model], forTagScheme: myTagScheme)
tagger.string = myString

tagger.enumerateTags(in: range, unit: .sentence,
                      scheme: myTagScheme, options: []) { (tag, tokenRange) -> Bool in

    // use the generated tags
    return true
}
```
Using Custom Models with NLTagger

```swift
let myTagScheme = NLTagScheme("MyTagScheme")

let tagger = NLTagger(tagSchemes: [myTagScheme])

tagger.setModels([model], forTagScheme: myTagScheme)

tagger.string = myString

tagger.enumerateTags(in: range, unit: .sentence,
                      scheme: myTagScheme, options: []){ (tag, tokenRange) -> Bool in

    // use the generated tags
    return true

}
Wade

Bookmark organization

Organize by topics, entities

- Apple announces WWDC 2018
- Best way to cook an Apple Pie
- San Francisco Giants team news
- Golden State Warriors team news
- Finance: Apple stock page on NASD
- Healthy Kitchen: Whole grain meals
- WSJ business page
- Wikipedia: San Francisco Giants
- Wikipedia: Golden State Warriors
- Wikipedia: Abraham Lincoln
- Wikipedia: Benjamin Franklin
- Wikipedia: Grand Canyon
- Trip planner: 10 things to see in LA
- Travelogue: Spending time in Oregon
- Great Barrier Reef: A trip across the P
- Hiking Shoes: The best choices in the
Demo
Swift APIs for NLP  Custom NLP models
Swift APIs for NLP

Custom NLP models

Performance
Natural Language

Available on all Apple platforms
Available on all Apple platforms
Natural Language

Available on all Apple platforms

Standardized text processing
Text Processing

Typical ML training

Training Data → Tokenization → Feature Extraction → Machine Learning Toolkit → CoreML Convertor
Text Processing
Typical ML inference

User Data → Output
Text Processing

Typical ML inference

User Data → Output
Text Processing

Typical ML inference

User Data → Tokenization Feature Extraction → Output
Standarized Text Processing
Natural Language
Standarized Text Processing

Natural Language

Training Data
Standarized Text Processing
Natural Language

Training Data → Create ML
Standarized Text Processing

Natural Language

Training Data -> Create ML

Natural Language

Tokenization
Feature Extraction

Machine Learning
Standarized Text Processing
Natural Language

Training Data → Create ML

Natural Language

Tokenization Feature Extraction
Machine Learning
Standarized Text Processing
Natural Language

Training Data -> Create ML -> ML

Natural Language

Tokenization
Feature Extraction
Machine Learning
Standarized Text Processing

Natural Language
Standarized Text Processing

Natural Language
Standarized Text Processing

Natural Language

User Data → Natural Language → Output
Optimized Models

Low latency

Optimized for Apple hardware
# Optimized Models

## Low latency

## Optimized for Apple hardware

<table>
<thead>
<tr>
<th></th>
<th>Open Source CRFSuite</th>
<th>Natural Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Named Entity Recognition</td>
<td>70 MB</td>
<td>1.4 MB</td>
</tr>
<tr>
<td>Chunking</td>
<td>30 MB</td>
<td>1.8 MB</td>
</tr>
</tbody>
</table>

Results on CoNLL training and test data
ML Algorithms

Text classification

```swift
let modelParameters = MLTextClassifier.ModelParameters(algorithm: .maxEnt(version: 1))
let modelParameters = MLTextClassifier.ModelParameters(algorithm: .CRF(version: 1))
```
ML Algorithms
Text classification

```swift
let modelParameters = MLTextClassifier.ModelParameters(algorithm: .maxEnt(version: 1))

let modelParameters = MLTextClassifier.ModelParameters(algorithm: .CRF(version: 1))
```
ML Algorithms
Text classification

```swift
let modelParameters = MLTextClassifier.ModelParameters(algorithm: .maxEnt(version: 1))
let modelParameters = MLTextClassifier.ModelParameters(algorithm: .CRF(version: 1))
```

Maximum Entropy
ML Algorithms

Text classification

```
let modelParameters = MLTextClassifier.ModelParameters(algorithm: .maxEnt(version: 1))
let modelParameters = MLTextClassifier.ModelParameters(algorithm: .CRF(version: 1))
```
ML Algorithms

Text classification

let modelParameters = MLTextClassifier.ModelParameters(algorithm: .maxEnt(version: 1))
let modelParameters = MLTextClassifier.ModelParameters(algorithm: .CRF(version: 1))
let model = try MLWordTagger(trainingWith: trainingData,
    tokenHeader: "tokens", labelColumn: "labels")

Conditional Random Field
let model = try MLWordTagger(trainingWith: trainingData,
    tokenHeader: "tokens", labelColumn: "labels")

Conditional Random Field
Data

- Validate training data
- Inspect training instances per class
Data
Data

Run different algorithms

Training
Set aside test data
Test on out-of-domain data
Evaluation → Training → Data → Evaluation

Fix by adding data
Retrain model
Workflow for ML

Data -> Training -> Evaluation -> Data
Swift APIs for NLP
Custom NLP models
Performance
Privacy
Privacy Preserving ML Applied to NLP
Natural Language Framework
Summary

New: Natural Language Framework

Custom NLP models

Performance

Privacy
## More Information

https://developer.apple.com/wwdc18/713

<table>
<thead>
<tr>
<th>Event</th>
<th>Location</th>
<th>Date &amp; Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine Learning Get-Together</td>
<td>Café Lounge</td>
<td>Wednesday 6:15PM</td>
</tr>
<tr>
<td>Natural Language Lab</td>
<td>Technology Lab 4</td>
<td>Thursday 1:00PM</td>
</tr>
</tbody>
</table>