Text and Linguistic Analysis

Session 215

Douglas Davidson
Natural Languages Group

Introduction

- Applications often deal with large amounts of text
- Knowledge about that text can help the user
- Mac OS X and iOS have sophisticated APIs for analyzing text
- Proper Unicode text handling is essential for international support

What You’ll Learn

- Iteration, matching, searching
- Regular expressions, Data Detectors, and linguistic APIs
- Putting it all together

Strings

- NSString (and CFString, toll-free bridged)
- Conceptually sequences of UTF-16 units (“characters”)
- Text processing operates on ranges within strings, not single characters

Attributed Strings

- NSAttributedString (and CFAttributedString, toll-free bridged)
- Has-a string

  `[attributedString string]`
- Decorated with attributes applied to ranges within the string
  - Font, color, underline, etc.

Character Clusters

- The smallest processing unit for most tasks
- Sometimes one character, sometimes two or more
- Composition
  - é = e + ´
  - 각 = ㄱ + ㅏ + ㄱ
- Surrogate pairs
  - 𠀋 U+2000B = 0xD840 + 0xDC0B
  - 😄 U+1F604 = 0xD83D + 0xDE04

Don’t Split Character Clusters

- Don’t use ranges in a string that split character clusters!

  Use `rangeOfComposedCharacterSequenceAtIndex:` or

  `rangeOfComposedCharacterSequencesForRange:`

- Otherwise you may end up with ´ or ▨ or ▨

Character Cluster Iteration

```
[string enumerateSubstringsInRange:range
  options:NSStringEnumerationByComposedCharacterSequences
  usingBlock:^(NSString *substring,
               NSRange substringRange,
               NSRange enclosingRange,
               BOOL *stop){

        [text addAttribute:NSForegroundColorAttributeName
               value:color range:substringRange];
}];
```

Character Cluster Iteration

```
[string enumerateSubstringsInRange:range
  options:NSStringEnumerationByComposedCharacterSequences
  usingBlock:^(NSString *substring,
               NSRange substringRange,
               NSRange enclosingRange,
               BOOL *stop){

        [text addAttribute:NSForegroundColorAttributeName
               value:color range:substringRange];
}];
```

Character Cluster Iteration

```
[string enumerateSubstringsInRange:range
  options:NSStringEnumerationByComposedCharacterSequences
  usingBlock:^(NSString *substring,
               NSRange substringRange,
               NSRange enclosingRange,
               BOOL *stop){

        [text addAttribute:NSForegroundColorAttributeName
               value:color range:substringRange];
}];
```

Character Cluster Iteration

`San José😄`

Character Cluster Iteration

San José😄

0x53

Character Cluster Iteration

San José😄

0x61

Character Cluster Iteration

San José😄

0x6e

Character Cluster Iteration

San José😄

0x20

Character Cluster Iteration

San José😄

0x4a

Character Cluster Iteration

San José😄

0x6f

Character Cluster Iteration

```
San José😄
```

0x73

Character Cluster Iteration

`San José😄`

0x65 0x301

Character Cluster Iteration

```
San José😄
```

0xd83d 0xde04

Words

- Appropriate processing unit for most transformation tasks
  - Letter-case mapping
  - Spell-checking
- Whitespace is not necessarily the only way to break words
  - 正しい日本語です = 正しい + 日本 + 語 + です
  - ภาษาไทย = ภาษา + ไทย
  - Mac用户将可以升级到Mountain Lion =

    Mac + 用户 + 将 + 可以 + 升级 + 到 + Mountain + Lion

Word Iteration

```
[string enumerateSubstringsInRange:range
  options:NSStringEnumerationByWords
  usingBlock:^(NSString *substring,
               NSRange substringRange,
               NSRange enclosingRange,
               BOOL *stop){

        [text addAttribute:NSForegroundColorAttributeName
               value:color range:substringRange];
}];
```

Paragraphs

- The maximum processing unit for all Unicode processing tasks
- Separated by newline, carriage return, paragraph break
- Especially important for bi-directional languages like Arabic and Hebrew
- Each paragraph has an overall text flow direction

Paragraph Iteration

```
[string enumerateSubstringsInRange:range
  options:NSStringEnumerationByParagraphs
  usingBlock:^(NSString *substring,
               NSRange substringRange,
               NSRange enclosingRange,
               BOOL *stop){

        [text addAttribute:NSForegroundColorAttributeName
               value:color range:substringRange];
}];
```

Processing a File by Lines

```
NSString *str = [NSString stringWithContentsOfURL:url
                 encoding:NSUTF8StringEncoding error:error];

[str enumerateLinesUsingBlock:^(NSString *line, BOOL *stop){
    // do something with line
}];
```

String Search

```
NSRange matchRange =
  [string rangeOfString:@"resume"
          options:NSCaseInsensitiveSearch |
                 NSDiacriticInsensitiveSearch |
                 NSWidthInsensitiveSearch
          range:range];

if (matchRange.location != NSNotFound) {
  // found a match
}
```

String Matching

```
NSRange matchRange =
  [string rangeOfString:@"resume"
          options:NSAnchoredSearch |
                  NSCaseInsensitiveSearch |
                  NSDiacriticInsensitiveSearch |
                  NSWidthInsensitiveSearch
          range:range];

if (matchRange.location != NSNotFound) {
  // found a match
}
```

String Search and Replace

```
NSString *modifiedString =
  [string stringByReplacingOccurrencesOfString:
    @"resume" withString:@"résumé"
    options:NSDiacriticInsensitiveSearch
    range:range];               // immutable strings
```

String Search and Replace

```
NSString *modifiedString =
  [string stringByReplacingOccurrencesOfString:
    @"resume" withString:@"résumé"
    options:NSDiacriticInsensitiveSearch
    range:range];             // immutable strings

[mutableString replaceOccurrencesOfString:
    @"resume" withString:@"résumé"
    options:NSDiacriticInsensitiveSearch
    range:range];             // mutable strings
```

Character Sets

- NSCharacterSet (and CFCharacterSet, toll-free bridged)
- Conceptually bitmap of UTF-32 values from 0x00000 to 0x10FFFF
- Many predefined examples (whitespace, punctuation, letter, etc.)
- Mutable and immutable variants
- Can create with arbitrary sets of characters

```
– characterSetWithRange:
– characterSetWithCharactersInString:
```

Character Set Search

```
NSRange matchRange =
  [string rangeOfCharacterFromSet:characterSet
          options:0
          range:range];

if (matchRange.location != NSNotFound) {
  // found a match
}
```

Character Set Matching

```
NSRange matchRange =
  [string rangeOfCharacterFromSet:characterSet
          options:NSAnchoredSearch
          range:range];

if (matchRange.location != NSNotFound) {
  // found a match
}
```

Demo

Regular Expressions

- Specify strings, character sets, word boundaries, and more
- Combined and repeated in arbitrary ways
- Also subexpressions, backreferences, and other non-regular features

Example Regular Expression

```
\b  (i|o)  (f|n)  \b
```

Example Regular Expression

```
\b  (i|o)  (f|n)  \b
```

word break

Example Regular Expression

```
\b  (i|o)  (f|n)  \b
```

word break

subexpression
(1st capture group)

Example Regular Expression

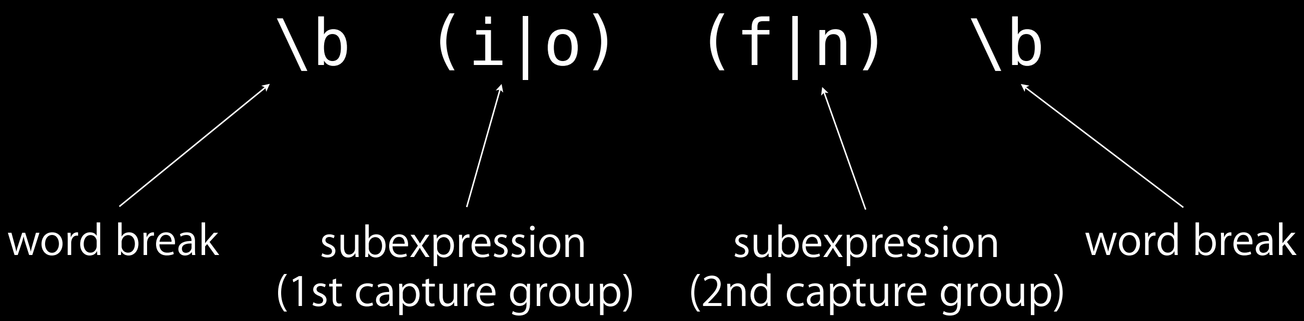
```
\b  (i|o)  (f|n)  \b
```

word break

subexpression
(1st capture group)

subexpression
(2nd capture group)

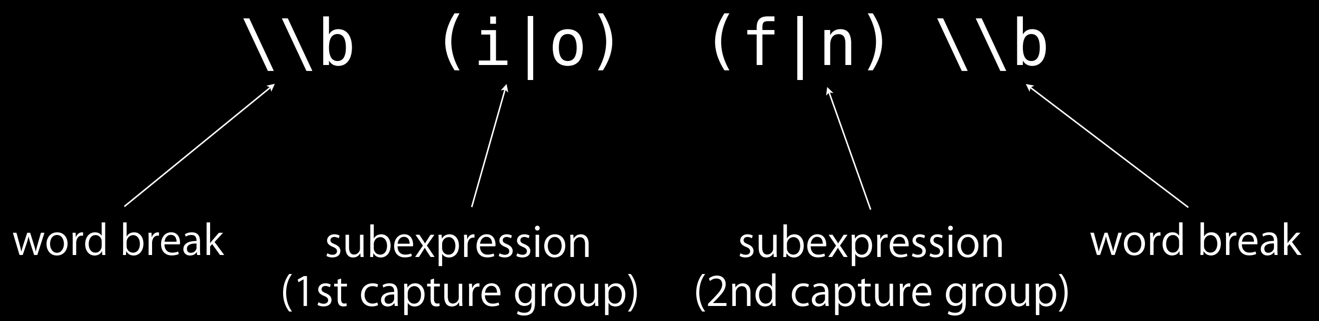
Example Regular Expression



```
\b  (i|o)  (f|n)  \b
```

word break
subexpression (1st capture group)
subexpression (2nd capture group)
word break

Example Regular Expression



Cocoa Regular Expressions

- Use standard ICU regular expression syntax
- Fully Unicode compliant
- All of the usual options (and more) are available

String Searching

```
NSRange matchRange =
  [string rangeOfString:@"\\b(i|o)(f|n)\\b"
          options:NSRegularExpressionSearch |
                 NSCaseInsensitiveSearch
          range:range];

if (matchRange.location != NSNotFound) {
  // found a match
}
```

String Searching

```
NSRange matchRange =
  [string rangeOfString:@"\\b(i|o)(f|n)\\b"
          options:NSRegularExpressionSearch |
                  NSCaseInsensitiveSearch
          range:range];

if (matchRange.location != NSNotFound) {
  // found a match
}
```

String Searching

```
NSRange matchRange =
  [string rangeOfString:@"\\b(i|o)(f|n)\\b"
         options:NSRegularExpressionSearch |
                NSCaseInsensitiveSearch
         range:range];

if (matchRange.location != NSNotFound) {
  // found a match
}
```

String Searching

```
NSRange matchRange =
  [string rangeOfString:@"\\b(i|o)(f|n)\\b"
          options:NSRegularExpressionSearch |
                  NSCaseInsensitiveSearch
          range:range];

if (matchRange.location != NSNotFound) {
  // found a match
}
```

NSRegularExpression

```
NSError *error = nil;

NSRegularExpression *regex =
 [NSRegularExpression
  regularExpressionWithPattern:@"\\b(i|o)(f|n)\\b"
  options:NSRegularExpressionCaseInsensitive
  error:&error];
```

NSRegularExpression

```
NSError *error = nil;

NSRegularExpression *regex =
 [NSRegularExpression
  regularExpressionWithPattern:@"\\b(i|o)(f|n)\\b"
  options:NSRegularExpressionCaseInsensitive
  error:&error];
```

NSRegularExpression

```
NSError *error = nil;

NSRegularExpression *regex =
 [NSRegularExpression
  regularExpressionWithPattern:@"\\b(i|o)(f|n)\\b"
  options:NSRegularExpressionCaseInsensitive
  error:&error];
```

Regular Expression Iteration

```
[regex enumerateMatchesInString:string
  options:0 range:range
  usingBlock:^(NSTextCheckingResult *match,
               NSMatchingFlags flags, BOOL *stop){

        [text addAttribute:NSForegroundColorAttributeName
               value:color range:[match range]];

}];
```

Regular Expression Iteration

```
If into in onto of often on and ON.
```

Regular Expression Iteration

```
If into in onto of often on and ON.
```

Regular Expression Iteration

```
If into in onto of often on and ON.
```

Regular Expression Iteration

```
If into in onto of often on and ON.
```

Regular Expression Iteration

```
If into in onto of often on and ON.
```

Regular Expression Iteration

```
If into in onto of often on and ON.
```

Match Objects

- Objects of class NSTextCheckingResult

```
@property NSTextCheckingType resultType;
```

```
@property NSRange range;
```

  - This is the overall range

```
- (NSRange)rangeAtIndex:(NSUInteger)idx;
```

  - These are the ranges of capture groups

Regular Expression Ranges

```
[regex enumerateMatchesInString:string
  options:0 range:range
  usingBlock:^(NSTextCheckingResult *match,
               NSMatchingFlags flags, BOOL *stop){

    NSRange matchRange = [match range];
    NSRange firstHalfRange =
                         [match rangeAtIndex:1];
    NSRange secondHalfRange =
                         [match rangeAtIndex:2];

}];
```

Regular Expression Ranges

```
[regex enumerateMatchesInString:string
  options:0 range:range
  usingBlock:^(NSTextCheckingResult *match,
               NSMatchingFlags flags, BOOL *stop){

    NSRange matchRange = [match range];
    NSRange firstHalfRange =
                         [match rangeAtIndex:1];
    NSRange secondHalfRange =
                         [match rangeAtIndex:2];

}];
```

Regular Expression Ranges

```
[regex enumerateMatchesInString:string
  options:0 range:range
  usingBlock:^(NSTextCheckingResult *match,
               NSMatchingFlags flags, BOOL *stop){

    NSRange matchRange = [match range];
    NSRange firstHalfRange =
                         [match rangeAtIndex:1];
    NSRange secondHalfRange =
                         [match rangeAtIndex:2];

}];
```

Additional Methods

```
– matchesInString:options:range:
– numberOfMatchesInString:options:range:
– firstMatchInString:options:range:
– rangeOfFirstMatchInString:options:range:
```

Additional Methods

```
 – matchesInString:options:range:
 – numberOfMatchesInString:options:range:
 – firstMatchInString:options:range:
 – rangeOfFirstMatchInString:options:range:
```

```
NSRange matchRange =
  [regex rangeOfFirstMatchInString:string
         options:0 range:searchRange];

if (matchRange.location != NSNotFound) {
  // found a match
}
```

Processing a File by Lines

```
NSString *str = [NSString stringWithContentsOfURL:url
                    encoding:NSUTF8StringEncoding error:error];

[str enumerateLinesUsingBlock:^(NSString *line, BOOL *stop){
    NSRange matchRange =
      [regex rangeOfFirstMatchInString:line
          options:0 range:NSMakeRange(0, [line length])];

    if (matchRange.location != NSNotFound) {
        printf("%s\n", [line UTF8String]);
    }
}];
```

Processing a File by Lines

```
NSString *str = [NSString stringWithContentsOfURL:url
                 encoding:NSUTF8StringEncoding error:error];

[str enumerateLinesUsingBlock:^(NSString *line, BOOL *stop){
    NSRange matchRange =
      [regex rangeOfFirstMatchInString:line
          options:0 range:NSMakeRange(0, [line length])];

    if (matchRange.location != NSNotFound) {
        printf("%s\n", [line UTF8String]);
    }
}];
```

Search and Replace

```
NSString *modifiedString =
  [regex stringByReplacingMatchesInString:string
    options:0
    range:range
    withTemplate:@"$2$1"];    // immutable strings
```

Search and Replace

```
NSString *modifiedString =
  [regex stringByReplacingMatchesInString:string
    options:0
    range:range
    withTemplate:@"$2$1"];    // immutable strings

[regex replaceMatchesInString:mutableString
    options:0
    range:range
    withTemplate:@"$2$1"];    // mutable strings
```

Search and Replace

```
NSString *modifiedString =
  [regex stringByReplacingMatchesInString:string
    options:0
    range:range
    withTemplate:@"$2$1"];    // immutable strings

[regex replaceMatchesInString:mutableString
    options:0
    range:range
    withTemplate:@"$2$1"];    // mutable strings
```

Template Replacement

```
If into in onto of often on and ON.
```

Template Replacement

```
If into in onto of often on and ON.

fI into ni onto fo often no and NO.
```

String Search and Replace

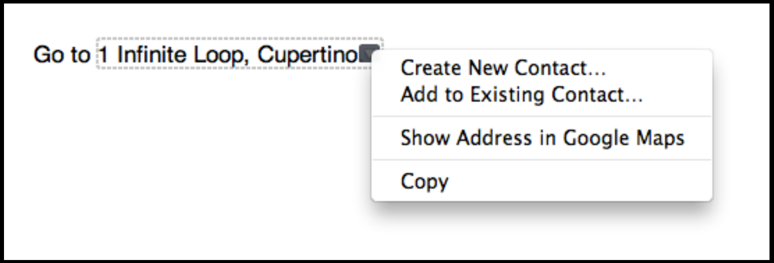
```
NSString *modifiedString =
  [string stringByReplacingOccurrencesOfString:
    @"\\b(i|o)(f|n)\\b" withString:@"$2$1"
    options:NSRegularExpressionSearch
    range:range];             // immutable strings

[mutableString replaceOccurrencesOfString:
    @"\\b(i|o)(f|n)\\b" withString:@"$2$1"
    options:NSRegularExpressionSearch
    range:range];             // mutable strings
```

Data Detectors

- Locate URLs, email addresses, phone numbers, dates, addresses, etc.
- Can handle many international formats
- Made available via NSRegularExpression subclass

Data Detectors

- Locate URLs, email addresses, phone numbers, dates, addresses, etc.
- Can handle many international formats
- Made available via NSRegularExpression subclass


Go to 1 Infinite Loop, Cupertino

NSDataDetector

```
NSError *error = nil;

NSDataDetector *detector =
   [NSDataDetector
     dataDetectorWithTypes:
         NSTextCheckingTypeLink |
                        NSTextCheckingTypePhoneNumber
     error:&error];
```

NSDataDetector

```
NSError *error = nil;

NSDataDetector *detector =
   [NSDataDetector
     dataDetectorWithTypes:
          NSTextCheckingTypeLink |
                         NSTextCheckingTypePhoneNumber
     error:&error];
```

Data Detector Types

- Dates

  `NSTextCheckingTypeDate`
- Addresses

  `NSTextCheckingTypeAddress`
- URLs

  `NSTextCheckingTypeLink`
- Phone numbers

  `NSTextCheckingTypePhoneNumber`

Data Detector Iteration

```
[detector enumerateMatchesInString:string
 options:0 range:range
 usingBlock:^(NSTextCheckingResult *match,
                  NSMatchingFlags flags, BOOL *stop){

        [text addAttribute:NSForegroundColorAttributeName
                  value:color range:[match range]];

}];
```

Getting Results

- NSTextCheckingResult objects with different resultType
- More NSTextCheckingResult properties:

```
	@property NSDate *date;
	@property NSDictionary *components;
	@property NSURL *URL;
	@property NSString *phoneNumber;
```

Data Detector Results

```
[detector enumerateMatchesInString:string
 options:0 range:range
 usingBlock:^(NSTextCheckingResult *match,
              NSMatchingFlags flags, BOOL *stop){
  NSTextCheckingType t = [match resultType];

  if (t == NSTextCheckingTypeLink) {
    NSURL *url = [match URL];
    // do something with url
  } else if (t == NSTextCheckingTypePhoneNumber) {
    NSString *phoneNumber = [match phoneNumber];
    // do something with phone number
  }
}];
```

Data Detector Results

```
[detector enumerateMatchesInString:string
 options:0 range:range
 usingBlock:^(NSTextCheckingResult *match,
                NSMatchingFlags flags, BOOL *stop){
  NSTextCheckingType t = [match resultType];

  if (t == NSTextCheckingTypeLink) {
    NSURL *url = [match URL];
    // do something with url
  } else if (t == NSTextCheckingTypePhoneNumber) {
    NSString *phoneNumber = [match phoneNumber];
    // do something with phone number
  }
}];
```

Data Detector Results

```
[detector enumerateMatchesInString:string
 options:0 range:range
 usingBlock:^(NSTextCheckingResult *match,
              NSMatchingFlags flags, BOOL *stop){
  NSTextCheckingType t = [match resultType];

  if (t == NSTextCheckingTypeLink) {
    NSURL *url = [match URL];
    // do something with url
  } else if (t == NSTextCheckingTypePhoneNumber) {
    NSString *phoneNumber = [match phoneNumber];
    // do something with phone number
  }
}];
```

Additional Methods

```
– matchesInString:options:range:
– numberOfMatchesInString:options:range:
– firstMatchInString:options:range:
– rangeOfFirstMatchInString:options:range:
```

Additional Methods

```
 – matchesInString:options:range:
 – numberOfMatchesInString:options:range:
 – firstMatchInString:options:range:
 – rangeOfFirstMatchInString:options:range:
```

```
NSRange matchRange =
  [detector rangeOfFirstMatchInString:string
            options:0 range:searchRange];

if (matchRange.location != NSNotFound) {
  // found a match
}
```

Demo

Linguistic Tagging

- Word and sentence boundaries
- Token type
  - Word, punctuation, whitespace, etc.
- Language
  - en, fr, de, ja, zh-Hans, etc.
- Script
  - Latn, Cyrl, Arab, Jpan, Hans, etc.

Linguistic Tagging

- Lexical class
  - Noun, verb, adjective, etc.
- Lemma
  - Root form of word
- Named entities
  - Personal name, place name, organization name

Current Language Support

- Lexical class
  - OS X—English, French, German, Italian, Spanish
  - iOS—English
- Lemma
  - OS X—English, French, German, Italian, Spanish
  - iOS—English
- Named entities
  - OSX and iOS—English
- Method to determine supported schemes for a given language
  - `availableTagSchemesForLanguage:`

NSLinguisticTagger

```
NSLinguisticTagger *tagger =
  [[NSLinguisticTagger alloc]
    initWithTagSchemes:
        [NSArray arrayWithObjects:
            NSLinguisticTagSchemeTokenType,
            NSLinguisticTagSchemeLexicalClass,
            NSLinguisticTagSchemeNameType,
            NSLinguisticTagSchemeNameTypeOrLexicalClass,
            NSLinguisticTagSchemeLemma, nil]
    options:0];

[tagger setString:string];
```

NSLinguisticTagger

```
NSLinguisticTagger *tagger =
  [[NSLinguisticTagger alloc]
    initWithTagSchemes:
        [NSArray arrayWithObjects:
            NSLinguisticTagSchemeTokenType,
            NSLinguisticTagSchemeLexicalClass,
            NSLinguisticTagSchemeNameType,
            NSLinguisticTagSchemeNameTypeOrLexicalClass,
            NSLinguisticTagSchemeLemma, nil]
    options:0];

[tagger setString:string];
```

NSLinguisticTagger

```
NSLinguisticTagger *tagger =
  [[NSLinguisticTagger alloc]
    initWithTagSchemes:
        [NSArray arrayWithObjects:
            NSLinguisticTagSchemeTokenType,
            NSLinguisticTagSchemeLexicalClass,
            NSLinguisticTagSchemeNameType,
            NSLinguisticTagSchemeNameTypeOrLexicalClass,
            NSLinguisticTagSchemeLemma, nil]
    options:0];

[tagger setString:string];
```

NSLinguisticTagger

```
NSLinguisticTagger *tagger =
  [[NSLinguisticTagger alloc]
    initWithTagSchemes:
        [NSArray arrayWithObjects:
            NSLinguisticTagSchemeTokenType,
            NSLinguisticTagSchemeLexicalClass,
            NSLinguisticTagSchemeNameType,
            NSLinguisticTagSchemeNameTypeOrLexicalClass,
            NSLinguisticTagSchemeLemma, nil]
    options:0];

[tagger setString:string];
```

NSLinguisticTagger

```
NSLinguisticTagger *tagger =
  [[NSLinguisticTagger alloc]
    initWithTagSchemes:
        [NSArray arrayWithObjects:
            NSLinguisticTagSchemeTokenType,
            NSLinguisticTagSchemeLexicalClass,
            NSLinguisticTagSchemeNameType,
            NSLinguisticTagSchemeNameTypeOrLexicalClass,
            NSLinguisticTagSchemeLemma, nil]
    options:0];

[tagger setString:string];
```

NSLinguisticTagger

```
NSLinguisticTagger *tagger =
  [[NSLinguisticTagger alloc]
    initWithTagSchemes:
        [NSArray arrayWithObjects:
            NSLinguisticTagSchemeTokenType,
            NSLinguisticTagSchemeLexicalClass,
            NSLinguisticTagSchemeNameType,
            NSLinguisticTagSchemeNameTypeOrLexicalClass,
            NSLinguisticTagSchemeLemma, nil]
    options:0];

[tagger setString:string];
```

NSLinguisticTagger

```
NSLinguisticTagger *tagger =
  [[NSLinguisticTagger alloc]
    initWithTagSchemes:
        [NSArray arrayWithObjects:
            NSLinguisticTagSchemeTokenType,
            NSLinguisticTagSchemeLexicalClass,
            NSLinguisticTagSchemeNameType,
            NSLinguisticTagSchemeNameTypeOrLexicalClass,
            NSLinguisticTagSchemeLemma, nil]
    options:0];

[tagger setString:string];
```

Linguistic Tagger Iteration

```
[tagger enumerateTagsInRange:range
        scheme:NSLinguisticTagSchemeLexicalClass
        options:NSLinguisticTaggerOmitWhitespace
        usingBlock:^(NSString *tag, NSRange tokenRange,
                     NSRange sentenceRange, BOOL *stop){

       if (tag == NSLinguisticTagNoun)
          [text addAttribute:NSForegroundColorAttributeName
               value:color range:tokenRange];

}];
```

Linguistic Tagger Iteration

```
[tagger enumerateTagsInRange:range
        scheme:NSLinguisticTagSchemeLexicalClass
        options:NSLinguisticTaggerOmitWhitespace
        usingBlock:^(NSString *tag, NSRange tokenRange,
                     NSRange sentenceRange, BOOL *stop){

       if (tag == NSLinguisticTagNoun)
          [text addAttribute:NSForegroundColorAttributeName
               value:color range:tokenRange];

}];
```

Linguistic Tagger Iteration

```
[tagger enumerateTagsInRange:range
        scheme:NSLinguisticTagSchemeLexicalClass
        options:NSLinguisticTaggerOmitWhitespace
        usingBlock:^(NSString *tag, NSRange tokenRange,
                     NSRange sentenceRange, BOOL *stop){

       if (tag == NSLinguisticTagNoun)
          [text addAttribute:NSForegroundColorAttributeName
               value:color range:tokenRange];

}];
```

Linguistic Tagger Iteration

```
[tagger enumerateTagsInRange:range
        scheme:NSLinguisticTagSchemeLexicalClass
        options:NSLinguisticTaggerOmitWhitespace
        usingBlock:^(NSString *tag, NSRange tokenRange,
                     NSRange sentenceRange, BOOL *stop){

       if (tag == NSLinguisticTagNoun)
          [text addAttribute:NSForegroundColorAttributeName
               value:color range:tokenRange];

}];
```

Linguistic Tagger Iteration

```
We said to him, “Hello!”
```

Linguistic Tagger Iteration

```
We said to him, “Hello!”
```

pronoun

we

Linguistic Tagger Iteration

```
We said to him, “Hello!”
```

verb

say

Linguistic Tagger Iteration

```
We said to him, “Hello!”
```

preposition

to

Linguistic Tagger Iteration

```
We said to him, “Hello!”
```

pronoun

he

Linguistic Tagger Iteration

```
We said to him, “Hello!”
```

punctuation

Linguistic Tagger Iteration

```
We said to him, “Hello!”
```

open quote

Linguistic Tagger Iteration

```
We said to him, “Hello!”
```

interjection

hello

Linguistic Tagger Iteration

```
We said to him, “Hello!”
```

sentence terminator

Linguistic Tagger Iteration

```
We said to him, “Hello!”
```

close quote

Additional Methods

```
– tagAtIndex:scheme:tokenRange:sentenceRange:
– tagsInRange:scheme:options:tokenRanges:
```

Additional Methods

```
- tagAtIndex:scheme:tokenRange:sentenceRange:
- tagsInRange:scheme:options:tokenRanges:
```

```
NSString *tag =
  [tagger tagAtIndex:idx
          scheme:NSLinguisticTagSchemeLexicalClass
          tokenRange:NULL
          sentenceRange:NULL];

  if (tag == NSLinguisticTagNoun) {
      // found a noun
  }
```

Specifying Language

```
NSDictionary *map = [NSDictionary dictionaryWithObject:
      [NSArray arrayWithObjects:@"en", nil] forKey:@"Latn"];

NSOrthography *orthography = [NSOrthography
      orthographyWithDominantScript:@"Latn" languageMap:map];

[tagger setOrthography:orthography range:range];
```

String Tagging

```
[string enumerateLinguisticTagsInRange:range
        scheme:NSLinguisticTagSchemeLexicalClass
        options:NSLinguisticTaggerOmitWhitespace
        orthography:orthography
        usingBlock:^(NSString *tag, NSRange tokenRange,
                     NSRange sentenceRange, BOOL *stop){

       if (tag == NSLinguisticTagNoun)
          [text addAttribute:NSForegroundColorAttributeName
               value:color range:tokenRange];

}];
```

Demo

Applications

- Improved text checking and correction
- Provide contextual information for words
- Identify names in text
- Improved indexing

Demo

Jennifer Moore
Natural Languages Group

Summary

- Examine ranges within NSStrings
- Use blocks to iterate over ranges
- Other APIs will search for ranges
- Different types of ranges provided by various APIs
  - NSString
  - NSCharacterSet
  - NSRegularExpression
  - NSDataDetector
  - NSLinguisticTagger

More Information

Jake Behrens
Frameworks Evangelist
behrens@apple.com

Documentation
String Programming Guide for Cocoa
http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/Strings/introStrings.html

ICU Regular Expression Syntax
http://userguide.icu-project.org/strings/regexp

Apple Developer Forums
http://devforums.apple.com

Related Sessions

Keyboard Input in iOS	Russian Hill Wednesday 2:00PM
Introduction to Attributed Strings for iOS	Mission Wednesday 3:15PM
Advanced Attributed Strings for iOS	Mission Thursday 10:15AM
Internationalization Tips & Tricks	Marina Friday 10:15AM

Labs

Attributed Strings & Text Lab	Essentials Lab A Thursday 11:30AM
Internationalization Lab	Application Frameworks Lab A Friday 11:30AM

WWDC2012

The last 3 slides after the logo are intentionally left blank for all presentations.

The last 3 slides after the logo are intentionally left blank for all presentations.

The last 3 slides after the logo are intentionally left blank for all presentations.