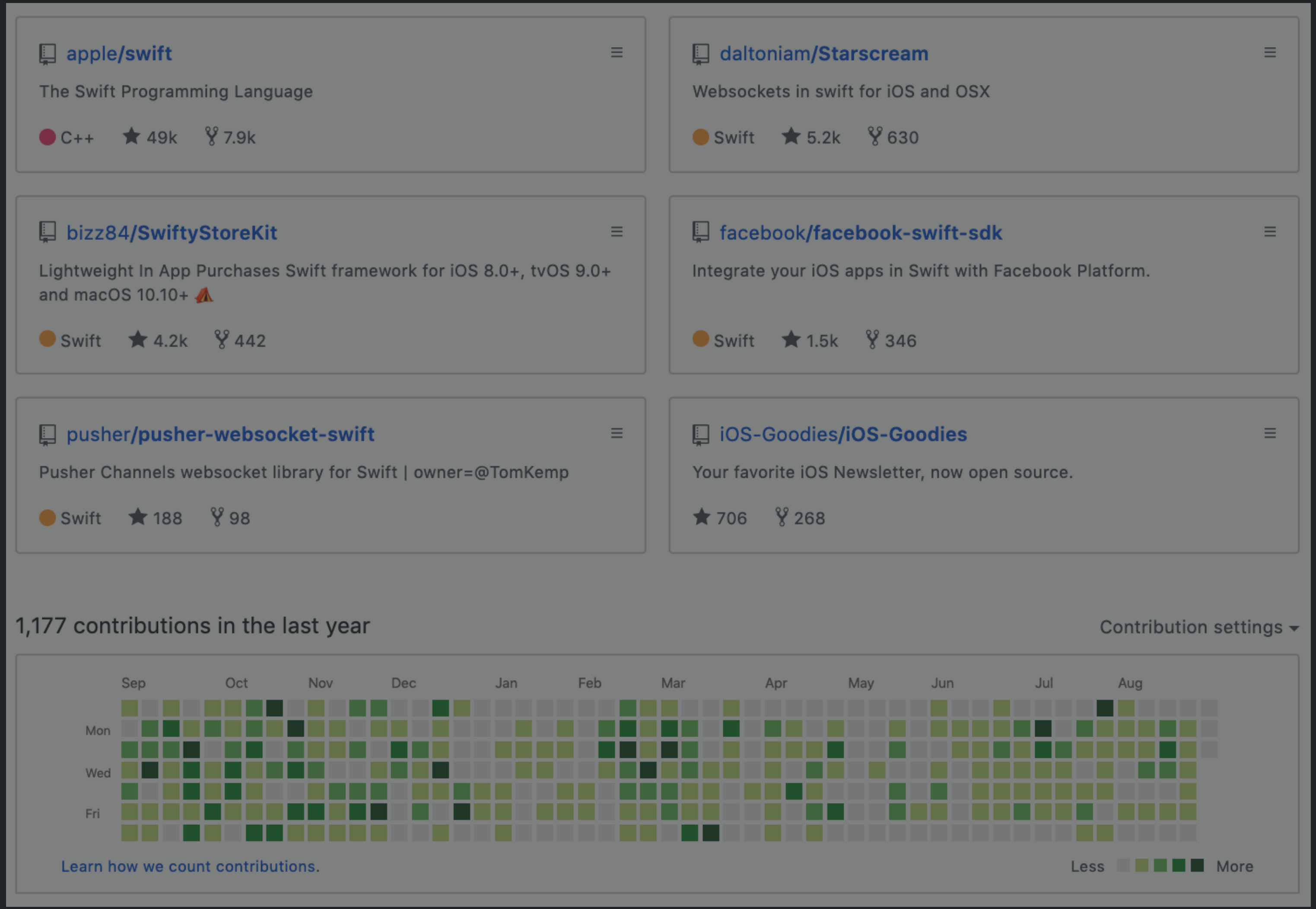
# NLP in Swift

Vadim Drobinin | @valzevul

## About me

- — Stealth startup (London, UK)
- — drobinin.com/talks & VK University
- — foodncities.com & thestirred.art
- — github.com/valzevul



apple/swift
The Swift Programming Language
C++ 49k 7.9k

daltoniam/Starscream
Websockets in swift for iOS and OSX
Swift 5.2k 630

bizz84/SwiftyStoreKit
Lightweight In App Purchases Swift framework for iOS 8.0+, tvOS 9.0+ and macOS 10.10+ ⛵
Swift 4.2k 442

facebook/facebook-swift-sdk
Integrate your iOS apps in Swift with Facebook Platform.
Swift 1.5k 346

pusher/pusher-websocket-swift
Pusher Channels websocket library for Swift | owner=@TomKemp
Swift 188 98

iOS-Goodies/iOS-Goodies
Your favorite iOS Newsletter, now open source.
706 268

1,177 contributions in the last year

## Agenda

- NLP in a nutshell
- Why frontend?
- Basics
- 2019 Edition
- Deep stuff
- Summary

# In a nutshell

Natural Language Processing (NLP)* explores interactions between computers and human languages.

* Not to be confused with Neuro-linguistic programming (also NLP), a pseudoscientific approach to communication, personal development, and psychotherapy.

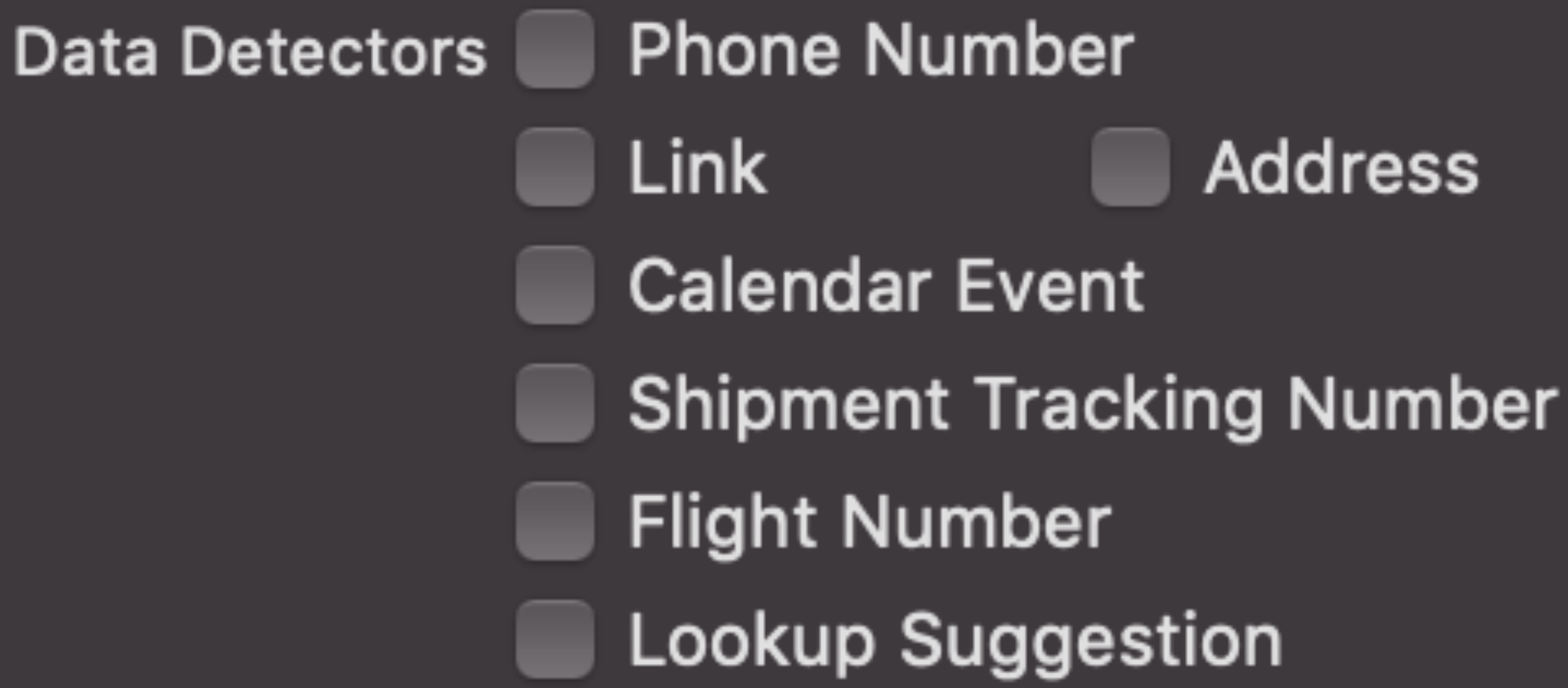
NLP
…Grammar induction…
…PoC-tagging…
IF

## Major tasks

- Syntax
  - Grammar induction, Lemmatization, Morphological segmentation, Part-of-speech tagging, Parsing, Sentence breaking, Stemming, Word segmentation, Terminology extraction
- Semantics
  - Lexical semantics, Distributional semantics, Machine translation, Named entity recognition (NER), Natural language generation, Natural language understanding, Optical character recognition, Question answering, Recognizing Textual entailment, Relationship extraction, Sentiment analysis, Topic segmentation and recognition, Word sense disambiguation
- Discourse
  - Automatic summarization, Coreference resolution, Discourse analysis
- Speech
  - Speech recognition, Speech segmentation, Text-to-speech
- Dialogue

## Major tasks

- Syntax
  - Lemmatization, Part-of-speech tagging, Sentence breaking, Word segmentation
- Semantics
  - Named entity recognition (NER), Question answering, Relationship extraction, Sentiment analysis



Data Detectors ☐ Phone Number
☐ Link ☐ Address
☐ Calendar Event
☐ Shipment Tracking Number
☐ Flight Number
☐ Lookup Suggestion

## Why frontend?

- Cheap
- Fast
- No vendor dependencies[1]
- Private

[1] Well, except Apple

## 💡 Basics

## NSLinguisticTagger

- iOS 5.0+
- Analyze natural language text to tag part of speech and lexical class, identify names, perform lemmatization, and determine the language and script.

## NaturalLanguage.framework

- iOS 12.0+
- class NLTokenizer:
  A tokenizer that segments natural language text into semantic units.
- class NLTagger:
  A tagger that analyzes natural language text.

## NLTagger.Options

- .omitWords
- .omitPunctuation
- .omitWhitespace
- .omitOther
- .joinNames
- .joinContractions

## NLTagScheme

- .tokenType
- .lexicalClass
- .nameType
- .nameTypeOrLexicalClass
- .lemma
- .language
- .script

## 🚀 Still Basics

## Set up

```swift
import NaturalLanguage

let schemes: [NLTagScheme] = [.language, .lemma, .lexicalClass]
let tagger = NLTagger(tagSchemes: schemes)
let options: NLTagger.Options = [.joinNames, .omitWhitespace]
```

## 1. Lemmatization

```swift
let quote = "Silver coins were minted at Aberystwyth Castle."
lemmatize(for: quote)
/*
    Silver -> silver
    coins -> coin
    were -> be
    minted -> mint
    at -> at
*/
```

## 1. Lemmatization

```swift
func lemmatize(for text: String) {
  tagger.string = text
  let range = text.startIndex..<text.endIndex
  tagger.enumerateTags(in: range, unit: .word, scheme: .lemma,
                       options: options) { (tag, range) -> Bool in
                        if let lemma = tag?.rawValue {
                          print("\(text[range]) -> \(lemma)")
                        }
                        return true
  }
}
```

## 2. Part-of-speech tagging

```
let quote = "Pressure on me: You are Welsh, they said;"

extractPoS(text: quote)
// 'Pressure' -> Noun, 'on' -> Preposition,
// 'me' -> Pronoun, ':' -> Punctuation, etc
```

## 2. Part-of-speech tagging

```swift
func extractPoS(text: String) {
  tagger.string = text
  tagger.enumerateTags(in: text.startIndex..<text.endIndex, unit: .word,
                       scheme: .lexicalClass, options: options) {
                        (tag, range) -> Bool in
                        if let tag = tag?.rawValue {
                          print("\(text[range]) -> \(tag)")
                        }
                        return true
  }
}
```

## 3. Dominant Language

```swift
let quote = """
Добро пожаловать в Аберистуит.
"""

dominantLanguage(for: quote) // -> "ru"
```

## 3. Dominant Language

```swift
func dominantLanguage(for text: String) {
  tagger.string = text
  let rawString = tagger.dominantLanguage?.rawValue
  if let languageName = rawString {
    print(languageName)
  }
}
```

## 4. Sentence breaking

- What about splitting by dots and capital letters?
- ["J", "Bond will complain to M", "I", "6"]

## 4. Sentence breaking

```swift
let quote = """
In Mr. Douglas Adams' dictionary of toponymic
neologisms, an Aberystwyth (or A.B.E.R.Y.S.T.W.Y.T.H)
is "A nostalgic yearning which is in itself more
pleasant than the thing being yearned for".
"""

tokenize(text: quote, for: .sentence) // -> 1
```

## 4. Sentence breaking

```swift
func tokenize(text: String, for unit: NLTokenUnit) {
  let tokenizer = NLTokenizer(unit: unit)
  tokenizer.string = text
  let range = quote.startIndex..<quote.endIndex
  let tokens = tokenizer.tokens(for: range)
  print(tokens.count)
}
```

## 5. Word segmentation

- Just `split()` by not letters, eh?
- ["Don", "t", "do", "that"]
- 
```
tokenize(text: "アベリステゥイスが大好き",
    for: .word) // -> 3, [アベリステゥイ, スが, 大好き]
```

## 6. Named entity recognition

```swift
let quote = """
The relatively obscure children's novel, Mr. Bass's Planetoid,
by Eleanor Cameron, has a character who claims
to be from Aberystwyth.
"""

extractNER(text: quote)

// -> Mr. Bass: PersonalName
// -> Eleanor Cameron: PersonalName
// -> Aberystwyth: PlaceName
```

## 6. Named entity recognition

```swift
func extractNER(text: String) {
  tagger.string = text
  let tags: [NLTag] = [.personalName, .placeName, .organizationName]
  let range = text.startIndex..<text.endIndex
  tagger.enumerateTags(in: range, unit: .word, scheme: .nameType,
                       options: options) { (tag, range) -> Bool in
                        if let tag = tag, tags.contains(tag) {
                          print("\(text[range]) -> \(tag.rawValue)")
                        }
                        return true
  }
}
```

## 📱 iOS 13+

## Sentiment analysis

```swift
let tagger = NLTagger(tagSchemes: [.sentimentScore])

func sentiment(for quote: String) {
    tagger.string = quote
    let (sentiment, _) = tagger.tag(at: quote.startIndex, unit: .sentence,
                                    scheme: .sentimentScore)
    if let sentimentScore = sentiment?.rawValue {
        if sentimentScore > 0 { print("😀") }
        else if sentimentScore == 0 { print("😐") }
        else { print("😭") }
    }
}
```

## Sentiment analysis

```
sentiment(for: "Such an awesome breeze you have here!") // -> 😀
sentiment(for: "What a day, isn't it?") // -> 😀
sentiment(for: "I hate that stormy wind!") // -> 😭
sentiment(for: "The weather is ok.") // -> 😐
```

## Text Suggestions

```swift
func findSimilar(for word: String) {
    let embedding = NLEmbedding.wordEmbedding(for: .english)
    embedding?.enumerateNeighbors(for: word.lowercased(),
                                  maximumCount: 5) {
        (string, distance) -> Bool in
        print(string)
        return true
    }
}

findSimilar(for: "rarebit") // -> sauce, cheese, toast, bread, mustard
findSimilar(for: "swift") // -> fast, fleet, speedy, quick, rapid
```

## Still iOS 13+

- Custom Word Embeddings
- Text Catalog

## 💀 Deep stuff

## Information Extraction

- TF-IDF (Term frequency - Inverse Document Frequency)
- Let's count the importance of each word and offset it by how common it is
- Let's use the most important words to build topic clusters

## Term Frequency

$$tf(t, d) = \frac{\text{number of occurrences of term in document}}{\text{total number of all words in document}}$$

## Term Frequency

Hello, my name is Vadim. Vadim Vadim. The city stands on the hill.

## Term Frequency

$$\overbrace{\text{Hello}}^{\frac{1}{8}=0.125}\text{, my }\overbrace{\text{name}}^{0.125}\text{ is }\overbrace{\text{Vadim.}}^{0.375}\text{ Vadim Vadim. The }\overbrace{\text{city}}^{0.125}\ \overbrace{\text{stands}}^{0.125}\text{ on the }\overbrace{\text{hill}}^{0.125}.$$

## Term Frequency

$$\overbrace{\overbrace{\text{Hello,}}^{0.125}\text{ my }\overbrace{\text{name}}^{0.125}\text{ is }\overbrace{\text{Vadim.}}^{0.375}}^{\frac{0.125+0.125+0.375}{3}=0.208}\overbrace{\overbrace{\text{Vadim}}^{0.375}\ \overbrace{\text{Vadim.}}^{0.375}}^{\frac{0.375+0.375}{2}=0.375}\ \overbrace{\text{The }\overbrace{\text{city}}^{0.125}\ \overbrace{\text{stands}}^{0.125}\text{ on the }\overbrace{\text{hill}}^{0.125}.}^{\frac{0.125+0.125+0.0125}{3}=0.125}$$

## Inverse Document Frequency

$$idf(t, d) = \log_{10}\left(\frac{\text{number of documents containing the term, t}}{\text{how many times the term, t, appears}}\right)$$

## Inverse Document Frequency

This is Aberystwyth. Aberystwyth is cool. There is a university.

## Inverse Document Frequency

$$\text{This } \overbrace{\text{city}}^{\log_{10}\frac{3}{1}=log_{10}3=0.477} \text{ is } \overbrace{\text{Aberystwyth.}}^{0.176} \overbrace{\text{Aberystwyth}}^{0.176} \text{ is } \overbrace{\text{cool.}}^{0.477} \text{ There is a } \overbrace{\text{university.}}^{0.477}$$

## Inverse Document Frequency

$$\overbrace{\text{This } \overbrace{\text{city}}^{0.477} \text{ is } \overbrace{\text{Aberystwyth}}^{0.176}\text{.}}^{\frac{0.477+0.176}{2}=0.32} \; \overbrace{\overbrace{\text{Aberystwyth}}^{0.176} \text{ is } \overbrace{\text{cool}}^{0.477}\text{.}}^{\frac{0.176+0.477}{2}=0.32} \; \overbrace{\text{There is a } \overbrace{\text{university}}^{0.477}\text{.}}^{\frac{0.477}{1}=0.477}$$

## TF-IDF

$$tf(\text{This is Aberystwyth}) = 0.5$$

$$idf(\text{This is Aberystwyth}) = 0.32$$

$$tfidf(\text{This is Aberystwyth}) = 0.5 * 0.32 = 0.16$$

## TF-IDF

```swift
func uniqueWords(words: [String]) -> [String] {
  return Array(Set(words))
}

func prettify(article: String) -> [String] {
  return tokenize(text: article, for: .word).filter { str -> Bool in
    !stopwords.contains(str) && str.count > 2
  }
}
```

## TF-IDF

```
func countWords(words: [String]) -> [String: Double] {
  var results = [String: Double]()
  let uniqueW = uniqueWords(words: words)
  for word in words {
    if uniqueW.contains(word) {
      results[word] = (results[word] ?? 0) + 1
    }
  }
  return results
}
```

## TF-IDF

```swift
let allWords = prettify(article: article.lowercased())
let allSentences = tokenize(text: article, for: .sentence)
let allUniqueWords = uniqueWords(words: allWords)
var allUniqueWordsFrequency = countWords(words: allWords)
```

## TF-IDF

```swift
func termFrequency() -> [String: Double] {
  var currentFrequency = [String: Double]()
  for word in allUniqueWordsFrequency.keys {
    let wordF = allUniqueWordsFrequency[word] ?? 0.0
    currentFrequency[word] = wordF / Double(allWords.count)
  }
  return countForSentences(sentences: allSentences,
                                      values: currentFrequency)
}
```

## TF-IDF

```swift
func inverseDocumentFrequence() -> [String: Double] {
  var inverseFrequency = [String: Double]()
  for word in allUniqueWordsFrequency.keys {
    let wordF = allUniqueWordsFrequency[word] ?? 0.0
    let inverseF = Double(allSentences.count) / wordF
    inverseFrequency[word] = log10(inverseF)
  }
  return countForSentences(sentences: allSentences,
                           values: inverseFrequency)
}
```

## TF-IDF

```swift
func countForSentences(sentences: [String],
                       values: [String: Double]) -> [String: Double] {
  var sentenceFrequences = [String: Double]()
  for sentence in sentences {
    var sum = 0.0
    let sentenceWords = prettify(article: sentence.lowercased())
    for word in sentenceWords {
      sum += (values[word.lowercased()] ?? 0)
    }
    sentenceFrequences[sentence] = sum / Double(sentenceWords.count)
  }
  return sentenceFrequences
}
```

## TF-IDF

```swift
func TFIDF() -> [String: Double] {
  let tf = termFrequency()
  let idf = inverseDocumentFrequence()
  var tfidf = [String: Double]()
  for word in tf.keys {
    guard let tf = tf[word], let idf = idf[word] else { continue }
    tfidf[word] = tf * idf
  }
  return tfidf
}
```

## TF-IDF

```swift
let sortedResults = TFIDF().sorted { (lhr, rhr) -> Bool in
  return lhr.value > rhr.value
}

print(sortedResults[0])
print(sortedResults[1])
print(sortedResults[2])
```

## TF-IDF

Aberystwyth is a university town and tourist destination, and forms a cultural link between North Wales and South Wales. Constitution Hill, scaled by the Aberystwyth Cliff Railway, gives access to panoramic views and to other attractions at the summit, including a camera obscura. Scenic Mid Wales landscape within easy reach of the town includes the wilderness of the Cambrian Mountains, whose valleys contain forests and meadows which have changed little in centuries. A convenient way to access the interior is by the preserved narrow-gauge Vale of Rheidol Railway. Although the town is relatively modern, there are a number of historic buildings, including the remains of the castle and the Old College of Aberystwyth University nearby. The Old College was originally built and opened in 1865 as a hotel, but after the owner's bankruptcy the shell of the building was sold to the university in 1867.[13]

The new university campus overlooks Aberystwyth from Penglais Hill to the east of the town centre. The station, a terminus of the main railway, was built in 1924 in the typical style of the period, mainly in a mix of Gothic, Classical Revival, and Victorian architecture. The town is the unofficial capital of Mid Wales, and several institutions have regional or national offices there. Public bodies located in the town include the National Library of Wales, which incorporates the National Screen and Sound Archive of Wales, one of six British regional film archives. The Royal Commission on the Ancient and Historical Monuments of Wales maintains and curates the National Monuments Record of Wales (NMRW), providing the public with information about the built heritage of Wales. Aberystwyth is also the home to the national offices of UCAC and Cymdeithas yr Iaith Gymraeg (Welsh Language Society), and the site of the Institute of Grassland and Environmental Research. The Welsh Books Council and the offices of the standard historical dictionary of Welsh, Geiriadur Prifysgol Cymru, are also located in the town.

## TF-IDF

- Aberystwyth is a university town and tourist destination, and forms a cultural link between North Wales and South Wales.
  **(tfidf = 0.014)**
- The town is the unofficial capital of Mid Wales, and several institutions have regional or national offices there.
  **(tfidf = 0.013)**
- The Royal Commission on the Ancient and Historical Monuments of Wales maintains and curates the National Monuments Record of Wales (NMRW), providing the public with information about the built heritage of Wales.
  **(tfidf = 0.012)**

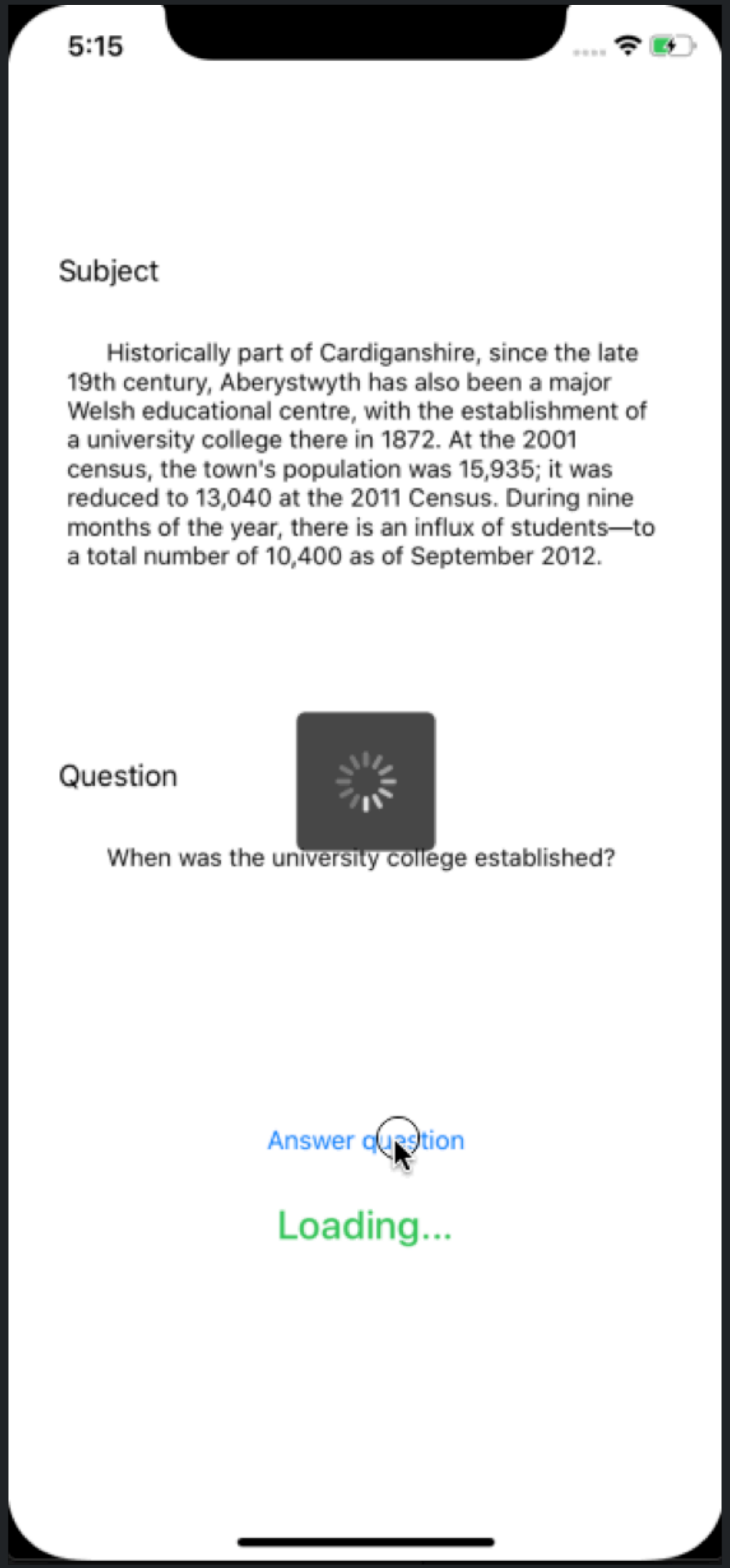
## TF-IDF

**Aberystwyth is a university town and tourist destination, and forms a cultural link between North Wales and South Wales.** Constitution Hill, scaled by the Aberystwyth Cliff Railway, gives access to panoramic views and to other attractions at the summit, including a camera obscura. Scenic Mid Wales landscape within easy reach of the town includes the wilderness of the Cambrian Mountains, whose valleys contain forests and meadows which have changed little in centuries. A convenient way to access the interior is by the preserved narrow-gauge Vale of Rheidol Railway. Although the town is relatively modern, there are a number of historic buildings, including the remains of the castle and the Old College of Aberystwyth University nearby. The Old College was originally built and opened in 1865 as a hotel, but after the owner's bankruptcy the shell of the building was sold to the university in 1867.[13]

The new university campus overlooks Aberystwyth from Penglais Hill to the east of the town centre. The station, a terminus of the main railway, was built in 1924 in the typical style of the period, mainly in a mix of Gothic, Classical Revival, and Victorian architecture. **The town is the unofficial capital of Mid Wales, and several institutions have regional or national offices there.** Public bodies located in the town include the National Library of Wales, which incorporates the National Screen and Sound Archive of Wales, one of six British regional film archives. **The Royal Commission on the Ancient and Historical Monuments of Wales maintains and curates the National Monuments Record of Wales (NMRW), providing the public with information about the built heritage of Wales.** Aberystwyth is also the home to the national offices of UCAC and Cymdeithas yr Iaith Gymraeg (Welsh Language Society), and the site of the Institute of Grassland and Environmental Research. The Welsh Books Council and the offices of the standard historical dictionary of Welsh, Geiriadur Prifysgol Cymru, are also located in the town.

## Question answering

- — BERT SQuAD (pre-trained by Apple)
- — Tokenize everything
- — Map tokens into the model's input format
- — Call the BERT model's `prediction(from:)` method
- — Locate and extract the answer by analyzing the BERT model's output



Subject

Historically part of Cardiganshire, since the late 19th century, Aberystwyth has also been a major Welsh educational centre, with the establishment of a university college there in 1872. At the 2001 census, the town's population was 15,935; it was reduced to 13,040 at the 2011 Census. During nine months of the year, there is an influx of students—to a total number of 10,400 as of September 2012.

Question

When was the university college established?

## One more thing...

- NLP is simple if you know what you need
- You don't always need fancy APIs to work with NLP
- Playgrounds: https://github.com/valzevul/NLPinSwift

## Not enough?

— Custom PoS models
— CoreML and CreateML
— `NLEmbedding` and deep dive into NLP algorithms
— Text Classification
— Text Generation

## Not enough?

- **Slides of this talk: https://drobinin.com/talks/2019/nlp-in-swift/**
- developer.apple.com: finding answers
- WWDC 2019 (232): Advances in Natural Language Framework
- WWDC 2019 (704): CoreML 3 framework

# Questions?

drobinin.com | @valzevul