Advances in Natural Language Framework

Vivek Kumar Rangarajan Sridhar, NLP Technologies
Doug Davidson, NLP Technologies
Amy Frost
I knew there was a reason I married you.❤️

Adam Goosoff
I missed the meeting. Did I actually miss anything?

Karla Gonzales
Thanks for chaperoning the trip to the museum. The kids...

Blair Lockhart
I need a new workout playlist.
😊 Yes, I’m blaming the playlist.

Jason Comet
Catching up on texts. I left my phone at home today.😢

Eric Townley
After careful consideration, I’ve decided I’m an autumn.

Ryan Notch
Sounds good.👍 I haven’t been there in forever.

Andrew Goldfarb
Happy birthday, pal.🎂 Hope you’re doing something fun.
Natural language output
New York-London in 3 ½ Hours? Supersonic Travel May Be Back
Natural Language

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

Text  Intelligence

Linguistics  Machine Learning
Sentiment Analysis

Text Classification
“We had a fun time in Hawaii with the family!”

0.8
“We had a not so fun time in Hawaii cause mom twisted her ankle.”

-0.8
import NaturalLanguage

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

tagger.string = string

let (sentiment, _) = tagger.tag(at: string.startIndex, unit: .paragraph, scheme: .sentimentScore)
import NaturalLanguage

let tagger = NLLTagger(tagSchemes: [.sentimentScore])
tagger.string = string

let (sentiment, _) = tagger.tag(at: string.startIndex, unit: .paragraph, scheme: .sentimentScore)
import NaturalLanguage

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

let (sentiment, _) = tagger.tag(at: string.startIndex, unit: .paragraph, scheme: .sentimentScore)
import NaturalLanguage

let tagger = NLPParser(tagSchemes: [.sentimentScore])

tagger.string = string

let (sentiment, _) = tagger.tag(at: string.startIndex, unit: .paragraph, scheme: .sentimentScore)
import NaturalLanguage

let tagger = NLLTagger(tagSchemes: [.sentimentScore])
tagger.string = string

let (sentiment, _) = tagger.tag(at: string.startIndex, unit: .paragraph, scheme: .sentimentScore)
import NaturalLanguage

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

let (sentiment, _) = tagger.tag(at: string.startIndex, unit: .paragraph, scheme: .sentimentScore)
Brie de Meaux (French)

- Le Roi des Fromages...
- Sweet and buttery taste!
- Has hints of mushroom and truffle.
Brie de Meaux (French)

- Le Roi des Fromages...
- Sweet and buttery taste!
- Has hints of mushroom and truffle.
Performance

Cheesë

Brie de Meaux (French)

- Le Roi des Fromages...
- Sweet and buttery taste!
- Has hints of mushroom and truffle.

7 Languages
Language Assets
import NaturalLanguage

NLTagger.requestAssets(for: .french, tagScheme: .sentimentScore) { (result, error) in
    // handle result
}
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Timothy and I visited Switzerland this summer and fell in love with Gruyère cheese
Timothy and I visited Switzerland this summer and fell in love with Gruyère cheese.
Timothy and I visited Switzerland this summer and fell in love with Gruyère cheese.
Text Catalog

Word Tagging

Gruyère
Emmantaler
Schabziger
Bündner Bergkäse
...
Camembert de Normandie
Faisselle
Cantal
Cabécou

Swiss Cheese
Swiss Cheese
Swiss Cheese
Swiss Cheese
...
French Cheese
French Cheese
French Cheese
French Cheese

Create ML

MLGazetteer

Text Catalog
import CreateML

let entities = [
    "Italian Cheese": ["Parmigiano-Reggiano", "Pecorino Romano", "Montasio", ...],
    "Mexican Cheese": ["Cotija", "Añejo", "Chihuahua", "Queso de cuajo", ...],
    "American Cheese": ["Monterey Jack", "Colby cheese", "Colorado Blackie", ...],
    "Greek Cheese": ["Feta", "Kasseri", "Manouri", "Kefalotyri", ...], ...
]

let gazetteer = try MLGazetteer(dictionary: entities)

try gazetteer.write(to: url)
import CreateML

let entities = 
    
    "Italian Cheese": 
    ["Parmigiano-Reggiano", "Pecorino Romano", "Montasio", ...],
    "Mexican Cheese": 
    ["Cotija", "Añejo", "Chihuahua", "Queso de cuajo", ...],
    "American Cheese": 
    ["Monterey Jack", "Colby cheese", "Colorado Blackie", ...],
    "Greek Cheese": 
    ["Feta", "Kasserī", "Manouri", "Kefalotyri", ...], ...

let gazetteer = try MLGazetteer(dictionary: entities)

try gazetteer.write(to: url)
import CreateML

let entities = [
  "Italian Cheese": ["Parmigiano-Reggiano", "Pecorino Romano", "Montasio", ...],
  "Mexican Cheese": ["Cotija", "Añejo", "Chihuahua", "Queso de cuajo", ...],
  "American Cheese": ["Monterey Jack", "Colby cheese", "Colorado Blackie", ...],
  "Greek Cheese": ["Feta", "Kasseri", "Manouri", "Kefalotyri", ...]
]

let gazetteer = try MLGazetteer(dictionary: entities)

try gazetteer.write(to: url)
import CreateML

let entities = [
    "Italian Cheese": ["Parmigiano-Reggiano", "Pecorino Romano", "Montasio", ...],
    "Mexican Cheese": ["Cotija", "Añejo", "Chihuahua", "Queso de cuajo", ...],
    "American Cheese": ["Monterey Jack", "Colby cheese", "Colorado Blackie", ...],
    "Greek Cheese": ["Feta", "Kasseri", "Manouri", "Kefalotyri", ...], ...
]

let gazetteer = try MLGazetteer(dictionary: entities)

try gazetteer.write(to: url)
import CreateML

let entities = [
    "Italian Cheese": ["Parmigiano-Reggiano", "Pecorino Romano", "Montasio", ...],
    "Mexican Cheese": ["Cotija", "Añejo", "Chihuahua", "Queso de cuajo", ...],
    "American Cheese": ["Monterey Jack", "Colby cheese", "Colorado Blackie", ...],
    "Greek Cheese": ["Feta", "Kasseri", "Manouri", "Kefalotyri", ...]
]

let gazetteer = try MLGazetteer(dictionary: entities)

try gazetteer.write(to: url)
import CreateML

let entities = 
    
    
    let gazetteer = try MLGazetteer(dictionary: entities)

try gazetteer.write(to: url)
Under the Hood

Gruyère
Emmantaler
Schabziger
Bündner Bergkäse
...
Camembert de Normandie
Faisselle
Cantal
Cabécou

Create ML

MLGazetteer

Text Catalog

Swiss Cheese
Swiss Cheese
Swiss Cheese
Swiss Cheese
...
French Cheese
French Cheese
French Cheese
French Cheese
Under the Hood

Gruyère  
Emmentaler  
Schabziger  
Bündner Bergkäse  
...  
Camembert de Normandie  
Faisselle  
Cantal  
Cabécou  

Swiss Cheese  
Swiss Cheese  
Swiss Cheese  
Swiss Cheese  
...  
French Cheese  
French Cheese  
French Cheese  
French Cheese  

Create ML  
MLGazetteer  
Text Catalog  

Natural Language
Under the Hood

Gruyère  
Emmantaler  
Schabziger  
Bündner Bergkäse  
Camembert de Normandie  
Faisselle  
Cantal  
Cabécou  

Swiss Cheese  
Swiss Cheese  
Swiss Cheese  
Swiss Cheese  
French Cheese  
French Cheese  
French Cheese  
French Cheese  

Create ML

MLGazetteer

Text Catalog

Natural Language

Create ML

MLGazetteer

Text Catalog

Natural Language
Text Catalog
Usage with a tagger

```swift
import NaturalLanguage

let gazetteer = try! NLGazetteer(contentsOf: url)

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

tagger.setGazetteers([gazetteer], for: .nameTypeOrLexicalClass)
```
import NaturalLanguage

let gazetteer = try! NLGazetteer(contentsOf: url)

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

tagger.setGazetteers([gazetteer], for: .nameTypeOrLexicalClass)
import NaturalLanguage

let gazetteer = try! NLGazetteer(contentsOf: url)

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

tagger.setGazetteers([gazetteer], for: .nameTypeOrLexicalClass)
import NaturalLanguage

let gazetteer = try! NLGazetteer(contentsOf: url)

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

tagger.setGazetteers([gazetteer], for: .nameTypeOrLexicalClass)
import NaturalLanguage

let gazetteer = try! NLGazetteer(contentsOf: url)

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

tagger.setGazetteers([gazetteer], for: .nameTypeOrLexicalClass)
import NaturalLanguage

let gazetteer = try! NLGazetteer(contentsOf: url)

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

tagger.setGazetteers([gazetteer], for: .nameTypeOrLexicalClass)
Brie de Meaux (French)

- Le Roi des Fromages...
- Sweet and buttery taste!
- Has hints of mushroom and truffle.

Lighter than Camembert or Vacherin
Brie de Meaux (French)

> Le Roi des Fromages...
> Sweet and buttery taste!
> Has hints of mushroom and truffle.

Lighter than Camembert or Vacherin
Word Embedding
Word Embedding

Neural Networks
Word Embedding

Neural Networks

Embedding Layer

$\mathbf{x}$

$\mathbf{W}$

$b$
Word Embeddings
Embedding
Concept
Embedding
Concept
Embedding
Concept

Mapping
## Embedding

### Concept

![Various icons representing different concepts](image)

---

### Mapping

<table>
<thead>
<tr>
<th>Concept</th>
<th>Vector Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="icon" alt="Icon" /></td>
<td><img src="values" alt="Values" /></td>
</tr>
<tr>
<td><img src="icon" alt="Icon" /></td>
<td><img src="values" alt="Values" /></td>
</tr>
<tr>
<td><img src="icon" alt="Icon" /></td>
<td><img src="values" alt="Values" /></td>
</tr>
<tr>
<td><img src="icon" alt="Icon" /></td>
<td><img src="values" alt="Values" /></td>
</tr>
<tr>
<td><img src="icon" alt="Icon" /></td>
<td><img src="values" alt="Values" /></td>
</tr>
</tbody>
</table>

---

### Vector Representation

<table>
<thead>
<tr>
<th>Concept</th>
<th>Vector Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="icon" alt="Icon" /></td>
<td>0.23 0.32 -0.45</td>
</tr>
<tr>
<td><img src="icon" alt="Icon" /></td>
<td>0.28 0.40 -0.39</td>
</tr>
<tr>
<td><img src="icon" alt="Icon" /></td>
<td><img src="values" alt="Values" /></td>
</tr>
<tr>
<td><img src="icon" alt="Icon" /></td>
<td>0.75 0.02 -0.95</td>
</tr>
<tr>
<td><img src="icon" alt="Icon" /></td>
<td>0.81 0.05 -0.88</td>
</tr>
</tbody>
</table>
# Embedding

## Concept

<table>
<thead>
<tr>
<th>Concept</th>
<th>Vector Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Image" /></td>
<td>0.23</td>
</tr>
<tr>
<td><img src="image2" alt="Image" /></td>
<td>0.28</td>
</tr>
<tr>
<td><img src="image3" alt="Image" /></td>
<td>...</td>
</tr>
<tr>
<td><img src="image4" alt="Image" /></td>
<td>0.75</td>
</tr>
<tr>
<td><img src="image5" alt="Image" /></td>
<td>0.81</td>
</tr>
</tbody>
</table>
# Embedding

## Concept

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>![Image]</td>
<td>0.23</td>
<td>0.32</td>
<td>-0.45</td>
</tr>
<tr>
<td>![Image]</td>
<td>0.28</td>
<td>0.40</td>
<td>-0.39</td>
</tr>
<tr>
<td>![Image]</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>![Image]</td>
<td>0.75</td>
<td>0.02</td>
<td>-0.95</td>
</tr>
<tr>
<td>![Image]</td>
<td>0.81</td>
<td>0.05</td>
<td>-0.88</td>
</tr>
</tbody>
</table>

Vector Representation

**Similarity in Vector Space**
Embedding

Concept

<table>
<thead>
<tr>
<th>Concept</th>
<th>0.23</th>
<th>0.32</th>
<th>-0.45</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.28</td>
<td>0.40</td>
<td>-0.39</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.75</td>
<td>0.02</td>
<td>-0.95</td>
</tr>
<tr>
<td></td>
<td>0.81</td>
<td>0.05</td>
<td>-0.88</td>
</tr>
</tbody>
</table>

Vector Representation

Similarity in Vector Space
## Embedding

### Concept

<table>
<thead>
<tr>
<th>Concept</th>
<th>Vector Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shoe</td>
<td>0.23</td>
</tr>
<tr>
<td>Shoe</td>
<td>0.28</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Lamp</td>
<td>0.75</td>
</tr>
<tr>
<td>Table</td>
<td>0.81</td>
</tr>
</tbody>
</table>

*Similarity in Vector Space*
Embedding

Concept

<table>
<thead>
<tr>
<th>Concept</th>
<th>Vector Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shoe</td>
<td>0.23</td>
</tr>
<tr>
<td>Heel</td>
<td>0.28</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Lamp</td>
<td>0.75</td>
</tr>
<tr>
<td>Lamp</td>
<td>0.81</td>
</tr>
</tbody>
</table>

Similarity in Vector Space
Embedding

Types

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>![Shoe]</td>
<td>0.23</td>
<td>0.32</td>
</tr>
<tr>
<td>![High Heel]</td>
<td>0.28</td>
<td>0.40</td>
</tr>
<tr>
<td>![...]</td>
<td>![...]</td>
<td>![...]</td>
</tr>
<tr>
<td>![Lamp]</td>
<td>0.75</td>
<td>0.02</td>
</tr>
<tr>
<td>![Arrow]</td>
<td>0.81</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Vector Representation
Embedding

Types

Images
Words
Phrases
Song titles
Product names

Mapping

Vector Representation

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>string1</td>
<td>0.23</td>
<td>0.32</td>
</tr>
<tr>
<td>string2</td>
<td>0.28</td>
<td>0.40</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>stringN</td>
<td>0.81</td>
<td>0.05</td>
</tr>
</tbody>
</table>
thunderstorm
Embedding

Get vector for word
Compute distance between two words
Get nearest neighbors for word
Get nearest neighbors for vector
OS Embedding

- English
- Spanish
- French
- Italian
- German
- Portuguese
- Simplified Chinese
Custom Word Embeddings
Custom Word Embeddings

Domains
Vocabulary
Languages
Custom Word Embeddings

word2vec  GloVe  fasttext  Custom Neural Network
Demo

Word Embeddings

Doug Davidson, NLP Technologies
import NaturalLanguage

guard let embedding = NLEmbedding.wordEmbedding(for: .english) else { return }

guard let vector = embedding.vector(for: string) else { return }

let distance = embedding.distance(between: word1, and: word2)

embedding.enumerateNeighbors(for: string, maximumCount: 5) { (string, distance) -> Bool in
  // make use of string and distance
  return true
}
import NaturalLanguage

guard let embedding = NLEmbedding.wordEmbedding(for: .english) else { return }

guard let vector = embedding.vector(for: string) else { return }

let distance = embedding.distance(between: word1, and: word2)

embedding.enumerateNeighbors(for: string, maximumCount: 5) { (string, distance) -> Bool in
    // make use of string and distance
    return true
}
import NaturalLanguage

guard let embedding = NLEmbedding.wordEmbedding(for: .english) else { return }

guard let vector = embedding.vector(for: string) else { return }

let distance = embedding.distance(between: word1, and: word2)

embedding.enumerateNeighbors(for: string, maximumCount: 5) { (string, distance) -> Bool in
    // make use of string and distance
    return true
}
import NaturalLanguage

guard let embedding = NLEmbedding.wordEmbedding(for: .english) else { return }

guard let vector = embedding.vector(for: string) else { return }

let distance = embedding.distance(between: word1, and: word2)

embedding.enumerateNeighbors(for: string, maximumCount: 5) { (string, distance) —> Bool in
  // make use of string and distance
  return true
}
import NaturalLanguage

guard let embedding = NLEmbedding.wordEmbedding(for: .english) else { return }

guard let vector = embedding.vector(for: string) else { return }

let distance = embedding.distance(between: word1, and: word2)

embedding.enumerateNeighbors(for: string, maximumCount: 5) { (string, distance) -> Bool in
  // make use of string and distance
  return true
}
import NaturalLanguage

guard let embedding = NLEmbedding.wordEmbedding(for: .english) else { return }

guard let vector = embedding.vector(for: string) else { return }

let distance = embedding.distance(between: word1, and: word2)

embedding.enumerateNeighbors(for: string, maximumCount: 5) { (string, distance) —> Bool in
  // make use of string and distance
  return true
}
import NaturalLanguage

guard let embedding = NLEmbedding.wordEmbedding(for: .english) else { return }

guard let vector = embedding.vector(for: string) else { return }

let distance = embedding.distance(between: word1, and: word2)

embedding.enumerateNeighbors(for: string, maximumCount: 5) { (string, distance) -> Bool in
  // make use of string and distance
  return true
}
import NaturalLanguage

guard let embedding = NLEmbedding.wordEmbedding(for: .english) else { return }

guard let vector = embedding.vector(for: string) else { return }

let distance = embedding.distance(between: word1, and: word2)

embedding.enumerateNeighbors(for: string, maximumCount: 5) { (string, distance) in
    // make use of string and distance
    return true
}
import CreateML

let vectors = [
    "Camembert": [0.118, 0.013, -0.063, -0.020, -0.100, 0.088, ...],
    "Brie": [0.038, 0.008, -0.051, 0.065, -0.198, 0.024, ...],
    "Cirrus": [0.128, 0.127, 0.021, -0.042, -0.057, 0.055, ...],
    "Neufchâtel": [0.308, 0.094, -0.011, 0.155, -0.005, 0.021..., ...

let embedding = try MLWordEmbedding(dictionary: vectors)

try embedding.write(to: url)
import CreateML

let vectors = 
  
  
  "Camembert": [0.118, 0.013, -0.063, -0.020, -0.100, 0.088, …],
  "Brie": [0.038, 0.008, -0.051, 0.065, -0.198, 0.024, …],
  "Cirrus": [0.128, 0.127, 0.021, -0.042, -0.057, 0.055, …],
  "Neufchâtel": [0.308, 0.094, -0.011, 0.155, -0.005, 0.021…], …

let embedding = try MLWordEmbedding(dictionary: vectors)

try embedding.write(to: url)
import CreateML

let vectors = 
    
    
    let embedding = try MLWordEmbedding(dictionary: vectors)

try embedding.write(to: url)
import CreateML

let vectors = 
    
    "Camembert": [0.118, 0.013, -0.063, -0.020, -0.100, 0.088, ...],
    "Brie": [0.038, 0.008, -0.051, 0.065, -0.198, 0.024, ...],
    "Cirrus": [0.128, 0.127, 0.021, -0.042, -0.057, 0.055, ...],
    "Neufchâtel": [0.308, 0.094, -0.011, 0.155, -0.005, 0.021..., ...]

let embedding = try MLWordEmbedding(dictionary: vectors)

try embedding.write(to: url)
import CreateML

let vectors = [
    "Camembert": [0.118, 0.013, -0.063, -0.020, -0.100, 0.088, ...],
    "Brie": [0.038, 0.008, -0.051, 0.065, -0.198, 0.024, ...],
    "Cirrus": [0.128, 0.127, 0.021, -0.042, -0.057, 0.055, ...],
    "Neufchâtel": [0.308, 0.094, -0.011, 0.155, -0.005, 0.021...], ...
]

let embedding = try MLWordEmbedding(dictionary: vectors)

try embedding.write(to: url)
import CreateML

let vectors = [
    "Camembert": [0.118, 0.013, -0.063, -0.020, -0.100, 0.088, ...],
    "Brie": [0.038, 0.008, -0.051, 0.065, -0.198, 0.024, ...],
    "Cirrus": [0.128, 0.127, 0.021, -0.042, -0.057, 0.055, ...],
    "Neufchâtel": [0.308, 0.094, -0.011, 0.155, -0.005, 0.021..., ...]
]

let embedding = try MLWordEmbedding(dictionary: vectors)

try embedding.write(to: url)
Under the Hood

Automatic compression
Efficient representation for fast k-NN
Under the Hood

Automatic compression
Efficient representation for fast k-NN

<table>
<thead>
<tr>
<th>Embedding</th>
<th>Vocabulary</th>
<th>Dimension</th>
<th>Size (MB)</th>
<th>20-NN time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GloVe</td>
<td>400,000</td>
<td>300</td>
<td>31</td>
<td>2.6</td>
</tr>
<tr>
<td>fastText</td>
<td>1,000,000</td>
<td>300</td>
<td>56</td>
<td>1.5</td>
</tr>
</tbody>
</table>
Podcast Embeddings
Podcast Embeddings

Recommender system embeddings
Podcast Embeddings

Recommender system embeddings

<table>
<thead>
<tr>
<th>String</th>
<th>0.23</th>
<th>0.32</th>
<th>-0.45</th>
</tr>
</thead>
<tbody>
<tr>
<td>string2</td>
<td>0.28</td>
<td>0.40</td>
<td>-0.39</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>stringN</td>
<td>0.81</td>
<td>0.05</td>
<td>-0.88</td>
</tr>
</tbody>
</table>
Podcast Embeddings
Podcast Embeddings

66K podcast embeddings from Apple Media
Podcast Embeddings

66K podcast embeddings from Apple Media

167MB → 3MB
Transfer Learning
Text Classification
Custom Text Classification
Custom Text Classification

Annotated training data
Custom Text Classification

Annotated training data ➔ Create ML
Custom Text Classification

Annotated training data → Create ML → Natural Language
Custom Text Classification

Annotated training data → Create ML

Natural Language
Custom Text Classification

- Annotated training data
- Create ML
- Natural Language
Transfer Learning
Text Classification

Smaller annotated training data
Transfer Learning

Text Classification

Smaller annotated training data + Pre-trained model
Transfer Learning
Text Classification

Smaller annotated training data + Pre-trained model → Create ML
Transfer Learning
Text Classification

Smaller annotated training data + Word Embedding → Create ML → Natural Language
import CreateML

let modelParameters = 
MLTextClassifier.ModelParameters(algorithm: .transferLearning(version: 1))
import CreateML

let modelParameters = 
MLTextClassifier.ModelParameters(algorithm: .transferLearning(version: 1))
import CreateML

let modelParameters = 
MLTextClassifier.ModelParameters(algorithm: .transferLearning(.staticEmbedding, version: 1))
Transfer Learning
Text Classification

```swift
import CreateML

let modelParameters = 
MLTextClassifier.ModelParameters(algorithm: .transferLearning(.staticEmbedding, version: 1))
```

OS Embeddings
import CreateML

let modelParameters = MLTextClassifier.ModelParameters(algorithm: .transferLearning(.customEmbedding(url), version: 1))
Transfer Learning
Text Classification

import CreateML

let modelParameters = 
MLTextClassifier.ModelParameters(algorithm: .transferLearning(.customEmbedding(url), version: 1))
USA leads in *apple* production

iPhone was introduced by *Apple* in 2007
Transfer Learning

Embeddings

USA leads in *apple* production

iPhone was introduced by *Apple* in 2007

| apple  | -0.83 | 0.32 | 0.48 | -0.89 | 0.41 | 0.78 | ... | ... | 0.68 |
Transfer Learning
Embeddings

USA leads in apple production

iPhone was introduced by Apple in 2007
import CreateML

let modelParameters = MLTextClassifier.ModelParameters(algorithm: .transferLearning(.dynamicEmbedding, version: 1))

Dynamic Embedding using Neural Network
Transfer Learning
Text Classification

```swift
import CreateML

let modelParameters = 
MLTextClassifier.ModelParameters(algorithm: .transferLearning(.dynamicEmbedding, version: 1))
```

Dynamic Embedding
using Neural Network
Demo
Transfer Learning for Text Classification
Supported Languages

Transfer Learning

Static Embedding

- English
- French
- Italian
- German
- Spanish
- Portuguese
- Simplified Chinese

Dynamic Embedding

- English
- Spanish
Guidelines

Data

Understand target domain

Prepare representative data

Cover variation expected

Create exclusive data splits of sufficient size
Guidelines

Data

- Understand target domain
- Prepare representative data
- Cover variation expected
- Create exclusive data splits of sufficient size
Guidelines

Data

- Understand target domain
- Prepare representative data
- Cover variation expected
- Create exclusive data splits of sufficient size
Guidelines

Data

Understand target domain

Prepare representative data

Cover variation expected

Create exclusive data splits of sufficient size
Guidelines

Algorithms
Guidelines

Algorithms

Start with maxEnt classifier
Guidelines

Algorithms

Start with maxEnt classifier

Training

<table>
<thead>
<tr>
<th>Statement</th>
<th>Sentiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>I love ice cream...can’t wait for summer</td>
<td>+</td>
</tr>
<tr>
<td>I hate cold weather</td>
<td>-</td>
</tr>
<tr>
<td>We are really happy with our new home</td>
<td>+</td>
</tr>
<tr>
<td>We are extremely unhappy with the insurance company</td>
<td>-</td>
</tr>
</tbody>
</table>
Guidelines

Algorithms

Start with maxEnt classifier

Training

I love ice cream...can’t wait for summer +
I hate cold weather -
We are really happy with our new home +
We are extremely unhappy with the insurance company -
Guidelines

Algorithms

Start with maxEnt classifier

Inference

I love mystery novels  +
I hate driving on rainy days -
I'm so happy that mom got me a scooter +
She was really unhappy when she spoke with me -
Guidelines

Algorithms

Start with maxEnt classifier
**Guidelines**

**Algorithms**

Start with maxEnt classifier

**Inference**

<table>
<thead>
<tr>
<th>Sentiment</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>I enjoy mystery novels</td>
</tr>
<tr>
<td>-</td>
<td>I despise driving on rainy days</td>
</tr>
</tbody>
</table>
Guidelines

Algorithms

Start with maxEnt classifier
Compare with transfer learning

Inference

I enjoy mystery novels

I despise driving on rainy days

love
happy

hate
unhappy
Summary

Sentiment Analysis
Text Catalogs
Word Embeddings
Transfer Learning for Text Classification
More Information

developer.apple.com/wwdc19/232

<table>
<thead>
<tr>
<th>What’s New in Machine Learning</th>
<th>WWDC 2019</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create ML for Activity, Text and Recommendations</td>
<td>WWDC 2019</td>
</tr>
</tbody>
</table>