Natural Language Processing in Scala

Ignacio Cases and David Hall

ScalaDays 2015
A Talk in Two Parts

• Epic: Sentence-level linguistic analysis
• Semantics: Learning Compositional Semantics
Natural Language Processing with Epic

David Hall
Some fruit visionaries say the Fuji could someday tumble the Red Delicious from the top of America's apple heap.

It certainly won’t get there on looks.
Some fruit visionaries say the Fuji could someday tumble the Red Delicious from the top of America’s apple heap.
Natural Language Processing

Ambiguity and (Missing) Context

Combinatorial Structure

$O(n^n)$
I made her duck
Named Entity Recognition

Chez Panisse, Berkeley, CA - A bowl of Churchill-Brenneis Orchards Page mandarins and Medjool dates twitpic.com/990erv

Person Organization Location Misc
First, some preprocessing

```scala
> val document: String = ??? // or just a File, or URL
> val tokenized: Seq[Seq[String]] =
  epic.preprocess.preprocess(document)
> val tokens = epic.preprocess.tokenize(sentence)
```
NER with Epic

```scala
> import epic.models.NerSelector
> val nerModel = NerSelector.loadNer("en").get
> val tokens = epic.preprocess.tokenize("Almost 20 years ago, Bill Watterson walked away from "Calvin & Hobbes.""
> println(nerModel.bestSequence(tokens).render)

Almost 20 years ago, [PER: Bill Watterson] walked away from ` ` [LOC: Calvin & Hobbes].''

Not a location!
Annotate a bunch of data?

My favorite authors are ____________, ____________, and the dude that wrote Calvin and Hobbes.
Building an NER system

```scala
> val data: IndexedSeq[Segmentation[Label, String]] = ???
> val system = SemiCRF.buildSimple(data,
>                                   startLabel,
>                                   outsideLabel)
> println(system.bestSequence(tokens).render)

Almost 20 years ago, [PER:Bill Watterson] walked away from ` ` [MISC:Calvin & Hobbes] . ''
List of newspaper comic strips A–F

From Wikipedia, the free encyclopedia

Parent article: List of comic strips; Siblings: A–F • G–O • P–Z

See also: Category:Comic strips, Category:Comic strips by country, cartoonist, webcomic and list of webcomics

http://en.wikipedia.org/wiki/List_of_newspaper_comic_strips_A%E2%80%93F
Gazetteers

C [edit]

- *Café con Leche* (2006–) by Charles Gary (US)
- *Candorville* (2003–) by Darrin Bell (US)
- *Candy* (1944–1971) by Harry Sahle and later Tom Dorr
- *The Candy Man* [disambiguation needed] (1981–) by Bill Murray
- *Cap Stubbs and Tippie* (1918–1966) by Edwina Dumm (US)
- *The Captain and the Kids* (see *The Katzenjammer Kids*)
- *Captain Easy* (1929–1988) originally by Roy Crane (US)
- *Captain Kate* (1967–1971) by Jerry Skelly, Hale Skelly and Archie Goodwin (US)
- *Captain RibMan* by John Sprengelmeyer and Rich Davis – webcomic
Using your own gazetteer

> val data: IndexedSeq[Segmentation[Label, String]] = ???
> val myGazetteer: Map[Seq[String], Label] = ???
> val system = SemiCRF.buildSimple(data,
       startLabel,
       outsideLabel,
       gazetteer = myGazetteer)
Chez Panisse, Berkeley, CA - A bowl of Churchill-Brenneis Orchards Page mandarins and Medjool dates twitpic.com/990erv
Named Entity Recognition

\[ \text{score} (\text{Chez Panisse}) \]

+ \text{score} (\text{Berkeley, CA})
+ \text{score} (\text{A bowl of})
+ \text{score} (\text{Churchill-Brenneis Orchards})
+ \text{score} (\text{Page mandarins and medjool dates})
Features

= 

\[ w(\text{starts-with-Chez}) + w(\text{starts-with-C...}) + w(\text{ends-with-P...}) + w(\text{starts-sentence}) + w(\text{shape:Xxx Xxxx}) + w(\text{two-words}) + w(\text{in-gazetteer}) \]
val dsl = new WordFeaturizer.DSL[L](counts) with SurfaceFeaturizer.DSL
import dsl._

word(begin) // word at the beginning of the span
+ word(end - 1) // end of the span
+ word(begin - 1) // before (gets things like Mr.)
+ word(end) // one past the end
+ prefixes(begin) // prefixes up to some length
+ suffixes(begin)
+ length(begin, end) // names tend to be 1-3 words
+ gazetteer(begin, end)
+ word(begin) * word(end) // product features with *
Imperative Featurization: CoreNLP

> // 1100 lines of this
> if (flags.use2W) {
>     featuresC.add(getWord(p2) + "-P2W");
>     featuresC.add(getWord(n2) + "-N2W");
> }

if (flags.useLC) {
    featuresC.add(cWord.toLowerCase() + "-CL");
    featuresC.add(pWord.toLowerCase() + "-PL");
    featuresC.add(nWord.toLowerCase() + "-NL");
}

if (flags.useUnknown) { // for true casing
    featuresC.add(c.get(CoreAnnotations.UnknownAnnotation.class) + "-UNKNOWN");
    featuresC.add(p.get(CoreAnnotations.UnknownAnnotation.class) + "-PUNKNOWN");
    featuresC.add(n.get(CoreAnnotations.UnknownAnnotation.class) + "-NUNKNOWN");
}
val dsl = new WordFeaturizer.DSL[L](counts) with SurfaceFeaturizer.DSL with BrownClusters.DSL

import dsl._

val featurizer = (unigrams(word + brown + shape, 2) + bigrams(shape, 1) + prefixes(7) + suffixes(7) + unigrams(props, 1))
Using your own featurizer

```scala
> val data: IndexedSeq[Segmentation[Label, String]] = ???
> val myFeaturizer = ???
> val system = SemiCRF.buildSimple(data,
    startLabel,
    outsideLabel,
    featurizer = myFeaturizer)
```
Some fruit visionaries say the Fuji could someday tumble the Red Delicious from the top of America's apple heap.
Ambiguity in NLP

We solved the problem with Scala.
We solved the problem with Scala.
Ambiguity in NLP

We solved the problem with Scala.
We solved the problem with Scala.
Ambiguity in the Constitution?

The president shall have the power to fill up all vacancies that may happen during the recess of the Senate.
Multilingual Parsing

Absolute Error Reduction
(over previous best “out-of-the-box” parser)

- Basque, 12.9
- Polish, 11.5
- Korean, 8.8
- Hungarian, 6.6
- Swedish, 2.8
- Hebrew, 0.2
- German, 0.1
- French, -0.7
- Arabic, -1.0
Epic Pre-built Models

- Parsing
  - English, Basque, French, German, Swedish, Polish, Korean
  - (working on Arabic, Chinese, Spanish)
- Part-of-Speech Tagging
  - English, Basque, French, German, Swedish, Polish
- Named Entity Recognition
  - English
- Sentence segmentation
  - English
  - (ok support for others)
- Tokenization
  - Above languages
Representing Annotations

• Lots of annotations!
  – Words
  – Sentences
  – Part of Speech Tags
  – Parses
  – Named Entities
  – Coreference
  – Speaker
  – ...

• Lots of annotations!
Annotation Dependencies

Sentences

Words

Part of Speech Tags

Parses

Named Entities

Coreference
trait AnnotatedDocument {
  def content: String
  def getWords: IndexedSeq[(Span, Word)]
  def getSentences: IndexedSeq[Span]
  def getPosTags: IndexedSeq[(Span, PosTag)]
  // ...
}
Typesafe Annotations?

> trait AnnotatedDocument {
  def content: String
  def getWords: IndexedSeq[(Span, Word)]
  def getSentences: IndexedSeq[Span]
  def getPosTags: IndexedSeq[(Span, PosTag)]
  // ...
  def posTagForWord(w: Int): PosTag
  def wordsForSentence(span: Span): IndexedSeq[...]
  //...
trait AnnotatedDocument {
  def content: String
  def getWords: Option[IndexedSeq[(Span, Word)]]
  def getSentences: Option[IndexedSeq[Span]]
  def getPosTags: Option[IndexedSeq[(Span, PosTag)]]
  // ...
  def posTagForWord(w: Int): Option[PosTag]
  def wordsForSentence(span: Span): Option[IndexedSeq[...]]
  // ...
}
Generic Documents

> trait Document[Annotation*] // not allowed in Scala!

> Document[Token, Sentence]
> Document[Token, Sentence, PosTag]
> Document[Token, PosTag, Parse]
> ...
Generic Documents

> trait Document[Annotation*] { // not legal scala!
    def annotations[T]
        (implicit compat: Contained[T, Annotations*])
        : Iterator[(Span, T)]

    def addLayer[T](anns: TraversableOnce[(Span, T)])
        : Document[T, Annotation*]
}
Generic Documents

- `Document[Token, Sentence]`
- `Document[Token, Sentence, PosTag]`
- `Document[Token, PosTag, Parse]`
Type Sets via Intersection Types

> Document[Token with Sentence]
> Document[Token with Sentence with PosTag]
> Document[Token with PosTag with Parse]
> Document[Token with PosTag with Parse]
  <: Document[Token]

> // Super hacky, but works surprisingly well
> // See also Christopher Vogt’s talk yesterday.
> // (Also, these are actually called Slabs.)
> trait Document[+Annotations] {
    def annotations[T >: Annotations]
        (implicit tt: TypeTag[T]): Iterator[(Span, T)]

    def addLayer[T](anns: TraversableOnce[(Span, T)])
        (implicit tt: TypeTag[T])
        : Document[Annotations with T]
}
Generic Documents

> def parser(doc: Document[Sentence with Token])
  : Document[ Sentence with Token with Parse]

> parser(??? : Document[Sentence]) // compile error!
> parser(??? : Document[Sentence with Token]) // ok

> parser(??? : Document[Sentence with Token with Entity])
> // --> Document[Sentence with Token with Parse]

> 🙁
Annotation Functions

> trait AnalysisFunction[In, +O] {
>     def apply[In <: I](doc: Doc[B,In]): Doc[In with O]
> }

> val parser: AnalysisFunction[Sentence with Token, Parse]
> val doc: Doc[Sentence with Token] = ???
> parser(doc):Doc[Sentence with Token with Parse]

> val doc2: Doc[Sentence] = ???
> parser(doc) // Compile time error!

> val doc3: Doc[Sentence with Token with Entity]
> parser(doc3):Doc[Sentence with Token with Entity with Parse]
AnalysisFunction

> trait AnalysisFunction[I,+O] {
>     def apply[In <: I](doc: Doc[In]):Doc[In with O]
>
>     def andThen[II >: (I with O), OO] (other: AnalysisFunction[II, OO])
>         : AnalysisFunction[I, O with OO] = ???
> }

> (sentenceSplitter andThen tokenizer andThen parser)
> : AnalysisFunction[Any, Sentence with Token with Parse]