Introduction to

Statistical Learning

Teaching a Computer English

Creighton Kirkendall
@ckirkendall
ckirkendall@gmail.com

What is learning?

What does this mean?
Assigning semantic value to data.

Have I seen this before?
Accumulate semantic information from experiences.

What should I do?
Using this meaning to make decisions?

Simple Case (Good & Bad)

Classifying data from known good and bad sources.

Known Good:
sample 1: red red blue green blue red red yellow
sample 2: red blue red red red green yellow

Known Bad:
sample 1: blue blue red yellow blue green blue
sample 2: blue red red blue blue blue green

New Data:
red blue red yellow red green yellow

Rules Based

Classifying data from known good and bad sources.

```
(if (> (count reds) (count blues))
   :good
   :bad)
```

What if they are equal?
What about yellow and green?

New Data:
yellow red yellow yellow yellow blue

Statistical Classification
Bayes Theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Bayes Theorem
Example

$$P(Bad|red) = \frac{P(red|Bad)P(Bad)}{P(red)}$$

P(red|Bad) = 3/14
P(Bad) = 14/29
P(red) = 11/29

P(Bad|red) = 3/11 ~ 27.3%

What about multiple values?

red blue red yellow red green yellow

Statistical Classification
Bayesian Inference

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}$$

Statistical Classification
Bayesian Inference

$$P(H_i) = P(H|E_0..E_i) = \frac{P(E_i|H)P(H_{(i-1)})}{P(E_i)}$$

Calculating the Posterior Probability

Statistical Classification
Bayesian Inference

Data: red yellow

P(H_0) = 0.5

P(red) = 11/29

P(red|Bad) = 3/14

P(yellow) = 3/29

P(yellow|Bad) = 1/14

$$P(Bad|red, yellow) = \frac{P(yellow|Bad)\frac{P(red|Bad)0.5}{P(red)}}{P(yellow)}$$

$$P(Bad|red, yellow) = \frac{\frac{1}{14}\left(\frac{\frac{3}{14}\times 0.5}{\frac{11}{29}}\right)}{\frac{3}{29}}$$

$$\frac{841}{4312} \cong .195$$

Statistical Classification
Bayesian Inference

Statistical Model

```
{:total-words 29
 :all {red 11/29, blue 11/29, green 4/29, yellow 3/29}
 :classes {:good {red 8/15, blue 1/5, green 2/15, yellow 2/15,
                  :genus.model/total 15/29}
           :bad {blue 4/7, red 3/14, yellow 1/14, green 1/7,
                 :genus.model/total 14/29}}}
```

Statistical Classification
Bayesian Inference

```
(defn posterior-prob
  [{:keys [classes all]} prior-prob word]
  (let [start-prob (/ 1 (count classes))]
    (into {}
          (for [[class word-probs] classes]
            (let [pe (get all word min-prob)
                  peh (get word-probs word min-prob)
                  ph (get prior-prob class start-prob)]
              [class (/ (* peh ph) pe)])))))


(defn bayesian-inference [model words]
  (reduce (partial posterior-prob model) {} words))
```

Statistical Classification
Bayesian Inference

My computer reads these books in about 3 seconds. Who needs CliffsNotes?

```
(deftest shakespeare-dickens-test
  (let [config {:training-data
                {:dickens
                 [(io/resource "dickens/hard-times.txt")
                  (io/resource "dickens/olivertwist.txt")
                  (io/resource "dickens/twocities.txt")]
                 :shakespeare
                 [(io/resource "shakespeare/hamlet.txt")
                  (io/resource "shakespeare/macbeth.txt")
                  (io/resource "shakespeare/midsummer.txt")]}}
        model (model/create-model config)
        sample1 (->> (io/resource "samples/great-exp-sample.txt")
                     io/reader
                     line-seq
                     (interpose " ")
                     (apply str))]
    (is (= :dickens (classify model sample1)))))
```

Bayesian Inference Applications

- Bayesian inference is good to use when you have independent categories and existing training data.
    - Spam Filtering - Shakespeare and Dickens are much more alike than your average email and spam.
    - Log Files - monitoring good, bad and new states and notifying when bad or new states are found.
    - Political Leaning - List of known liberal/progressive texts and list of known conservative/libertarian texts.
    - Sentiment analysis

Now for something a bit more complex.

Teaching a Computer English

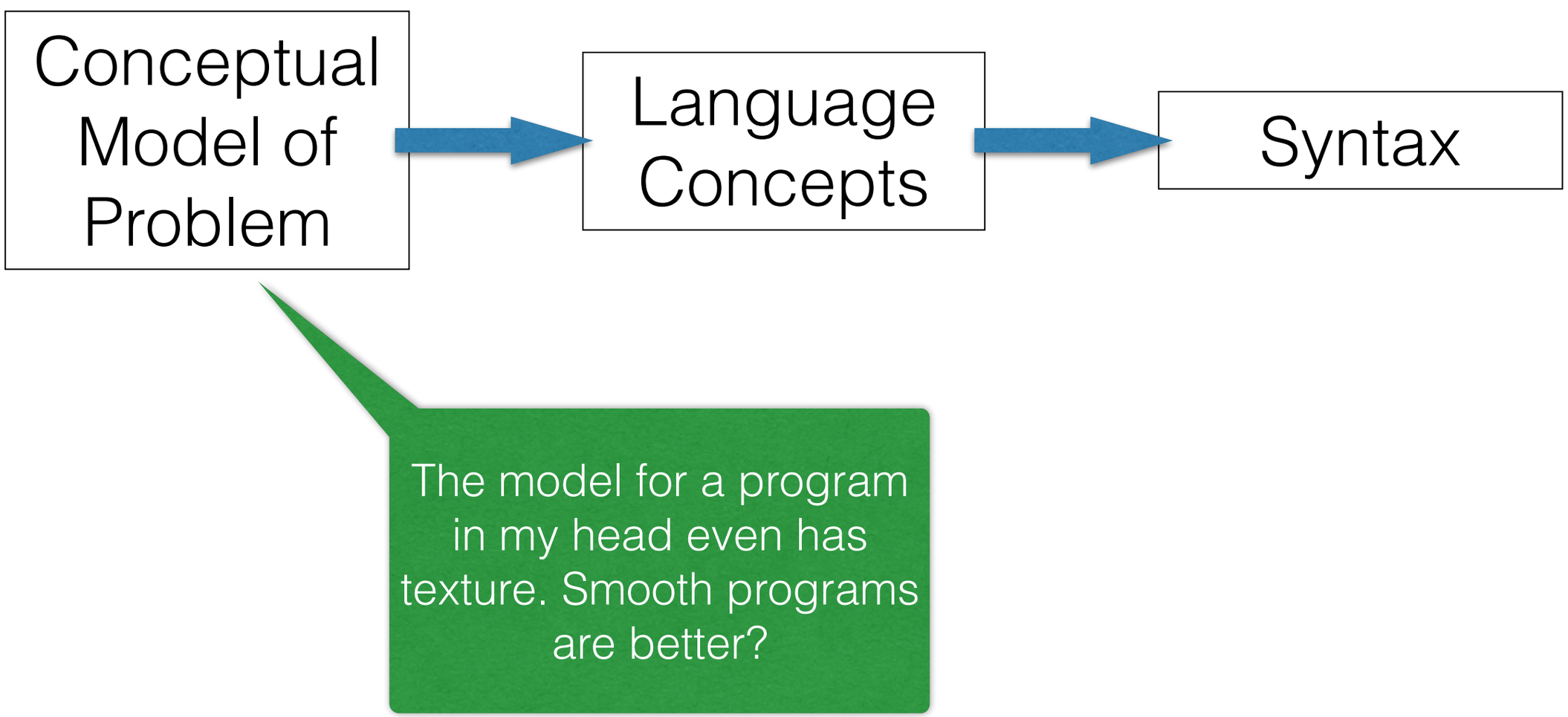
Story Time

My first Clojure program and dyslexia

Driving Question

Why am I good with programming languages but struggle with basic english structure and grammar?

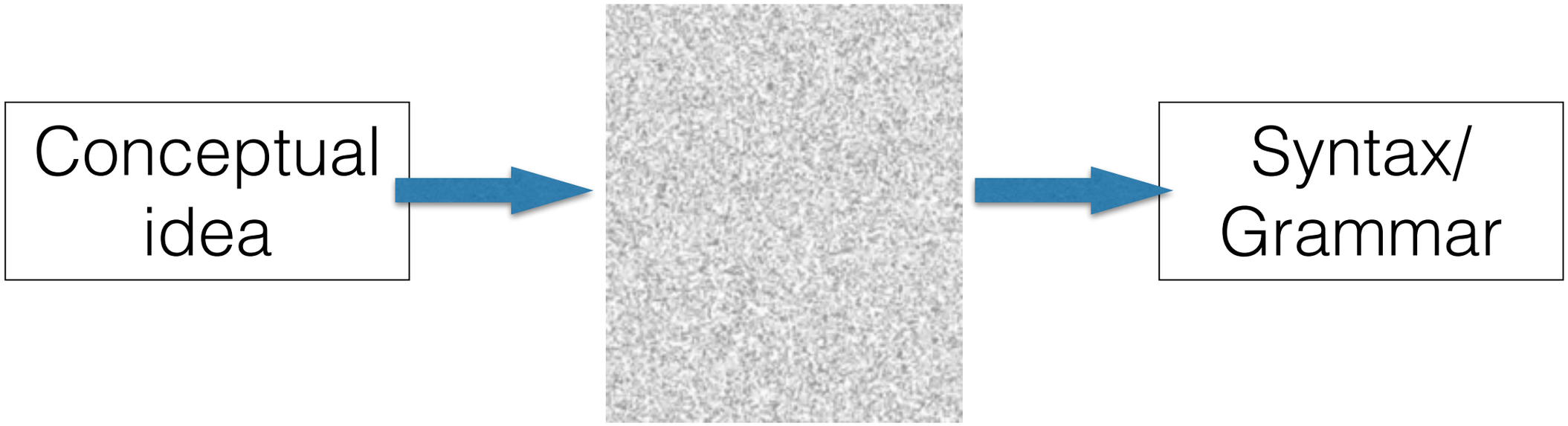
Programing


Conceptual Model of Problem
Language Concepts
Syntax
The model for a program in my head even has texture. Smooth programs are better?

Dyslexia and English

Dyslexia is breakdown in either the assignment of semantic meaning or the retrieval of that meaning when presented with physical representation of text.

Dyslexia and English



Types of Problems

Afterwards, got to the party at there house.

Sally base her decision on the outcome.

I wanted a conceptual model of english.

The tools I had were Mathematics and Computer Science.

I wanted to understand better what the static was and how it might be bypassed.

I tried a few times to build a part of speech tagger in Java but failed to finish because I got bogged down and struggled with how to represent things.

My first Clojure program was part of speech tagger.
It was a little over 100 lines of code.

While the code is a bit longer and cleaner now much it bears a striking resembles to the original code that started it all.

What is part of speech tagger?

Take a sentence and assign each word the correct part of speech, such as none, verb or past participle.

Example: Sally Cornelly went to their house.

NNP/Sally NNP/Cornelly VBD/went TO/to
PRP$/their NN/house ./.

How is a POS tagger Different from Bayesian Classifier?

- Highly depended variables. Bayesian mathematics assume all data is independent and its classification is not related to the classification of any other data.

- English is context based the semantic meaning of a word is very dependent on the words around it.

  He restricted access.
  He entered restricted air space.

Building the Model

- How do we represent the context sensitive nature in a model.

- In the bayesian model we tracked the relationship between word and class through the probability of it being in that class.

- In this model we need to represent the relationship of word to it's tokens (or class) and how it relates to the surrounding words.

Building the Model

Let's track two main relationships.

- P(T|W) - Probability of a word being a given part of speech or token.
- P(T|PT) - Probability of token following another token.

Similar Model:

A Maximum Entropy Model for Part-Of-Speech Tagging

Adwait Ratnaparkhi

University of Pennsylvania

Dept. of Computer and Information Science

http://www.aclweb.org/anthology/W96-0213

Building the Model
Training Data

The Open American National Corpus

The Open American National Corpus (OANC) is a massive electronic collection of American English, including texts of all genres and transcripts of spoken data produced from 1990 onward. All data and annotations are fully open and unrestricted for any use.

15 million words of contemporary english tagged with meta data including part of speech.

http://www.anc.org/

Building the Model
Training Data

What does the data look like?

<xces:tok affix="ed" base="deploy" msd="VBN">deployed</xces:tok>
<xces:tok affix=" " base="as" msd="IN">as</xces:tok> <xces:tok
base="part" msd="NN">part</xces:tok> <xces:tok base="of" msd="IN">of</
xces:tok> <xces:tok base="nato" msd="NNP">NATO</xces:tok><xces:tok
base="'s" msd="POS">'s</xces:tok> <xces:tok base="sfor"
msd="NNP">SFOR</xces:tok> <xces:tok affix="s" base="force"
msd="NNS">forces</xces:tok> <xces:tok base="in" msd="IN">in</xces:tok>
<xces:tok affix="s" base="bosnium" msd="NNP">Bosnia</xces:tok>
<xces:tok base="and" msd="CC">and</xces:tok><xces:tok base=","
msd=",">,</xces:tok> <xces:tok base="in" msd="IN">in</xces:tok>
<xces:tok base="2003" msd="CD">2003</xces:tok><xces:tok base=","
msd=",">,</xces:tok> <xces:tok base="120" msd="CD">120</xces:tok>

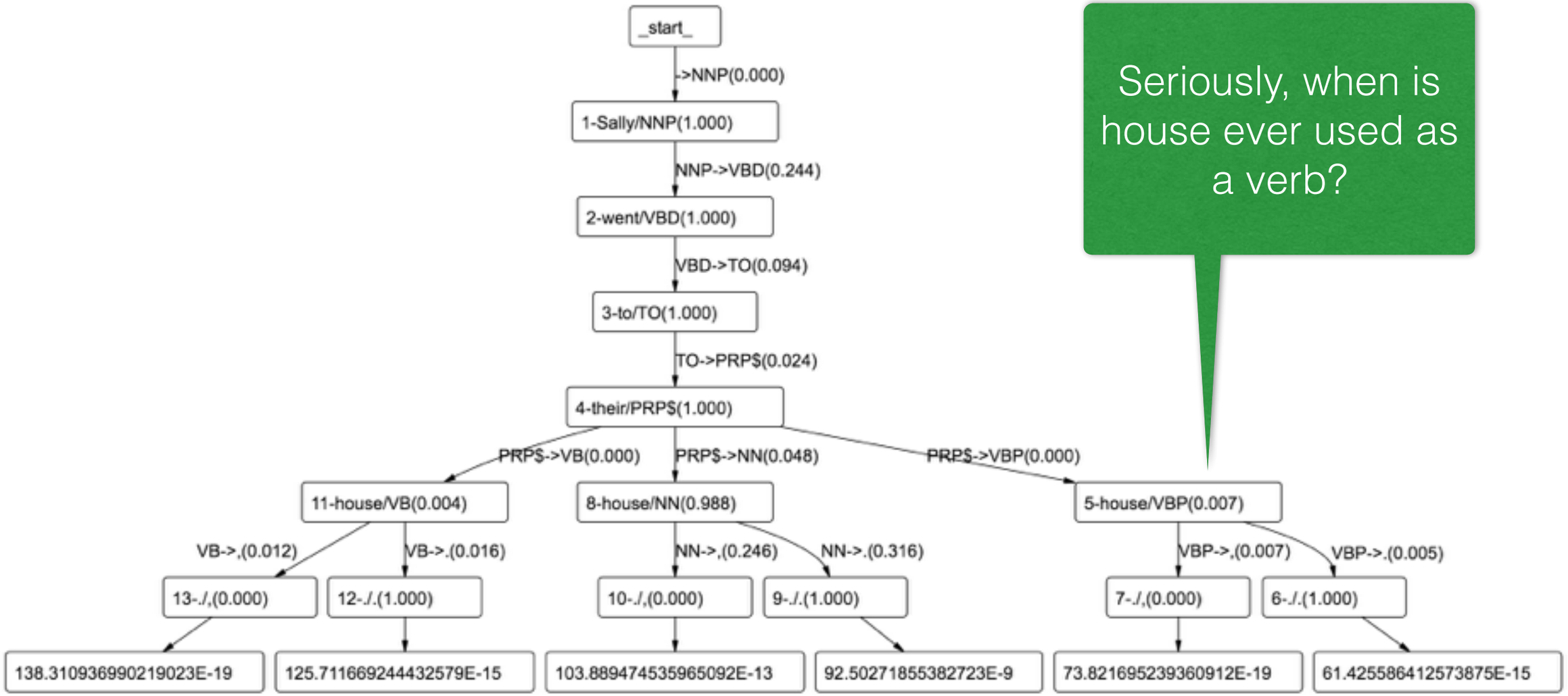
Building the Model

Structure:

```
user> (require '[ziad.pos :as pos])
nil
user> (get-in pos/oanc-model [:word-model "restricted"])
{"VBD" 2/19, "VBN" 16/19, "JJ" 1/19}
user> (get-in pos/oanc-model [:token-model "VBN"])
{nil 9/83506, "JJ" 298/41753, "VBN" 1072/41753, "POS" 240/41753, "UH"
651/41753, ")" 2/41753, "JJS" 33/83506, "PRP$" 74/41753, "NNP"
1357/83506, "WP$" 3/41753, "VB" 3803/41753, "CD" 87/41753, "EX" 46/41753,
"NNPS" 7/41753, "RBS" 35/41753, "VBZ" 4239/41753, "(" 11/83506, "WDT"
109/83506, :start 15/41753, "WRB" 37/83506, "WP" 25/41753, "VBD"
7307/83506, "CC" 610/41753, "TO" 65/41753, ":" 19/83506, "VBG"
1479/83506, "RB" 7793/41753, "," 173/83506, "MD" 123/41753, "RP" 9/41753,
"PRP" 1609/83506, "JJR" 14/41753, "VBP" 12450/41753, "DT" 1025/41753,
"''" 19/83506, "RBR" 113/83506, "NN" 1479/41753, "IN" 858/41753, "NNS"
1185/83506}
```

Using our Model

"Sally went to their house."
NNP/Sally VBD/went TO/to PRP$/their NN/house ./.



Seriously, when is house ever used as a verb?

The Search Problem

Fancy way of saying multiply all probabilities along a path.

Algorithm

Finding the probability of a path

$$P(t_1 ... t_n | w_1 ... w_n) = \prod_{i=1}^{n} p(t_i|w_i)p(t_i|t_{i-1})$$

Number of paths ~= t^n
where t is the number of tokens per word
and n is the number of words.

complexity = $O(t^n)$

Ouch!

Helicopters will patrol the temporary no-fly zone around New Jersey's MetLife Stadium Sunday, with F-16s based in Atlantic City ready to be scrambled if an unauthorized aircraft does enter the restricted airspace.

255,472 nodes
73,728 paths

Beam Search

Narrowing the Solution Space

Step 1: Find the optimal set of tokens t0…tn for words w0…wn where n=beam with

Step2: Find the optimal set of tokens t'1…t'n+1 for word w1…wn+1

Step3: if(t1 == t'1)
    w0 -> t1
    inc cur-index
else
    inc beam width
    goto Step 1

Complexity = O(n)

Beam Search

Narrowing the Solution Space

words: s1 w2 w3 w4 w5 w6

[s1 w2 w3] w4 w5 w6
[tx ty tz]

s1 **[w2 w3 w4]** w5 w6
[ta tz tw]

ty != ta
inc beam width

[s1 w2 w3 w4] w5 w6
[tx ty tz tb]

s1 **[w2 w3 w4 w5]** w6
tx [ty tz tb te]

ty = ty

w2 = ty

Special Cases

Statistical Models still need help sometimes.

```
(cond
  (and (= cur-word "'s")
       (#{"NNP" "NN"} (:tok cur-tok)))
  ["POS"]

  word-model
  (keys word-model)

(and (common/digit? (first cur-word))
     (= (last cur-word) \s))
["NNPS"]

;;Unknown words in caps are consider proper nouns
(common/upper-case? (first cur-word))
["NNP"]

(common/isNumber? cur-word)
["CD"]

(common/isHyphenated? cur-word)
(conj (mapcat #(word-toks % cur-tok model)
              (str/split cur-word #"-"))
      "JJ")

:else
(keys (get-in model [:rev-token-model (:tok cur-tok)])))
```

"'s" is mapped to the is token and a noun followed by "is" is very common. This makes it hard to detect possession.

Numbers are are infinite so detecting them you can't do it by word match.

Let’s see it in action!

https://ckirkendall.github.io/ziad/

Basic Grammar Checking

Basic Grammar Checking

The Model

We use the same training data as our pos tagger to build probability map of english tries [t1 t2 t3]

```
user=> (get-in oanc-model [:tri-model ["NN" "VBZ" "JJ"]])
1387/6353493
```

Basic Grammar Checking

Algorithm

For tokens [t1 t2 t3] if the probability of this occurring is less than some min probability mark tokens as having a grammar issue.

```
(defn grammar-check
  "Given set of tokens annotate grammar errors based
   on probability of a grammar tri existing."
  [toks model]
  (if (:tri-model model)
    (reduce
     (fn [stack tok]
       (let [cnt (count stack)]
         (if (>= cnt 2)
           (let [tri (conj (subvec stack (- cnt 2) cnt) tok)
                 sem-tri (create-semantic-tri tri)
                 prob (get-in model [:tri-model sem-tri] grammar-threshold)]
             (if (<= prob grammar-threshold)
               (into (subvec stack 0 (- cnt 2))
                     (map #(assoc % :grammar-issue true
                                    :tr-prob prob) tri))
               (conj stack tok)))
           (conj stack tok))))
     []
     toks)
    toks))
```

Where do I go from here?

Questions?

Resources:

- Ziad - Clojure/Script POS Tagger and Grammar Checker https://github.com/ckirkendall/ziad
- Genus - Clojure Bayesian Classifier https://github.com/ckirkendall/genus
- OANC Corpus - http://www.anc.org/
- Stanford NLP Library - http://nlp.stanford.edu/software/
- Stanford-Talk - Clojure wrapper for Stanford NLP https://github.com/gigasquid/stanford-talk