how to do things with words*

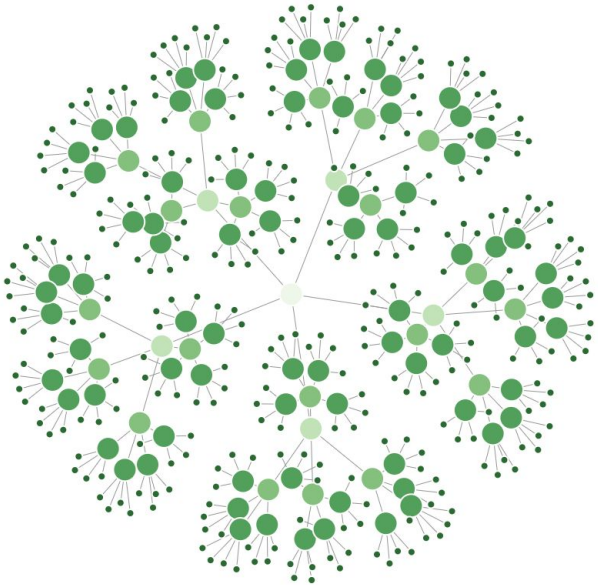
K. Hunter Wapman
hneutr.github.io
hunter.wapman@gmail.com

* title stolen from J. L. Austin's very good (and very readable!) series of lectures on performatives

this guy

- ms (“nlp”) → phd (w/ DBL) (less nlp)
- into words + structure in art
- previous work:
  a. can we detect puns?
     - today!
  b. can we help people be funny?
  c. how does style vary in time?
  d. webweb
- currently:
  a. narrative complexity
  b. hierarchies in dating apps

Cayley Tree (via webweb)



can we find puns?

task: locate the *pun* word

this is a *sequence to sequence* task

“atheism is a non-prophet institution”*

^		^	^	^		^		^	
0		0	0	0		1		0	

*George Carlin

https://i.ytimg.com/vi/YZ_mjtTCdcg/maxresdefault.jpg

https://i.pinimg.com/originals/1b/2b/18/1b2b18085c8924cbf8ff6c5042e6f82b.jpg

outline

1. a neural network approach
2. a sliding window approach

what are puns?

“a form of play that involves multiple meanings”

wikipedia says “word play”
wikipedia is wrong

puns can involve more than words

types of puns

homographic

“would you say a 14 layer neural network for detecting pools is on the *deep* end?”

(“pun word” spelled the same)

heterographic

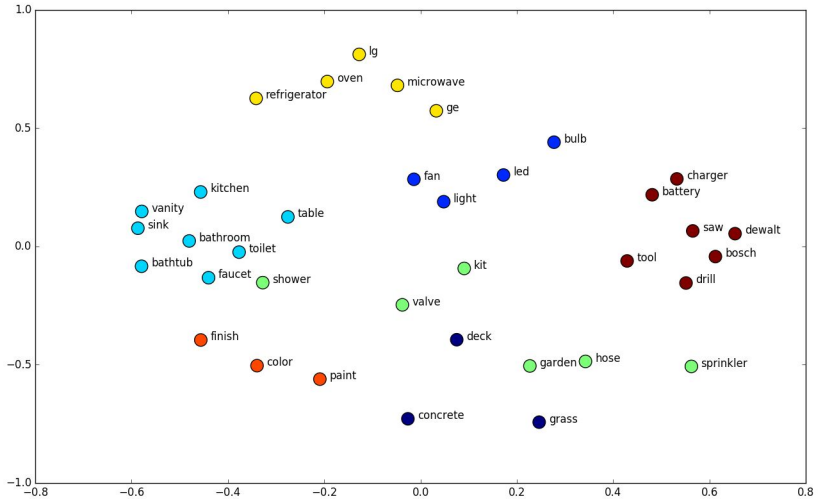
“cloud detection is a *cirrus* problem.”

(“pun word” spelled differently)

visual

https://i.pinimg.com/236x/42/48/c6/4248c6e911b3fa009b92d276ae521035--visual-puns-funny-design.jpg?b=t

a neural approach: word embeddings



Super briefly:

- take a big corpus
- find the contexts (words) a word appears in
- use this to represent a word as a vector

they capture *semantic* (“meaning”) relationships

← reduction from high dimensional space into 2D

https://shanelynnwebsite-mid9n9g1q9y8tt.netdna-ssl.com/wp-content/uploads/2018/01/word-vector-space-similar-words.png

a neural approach: input

“cloud detection is a *cirrus* problem.”

“cloud” → $\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ “detection” → $\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$

etc

input: [x1, x2, …, xn, y1, y2, …, yn, …]

Details on the embeddings we used:

- in our case, we used GloVe
- vectors had dimension 300

on input:

- had to “pad” the vector with empty (0) values so it was always the same length
- length → max length of pun in corpus

a neural approach: architecture

- Layer 1: Long Short-Term Memory (LSTM)
  - input:
    - [x1, x2, …, xn, y1, y2, …, yn, …]
  - output:
    - [prob(x), prob(y), …]
- Layer 2: softmax
  - input: [prob(x), prob(y), …]
  - output: x (or y, or etc)
    - the algorithm's guess at the pun word

but this didn’t work super well. why?

It’s often assumed that “neural networks will figure out the features”

this is really a crazy idea in text!

(and wordplay specifically)

There’s a lot “between the lines” in text.

between the lines of text

Example credit to Yejin Choi
https://i.kym-cdn.com/photos/images/original/000/610/809/13e.jpg

between the lines of text

what happened?

a. someone stabbed someone else over a cheeseburger
b. someone stabbed someone else with a cheeseburger
c. someone stabbed a cheeseburger
d. a cheeseburger stabbed someone
e. a cheeseburger stabbed another cheeseburger

Example credit to Yejin Choi
https://i.kym-cdn.com/photos/images/original/000/610/809/13e.jpg

characteristics of the problem

“cloud detection is a *cirrus* problem.”

this pun involves phonetics (how words sound)

but a pun can involve:

- idioms (cultural “phrases”)
- hyphenates/portmanteaus
- misspellings

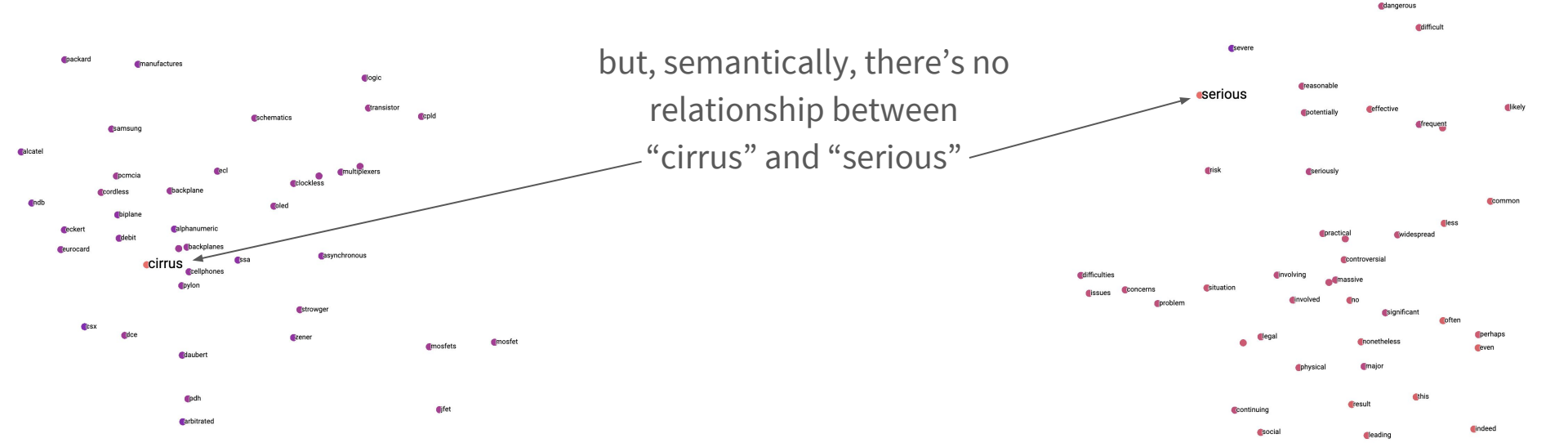
in other words: non-semantic information

a neural approach

“cloud detection is a *cirrus* problem.”

we’re feeding our neural net word embeddings

but, semantically, there’s no relationship between “cirrus” and “serious”



a sliding window approach: input

“cloud detection is a *cirrus* problem.”

idea:

- use the words *around* what you want to classify as features to classify it
  - can use anything about those words for a feature

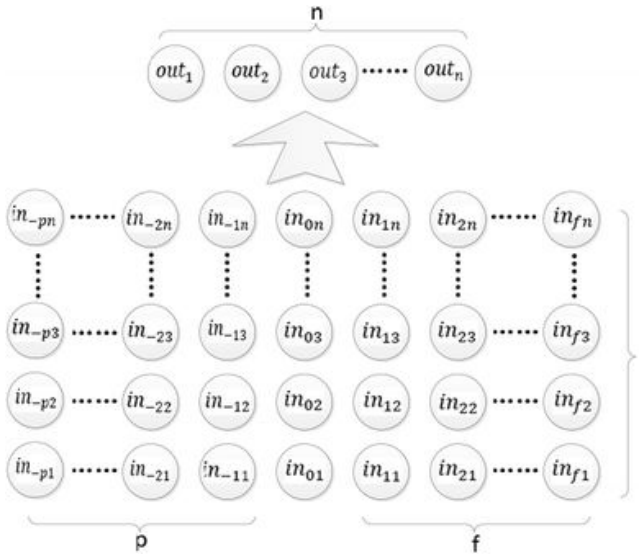
if the word is *cirrus* and the window is 2, these are our features:

cloud		
detection		
is	———	word-2 POS: verb
a	———	word-1 POS: article
cirrus	———	word POS: adjective
problem	———	word+1 POS: noun
<end>	———	word+2 POS: N/A

sliding window classifiers

Maximum Entropy Markov Model that generalizes logistic regression for multiclass classification

- used a lot for Part of Speech (POS) tagging (now with neural networks!)
- no padding of inputs
  - (really inputs all padded *identically*)
- allows us to add problem specific features
- we improved drastically by using the *lesk* distance between words
  - a “distance” between the senses of two words’ definitions

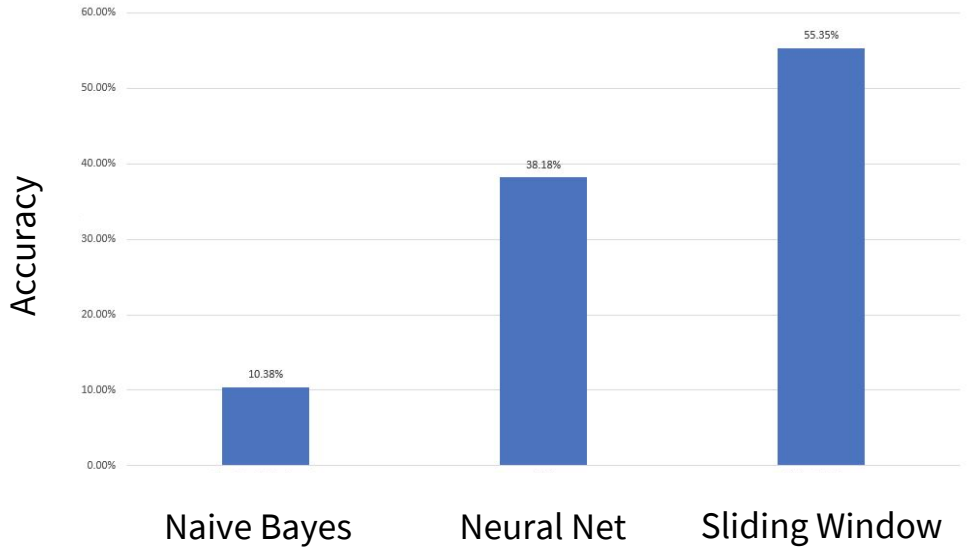


https://media.springernature.com/lw785/springer-static/image/art%3A10.1007%2Fs10772-016-9356-2/MediaObjects/10772_2016_9356_Fig1_HTML.gif

a sliding window approach: architecture

- step 1: MaxEnt/logistic regression:
  - input (in series):
    - [x features],
    - [y features]
  - output:
    - [prob(x), prob(y), …]
- step 2:
  - argmax([prob(x), prob(y), …])
    - the algorithm's guess at the pun word

Results


Accuracy
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
10.38%
38.18%
55.35%
Naive Bayes
Neural Net
Sliding Window

wrap-up

- we wanted to find the location of a “pun” word
- we tried using a neural network
  - it didn’t do very well because we didn’t give the classifier the information relevant to the problem
- we tried a sliding window classifier
  - it worked better because we could give the classifier the information relevant to the problem

takeaway:

characteristics of your data
will likely affect the success
of a given approach!

Gracias! Questions?

K. Hunter Wapman
hneutr.github.io
hunter.wapman@gmail.com

types of puns: “loose”

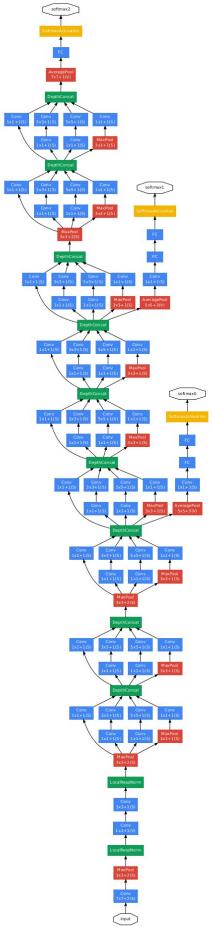
word choice *resonates*

“you’re barking up the wrong tree”

(the only conscionable kind of pun)

3. why didn’t the neural network… work?

we needed more layers, obviously



https://alexisbcook.github.io/2017/using-transfer-learning-to-classify-images-with-keras/

3. why didn’t the neural network… work?

It is often assumed that “neural networks will figure out the features”

ok. maybe. but:

… can they?
… how could they?
… will they?

5. what would I do differently now?

annotate the dataset with *preparatory*/support words

the idea is:

- a pun plays something (or things) previous in the sentence
- why not add that into the dataset?

this is an idea I stole from Sam F. Way:

- take an existing dataset and add to it

5. what would I do differently now?

What about multi-pun sentences?

don’t:

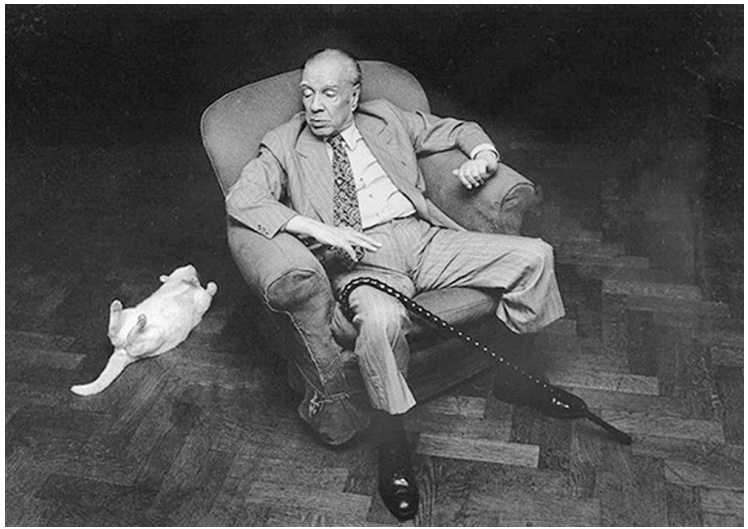
- try to find “the” pun word

do:

- identify pun *words* and their support

sliding window classifiers — what I like about them

- no padding of inputs
  - or really, inputs all padded *identically*
    - neural networks are reasonable for the library of babel
    - the real world is (thankfully!) not the library of babel.
- arbitrary features!
  - we improved drastically by just including the word's lemma as a feature...



https://www.theparisreview.org/interviews/4331/jorge-luis-borges-the-art-of-fiction-no-39-jorge-luis-borges