Welcome to Make-a-Bot #2!

- Install spaCy and its huge model: https://spacy.io/docs#install-spacy

- Example project is at: https://github.com/andkon/instantreviews

- Our home (and Slack invite button) is at: https://github.com/andkon/make-a-bot

Twitter Bots with spaCy

Make-a-Bot Meetup #2
July 27, 2016

An easy bot:

- Take a sentence.
- Find its subject and the verb operating on that subject.
- Make a new sentence out of it.

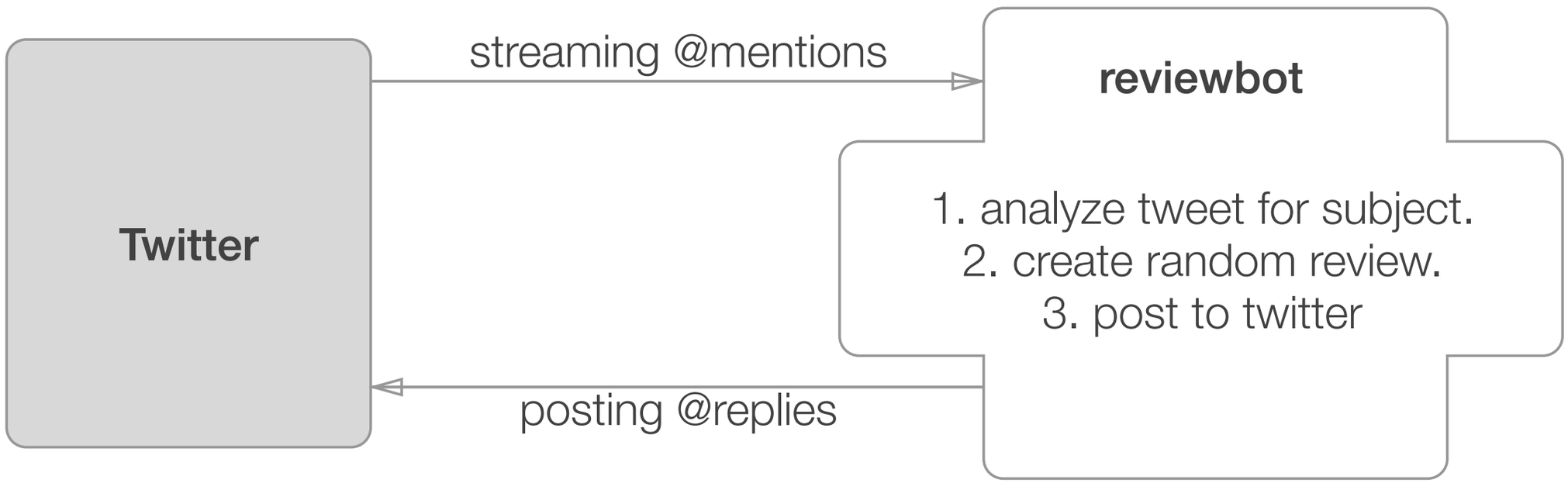
This could even be funny

- Input: “How much do you like candy?”
- Output: “I really like candy.”
- Angry Grandpa output: “I like candy as much as I like millennials, which isn’t a lot!”

NLP apps are rarely easy

- How do you actually analyze the text itself?
    - How do you understand what the heck the subject of the sentence is? Subjects aren't just nouns.
- How do you line up a pipeline of content?
- How do you publish this content?

The complete bot:



github.com/andkon/instantreviews

reviewbot.py is the fun part

- It uses spaCy to break apart the sentence you receive from Twitter.

- And then intelligently take the subject and make a new random review from it

Intro to spaCy

- https://spacy.io/docs#install-spacy

- I hope you installed it before coming here…

Analyzing a document

```
# python code


import spacy
en_nlp = spacy.load(‘en') # takes forever
doc = en_nlp(u“This is a sentence, and it’s about some
stuff.”) # needs unicode
```

Working with a doc

```
# looking through the doc


for token in doc:
    print t.dep_ + ", " + t.pos_ + "/" + t.tag_  + ": " + t.lower_

# output
nsubj, DET/DT: this
ROOT, VERB/VBZ: is
det, DET/DT: a
attr, NOUN/NN: sentence
punct, PUNCT/,: ,
cc, CONJ/CC: and
nsubj, PRON/PRP: it
conj, VERB/VBZ: ’s
prep, ADP/IN: about
det, DET/DT: some
pobj, NOUN/NN: stuff
punct, PUNCT/.: .
```

Important token properties

```
doc = en_nlp(u”I am taller than Fred.”)

doc[2]
> taller
doc[2].lemma_
> tall


# how is this token related to the root of the sentence?
doc[1].dep_
> ROOT
doc[0].dep_
> nsubj

# what is the part of speech of this token?
doc[0].pos_
> PRON
doc[1].pos_
> VERB
```

reviewbot.py

```
doc = en_nlp(u”What do you think of bananas?”)
bananas = doc[5]

bananas.pos_
> NOUN
bananas.dep_
> pobj

# so we’re looking for nouns that are the pobj!
for t in doc:
    if t.pos_ == “NOUN” and t.dep_ == “pobj”:
        reviewed_obj = t
```

But things aren’t that easy

```
reviewed_obj = “bananas”
> works


reviewed_obj = “Dark Horse”
> doesn’t happen, because it’s two tokens long
```

Solve with noun chunks

```
# a noun chunk is a noun phrase that doesn’t contain any
other noun phrases inside of it. Good for noticing proper
names.

for chunk in doc.noun_chunks:
    token = chunk.root
    if t.pos_ == “NOUN” and (t.dep_ == (“dobj” or “pobj”)):
        review_obj = chunk
    # now we make the review from the noun chunk!
```

Making the review

```
def reviewer(span):
    review_obj_string = span.text_with_ws
    adj = random.choice([“really like”, “really hate”])
    return “I %s %s.” % (adj, review_obj_string)
```

What's next?

- Integrate it with Twitter (lol didn't have time)
- Make it handle this sentence: "What do you think of my socks?" -> "I *** your socks"
- Make it run on Heroku