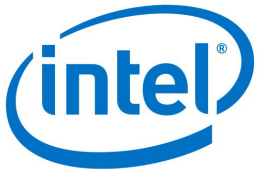
Developing smart chatbots

Angik Sarkar

About me

- Easy to remember…. “An Geek”
- Founder/CEO @Waylo
- PhD @Purdue, BS @IIT, India



@kyunbit

What is Waylo?

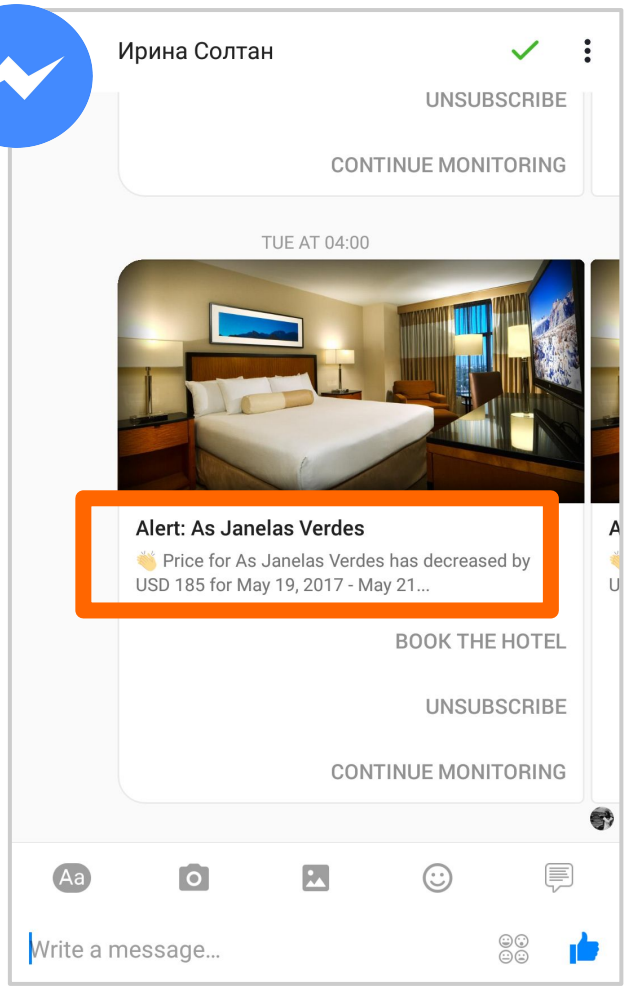
Track Hotel Rates

$ $

Book @

Predicted Rate

https://m.me/theWaylo



Agenda: Build a chatbot

- No-code solutions
- Free NLU platforms: wit.ai/api.ai
- Basics of NLP: NLTK
- RasaNLU/SpaCy for NLP in prod
- Human handover

Why build a chatbot?

In 5 years, every business will have a conversational interface

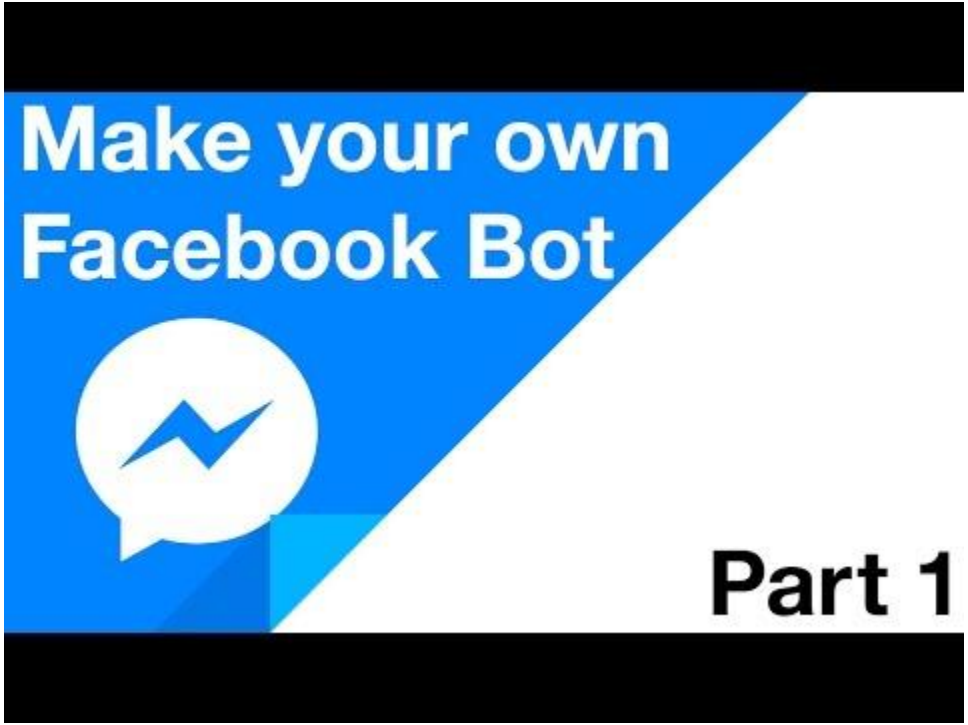
Code-free solutions

Building the brains of the bot

Quirks of api.ai

- Use it for Small Talk Domain
- Various ‘pre-built agents’. None work well unless trained with massive amount of data
- Use Weather domain if you want to extract date time/location with the highest accuracy

Let's first create a small talk bot

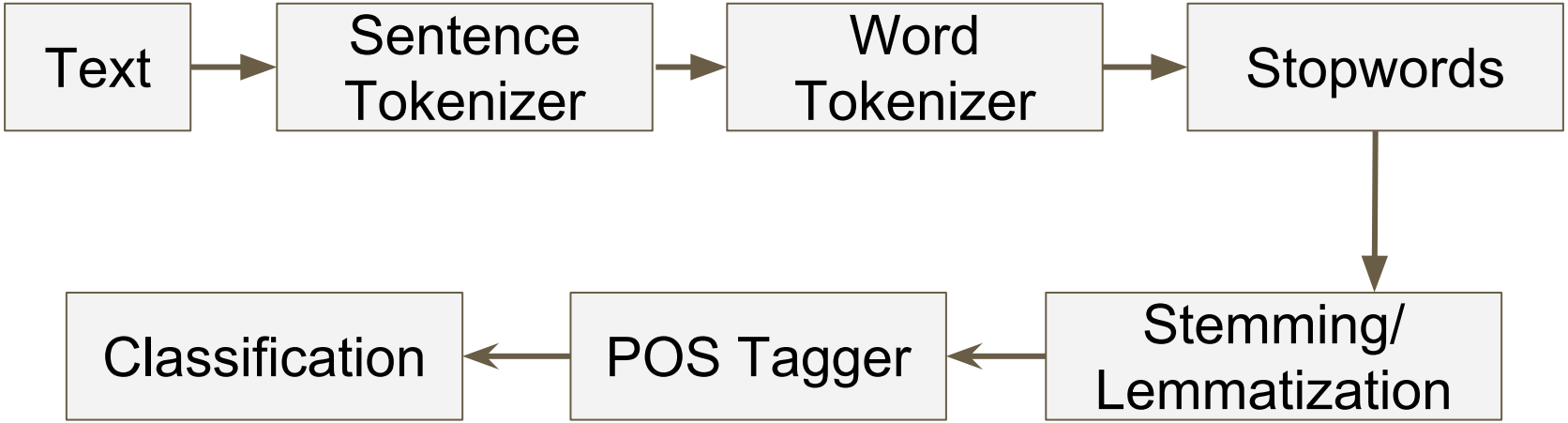


Waylo

Quirks of wit.ai

- More reliable than api.ai in detecting location and dates
- Does not have many Tier 2 cities in their database e.g. Interlaken
- Datetime works only in specific formats
  21-23 July won’t work; but July 21-23 works

Building the brains of your bot: NLP pipeline



Installing Python, NLTK

Mac https://conda.io/docs/install/full.html#os-x-anaconda-install

Linux https://www.digitalocean.com/community/tutorials/how-to-install-the-anaconda-python-distribution-on-ubuntu-16-04

NLP in JavaScript

- naturalNode: GITHUB
- nlp-compromise: GITHUB
- https://github.com/josephmisiti/awesome-machine-learning#javascript-nlp

Open-source packages for datetime

- chrono-node (Nodejs)
- Natty (Python/Java)
- *Duckling (Clojure, Wit.ai)*
- *Chronic (Ruby)*

Sentence Tokenizer

In [1]:
```
#import nltk
from nltk.tokenize import sent_tokenize
sentence = "Show me all theatre performances in San Francisco between 4th and 6th July. Preferably somewhere near Un
tokens = sent_tokenize(sentence)
print (len(tokens), tokens)
```

```
2 ['Show me all theatre performances in San Francisco between 4th and 6th July.', 'Preferably somewhere near Union
 Square']
```

Sentences segregated

Word Tokenizer

```
In [2]: from nltk.tokenize import word_tokenize
        sentence = "Show me all theatre performances in San Francisco between 4th and 6th July."
        tokens = word_tokenize(sentence)
        print (len(tokens), tokens)

        14 ['Show', 'me', 'all', 'theatre', 'performances', 'in', 'San', 'Francisco', 'between', '4th', 'and', '6th', 'Jul
        y', '.']
```

Words separated

Stopwords

In [3]:
```
#nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
tokens = [w for w in tokens if not w in stop_words]
print (len(tokens), tokens)
```

```
8 ['Show', 'theatre', 'performances', 'San', 'Francisco', '4th', '6th', 'July']
```

Words that convey no information removed

Remove Punctuation

In [6]:
```
import string
s = sentence.translate(str.maketrans('','',string.punctuation))
print(s)
```

```
Show me all theatre performances in San Francisco between 4th and 6th July
```

Stemming

```
In [4]: from nltk.stem import PorterStemmer
        ps = PorterStemmer()
        for t in tokens:
            print(ps.stem(t))

        Show
        theatr
        perform
        San
        Francisco
        4th
        6th
        Juli
```

Lemmatization

```
In [16]: # import nltk
         # nltk.download('wordnet')
         from nltk.stem.wordnet import WordNetLemmatizer
         wordnet_lemmatizer = WordNetLemmatizer()
         for t in tokens:
             print(wordnet_lemmatizer.lemmatize(t))

         Show
         theatre
         performance
         San
         Francisco
         4th
         6th
         July
         .
```

“Better” will be stemmed to “Better”;
but lemmatized to “Good”

Part-Of-Speech Tagging

In [21]:
```
import nltk
# nltk.download('averaged_perceptron_tagger')
tags = nltk.pos_tag(tokens)
print (tags)
```

```
[('Show', 'NNP'), ('theatre', 'VBZ'), ('performances', 'NNS'), ('San', 'NNP'), ('Francisco', 'NNP'), ('4th', 'CD'),
 ('6th', 'CD'), ('July', 'NNP'), ('.', '.')]
```

https://cs.nyu.edu/grishman/jet/guide/PennPOS.html

Chunking

In [20]:
```
# nltk.download('maxent_ne_chunker')
# nltk.download('words')
from nltk import word_tokenize, pos_tag, ne_chunk
from nltk.chunk import conlltags2tree, tree2conlltags

# chunk the sentence
ne_tree = ne_chunk(pos_tag(word_tokenize(sentence)))

# IOB transform
# B-{CHUNK_TYPE} – for the word in the Beginning chunk
# I-{CHUNK_TYPE} – for words Inside the chunk
# O – Outside any chunk

iob_tagged = tree2conlltags(ne_tree)
print (iob_tagged)
```

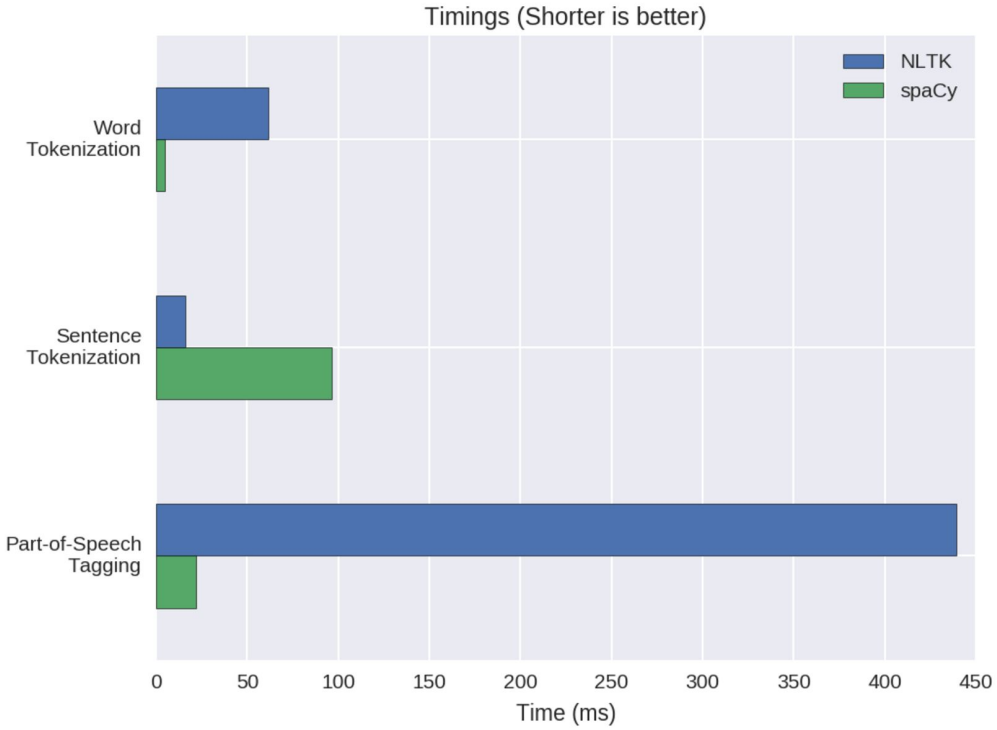
```
[('Show', 'VB', 'O'), ('me', 'PRP', 'O'), ('all', 'DT', 'O'), ('theatre', 'JJ', 'O'), ('performances', 'NNS', 'O'),
 ('in', 'IN', 'O'), ('San', 'NNP', 'B-GPE'), ('Francisco', 'NNP', 'I-GPE'), ('between', 'IN', 'O'), ('4th', 'CD',
 'O'), ('and', 'CC', 'O'), ('6th', 'CD', 'O'), ('July', 'NNP', 'O'), ('.', '.', 'O')]
```

Let's develop a bot

Example:

A Facebook Messenger bot for searching events in a city

Why use SpaCy



Source: thedataincubator

RasaNLU

- Higher level NLU package built on SpaCy and MITIE that allows you import data from api.ai/wit.ai/luis.ai and host the trained model on your server.
- You can start with free NLU services and move on to tune yourself when required.
- Increases speed to MVP.

AI will fail

- Use Hybrid AI + human
- Use sentiment analysis and intent classification to hand over to humans
- Chatbots allow effortless switching.

Happy Botting

Angik Sarkar

sarkara@theWaylo.com

@kyunbit

Waylo