Text Generator Project

What we're going to do today:

1. **Welcome**
2. Text generation
3. Coding a chain
4. Get started!
5. Next steps

Text generation

Ways to generate text

Nonsensical

- Concatenate random words from a dictionary
- Use a Markov chain to generate text based on the probabilities found in some source text
- Train a neural network on large datasets of text from books, and ask the neural network to generate text based on prediction.
- Store the parse trees for a large number of sentences and generate text based on possible sentence structures and possible words that can fill each grammatical role

Grammatical

Markov chain

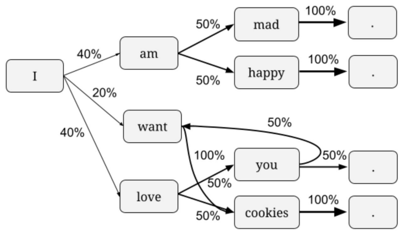
A way to model a sequence of events based on probabilities.

If the events are words, and the probabilities are the likelihood of one word following another word, then a Markov chain can model human language.

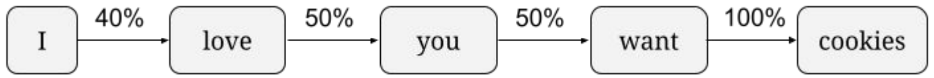
A toddler language:

- I am mad.
- I am happy.
- I love you.
- I want cookies.
- I love cookies.
- You want cookies.

Chain:



Possible output from chain:



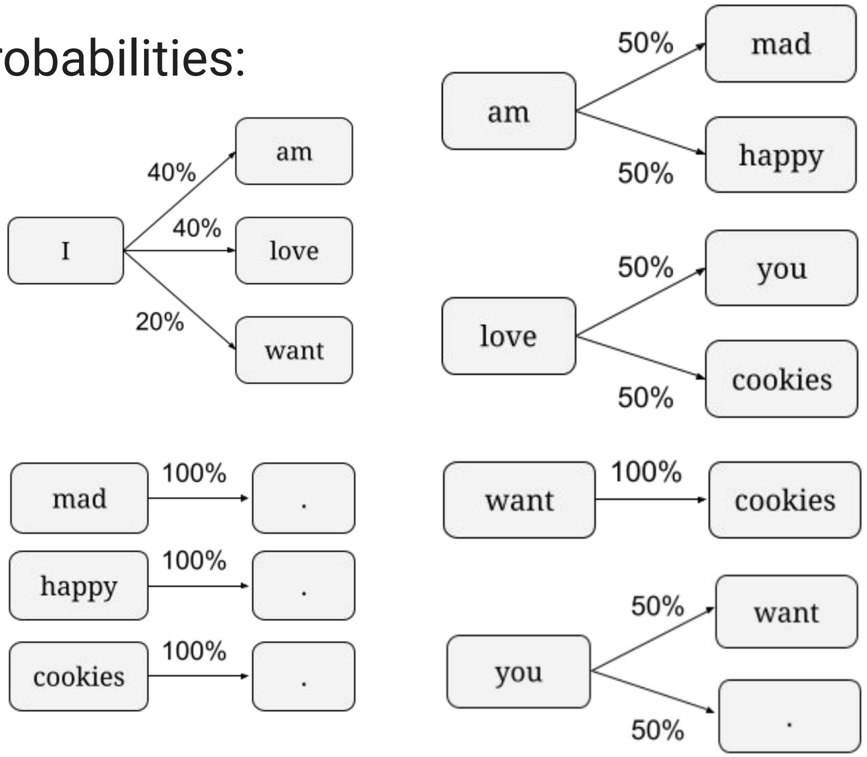
Building a Markov chain

A toddler language:

- I am mad.
- I am happy.
- I love you.
- I want cookies.
- I love cookies.
- you want cookies.

Words: I, am, mad, happy, love, you, want, cookies, .

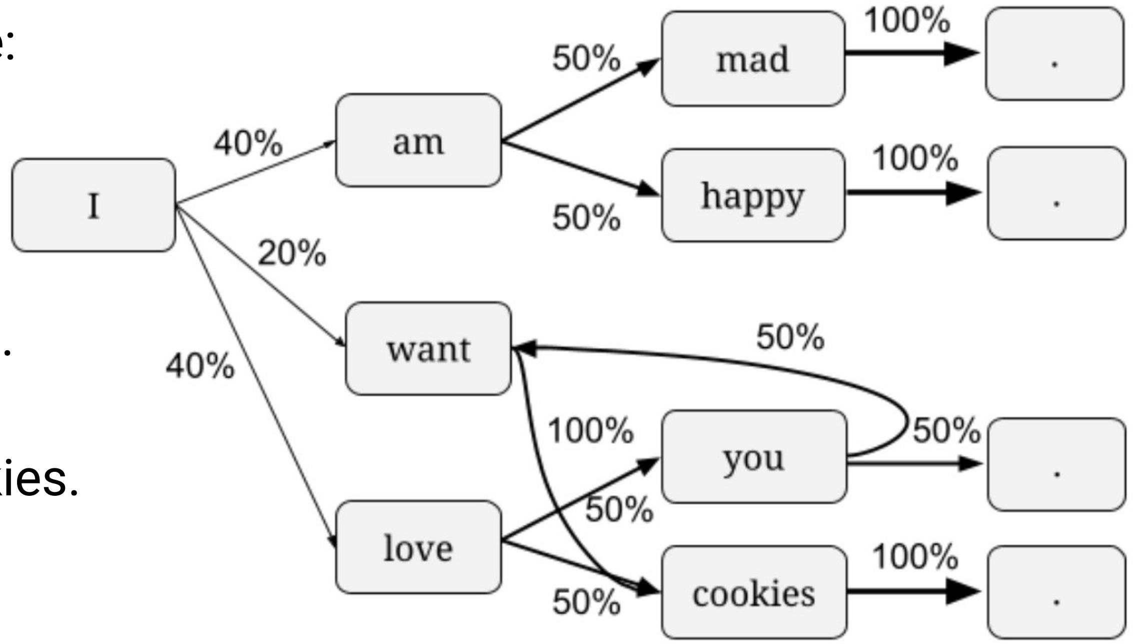
Probabilities:



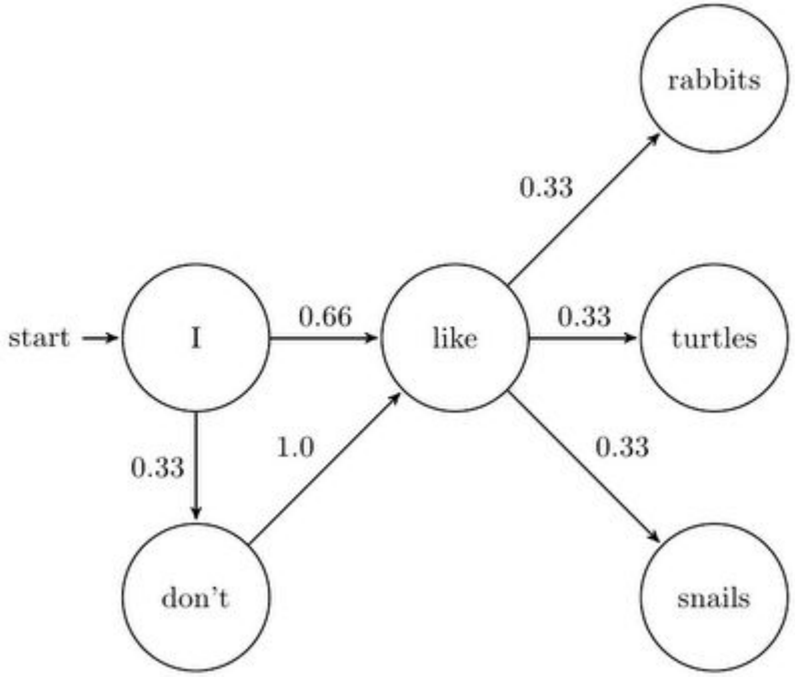
The full chain

A toddler language:

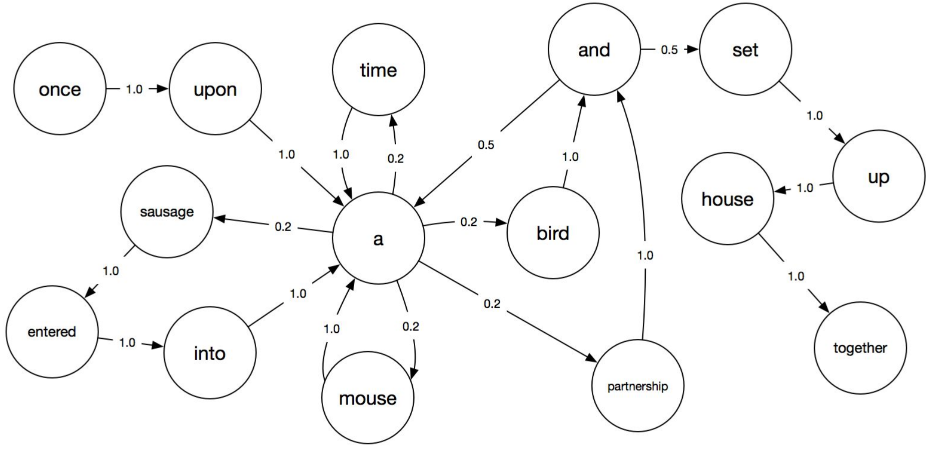
- I am mad.
- I am happy.
- I love you.
- I want cookies.
- I love cookies.
- you want cookies.



Other chains



https://www.awalsh128.com/text-generation-using-markov-chains/

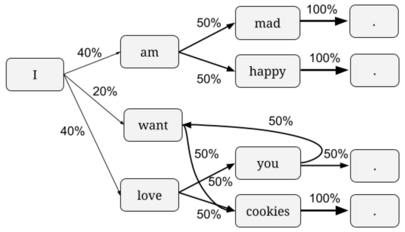


https://thagomizer.com/blog/2017/11/07/markov-models.html

Coding a chain

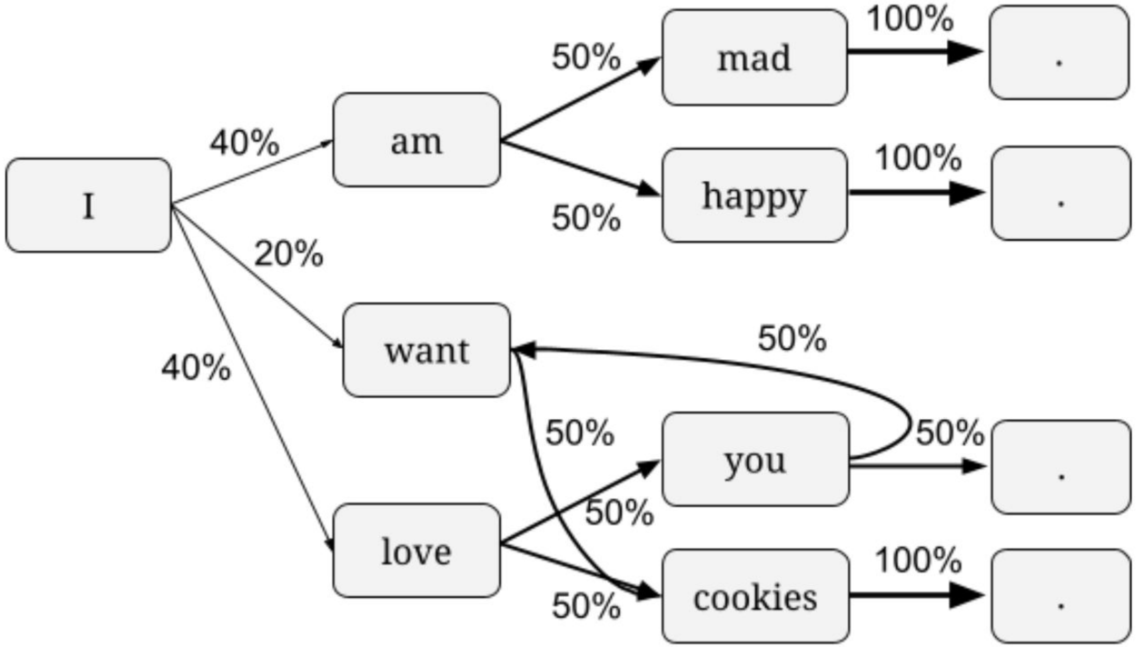
Generating text from a chain

1. Find input text and parse into sentences (or phrases)
2. Store sentences as lists of words
3. Based on the sentences, build a chain of words and the probabilities of the next word
4. Generate text by picking words probabilistically based on what words can show up after the current word



Representing a chain in code: Option # 1

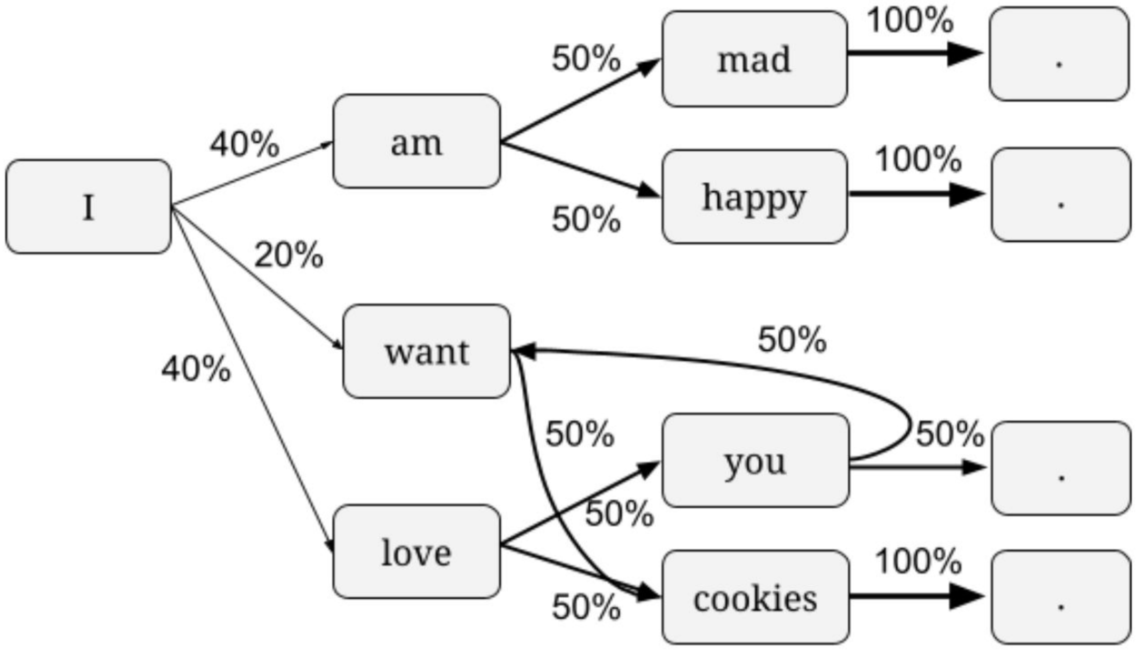
```
{"START": {"I":0.8, “you”: 0.2},
 "I": {"am": 0.4,
      "want": 0.2,
      "love": 0.4},
 "am": {"mad": 0.5,
       "happy": 0.5},
 "want": {"cookies": 1.0},
 "love": {"you": 0.5,
         "cookies": 0.5},
 "you": {"want": 0.5,
        ".": 0.5},
 "mad": {".": 1.0},
 "happy": {".": 1.0},
 "cookies": {".": 1.0}
}
```



Representing a chain in code: Option # 2

Winner!

```
{"START": ["I", "I", "I", "I",
"I", "you"],
 "I": ["am", "am", "want",
       "love", "love"],
 "am": ["mad", "happy"],
 "want": ["cookies"],
 "love": ["you", "cookies"],
 "you": ["want", "."],
 "mad": ["."],
 "happy": ["."],
 "cookies": [".", "."]
}
```



Generating next word in a chain

If current word is “I”, next word has historically been:

```
next_words = ["am", "am", "want", "love", "love"]
```

How could we select a next word in a way that reflects how often a next word has shown up historically?

```
random.choice(next_words)
```

The chain:

```
{"START": ["I"],
 "I": ["am", "am", "want",
       "love", "love"],
 "am": ["mad", "happy"],
 "want": ["cookies"],
 "love": ["you", "cookies"],
 "you": ["want", "."],
 "mad": ["."],
 "happy": ["."],
 "cookies": [".", "."]
}
```

Get Started!

Possible input sources

- sayings.txt: ~2400 wise (or not so wise) sayings.
- titles.txt: 6800 movie titles (which are on the shorter side, so your generated "sentences" will be quite short)
- composingprograms.txt: 3800 sentences from composingprograms.com, a textbook about Python, Scheme, and SQL.
- Project Gutenberg: An archive of public domain books. Find a book, then select “Plain Text UTF-8” link and copy the URL.

Breakout: Find text

1. Make a copy of the project notebook and change the sharing settings
2. **2.** Find an input source that you want to work with
3. Change the first code block to bring in that text source
4. Change the `split()` separator to work for your source text
5. Inspect the first few sentences to make sure they look as expected
6. If you have time, see if you can start on `wordify_sentences`
7. Ask any questions you run into!

Next Steps

👀 Next steps

- Attend Office hours on **Friday**
- Use Slack for support
- Submit projects by end of day **Sunday**
- Send peer review by end of day **Monday**

Any questions?