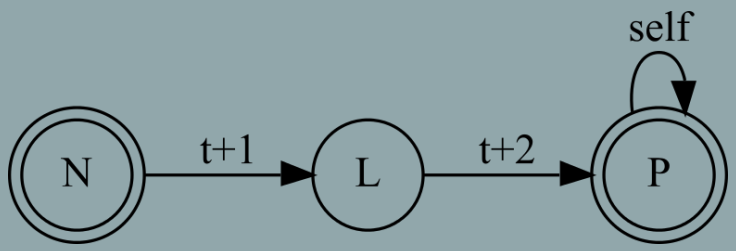
LING-362

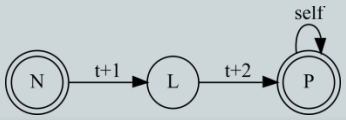
Introduction to **N**atural **L**anguage **P**rocessing

N-grams and language Models I



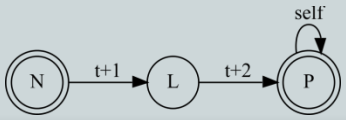
Mini hackathon results

```
Multichar_Symbols +N +V +PastPart +Past +PresPart +3 +1 +2 +Sg +Pl +F +M +Poss

LEXICON Root

Noun  ;
ProperNoun ;
ConjBase ;

LEXICON Noun

woman:shime    Ninf;
man+N:rOme    Ninf;
devil+N:diabolos    Ninf;
house+N:Hi    Ninf;
place+N:ma    Ninf;
foot+N:rat    Ninf;

LEXICON ProperNoun

Senoute  #;
sara  #;
biktOr  #;
hllaria  #;
60:se  #;
```



Mini hackathon results

- Covered:
  - 10 nouns
  - 6 verbs
  - 5 prepositions
  - 5 auxiliaries (plus null)
  - All persons (except 2sgF)
- And phonotactic rules!
  - Article lengthening
- Cleaned up solution in Code > Foma



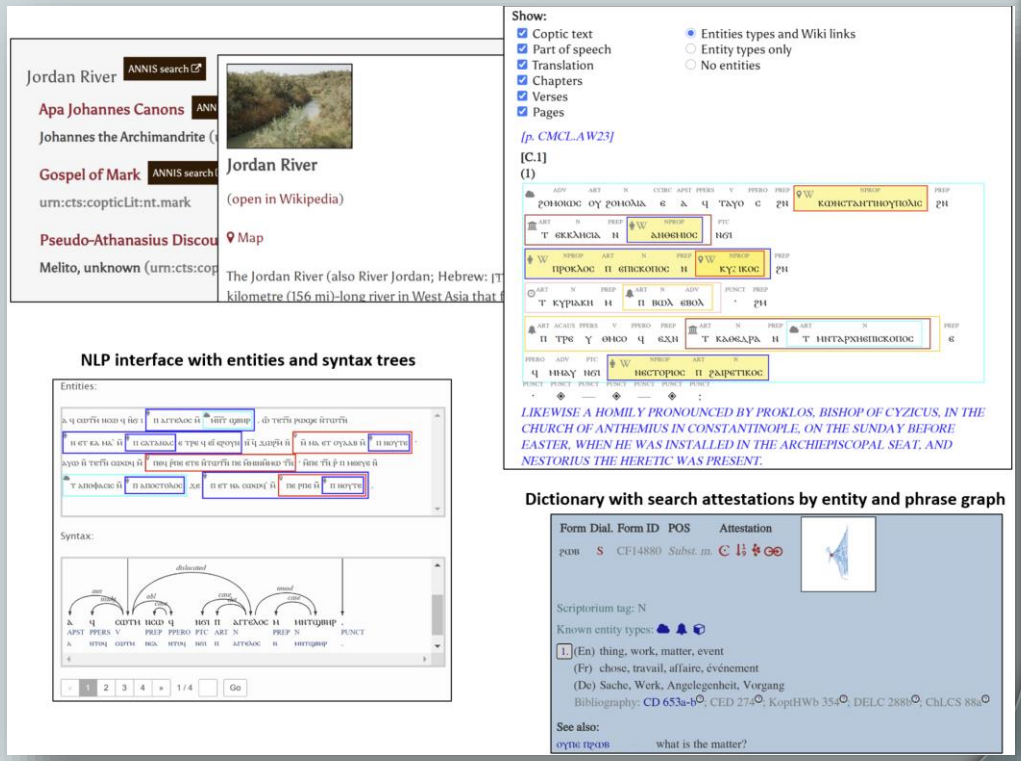
What is this for?

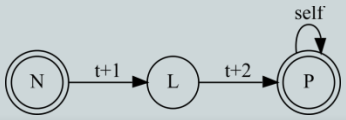
GEORGETOWN UNIVERSITY

COPTIC SCRIPTORIUM

NATIONAL ENDOWMENT FOR THE HUMANITIES

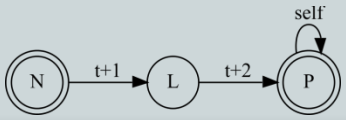
- Coptic Scriptorium - https://copticscriptorium.org/
  - Read and analyze Coptic texts
  - Online dictionary
  - NLP tools
  - Syntactic, morphological and semantic search





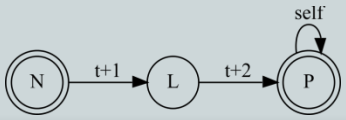
Grammar and usage

- Regular languages (and FSAs) have a formal grammar
- Transitions between states
- No knowledge about usage
  - No memory of **previous states**
  - No experience from **previous runs**



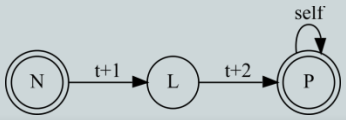
Memory and experience

- At the micro level, remembering previous context can be helpful:
  - Can we disambiguate? What if we could stop over-generation by considering earlier input?
  - Is this analysis correct?
    - Number: numb+er = numb+ADJ+COMP
  - How about now?
    - … the number is …
    - … is number than …
    - … more number one …



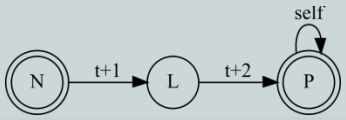
Memory and experience

- At a macro level, experience helps us even without context:
  - ***number***='digits' ***more often*** than ***number***='more numb'
- And with context, for **categories**:
  - Comparative ***more likely*** if ***than*** appears



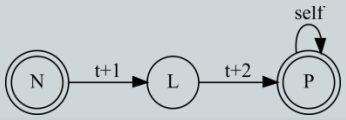
Brief review: probability

- Many questions in NLP are a matter of probability:
  - ***number*** is an adjective:
    - Possible (generated by FSM)
    - Unlikely (rare in corpus data)
    - Does occur somewhere!
    - Some contexts truly ambiguous
- Declared goal of most statistical approaches to NLP: be wrong as rarely as possible (but we will be wrong sometimes)



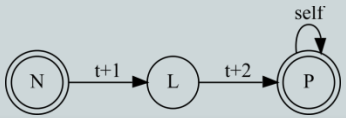
Frequentist probability

- For some scenarios we may have a rational expectation of exact probabilities:
  - Coin toss: p(heads) = 0.5 (or tails: 0.5)
  - Dice: p(⚄)= 1/6 (or not: 5/6)
  - **Note:**
    - **We know all possible outcomes**
    - **Probabilities sum up to 1!**



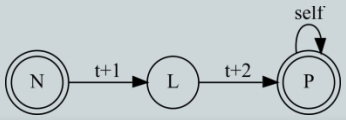
Frequentist probability

- For many empirical phenomena we do not have such numbers
  - p(rain tomorrow) = ?
  - p(‘*number’* is an adjective) = ?



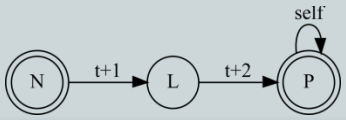
Frequentist probability

- We can estimate probabilities based on previous experience:
  - Guess October 6, 2021 will be like last October 6… (could have been a fluke that year?)
  - Maybe take average precipitation and temperature of last 10 years?
- More data typically means better predictions



Conditional probabilities

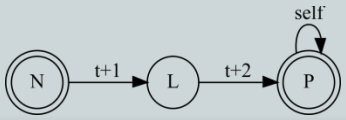
- Won't the weather today depend on yesterday?
  - Conditional probabilities are written like this:
    - p(rain today|rain yesterday)



Conditional probabilities

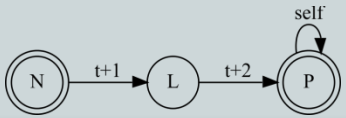
- If two events A, B, are **independent**, then:
  - p(A|B) = p(A)
- For independent events, the **chain rule** applies:
  - p(A&B) = p(A)*p(B)
- For example:
  - P(⚂ & ⚃) = ?

  1/6 * 1/6 = 1/36



Back to linguistic experience

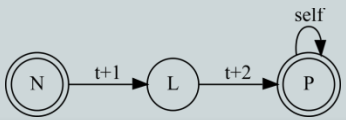
- Probabilities without context:
  - p(number=digits) > p(number='more numb')
- And with context:
  - p(comparative|'than' is next) > p(noun|'than' is next)
- How can we model ‘context’ more generally?



N-gram models

- A very simple (and efficient) way of modeling context is using **n-grams**
  - Decide on a useful context size, often 3 words
  - Save analyses not of individual words, but of words given the previous 2 words – **trigram model**

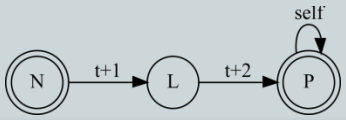
Image: googlesystem.blogspot.com



N-gram models

Why 3?

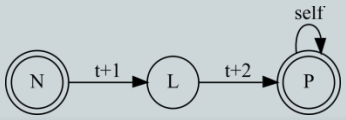
- Any number can be chosen
- Key consideration: amount of data available
  - Choosing long chains increases accuracy
  - Different answer for:
    - *Find him number than*
    - *Find him number of*
  - **But:** potentially few instances of each chain
  - **Data sparseness problem**



N-gram models

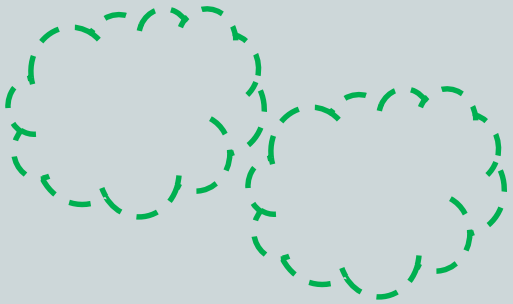
Rules of thumb:

- For typical 'million word' resources – trigrams are taken
- 'Low resource languages' with 10K samples – bigrams
- Gigaword corpora – 4, 5-grams (e.g. Google n-grams)

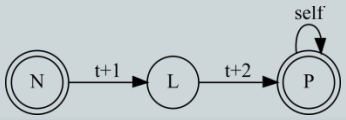


Are n-grams realistic?

- Humans use a lot of linguistic information not covered in n-grams
  - Can understand novel combinations: ***colorless green ideas***



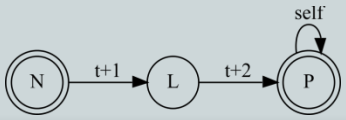
- But humans probably do store n-grams:
  - ***beam me …***
  - ***come out come out …***
  - ***a twist of …***



Experiment

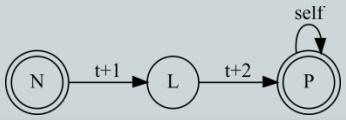
- once in a while
- in over the past
- as if the click
- got away with it

>11	once in line to
27<	all over the world
20<	as if they were
>20	scribble away and confide



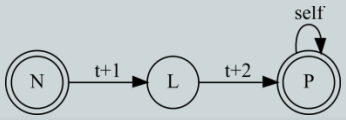
What are they good for?

- The idea of using n-grams to model language data is due to Markov (1913):
  - Looked at Russian orthography in Pushkin's *Eugene Onegin*
    - Frequency of characters followed normal distribution (despite very complex poetic meter)
    - Language is not random – strong deviation from independence
    - Phonotactics and morphology in a language's written script

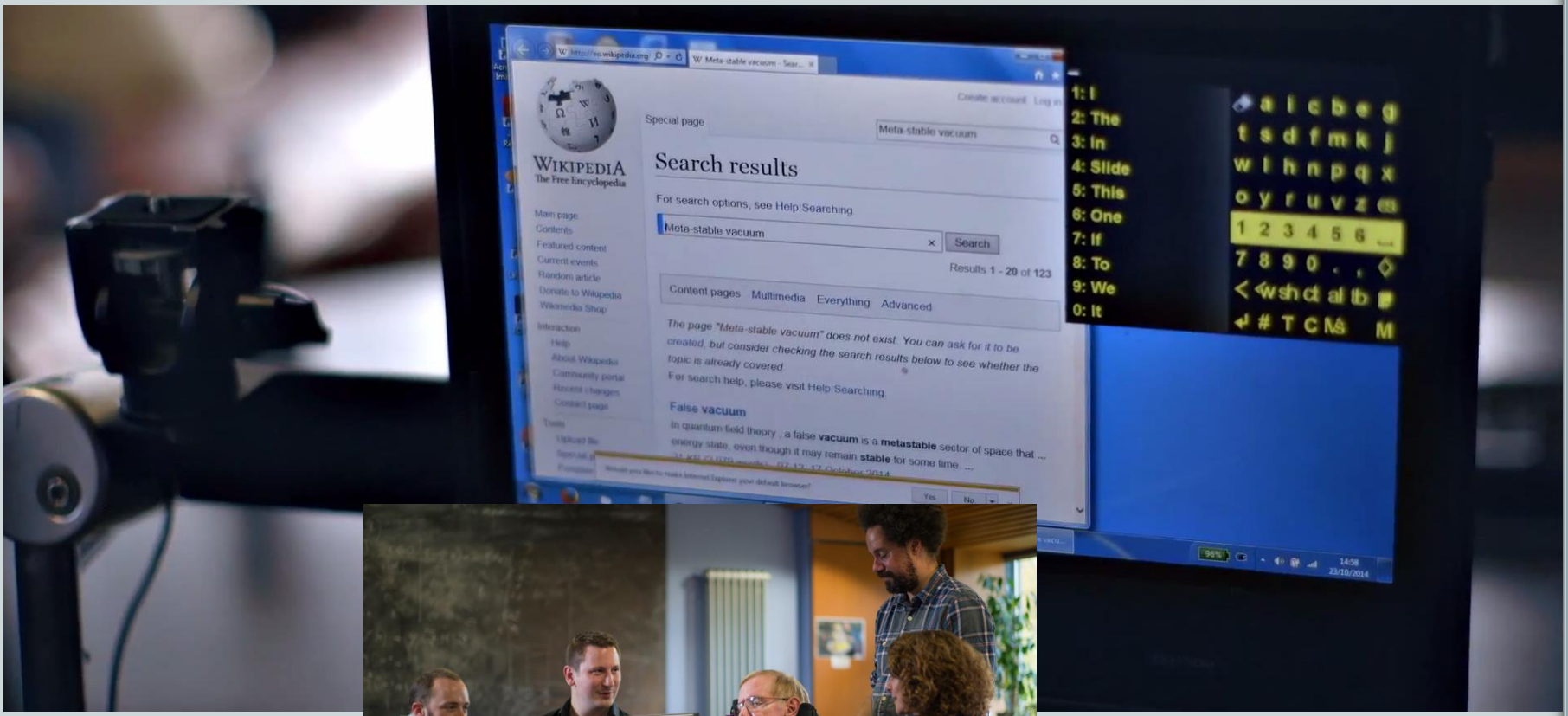


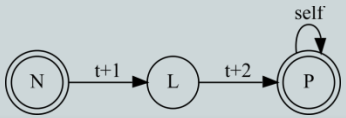
What are they good for?

- The extension to word models became obvious and was popularized in the 50s
- Today:
  - language models in machine translation
    - Given N possible translations outputted by a system…
    - Rank each by likelihood as given by the model
  - predictive keyboards (cellphones)
  - augmentative communication (for disabilities)
  - Optical Character Recognition (OCR)
  - Speech to text



Example - SwiftKey





Images: SwiftKey, PCWorld

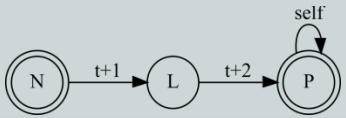
Building our own!

[Wikimedia]

- Consider the following texts, by Charles Dickens*:

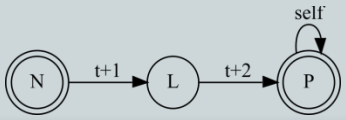
bigram model	trigram model	4-gram model
depose to linger yet, pointing upward ! ' are melting from me, pointing upward	signature under such circumstances, Mr. Mell, formerly poor pinched usher to my Middlesex magistrate	that I stole into the next street, and open a chemist's shop? Whether he could

*sort of



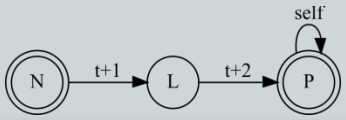
DIY natural language generation!

- Let's create Dickensian stories
- We need:
  - Dickensian training data and a way to read it
  - Random selection
  - A mechanism to choose best output for a story of a certain length
- Download:
  - *ngrams.py*
  - *dickens.txt*



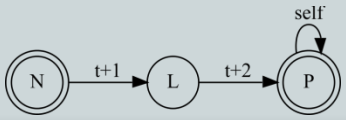
Reminder – reading files

```
parser = argparse.ArgumentParser()
parser.add_argument("file")

options = parser.parse_args()
training_file = options.file

a = open(training_file).read()

with open(training_file,'r') as f:

    training_data = f.read()

...

counts = get_counts(context_length, training_data)
```



Learning the frequencies

- To track frequencies of n-grams, we'll need a dictionary tracking counts like this:
  - Key: "In", "the"
    - Value: another dictionary:
      - Key: "beginning", value: 2
      - Key: "end", value: 5
      - …
  - The function get_counts() should build this dictionary



Learning the frequencies

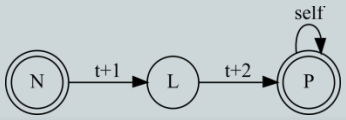
- It looks like we could use a **list** for the key:

```
freqs[["in","the"]] = {"end":5, "beginning": 2}
```

- Actually, this is forbidden:
  - Dictionary keys must be **immutable**
  - List values could be changed:
    - We could change the second list member of [“in”, “the”]

```
my_key = ["in","the"]
freqs[my_key] = 5
my_key[0] = "by"
```

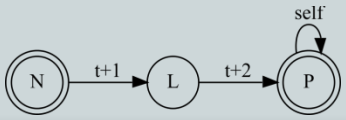
    - Dictionary would be broken



Tuples: not quite Lists

- Python has a cousin data-type to lists: **tuples**
  - Tuples are like lists, but they are **immutable**
  - Once defined they can't be changed
  - Look a lot like lists in **round** brackets:

```
# This is a list:
a = ["in", "the"]

# This is a tuple:
b = ("in", "the")
```

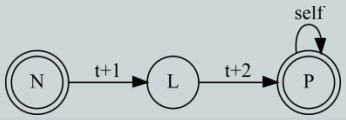


Tuples: not quite Lists

- In practice, tuples are used:
  - When an **immutable** data type is required
  - In contexts where something like a list:
    - Has a specified, invariable length
    - Each position has a predictable meaning

- Example: person's height, weight and age:

```
# Don't need to append or change these:
stats = (175, 72, 43)
```

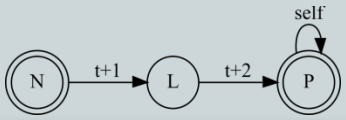


Another helper function: range()

- We'll need to count words up to the needed context size (last two words for trigrams)
- We can use the range() generator for this:

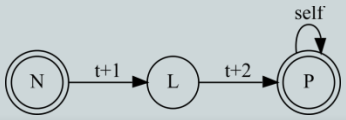
```
# Give us a list with the first 10 numbers
list(range(10))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```



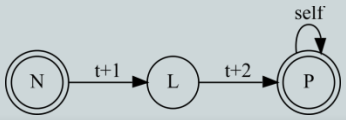
get_counts()

```
def get_counts(context_length, training_text):

    counts = {}

    tokens = word_tokenize(training_text)
    for i in range(len(tokens) - context_length):
        context = []
        next_token = tokens[i + context_length]
        for j in range(context_length):
            context.append(tokens[i + j])

        # Add 1 to frequency or create new dictionary item for this tuple
        if tuple(context) in counts:
            if next_token in counts[tuple(context)]:
                counts[tuple(context)][next_token] += 1
            else:
                counts[tuple(context)] = {next_token: 1}
        else:
            counts[tuple(context)] = {next_token: 1}

    return counts
```



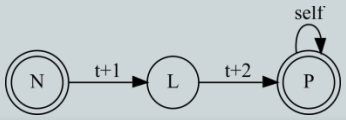
generate_from_file(context_length,training_file,output_length=10)

```
first_tokens = choice(counts.keys())  # Choose a random first context
output_list = list(first_tokens)
current_context = first_tokens

for i in range(output_length):
    next_context = max(counts[current_context], key=counts[current_context].get)
    temp = list(current_context)
    temp.pop(0)  # Remove first token in previous context
    temp.append(next_context)  # Add new token for the next context
    next_token = temp[-1]
    next_context = tuple(temp)

    current_context = next_context

    output_list.append(next_token)

print(" ".join(output_list))
```



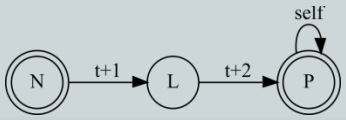
Spot the genre

these changes into our other midmarket power forms . Thanks again for your help on this . Carol St. Clair EB 3889 713-853-3989 (Phone) 713-646-3393 (Fax) carol.st.clair @ enron.com All , Please see the attached Interconnect Agreement with Questar . Transwestern will own and operate the interconnect . Questar may be able to purchase material , but some of



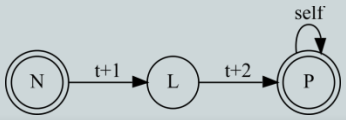
Spot the genre

Furies , and I 'll be as good as my word ; but speciously for Master Fenton . Well , on went he for a search , and away went I for foul clothes . But mark the sequel , Master Brook-I suffered the pangs of three several deaths : first , an intolerable fright to be detected with a jealous rotten bell-wether



Spot the genre

ambush in her system , ready , at the corner of the street , with his great kite at his back , a very monument of human misery . My aunt went on with a quiet enjoyment , in which there was very little affectation , if any ; drinking the warm ale . 'She 's the most ridiculous of mortals . But



At home

- We are not ready to expand the ngram code yet
- But you should find some time to practice Python!
- Try working through Chapter 3 of NLTK:
  - http://www.nltk.org/book/ch03.html
  - Learning about file I/O
  - Getting data from the internet
  - More practice with lists, regex, nltk, and more