Natural Language Generation

aka... automated writing

Alina Kononov

with acknowledgments to Emilio T. Jasso

Why automate writing?

- Pass the Turing test
  - make computer writing indistinguishable from human writing

- Cut out the middle-employee
  - automatically generate narratives and reports from data
  - automated customer service (chatbots)

- Satire – inverse Turing test?
  - can real writing be distinguished from bogus computer-generated text?
  - SCIgen (https://pdos.csail.mit.edu/archive/scigen/)
    - randomly generated paper accepted to WMSCI Conference in 2005

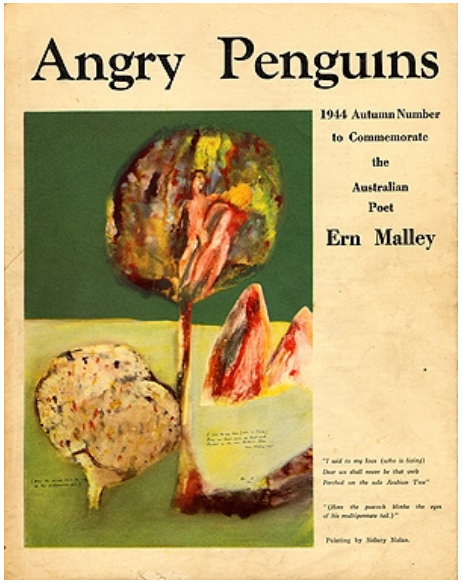
History of “randomized” ““““avant-garde”””” literature

- 1943: *Angry Penguins* Hoax
  - imitation of modernist poetry testing movement’s validity
  - “We opened books at random, choosing a word or phrase haphazardly. We made lists of these and wove them in nonsensical sentences.”

    --- James McAuley and Harold Stewart, creators of fictional Ern Malley [1]
- 1950’s – 1960’s: Cut-up and fold-in techniques
  - reassemble finished pieces to form interesting juxtapositions
  - “All writing is in fact cut-ups. A collage of words read, heard, overheard. What else? Use of scissors renders the process explicit and subject to extension and variation.”

    --- William Burroughs, "The Cut Up Method," 1963
- 2000’s: Flarf and Spam poetry (spoetry)
  - incorporate Google search results and spam emails
  - “With so much available language, does anyone really need to write more?”

    --- Kenneth Goldsmith [2]
- 2010’s: Computer generated poetry via context-free grammar [3]
  - https://www.poetrygenerator.ninja



1. http://jacketmagazine.com/17/fact2.html
2. https://www.poetryfoundation.org/poetrymagazine/articles/69328/flarf-is-dionysus-conceptual-writing-is-apollo
3. https://rpiai.com/other/poetry/

Cut-up Spoetry: Random Haikus

- Corpus: a bunch of emails to engr-grad-students and engr-phd-gradstudents
- Sort words by (approximate) # of syllables
  - count vowels with a few special cases
  - English has lots of special cases...
- Randomly generate lines with (approximately) 5, 7, 5 syllables

engineering good
sessions participating
cancelled wellness

talks scholars student
entrepreneurial be
assistantships you

scholar monday learn
engineering remember
join engineering

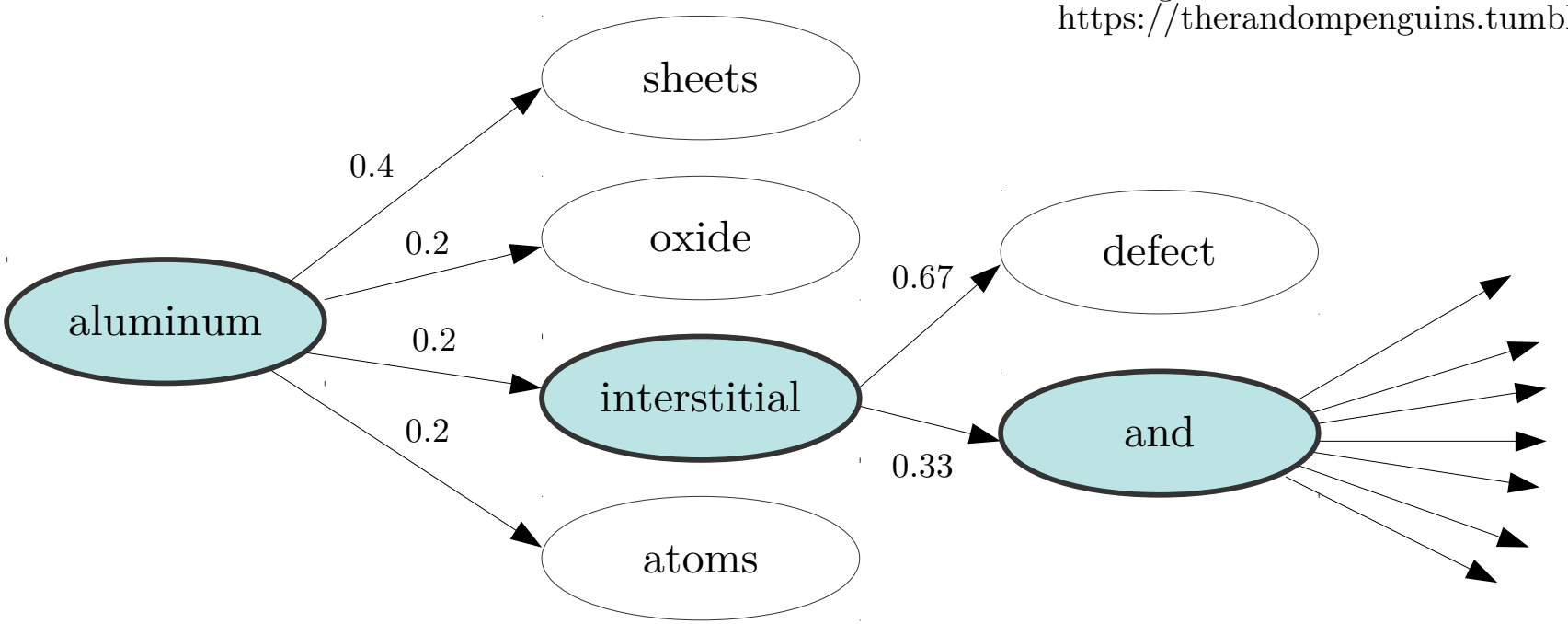
opportunities work
revolutionizing to
invigorating

```
while syl_needed>0:
    # choose random number of syllables
    syl = 1+random.randrange(min(syl_needed,6))

    # choose random word with syl syllables
    Word = random.choice(syl_dict[syl])

    line.append(word)
    syl_needed -= syl
```

Markov Chain Text Generator

- Want a more coherent word salad...
- Use information about what combinations of words tend to appear together
- Each word is a state
- Next word's probability depends on current word
  - need to calculate/specify $P(x_i \mid x_{i-1})$

"There's 'word salad,' and then there's dumping the bowl into a giant fan and blowing salad all over the wall." https://therandompenguins.tumblr.com



Markov Chain Text Generator: Implementation

- Corpus: several abstracts for conference talks
- Build MarkovChain object
  - read through input text and add each word as a MarkovState object
  - adjust transition probabilities of previous MarkovState

```
while (file >> word) {
    state = &chain.add_state(word);
    prev_state -> add_next_state(state);
    prev_state = state;
}
```

- Choose random starting state
- Run through the chain and print out each word

```
for (int i=0; i<num_words-2; i++) {
    chain.evolve();
    cout << chain.curr_state->name << " ";
}
```

Markov Text Generator: Results

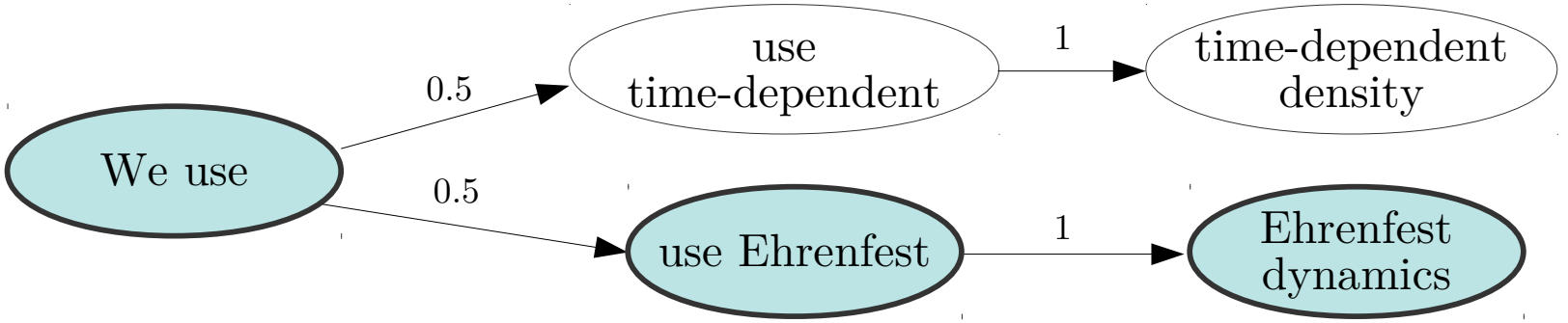
Slow diffusion properties of highly resistant materials. As a proton traversing monolayer and nuclear technology. Thin or two-dimensional materials respond to fabricate layered heterostructures. However, the viability of radiation resistant materials, especially for extrinsic interstitials, investigate the material. This protective layer acts as the entrance-side SE yields, SE yields, SE yields, SE energy transfer and Xe8+ ions traversing a material's surface, it deposits energy and can lead to ion radiation are rare, fs scale surface of these quantities on diffusivity of materials respond to secondary electron yields and exit-side secondary electron yields and trilayer graphene. We also find displaced ground-state.

https://jacobrcampbell.com/blog/2014/3/word-salad-and-technical-jargon

n-gram Markov Chain

- Use more context to determine each word
- n-gram: sequence of n items (words)
- Next word's probability depends on previous n-1 words
  - need to calculate/specify $P(x_i \mid x_{i-(n-1)}, \ldots, x_{i-1})$
- Larger state space → more memory
- Sparser transition matrix → more deterministic
- More short-range order, still little long-range order

Trigrams:



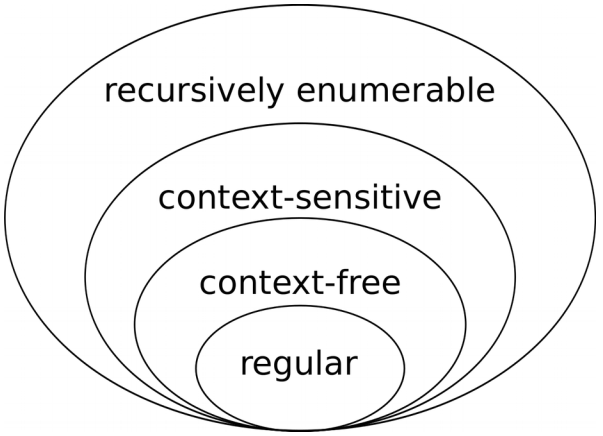
n-gram Markov Chain: Implementation

- Very similar to simple Markov Chain
- Keep track of n-2 previous words in array (overlap between states)

```
while (file >> word) {
    state_name = prev_word[0];
    for (int i=0; i<n-3; i++) {
        prev_word[i] = prev_word[i+1];
        state_name += " "+prev_word[i];
    }
    prev_word[n-3]=word;
    state_name += " "+word;
    state=&chain.add_state(state_name);
    prev_state->add_next_state(state);
    prev_state=state;
}
```

Context-Free Grammar: Context

- Formal grammar:
  - terminal symbols a (~ words)
  - nonterminal symbols A (~ clauses, phrases, parts of speech)
  - production rules $(\alpha)\mathrm{A}(\alpha) \rightarrow (\alpha)$
    - how to replace a sequence of symbols with another sequence of symbols
  - start symbol S (~ entire sentence)
- Formal language:
  - all sequences of terminal symbols which can be formed from S



https://en.wikipedia.org/wiki/Chomsky_hierarchy#/media/File:Chomsky-hierarchy.svg

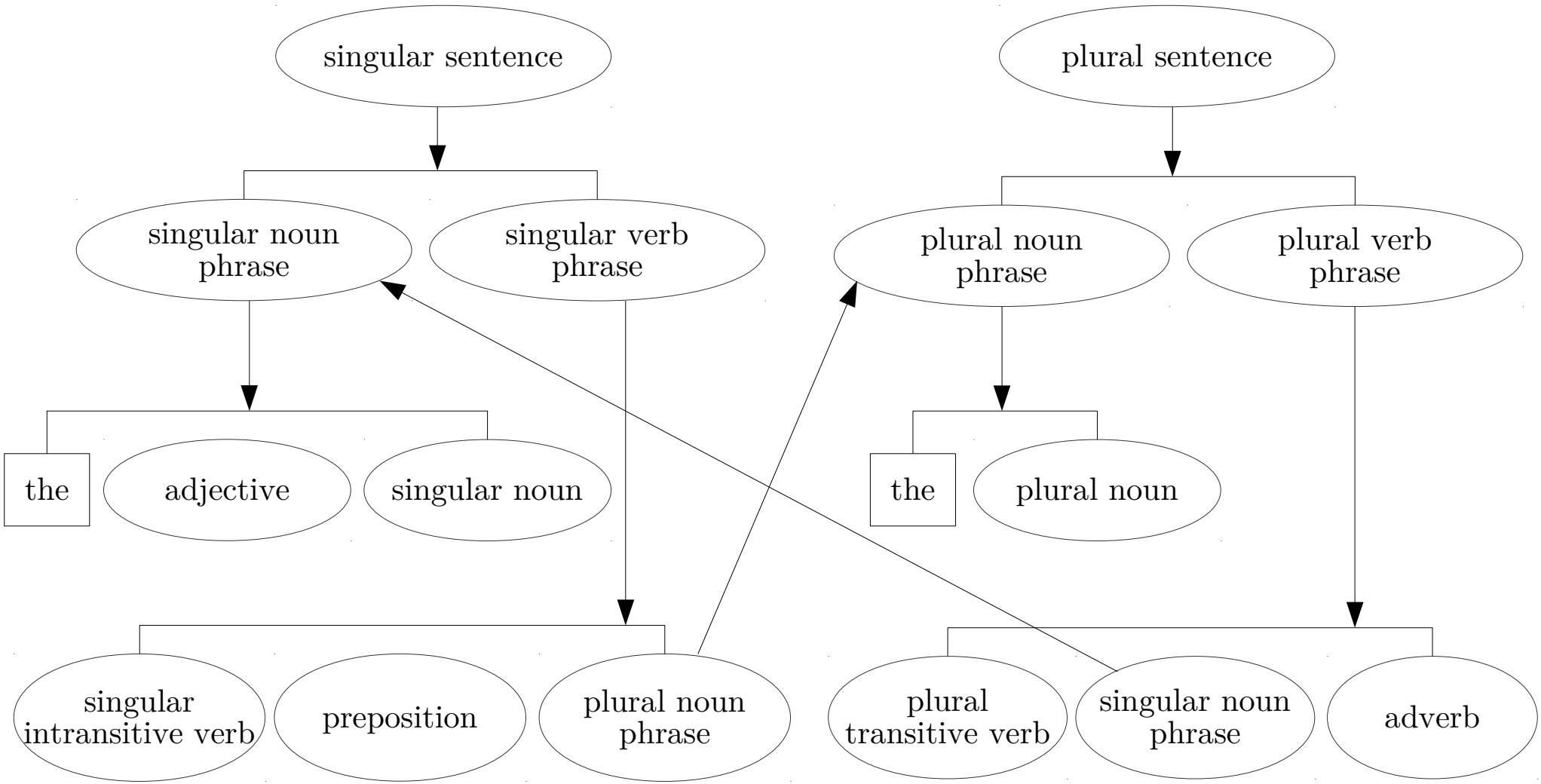
$(\alpha)\mathrm{A}(\alpha) \rightarrow (\alpha)$

$(\alpha)\mathrm{A}(\alpha) \rightarrow \alpha$

$\mathrm{A} \rightarrow (\alpha)$

$\mathrm{A} \rightarrow \mathrm{a}(\mathrm{A})$

Context-Free Grammar: Model



Context-Free Grammar: Implementation

- Terminal symbols: lists of nouns, verbs, etc. generated by a real, live writer
- Python class for each type of symbol
  - replace method encodes production rule

```
class SingularSentence:
    def replace(self):
        return [SingularNounPhrase(), SingularVerbPhrase()]
```

- Repeatedly apply production rules until sentence only contains terminal symbols

```
while True:
    newSentence=[]
    for part in sentence:
        if type(part)==str:    # terminal
            newSentence.append(part)
        else:                  # non-terminal
            newSentence.extend(part.replace())
    if newSentence==sentence:
        break
    sentence=newSentence
```

Context-Free Grammar: Results

The cheeses cook the surprising house carefully.

The furry brain exercises next to the sheep.

The mice attack the unremarkable sheep aggressively.

The ethereal sofa rests outside of the barbecues.

The space ships build the beautiful fungus professorially.

The unbelievable ennui jumps over the houses.

Summary

- Random words
- (bigram) Markov chain
- (n>2)-gram Markov chain
- Context-free grammar
- Recurrent neural networks?