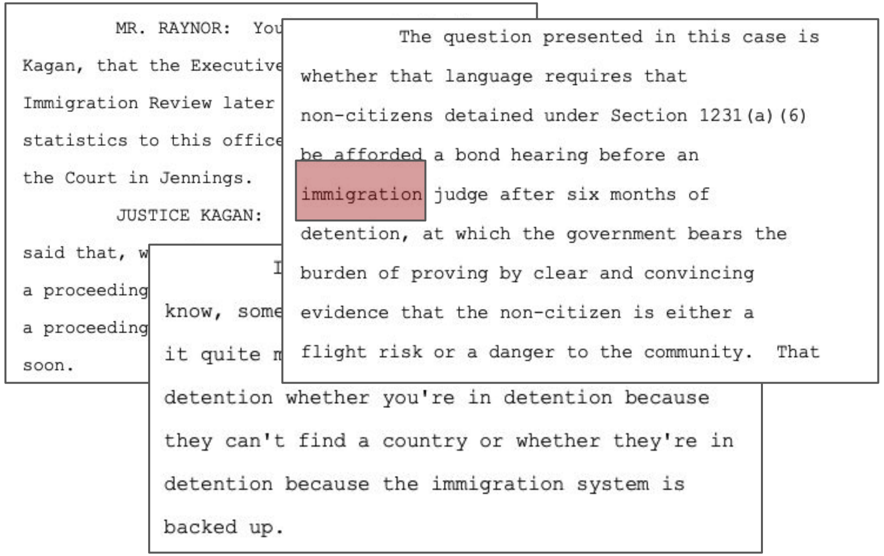
Moving from words to phrases when doing NLP (Part 1 of 2)

NLP+CSS 201 Tutorials
February 17, 2022

Abe Handler
Instructor
Department of Information Science
University of Colorado, Boulder
www.abehandler.com
@AbeHandler

Your NLP pipeline probably uses a **unigram bag of words**



MR. RAYNOR: You

Kagan, that the Executive

Immigration Review later

statistics to this office

the Court in Jennings.

JUSTICE KAGAN:

said that, w

a proceeding

a proceeding

soon.

I

know, some

it quite m

detention whether you're in detention because

they can't find a country or whether they're in

detention because the immigration system is

backed up.

The question presented in this case is

whether that language requires that

non-citizens detained under Section 1231(a)(6)

be afforded a bond hearing before an

immigration judge after six months of

detention, at which the government bears the

burden of proving by clear and convincing

evidence that the non-citizen is either a

flight risk or a danger to the community. That

Document 1 = {"immigration", "bond", "judge", "hearing"}

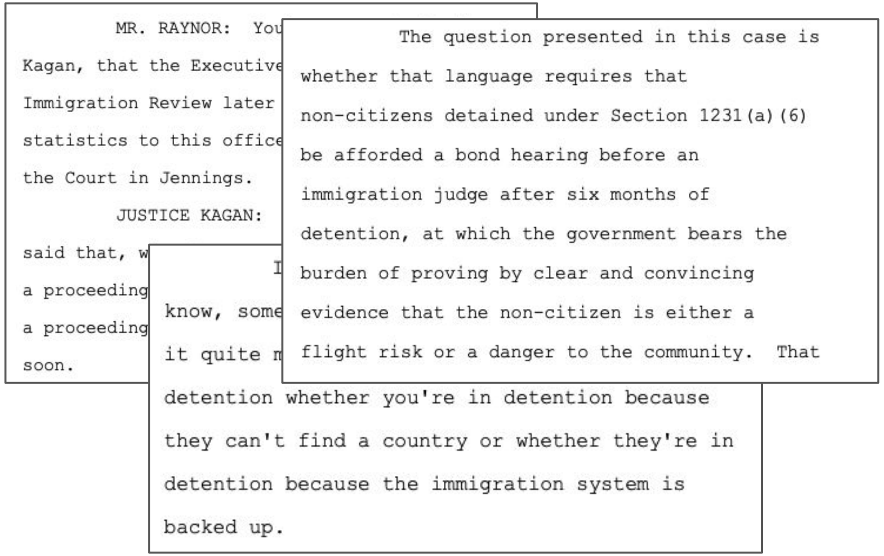
Document 2 = {"immigration", "because" ... "system" }

Document 3 = {"statistics", "review", "of, "immigration" }

[See Jurafsky and Martin, Speech and Language Processing, Figure 4.1]

[https://www.supremecourt.gov/oral_arguments/argument_transcripts/2021/19-896_3314.pdf]

The unigram bag of words is helpful, but has issues



MR. RAYNOR: You

Kagan, that the Executive

Immigration Review later

statistics to this office

the Court in Jennings.

JUSTICE KAGAN:

said that, w

a proceeding

a proceeding

soon.

I

know, some

it quite m

detention whether you're in detention because

they can't find a country or whether they're in

detention because the immigration system is

backed up.

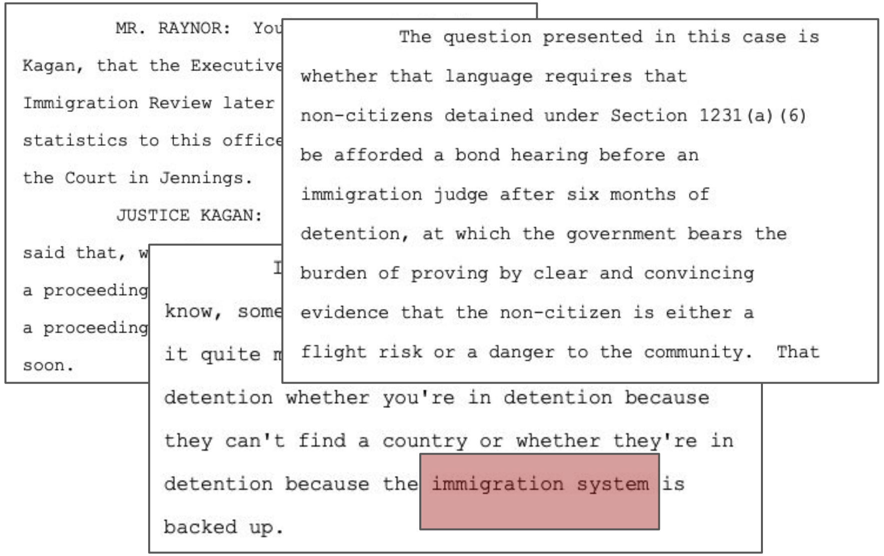
The question presented in this case is whether that language requires that non-citizens detained under Section 1231(a)(6) be afforded a bond hearing before an immigration judge after six months of detention, at which the government bears the burden of proving by clear and convincing evidence that the non-citizen is either a flight risk or a danger to the community. That

Document 1 = {"immigration", "bond", "judge", "hearing"}

Document 2 = {"immigration", "because" ... "system" }

Document 3 = {"statistics", "review", "of, "immigration" }

Unigram bag of words will miss domain-specific vocabulary



MR. RAYNOR: You

Kagan, that the Executive

Immigration Review later

statistics to this office

the Court in Jennings.

JUSTICE KAGAN:

said that, w

a proceeding

a proceeding

soon.

The question presented in this case is whether that language requires that non-citizens detained under Section 1231(a)(6) be afforded a bond hearing before an immigration judge after six months of detention, at which the government bears the burden of proving by clear and convincing evidence that the non-citizen is either a flight risk or a danger to the community. That

I

know, some

it quite m

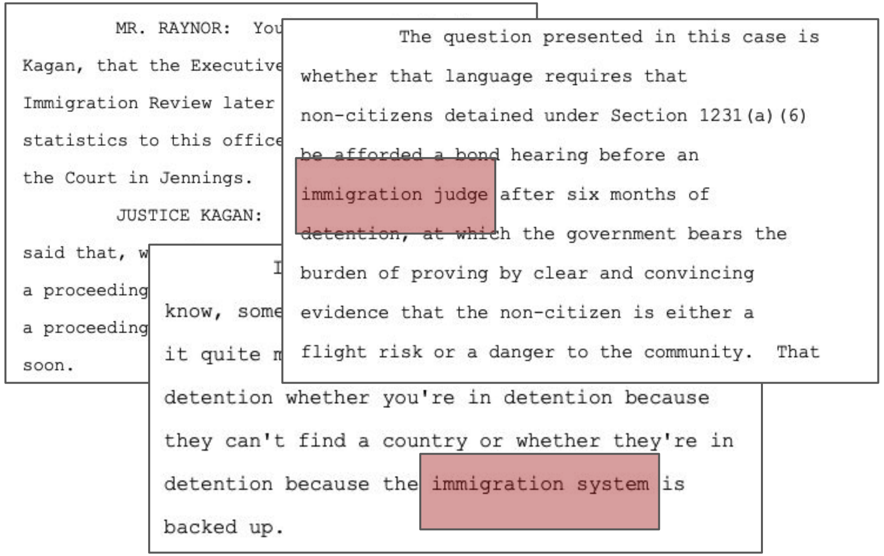
detention whether you're in detention because they can't find a country or whether they're in detention because the immigration system is backed up.

Document 1 = {"immigration", "bond", "judge", "hearing"}

Document 2 = {"immigration", "because" ... "system" }

Document 3 = {"statistics", "review", "of, "immigration" }

Unigram bag of words merges different phrases



MR. RAYNOR: You

Kagan, that the Executive

Immigration Review later

statistics to this office

the Court in Jennings.

JUSTICE KAGAN:

said that, w

a proceeding

a proceeding

soon.

The question presented in this case is

whether that language requires that

non-citizens detained under Section 1231(a)(6)

be afforded a bond hearing before an

immigration judge after six months of

detention, at which the government bears the

burden of proving by clear and convincing

evidence that the non-citizen is either a

flight risk or a danger to the community. That

I

know, some

it quite m

detention whether you're in detention because

they can't find a country or whether they're in

detention because the immigration system is

backed up.

Document 1 = {"immigration", "bond", "judge", "hearing"}

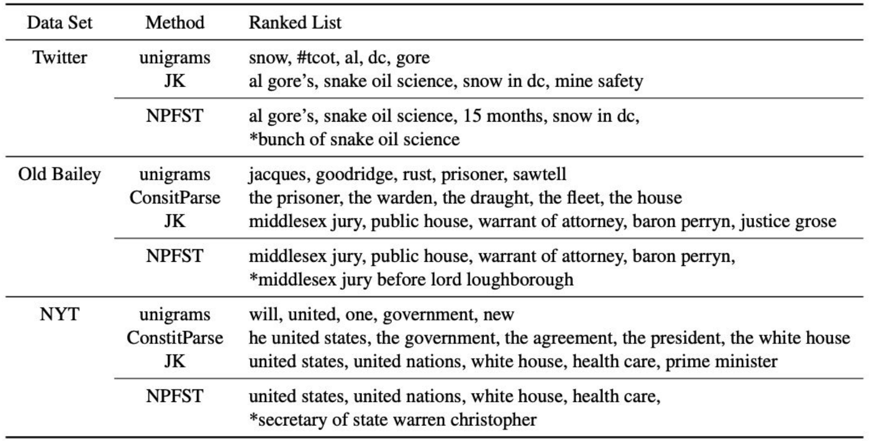
Document 2 = {"immigration", "because" ... "system" }

Document 3 = {"statistics", "review", "of, "immigration" }

Unigram bag of words can not capture key concepts in many domains (likely including yours)

Unigrams do not capture key concepts from diverse text data from:

- Twitter
- historical court records
- articles from *The New York Times*



Data Set	Method	Ranked List
Twitter	unigrams	snow, #tcot, al, dc, gore
	JK	al gore's, snake oil science, snow in dc, mine safety
	NPFST	al gore's, snake oil science, 15 months, snow in dc, *bunch of snake oil science
Old Bailey	unigrams	jacques, goodridge, rust, prisoner, sawtell
	ConsitParse	the prisoner, the warden, the draught, the fleet, the house
	JK	middlesex jury, public house, warrant of attorney, baron perryn, justice grose
	NPFST	middlesex jury, public house, warrant of attorney, baron perryn, *middlesex jury before lord loughborough
NYT	unigrams	will, united, one, government, new
	ConstitParse	he united states, the government, the agreement, the president, the white house
	JK	united states, united nations, white house, health care, prime minister
	NPFST	united states, united nations, white house, health care, *secretary of state warren christopher

[Handler et al. (2016), Bag of What?]

OK, you've convinced me I need phrases. How do I get them?

You have options

- Along with my coauthors, I explored your choices in a 2016 paper Bag of What?
    - Ideas from the paper went into the software phrasemachine

You have options

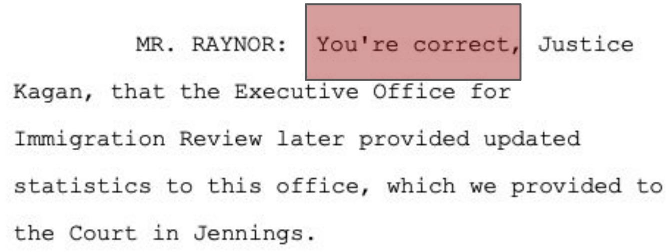
- n-grams
- Named entities
- Constituents
- Regular expression over part-of-speech tags

Our proposed method

You have options

- **n-grams**
- Named entities
- Constituents
- Regular expression over part-of-speech tags

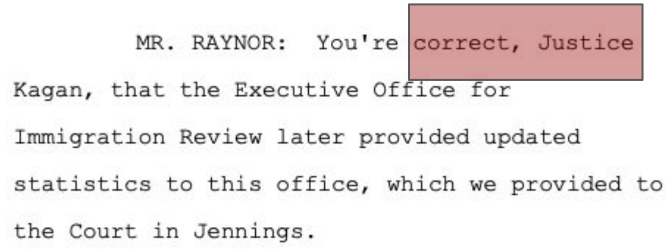
n-grams


MR. RAYNOR: You're correct, Justice Kagan, that the Executive Office for Immigration Review later provided updated statistics to this office, which we provided to the Court in Jennings.

2-grams

"you're correct"

n-grams

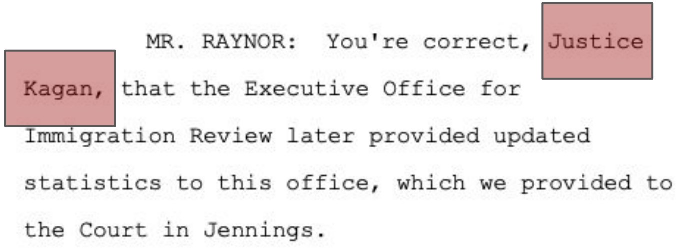

MR. RAYNOR: You're correct, Justice
Kagan, that the Executive Office for
Immigration Review later provided updated
statistics to this office, which we provided to
the Court in Jennings.

2-grams

"you're correct"

"correct justice"

n-grams


MR. RAYNOR: You're correct, Justice Kagan, that the Executive Office for Immigration Review later provided updated statistics to this office, which we provided to the Court in Jennings.

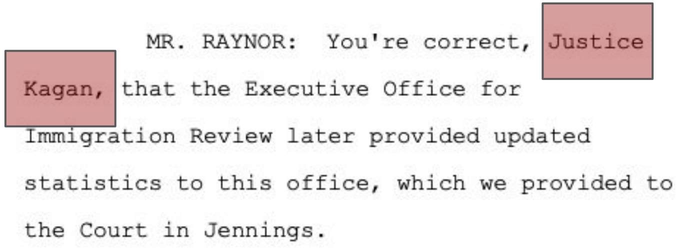
2-grams

"you're correct"

"correct justice"

"justice Kagan"

n-grams



MR. RAYNOR: You're correct, Justice Kagan, that the Executive Office for Immigration Review later provided updated statistics to this office, which we provided to the Court in Jennings.

2-grams

"You're correct"

"correct justice"

"justice Kagan"

Pro:

- n-grams are easy to compute.
- You don't need any annotation
  - That is the NLP word for what social scientists called "coding"; someone goes and codes/labels what is happening in the text
- special software or linguistic knowledge
- If you get all n-grams up to some large n, you will extract most phrases (high recall)

Con:

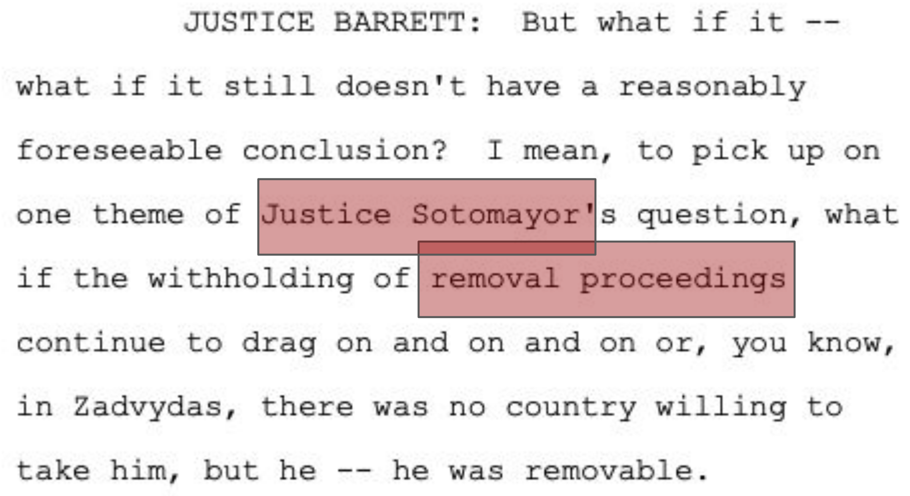
- Too many n-grams to store & index!
  - A sequence of length K will contain a total of K - n + 1 subsequences of length n (i.e. n-grams)
  - See Figure 2 in the paper

You have options

- n-grams
- **Named entities**
- Constituents
- Regular expression over part-of-speech tags

Named entities

Roughly: token spans automatically tagged by category (with a package)



```
          JUSTICE BARRETT:  But what if it --
what if it still doesn't have a reasonably
foreseeable conclusion?  I mean, to pick up on
one theme of Justice Sotomayor's question, what
if the withholding of removal proceedings
continue to drag on and on and on or, you know,
in Zadvydas, there was no country willing to
take him, but he -- he was removable.
```

Pro:

- Many of the phrases you want will be named entities (e.g. "Justice Sotomayor")
- Many packages for entity tagging

Con:

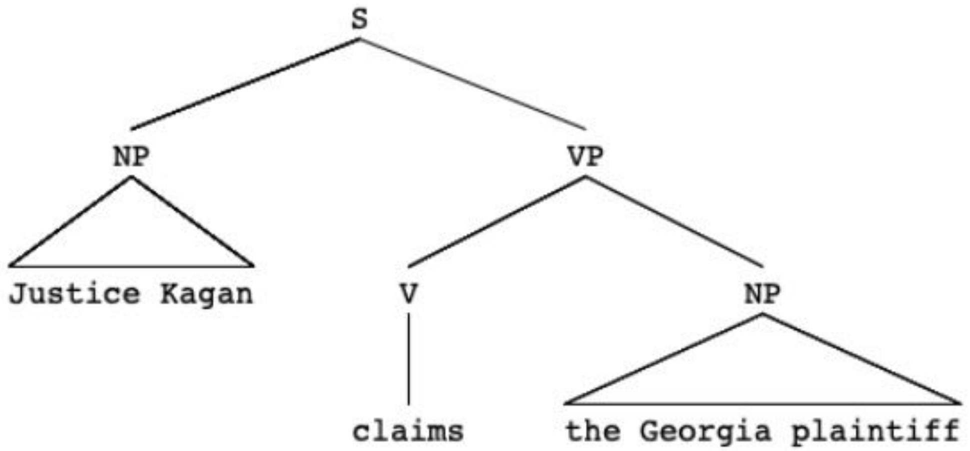
- In NLP there are implicit definitions (and annotation decisions) about what kinds of spans count as entities
  - Common entities: people, organizations, places...
- So not all of the phrases you want will be named entities (e.g. "removal proceedings")

You have options

- n-grams
- Named entities
- **Constituents**
- Regular expression over part-of-speech tags

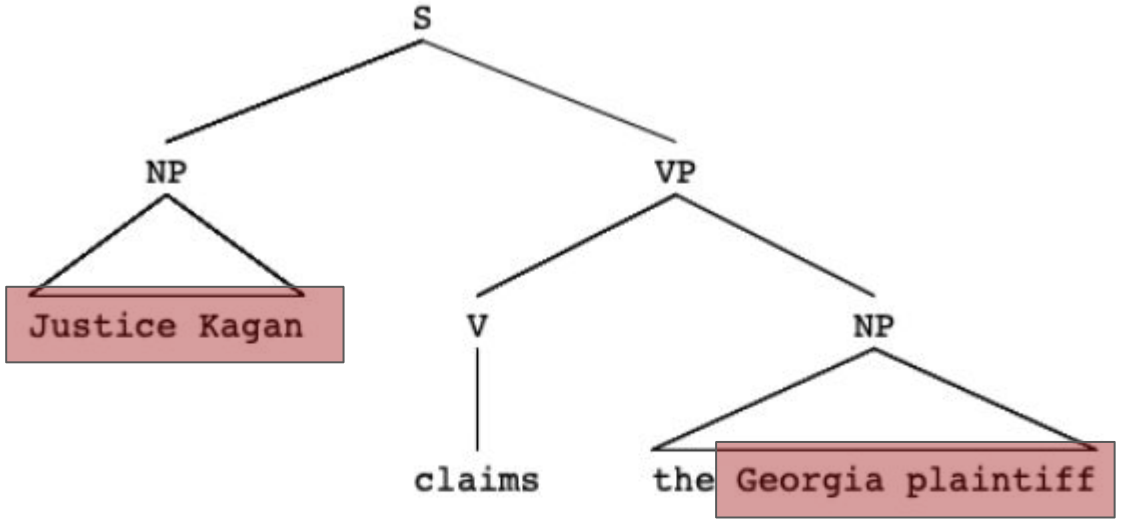
Constituent parse tree

A **constituent parse tree** is a representation of the syntactic structure of a sentence. Many NLP packages use machine learning to infer unobserved constituent parses (usually based on training data)



"Justice Kagan claims the Georgia plaintiff"

Can you use subtrees as phrases?



Pro:

- Build on / use much knowledge of constituents

Con:

- The phrases you want may not be constituents, as defined by annotation guidelines for the training data
  - e.g., "the Georgia plaintiff"
  - In practice, you are tied to annotation decisions behind the training data for your parser
    - These decisions may not be right for your project
- Parsers can be slow
  - If you have some data and want to answer a question, you may have to wait days for the parser on your laptop

You have options

- n-grams
- Named entities
- Constituent parsing
- **Regular expression over part-of-speech tags**

Our proposed approach

Regular expression over ~~part-of-speech tags~~ characters

aabda**aaabbbbc**

`a*b*c?`

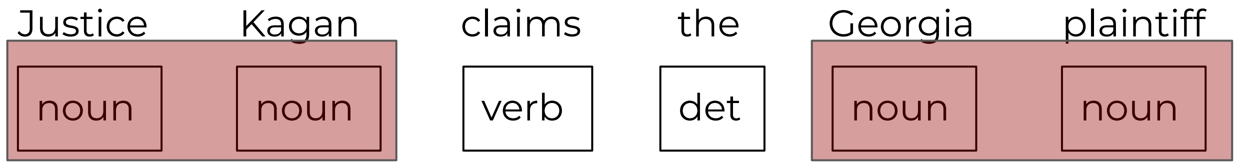
[See Jurafsky and Martin's textbook]

Regular expression over **part-of-speech tags**

Justice	Kagan	claims	the	Georgia	plaintiff
noun	noun	verb	det	noun	noun

[See Jurafsky and Martin's textbook]

Regular expression over part-of-speech tags

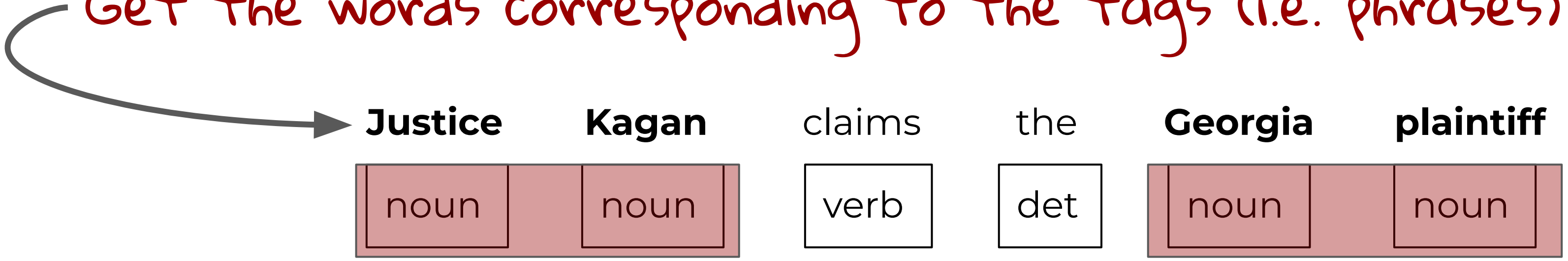


```
(adjective|noun)*noun
```

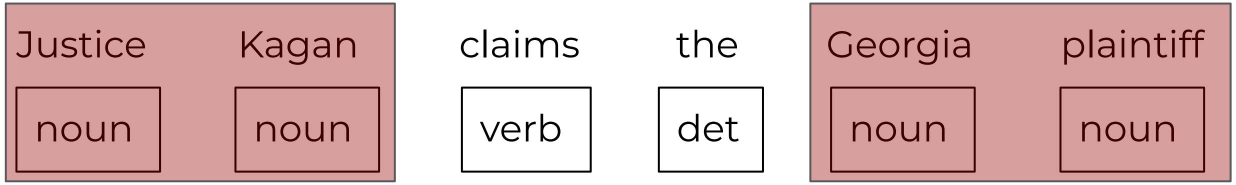
A simple example regular expression

Regular expression over part-of-speech tags

Get the words corresponding to the tags (i.e. phrases)



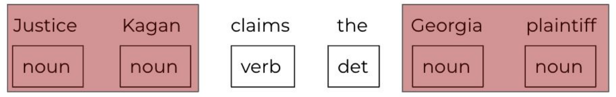
Regular expression over part-of-speech tags



```
(A|N)*N(PD*(A|N)*N)*
```

The full grammar used in our companion software
(Our paper also uses fast tools to find matches)

Regular expression over part-of-speech (POS) tags



```
(adjective|noun)*noun
```

A simple example regular expression

Pro:

- **Computationally light**
  - POS taggers are really fast
- **Annotation light**
  - Using POS tags has less reliance on annotation than entity or constituent extraction
  - POS taggers work on different kinds of text
- **But gets lots of important phrases**
  - The method has high recall with low yield

Con:

- Multiple overlapping and nested spans will match the same regular expression, which can be annoying in post-processing
- Not distributed or contextual (more on this shortly)

You can also roll your own "phrases" using a regular expression

Consider writing your own phrasefinding regular expression, based on both words and POS tags!

```
theory of the petitioner
theory of the government
theory of indemnity
theory of breach
theory of the court of appeals
theory of the labor management relations act
theory of federal jurisdiction
theory of the defendant
theory of guilt
```

```
(theory)(of)D*(A|N)*N)*
```

Consider writing your own phrasefinding regular expression!

Companion code shows one simple way to do this with phrasemachine
(there are other ways too)

```
theory of the petitioner
theory of the government
theory of indemnity
theory of breach
theory of the court of appeals
theory of the labor management relations act
theory of federal jurisdiction
theory of the defendant
theory of guilt
```

`(theory)(of)D*(A|N)*N)*`

Limitations of any discrete phrase extraction (cue Shufan)

- Not distributed (i.e. no embedding)
  - "immigration judge" and "immigration hearing" are similar, so representation should reflect that. Discrete phrase extraction can't do that.
- Not contextual
  - "UK immigration hearing" vs "immigration hearing in Texas"
  - The meaning of "immigration hearing" is slightly different in context, but there is no way to represent the context
- Phrase-BERT can address these **big** disadvantages, but it is maybe not the best place to start
  - My advertisement for discrete phrase extraction versus embeddings:
    - *"almost as good and a whole lot cheaper"*

Much more detail in our paper "Bag of What"

Matt Denny

Hanna Wallach

Brendan O'Connor

Alternately, `$ pip install phrasemachine`