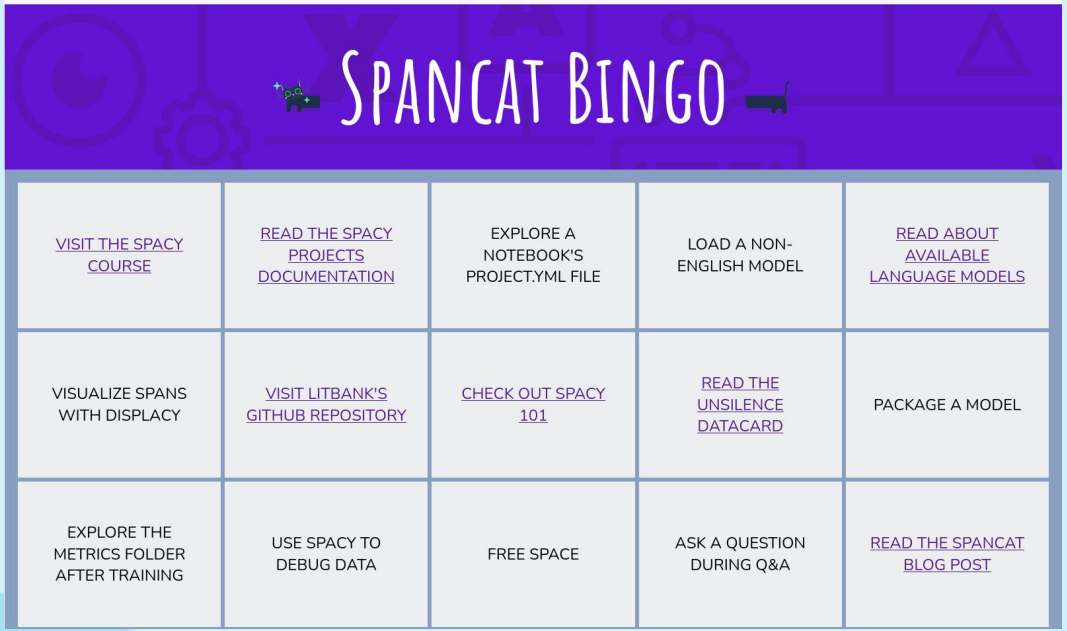
spaCy's Span Categorizer

Digital Humanities Applications

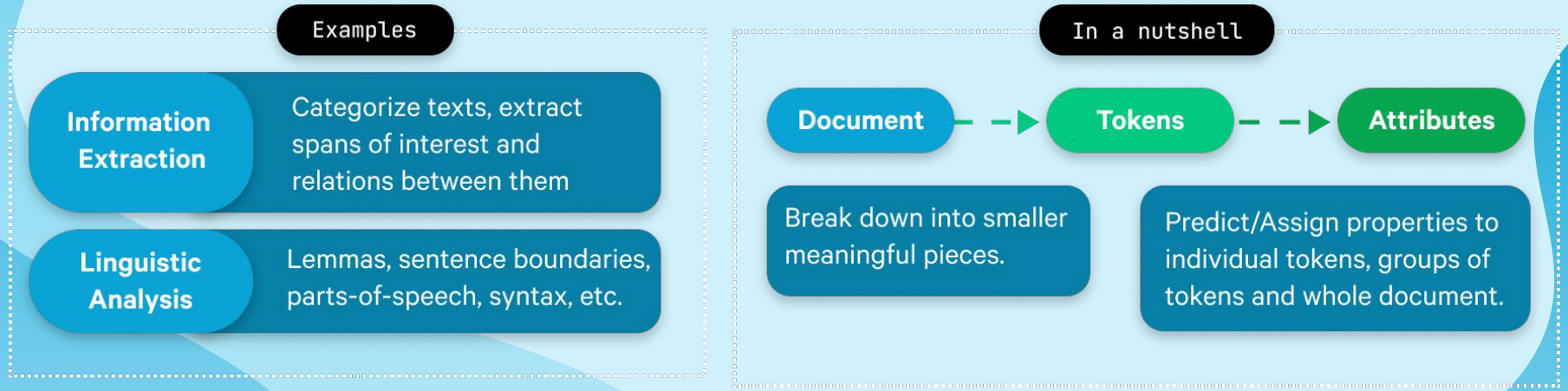
Overview

- spaCy Intro
- NER vs. Spancat
- Spancat Intro
- Spancat Use Cases
  - LitBank
  - UnSilence

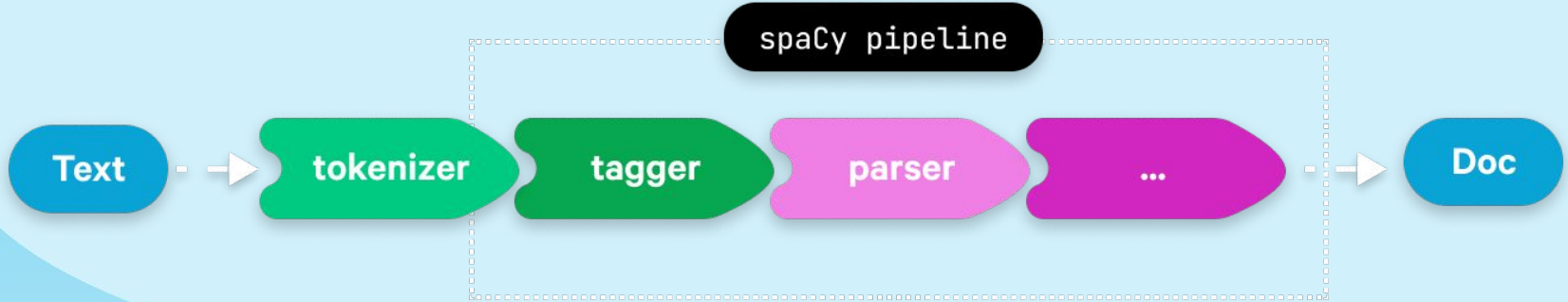


spancat-bingo.netlify.app

Natural Language Understanding



spaCy: Pipelines



spaCy provides a modular architecture to construct NLP pipelines that can be tailored towards individual needs.

spaCy: Pipelines and Docs

The nlp object

The processing pipeline object, contains all the different components.

```
import spacy

# Create a blank English nlp object
nlp = spacy.blank("en")
```

The doc object

The Doc lets you access information about the text in a structured way, and no information is lost.

```
# process a string of text with the nlp object
doc = nlp("Hello world!")

# Iterate over tokens in a Doc
for token in doc:
    print(token.text)
```

Output

```
Hello
world
!
```

spaCy: Tokens and Spans

The token object

Represent the tokens in a document – for example, a word or a punctuation character.

```
# Index into the Doc to get a single Token
token = doc[1]

# Get the token text via the .text attribute
print(token.text)
```

Output

```
world
```

The span object

A slice of the document consisting of one or more tokens. Doesn't contain data.

```
# A slice from the Doc is a Span object
span = doc[1:3]

# Get the span text via the .text attribute
print(span.text)
```

Output

```
world!
```

spaCy: Lexical Attributes

```
doc = nlp("It costs $5.")

print("Index: ", [token.i for token in doc])
print("Text: ", [token.text for token in doc])

print("is_alpha:", [token.is_alpha for token in doc])
print("is_punct:", [token.is_punct for token in doc])
print("like_num:", [token.like_num for token in doc])
```

Output

```
Index:    [0, 1, 2, 3, 4]
Text:     ['It', 'costs', '$', '5', '.']

is_alpha: [True, True, False, False, False]
is_punct: [False, False, False, False, True]
like_num: [False, False, False, True, False]
```

These attributes are also called **lexical attributes**: they refer to the entry in the vocabulary and don't depend on the token's context.

spaCy: Pipeline Components

- Models that enable spaCy to predict linguistic attributes in context
- Trained on annotated example texts
- Can be updated with more examples to fine-tune predictions

```bash
$ python -m spacy download en_core_web_sm
```

```python
import spacy

# Load a trained pipeline
nlp = spacy.load("en_core_web_sm")
```

Machine Learning

tagger

Assigns part-of-speech tags to tokens.

Machine Learning

parser

Analyzes syntactic structure and assigns dependency relations between tokens.

Machine Learning

ner

Identifies non-overlapping named entities.

spaCy: Part-of-Speech Tagging

```python
import spacy

# Load the small English pipeline
nlp = spacy.load("en_core_web_sm")

# Process a text
doc = nlp("She ate the pizza")

# Iterate over the tokens
for token in doc:
    # Print the text and the predicted part-of-speech tag
    print(token.text, token.pos_)
```

Output

```
She PRON
ate VERB
the DET
pizza NOUN
```

Let's take a look at the model's predictions. In this example, we're using spaCy to predict **part-of-speech tags**, the word types in context.

spaCy: Dependency Parsing

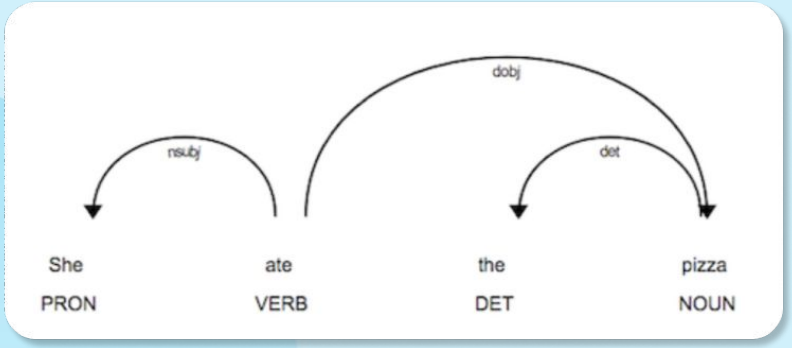
```
doc = nlp("She ate the pizza")

# Iterate over the tokens
for token in doc:
    print(token.text, token.pos_, token.dep_, token.head.text)
```

Output

```
She PRON nsubj ate
ate VERB ROOT ate
the DET det pizza
pizza NOUN dobj ate
```

In addition to the part-of-speech tags, we can also predict how the words are related. For example, whether a word is the subject of the sentence or an object.



spaCy: NER

```
# Process a text
doc = nlp("Apple is looking at buying U.K. startup for
$1 billion")

# Iterate over the predicted entities
for ent in doc.ents:
    # Print the entity text and its label
    print(ent.text, ent.label_)
```

Output

```
Apple ORG
U.K. GPE
$1 billion MONEY
```

Named entities are "real world objects" that are assigned a name – for example, a person, an organization or a country.
The `doc.ents` property lets you access the named entities predicted by the named entity recognition model.

spaCy: Span Labeling

```python
import spacy
from spacy.tokens import Span

text = "Welcome to the Bank of China."
nlp = spacy.blank("en")
doc = nlp(text)

doc.spans["sc"] = [
Span(doc, 3, 6, "ORG"),
Span(doc, 5, 6, "GPE"),
]
```

Unlike named entities, which have clear token boundaries and are often comprised of the same syntactic units, spans can be overlapping and composed of arbitrary phrases. The `doc.spans` property lets you access the predicted spans.

spaCy: Rule-Based and ML Components

rule-based & ML

lemmatizer

Assigns base forms to tokens.

```
doc = nlp("Apples are great.")
assert doc[0].lemma_ == "apple"
assert doc[1].lemma_ == "be"
```

Machine Learning

textcat

Predicts categories over a whole document.

```
doc = nlp("Apples are great.")
assert doc.cats["positive"] == 1.0
```

rule based

sentencizer

Custom sentence boundary detection logic without dependency parsing.

```
nlp.add_pipe("sentencizer")
doc = nlp("This is a sentence.
           This is another sentence.")
assert len(list(doc.sents)) == 2
```

spaCy Demo

https://github.com/adrianeboyd/workshop-dh2023/


https://spacy.io/api/spancategorizer
SpanCategorizer
SOURCE
CLASS,EXPERIMENTAL
V3.1
STRING NAME: spancat
BASE CLASS: TrainablePipe
TRAINABLE:
Pipeline component for labeling potentially overlapping spans of text
Spancat
COMPONENT

(Named) Entity Recognition

- **Proper names**
  - Also: dates, times, amounts

NER Example: CoNLL 2003

La petición del **Abogado General** tiene lugar después de que un juez del **Tribunal Supremo**
PER ORG

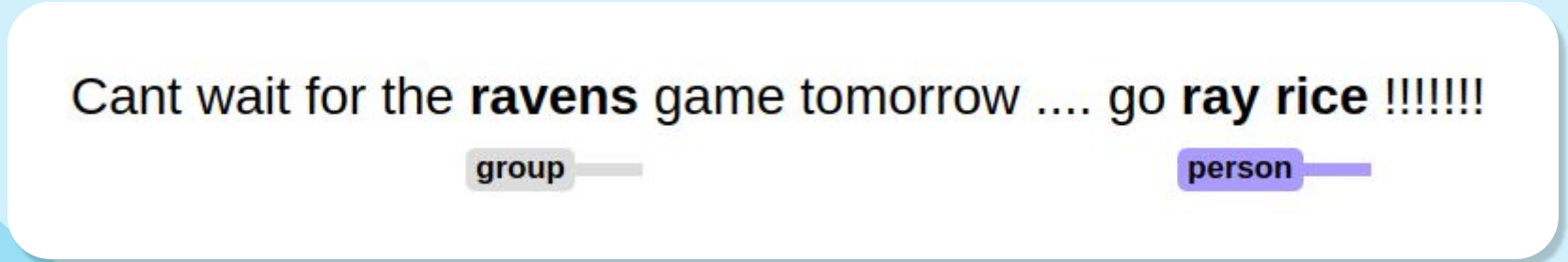
del estado de **Victoria** (**Australia**) se viera forzado a disolver un jurado popular y suspender
LOC LOC

el proceso ante el argumento de la defensa de que las personas que lo componían podían

haber obtenido información sobre el acusado a través de la página **CrimeNet** .
MISC

LOC, MISC, ORG, PER

NER Example: WNUT 2017



corporation, creative-work, group, location, person, product

(N)ER Example: Anatomical Entity Mention

Erythroblasts in the center of **blood islands** (spherical **cells** in Fig . 1e) weakly express

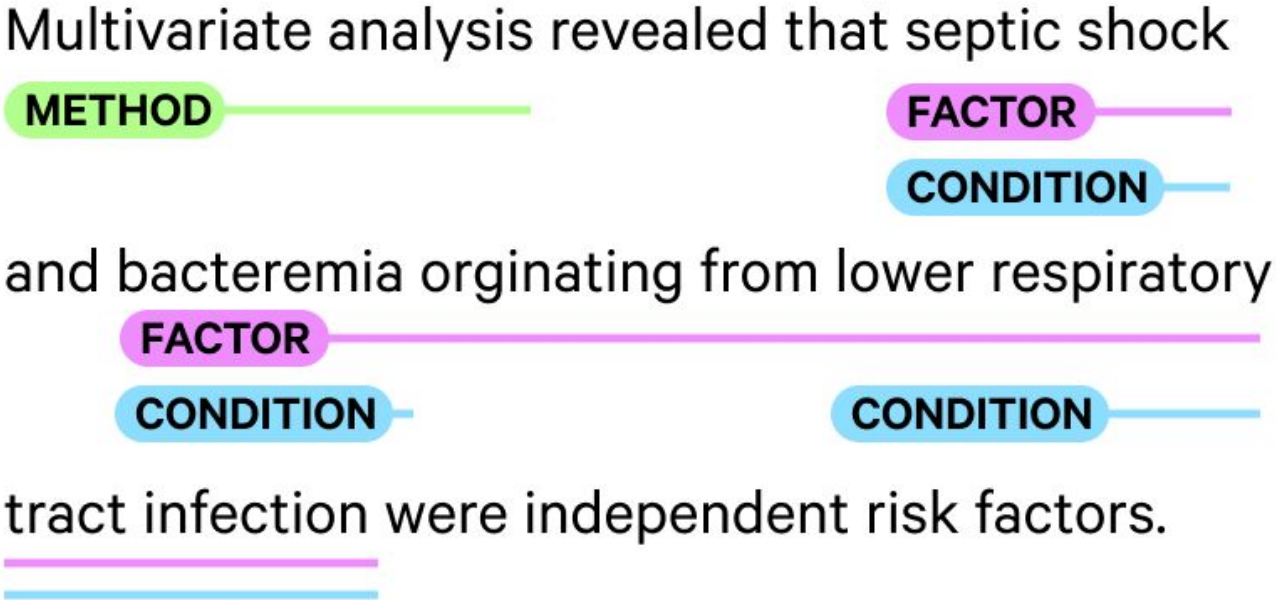
Cell Organism_substance Cell

CCN3 and their expression is mildly enhanced in later stage (data not shown) .

Anatomical_system, Cell, Cellular_component, Developing_anatomical_structure, Immaterial_anatomical_entity, Multi-tissue_structure, Organ, Organism_subdivision, Organism_substance, Pathological_formation, Tissue

Nested NER

Less NER-Like Spans



spaCy: NER vs. Spancat

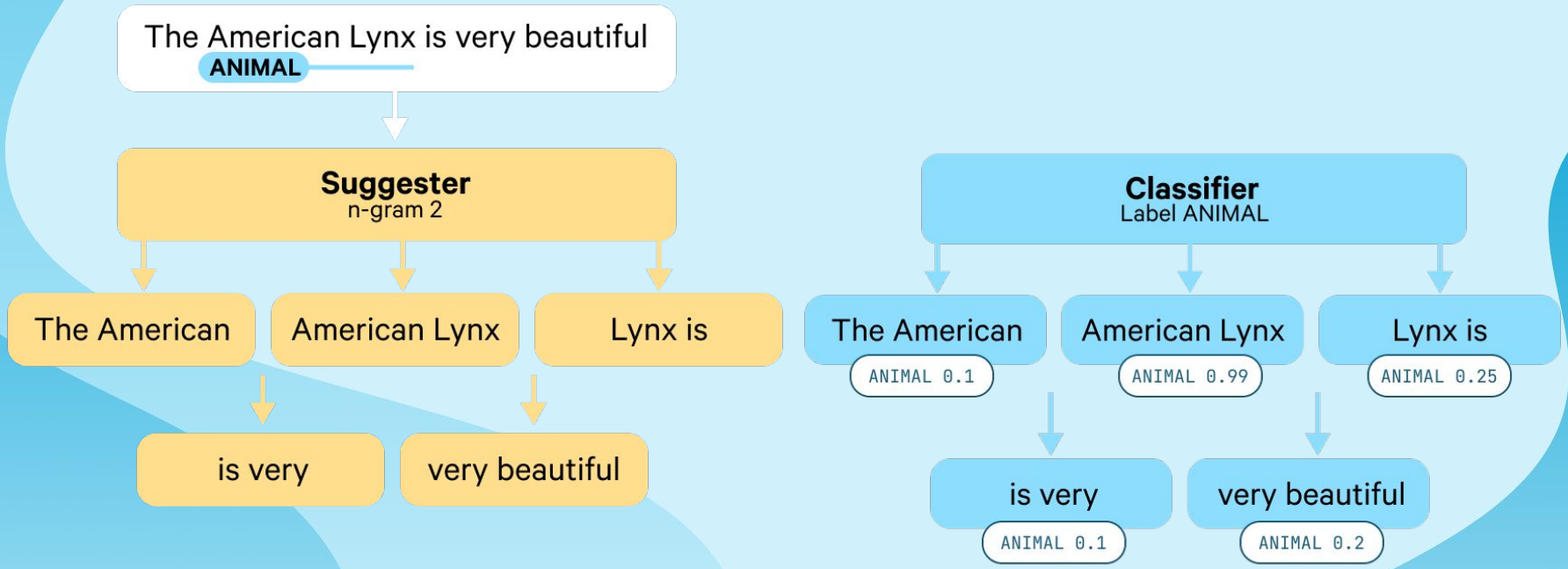
NER:

- Have non-overlapping (named) entities

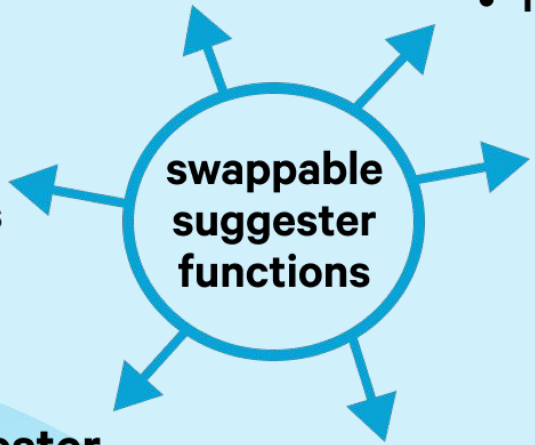
Spancat:

- Nested or overlapping spans
- Need for confidence scores
- Long spans or spans that cross sentence boundaries

Spancat Overview



Spancat Suggesters



Chunk suggester

- noun chunk iterator

Sentence suggester

- full sentences

Subtree suggester

- syntactic dependencies

SpanFinder

- machine learning approach
- learns start and end tokens

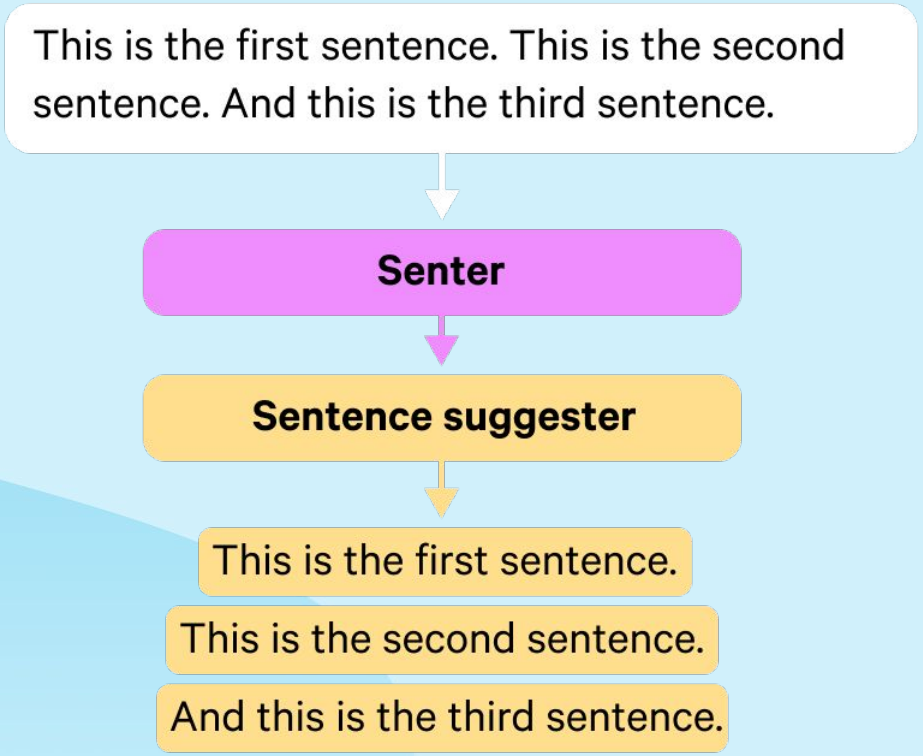
n-gram suggester

- a certain number of tokens

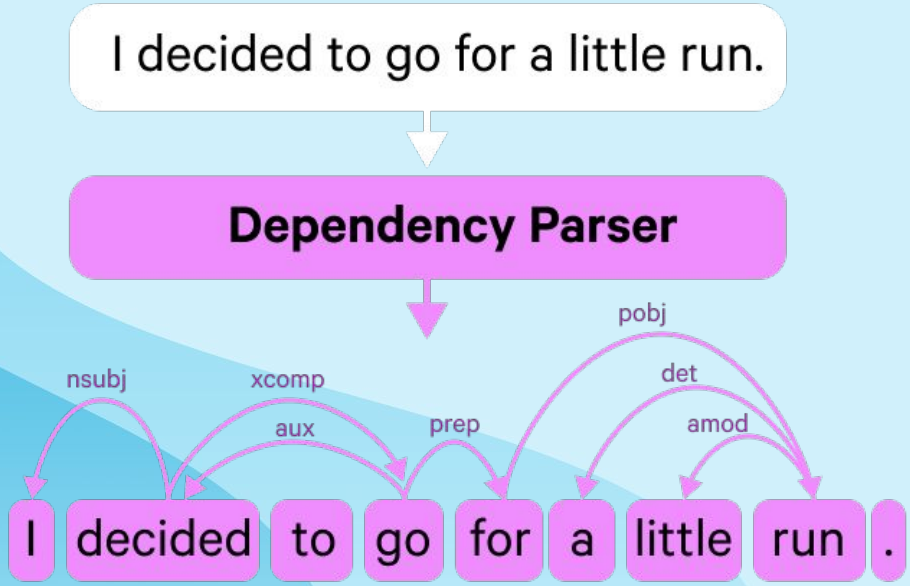
SpanRuler

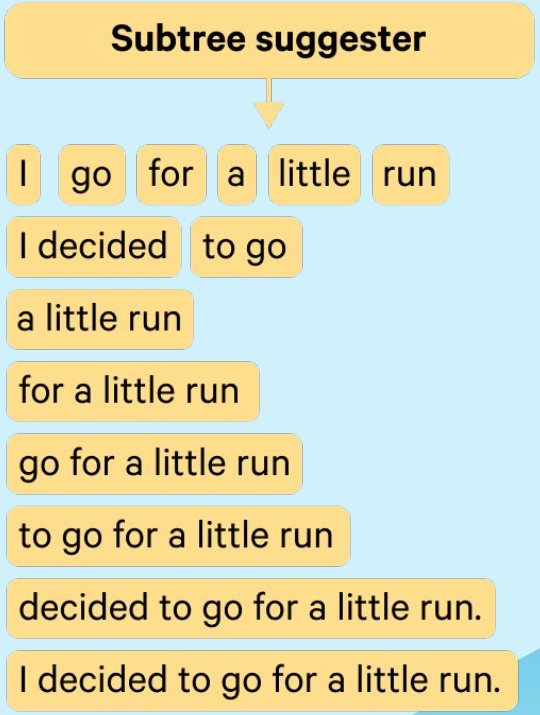
- rule-based matching

Spancat Sentence Suggester

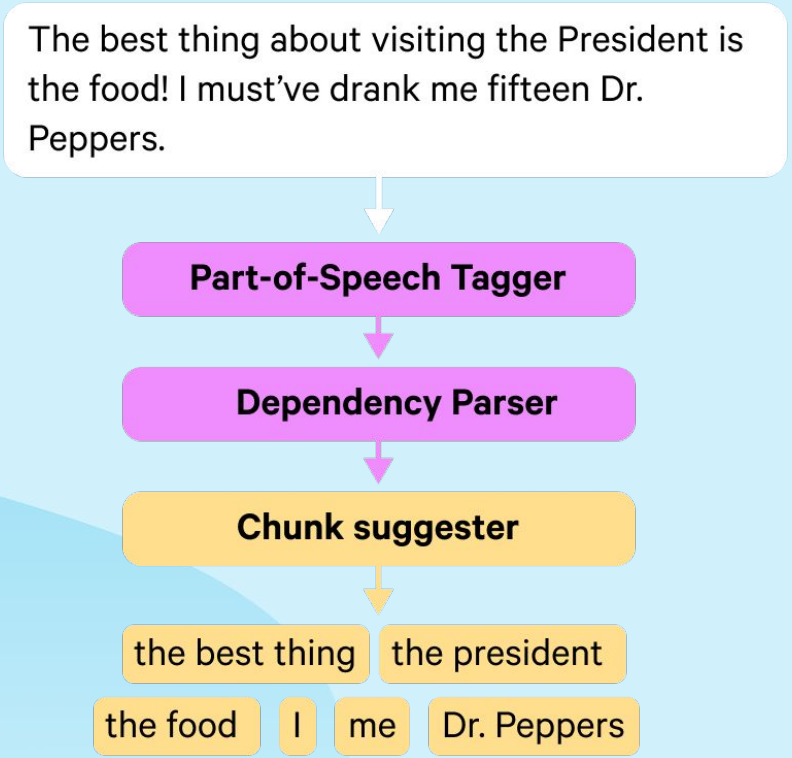


Spancat Subtree Suggester





Spancat Noun Chunk Suggester



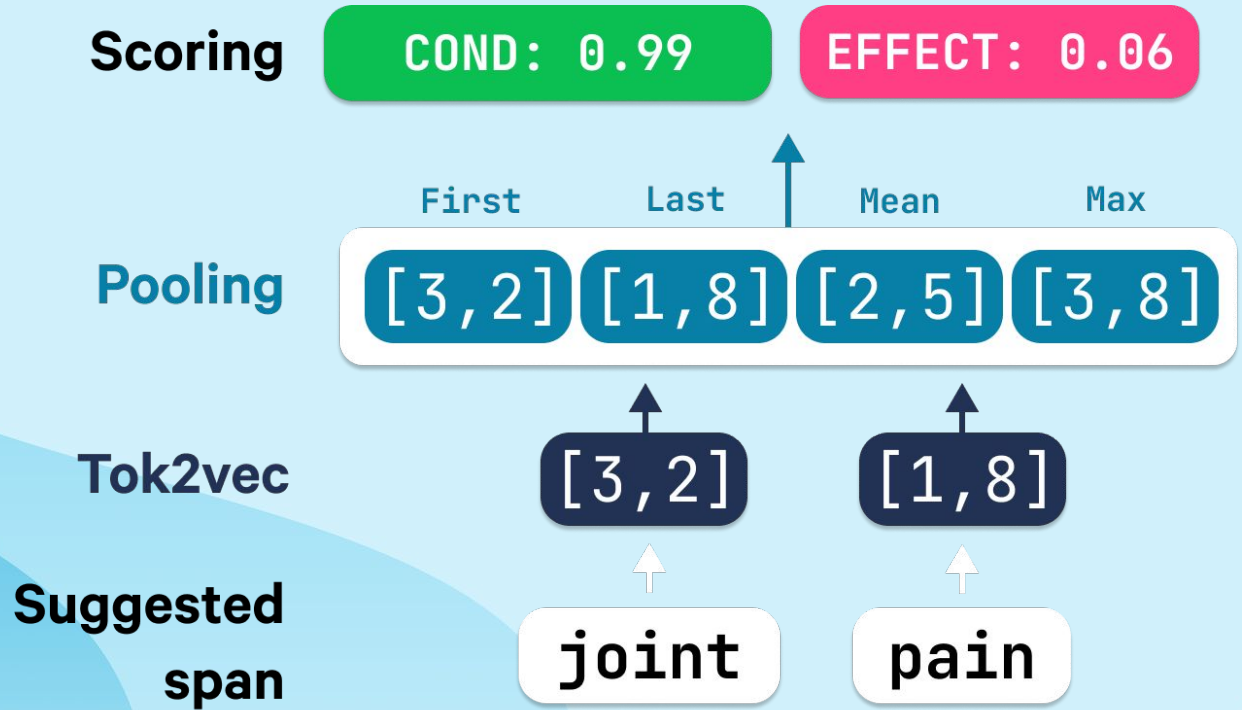
Spancat Suggester Requirements

- **N-gram**
  - Token segmentation (tokenizer)
- **Span ruler**
  - Rule-based spans (matcher patterns)
- **Sentences**
  - Rule-based sentence boundaries (sentencizer)
  - ML sentence boundaries (senter, parser)
- **Subtrees**
  - ML dependency parses (parser)
- **Span finder**
  - ML spans (span_finder)
- **Noun chunks**
  - ML part-of-speech tags + parses (morphologizer + parser)

Spancat Suggester Recommendations

- **N-gram**
  - Short spans
  - Lots of RAM
- **Span finder**
  - Longer spans
- **Span ruler**
  - High-recall patterns
- **Subtrees / noun chunks**
  - Spans are subtrees / noun chunks
  - High-quality core pipeline (en_core_web_lg)

Spancat Model



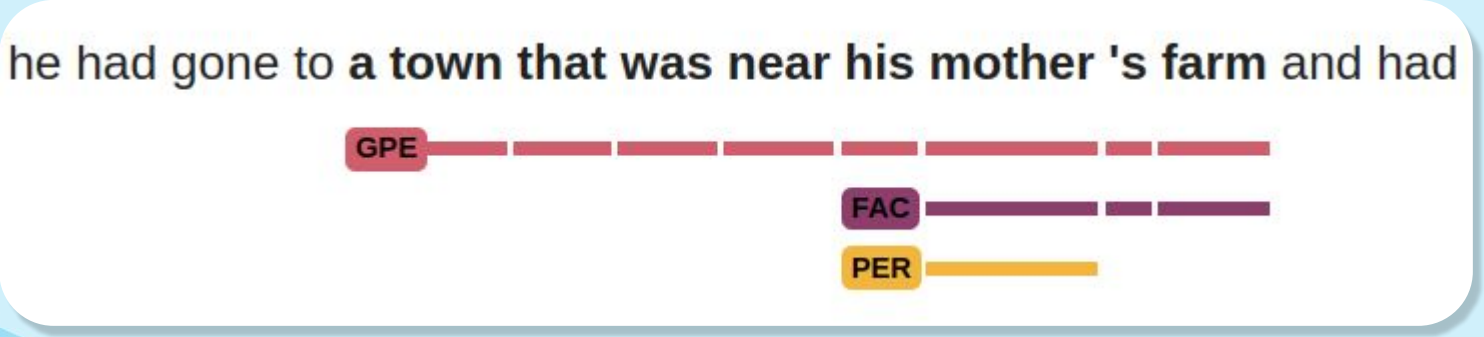
LitBank

- Named and non-named entities from literary texts

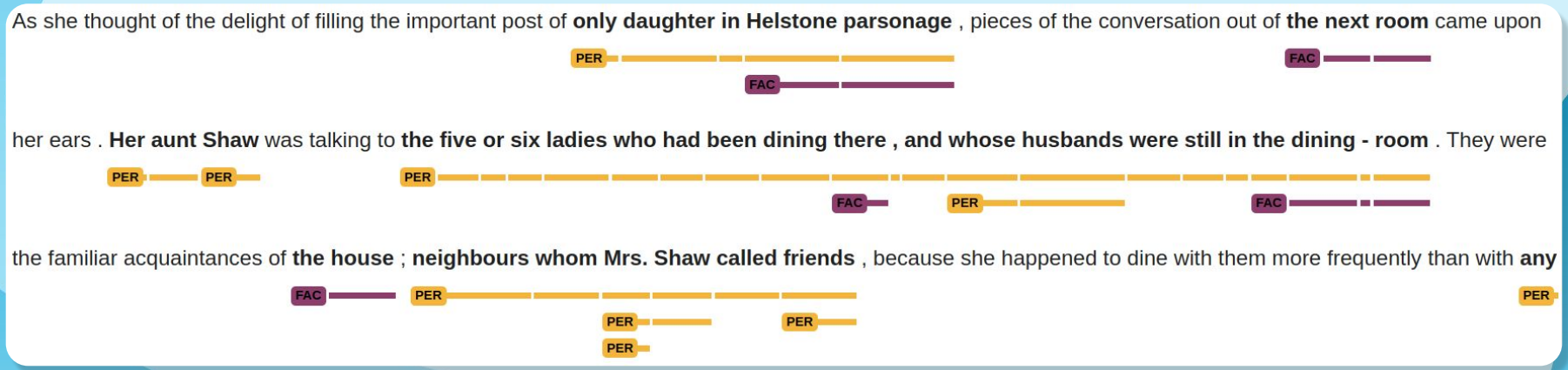
Entity type	Count	Examples
PER	9,383	my mother, Jarndyce, the doctor, a fool, his companion
FAC	2,154	the house, the room, the garden, the drawing-room, the library
LOC	1,170	the sea, the river, the country, the woods, the forest
GPE	878	London, England, the town, New York, the village
VEH	197	the ship, the car, the train, the boat, the carriage
ORG	130	the army, the Order of Elks, the Church, Blodgett College

David Bamman, Sejal Popat and Sheng Shen (2019), "An Annotated Dataset of Literary Entities," NAACL 2019.

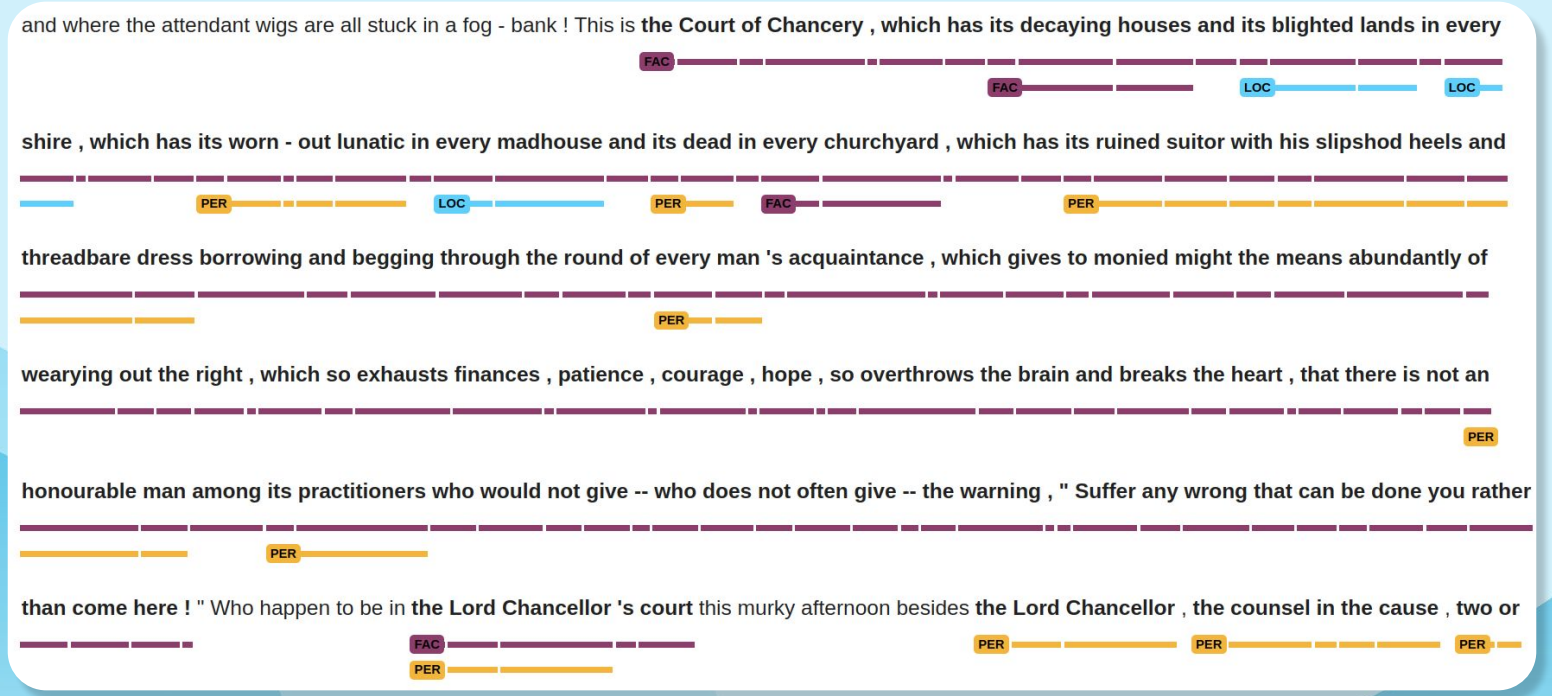
LitBank: Nested Spans



LitBank: Nested Spans



LitBank: Nested Spans, Long



LitBank: Spans

Span Type	Length	SD	BD	N
LOC	2.73	1.87	1.13	995
FAC	2.67	1.54	0.94	1746
GPE	1.57	2.69	1.19	687
PER	2.13	1.02	0.75	7392
VEH	2.35	3.26	1.30	182
ORG	3.67	2.53	1.51	111
Wgt. Average	2.25	1.34	0.86	-

LitBank: Spans

ℹ️ Over 90% of spans have lengths of 1 -- 6 (min=1, max=131).

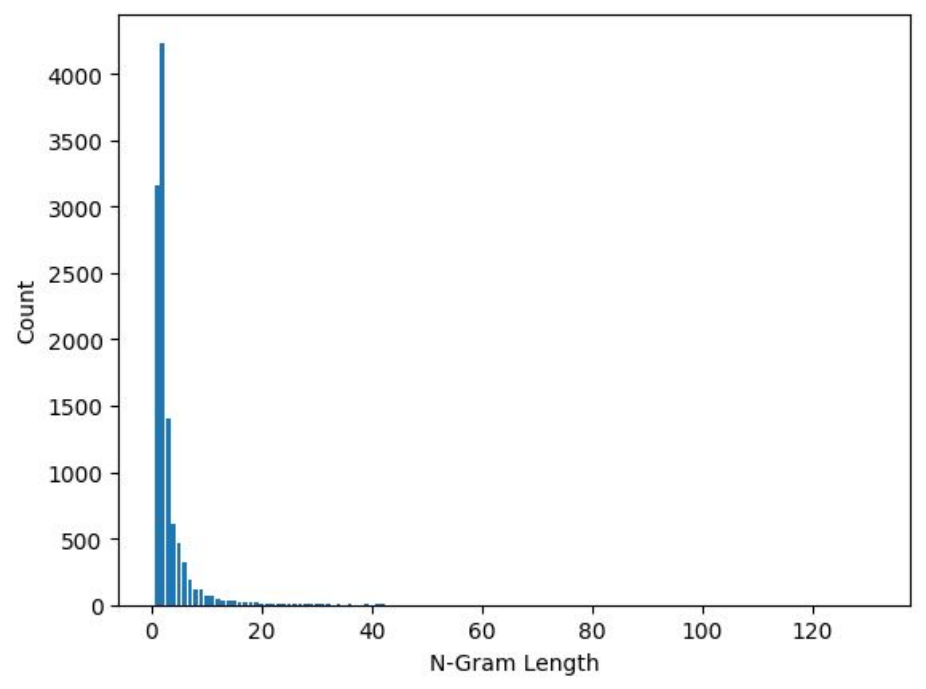
The most common span lengths are: 1 (28.44%), 2 (38.1%), 3 (12.67%), 4 (5.52%), 5 (4.17%), 6 (2.92%).

✔ Spans are distinct from the rest of the corpus

⚠️ Boundary tokens are not distinct from the rest of the corpus

✔ Good amount of examples for all labels

LitBank: Span Lengths



LitBank: Spancat CNN Configuration

- **Suggester:**
  - N-grams: 1-grams to 8-grams
- **Model:**
  - Default `tok2vec` with `en_core_web_lg` word vectors

LitBank: Spancat N-Gram CNN Results

	P	R	F
FAC	0.791	0.502	0.615
GPE	0.714	0.545	0.618
LOC	0.769	0.500	0.606
ORG	0.000	0.000	0.000
PER	0.830	0.725	0.773
VEH	0.000	0.000	0.000
Overall (micro)	0.814	0.661	0.730

LitBank: Spancat Transformer Configuration

- **Suggester:**
  - N-grams: 1-grams to 8-grams
- **Model:**
  - Default `transformer` with `roberta-base`

LitBank: Spancat N-Gram Transformer Results

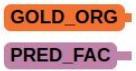
	P	R	F
FAC	0.802	0.661	0.725
GPE	0.767	0.881	0.820
LOC	0.672	0.683	0.678
ORG	0.500	0.154	0.235
PER	0.880	0.835	0.857
VEH	0.500	0.600	0.545
Overall (micro)	0.849	**0.800**	0.823

LitBank: N-Gram CNN Model Predictions

thing , and putting other considerations aside , I am **an orderly man** and do n't like that . This is by the way . Third reason :

Because I want **my boy Harry , who is over there at the hospital in London studying to become a doctor** , to have

something to amuse him and keep him out of mischief for a week or so . **Hospital** work must sometimes pall and grow rather



LitBank: Suggesters (CNN)

	# Suggestions	Recall	GPU RAM	F-Score	Speed
Litbank (dev)	1299				
N-Grams 1-8	178720	0.95	7 GB	0.730	9113
Subtree	34833	0.91	4 GB	0.719	9423
Span Finder	3527	0.71	4 GB	0.727	5420
Noun Chunk	5111	0.64	–	–	–

LitBank: Suggesters (TRF)

	# Suggestions	Recall	GPU RAM	F-Score	Speed
Litbank (dev)	1299				
N-Grams 1-8	178720	0.95	17 GB	0.823	6955
Subtree	33918	0.92	13 GB	0.817	5879
Span Finder	3860	0.88	9 GB	0.828	3997
Noun Chunk	5111	0.64	–	–	–

LitBank Demo

https://github.com/adrianeboyd/workshop-dh2023/

Revisit an NER Experiment using Span Categorization

Unsilencing Colonial Archives via Automated Entity Recognition

by Mrinalini Luthra, Konstantin Todorov, Charles Jeurgens, Giovanni Colavizza

Dutch East India Company (VOC) Archives

Named and Unnamed Persons

“...to Emancipate his slave girl called Maitac of Sumbauwa…”

"...and his possessions consisting mostly of clothes and household objects, a slave Boy..."

UnSilence: Spans

```
ℹ Over 90% of spans have lengths of 1 -- 5 (min=1, max=36). The most
common span lengths are: 1 (23.83%), 2 (27.35%), 3 (24.64%), 4 (11.17%), 5
(5.82%). If you are using the n-gram suggester, note that omitting infrequent
n-gram lengths can greatly improve speed and memory usage.
✔ Spans are distinct from the rest of the corpus
✔ Boundary tokens are distinct from the rest of the corpus
✔ Good amount of examples for all labels
✔ Examples without occurrences available for all labels
```

UnSilence: NER vs. Spancat

	Model	Precision	Recall	F1
NER	CRF Baseline (NER, Macro)	.37	.28	.32
	BERTje + Bi-LSTM-CRF (NER, Macro)	.47	.37	.41
	spaCy ner	.39	.28	.32
	ner_trf	.49	.45	.47
spancat	spancat	.70	.44	.54
	spancat_trf	.66	.51	.58

UnSilence: N-Grams vs. Span Finder

```
[components.spancat.suggester]
@misc = "spacy.ngram_range_suggester.v1"
min_size = 1
max_size = 6

[components.spancat.suggester]
@misc =
"spacy-experimental.span_finder_suggester.v1"
candidates_key =
${components.span_finder.predicted_key}
```

UnSilence Model Predictions

ende regteren te moogen genieten — aldus gedaan ende gepasseerd **in 'thospitaa** voorsz: ter -presentie

GOLD_Place

van **Sijmon Thomp Carelsz:,** en **Johannes Stevens clend als getuijgen;** die de minute deses in „vens

GOLD_Person GOLD_Person

PRED_Person PRED_Person

den testateur item den Executeur ende mij notaris hebben onderteekendt /:onderstond:/ Quod Attestor /:was

getektd **JJ„n V„n Visvliet nots:** publ: Accordeert **A. Summer** e

GOLD_Person GOLD_Person

UnSilence Demo

https://github.com/adrianeboyd/workshop-dh2023/

Thanks

- spaCy team including Ákos Kádár, Edward Schmuhl, Victoria Slocum
- NewNLP team: Toma Tasovac, Natalia Ermolaev, Andrew Janco, David Lassner, Nick Budak, Jajwalya Karajgikar

newnlp.princeton.edu

References and Links

- LitBank
  - David Bamman, Sejal Popat and Sheng Shen (2019), "An Annotated Dataset of Literary Entities," NAACL 2019.
  - https://github.com/dbamman/litbank
- UnSilence
  - Luthra, Mrinalini, Konstantin Todorov, Charles Jeurgens, and Giovanni Colavizza. "Unsilencing colonial archives via automated entity recognition." Journal of Documentation (2023).
  - https://github.com/budh333/UnSilence_VOC
- spaCy
  - spaCy docs: https://spacy.io
  - spaCy forum: https://github.com/explosion/spacy/discussions
  - Spancat: https://explosion.ai/blog/spancat

References and Links

- CoNLL 2003: Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition (Tjong Kim Sang & De Meulder, CoNLL 2003)
- WNUT 2017: Results of the WNUT2017 Shared Task on Novel and Emerging Entity Recognition (Derczynski et al., WNUT 2017)
- AnEM: Open-domain Anatomical Entity Mention Detection (Ohta et al., 2012)