Integrated Annotation of Event Structure, Object States, and Entity Coreference

Kyeongmin Rim(*), James Pustejovsky

December 6, 2023
CRAC 2023

Let’s make a meat & veggie casserole!

Cut carrots and zucchini into cubes.

Slice onions and saute in olive oil, add chopped vegetables and cook for 10 mins.

In a separate pan, saute minced meat, add canned tomatoes.

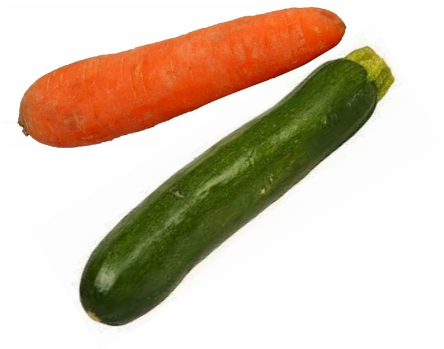
Transfer the meat into the pan with the vegetables, season to taste.

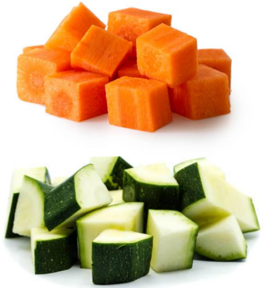
Let’s make a meat & veggie casserole!

Cut carrots and zucchini into cubes.

Let’s make a meat & veggie casserole!

Cut carrots and zucchini into cubes.

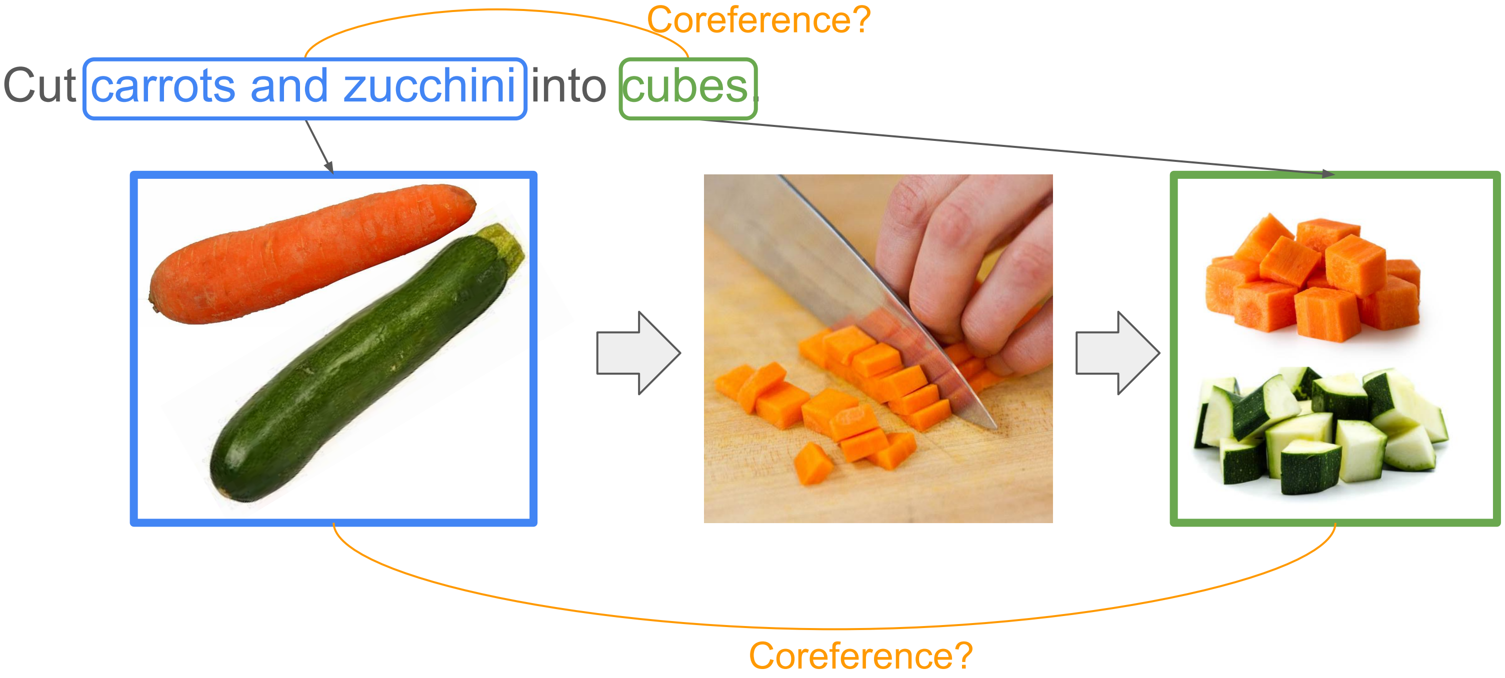




Let’s make a meat & veggie casserole!

Cut carrots and zucchini into cubes.

Let’s make a meat & veggie casserole!



Let’s make a meat & veggie casserole!

Cut carrots and zucchini into cubes.

Slice onions

Let’s make a meat & veggie casserole!

Cut carrots and zucchini into cubes.

Slice onions

???

Let’s make a meat & veggie casserole!

Cut carrots and zucchini into cubes.

Slice onions and saute in olive oil, add chopped vegetables and cook for 10 mins.

Let’s make a meat & veggie casserole!

Cut carrots and zucchini into cubes.

Slice onions and saute in olive oil, add chopped vegetables and cook for 10 mins.

In a separate pan, saute minced meat, add canned tomatoes.

Let’s make a meat & veggie casserole!

Cut carrots and zucchini into cubes.

Slice onions and saute in olive oil, add chopped vegetables and cook for 10 mins.

In a separate pan, saute minced meat, add canned tomatoes.

Transfer the meat into the pan with the vegetables, season to taste.

Let’s make a meat & veggie casserole!

Cut carrots and zucchini into cubes.

Slice onions and saute in olive oil, add chopped vegetables and cook for 10 mins.

In a separate pan, saute minced meat, add canned tomatoes.

Transfer the meat into the pan with the vegetables, season to taste.

Let’s make a meat & veggie casserole!

Cut carrots and zucchini into cubes.

Slice onions and saute in olive oil, add chopped vegetables and cook for 10 mins.

In a separate pan, saute minced meat, add canned tomatoes.

Transfer the meat into the pan with the vegetables, season to taste.

Coreferences?

Let's make a meat & veggie casserole!

Cut carrots and zucchini into cubes.

Slice onions and saute in olive oil, add chopped vegetables and cook for 10 mins.

In a separate pan, saute minced meat, add canned tomatoes.

Transfer the meat into the pan with the vegetables, season to taste.

Let’s make a meat & veggie casserole!

Cut carrots and zucchini into cubes.

Slice onions and saute in olive oil, add chopped vegetables and cook for 10 mins.

In a separate pan, saute minced meat, add canned tomatoes.

Transfer the meat into the pan with the vegetables, season to taste.

Now, given referencing is transitive relation, are they all coreference?

The problems

1. Are carrot before cubed and after cubed anaphoric (or coreference) at all?
2. When there's lots of hidden/drop entities, how can we improve coreference annotation?
3. Are coreference links transitive or commutative?

Previous study

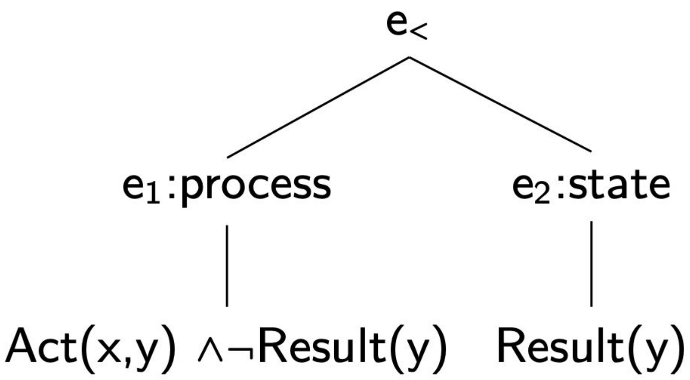
- Marta, Hovy, Martí. 2011. *Identity, non-identity, and near-identity: Addressing the complexity of coreference.*
    - Introduced “near-identity” as a type of anaphora
- Roesiger, Riester, Kuhn. 2018. *Bridging resolution: Task definition, corpus resources and rule-based experiments.*
    - Introduced “bridging resolution” as a new NLP task, defining “lexical bridging” as a form of set-member and part-whole relations
- Fang, Baldwin, Verspoor. 2022. *What does it take to bake a cake? The RecipeRef corpus and anaphora resolution in procedural text.*
    - Annotated recipe texts based on “conventional” coreference relations and previously defined bridging relations

Our approach: CuT and CuI

- “Near-identity” and “bridging” from the prior study don’t capture event dynamics in discourse development <u>over time</u> (or a sequence of events).
- Generative Lexicon provides semantic typings for introducing both transformation and bridging relationships
- To address the gap in previous theoretical models, we introduce a new type of coreference: Coreference-Under-Transformation, or <u>**CuT**</u>.
- As opposed to the traditional “full-identity” corefereces, which we will call Coreference-Under-Identity, or <u>**CuI**</u>

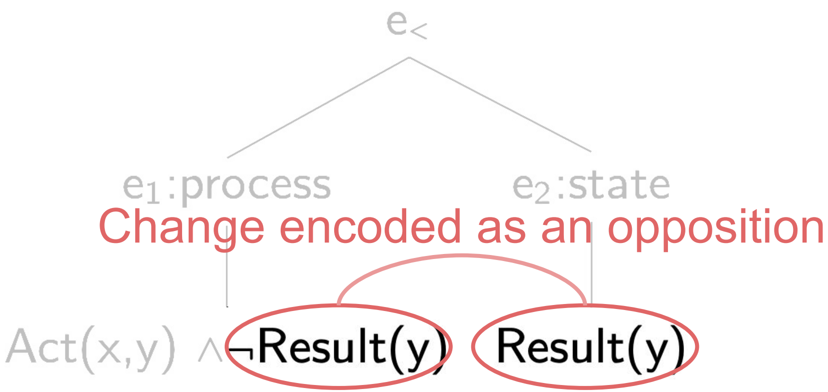
GL - Subevent Structure

- The role of events in argument structure
  - Prototypical transitive predicates encode causation, where one the **agent** acts on another the **theme** or **patient** and brings about a change in the latter.
  - These events are complex and are typically viewed as consisting of *(at least)* two subevents: the causing subevent and the resulting state.



GL - Subevent Structure

- The role of events in argument structure
  - Prototypical transitive predicates encode causation, where one the **agent** acts on another the **theme** or **patient** and brings about a change in the latter.
  - These events are complex and are typically viewed as consisting of *(at least)* two subevents: the causing subevent and the resulting state.



GL - Semantically Implicit Arguments

- Objects not expressed linguistically in the discourse, but which are created as a result of event predicates bringing about a change of state.
- This includes both abstract objects and concrete objects introduced into the discourse through linguistic or nonverbal communicative (situated) acts
  - Ex: Mary translated the book. (result: the translation)
- Introducing d“Generalized Result Nominals, GRN” that incorporates both
  - linguistically realized surface expressions in language, and
  - a formal representation of the creation resulting from any state-changing predicate

Generalized Result Nominal

- Any event predicate inducing a modification, transformation, or creation/destruction of an object, introduces an additional argument denoting the change itself, the GRN.
- **Lexically Encoded Result**: Creation predicates encode the changing argument as a lexically encoded test e.g., build. The object of a creation predicate is itself the GRN.

Process-Oriented Event Model (POEM)

- Simplified from GL event structure, POEM treats events as simple I/O processes, where input and output can be both explicit/verbal or implicit/drop.
- Events link *pre-states* and *post-states* of an entity as CuT.
- **Hence, every action event essentially creates a anaphoric link.**
  - Pre-state → event → Post-state
  - Input → event → Output
- We use a *phantom*, “generalized result” node, making sure events always have I/O, regardless of textual extent and syntactic structure.
  - ENTITY → VERBing → RES.VERB
- We model linguistically present results as re-entrance of the generalized results (as a CuI)

Let’s go back to the casserole example

Cut carrots and zucchini into cubes.

Slice onions

POEM

Let’s go back to the casserole example

Cut carrots and zucchini into cubes.

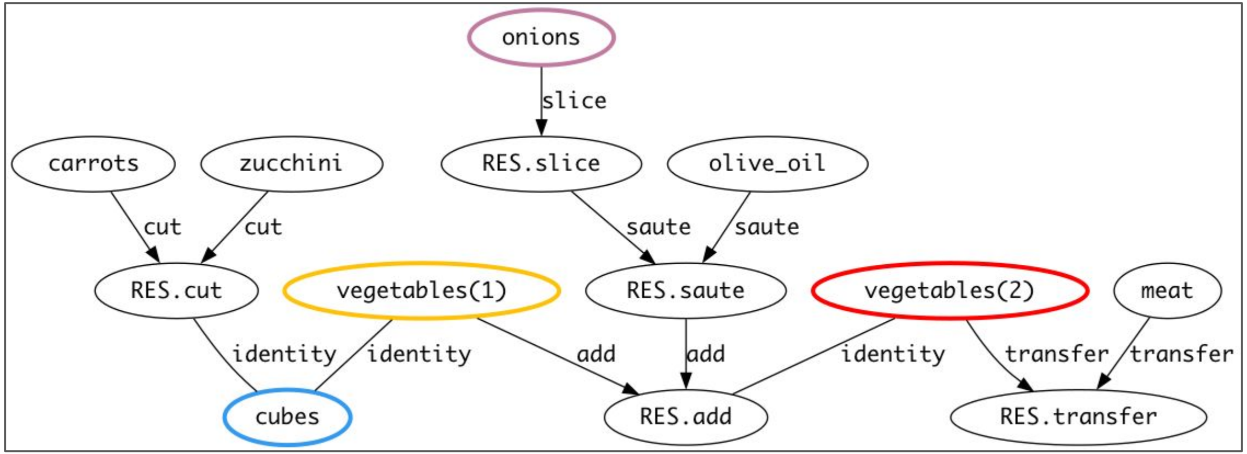
Slice onions (into RES.slice)

Let's go back to the casserole example

Cut carrots and zucchini into cubes.

Slice onions and saute in olive oil, add chopped vegetables and cook for 10 mins.

… Transfer the meat into the pan with the vegetables, season to taste.



More event semantics

- Temporal relations
- Subevent relations
- Event anaphora
- Light verb construction (LVC, multi-word verbs)
  - Ex: Bring to a boil
- Implicit Events (Nominal events)
  - Ex: That book bored me terribly.

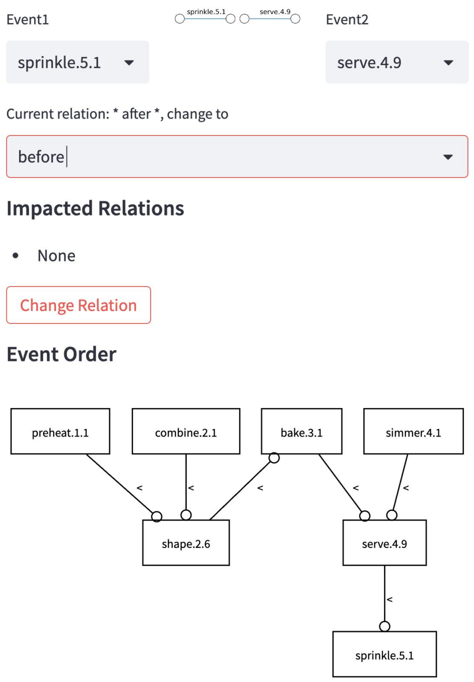
CuT Labeler (CUTLER) annotation environment

- Pairwise annotation for temporal ordering
- Event-by-event annotation for argument linking (CuT annotation)
- Table-based CuI (traditional coreference) annotation
- Real-time graph generation

Event-event relations in CUTLER (temporal ordering)

- 4 temporal relations and light-verb construction (LVC)
- LVC
  - [bring] to a [boil] - boil is the head
- Temporal ordering
  - independent: no direct temporal relation between the event pair
  - before/after: one event must be finished before the other starts
  - beginning/begun_by: the beginning of one event is conditioned on the end of the other (e.g., do X immediately after Y)
  - ending/ended_by: the end of one event is conditioned on the end of the other (e.g., do X until Y)
- Temporal relation names are selected from TimeML's (Pustejovsky et al., LREC 2010) TLINK, but the logic is largely based on Allen, ACM 26(11), 1983.

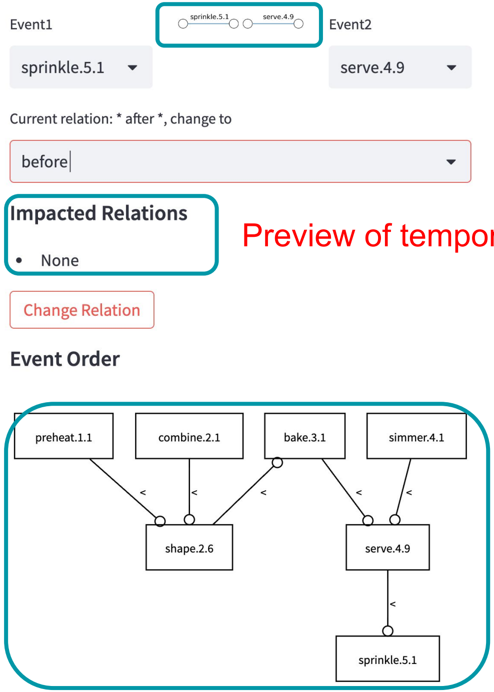
Event-event relations in CUTLER (temporal ordering)



Event-event relations in CUTLER (temporal ordering)

Pairwise interval visualization

Relation Picker



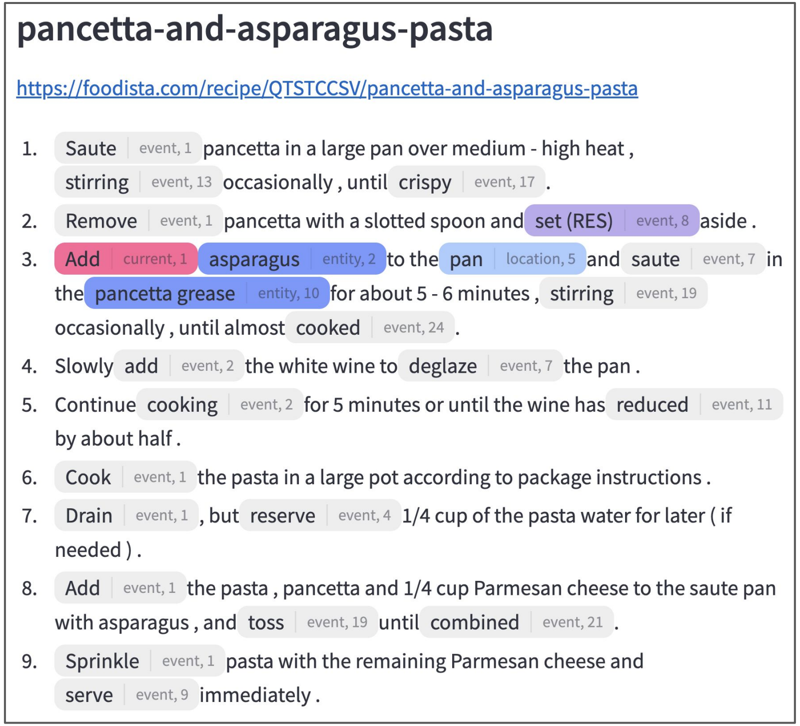
Preview of temporal closure reasoning

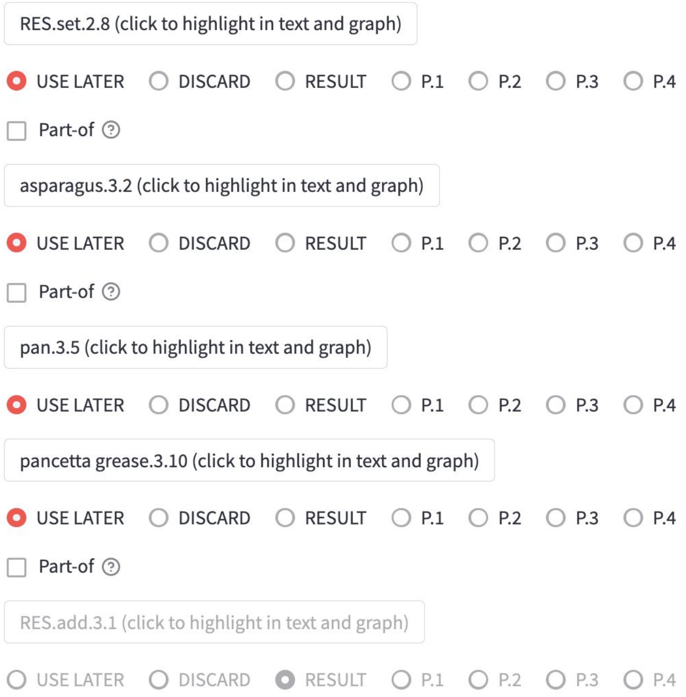
Doc-level topological sort

Event-entity relations in CUTLER (argument structure)

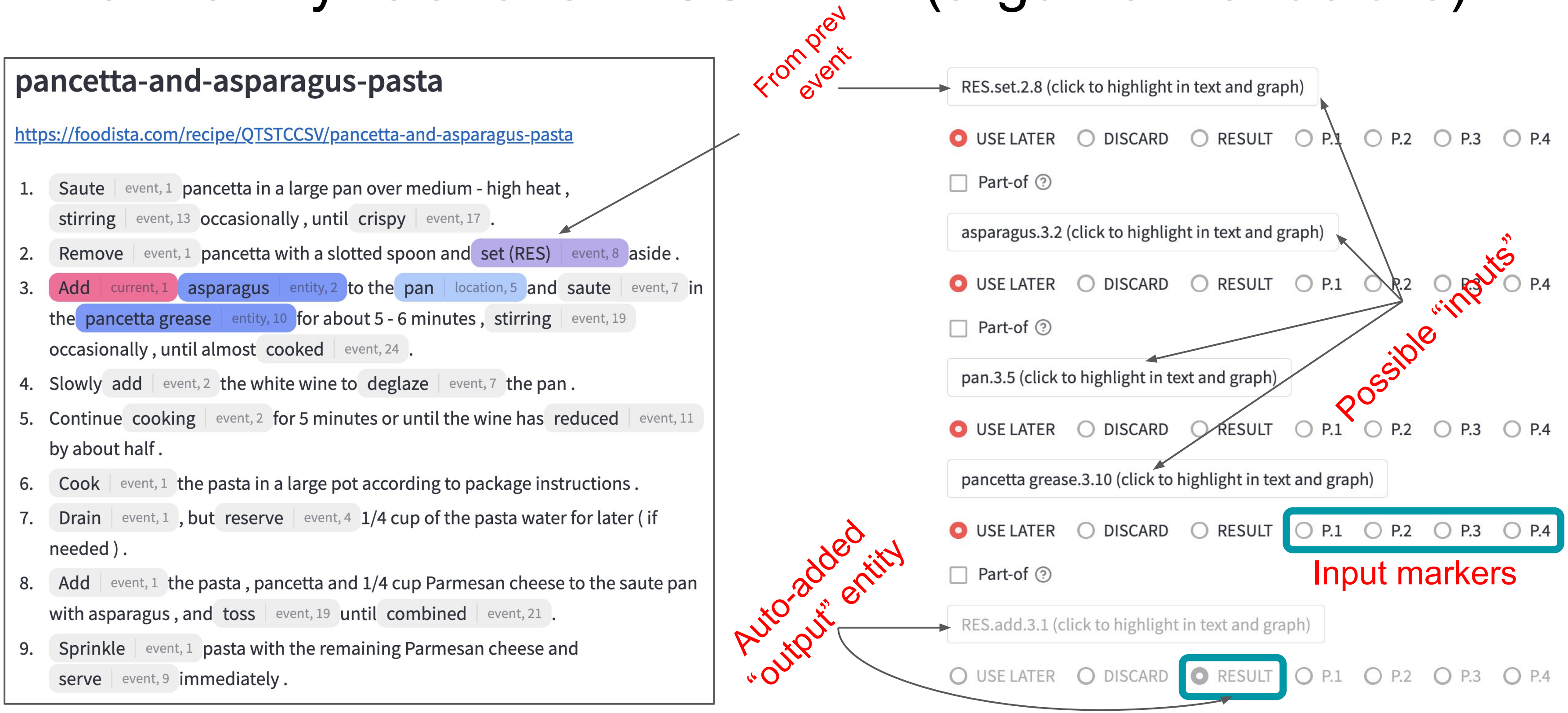
- Annotators handle one event at a time - no global document-level cognitive load
- At each event, annotators are presented with a subset of entities that are
    a. not consumed in previous events,
    b. new entities found around the current event span
- Thematic roles are simplified to just inputs and outputs.
- A link between an input and an output of an event constitute a CuT relation

Event-entity relations in CUTLER (argument structure)





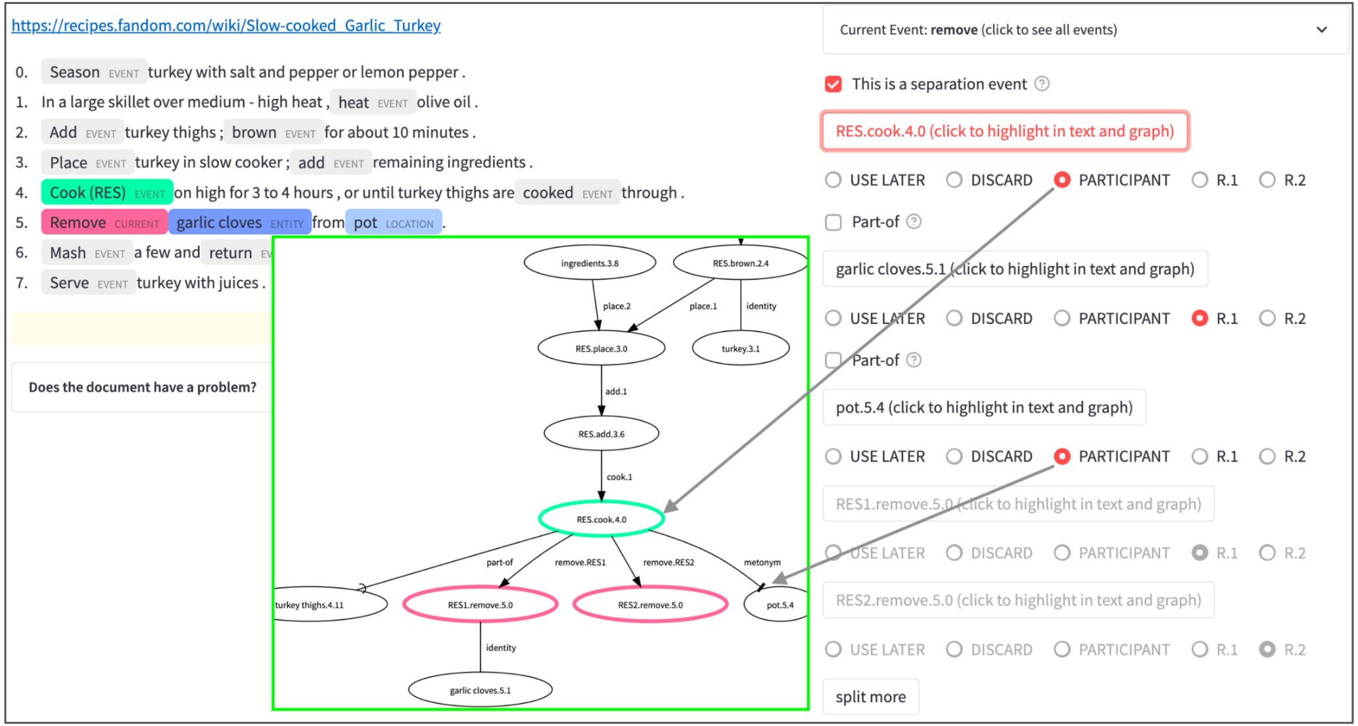
Event-entity relations in CUTLER (argument structure)



Event-entity relations in CUTLER (coreference links)

- When linking an entity to an event, annotators can give an integer group to the relation.
- All entities in the same integer group are considered as a kind of CuI.
- Subtypes of CuI then can be additionally annotated.

Event-entity relations in CUTLER (coreference links)



CUTLER Input requirement

- CUTLER do not support span annotation of events and entities.
  - In our annotation work, we use a separately trained NER model with human correction to get relevant mention spans and labels.
- Input documents must be pre-tokenized with sentences and words.

CUTLER output format

- CUTLER stores annotation in
  - CSV - each line is a triple of relation (span1,relation, span2)
  - png - graphviz-based graph generated from the CSV