Episode Book

EPISODE #5936

Training a Model When You Don't Know the Labels Yet

PUBLISHED	13 October 2026 at 17:17 UTC
DURATION	26:33
LISTEN	myweirdprompts.com/episode/unbounded-label-annotation-schema-62956/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.4
SOURCES	22 cited by the research (listed at the end)
BOOK GENERATED	19 October 2026 at 17:41 UTC

SYNOPSIS

What happens when you need to train an extraction model but the label set doesn't exist yet? This episode works through a listener's parametric-search project — scraping vendor spec sheets to build a filterable home inventory of adapters, cables, and power supplies — where the schema grows during annotation itself. We dig into attribute value extraction, why standard NER assumes a closed world, and how schema-conditioned models like GLiNER let you pass labels in as text at inference time with no retraining. Along the way: a 300M-parameter encoder that beats a 13B model on out-of-domain benchmarks, why annotation tooling never caught up to the research, and the anchoring risk hiding inside model-assisted labeling.

TRANSCRIPT



Corn

I want to start with the adapter story, because it's the whole prompt in miniature.



Herman

The voltage, the amperage, the plug type.



Corn

He listed five parameters he could imagine wanting. Plug type, DC voltage, current, polarity, barrel type. Then he said the only three he'd actually need are voltage, amperage, plug type. He wrote that in the same paragraph he wrote the five.



Herman

And that's not a contradiction. That's the discovery. He found out what he needed by looking at the pile.



Corn

Right. Daniel sent us another worked example. Same series we've been running, the how-would-you-actually-train-this episodes, and this one is the parametric search idea again but pointed at his own shelf. Scrape vendor pages, scrape spec sheets, pull the structured parameters out, feed them into the home inventory system, and then be able to filter the way Grainger lets you filter.



Herman

Except there's a twist in the middle of it. He can't tell you the label set ahead of time.



Corn

He says it plainly. He moves through items, searches his existing labels, and if nothing fits, he makes a new one. And a new one isn't just a word. It's a word plus a data primitive. Is this Boolean, is this an enum, is this a string. So the schema grows while the



Corn (continued)

annotation is happening.



Herman

Which is what makes it interesting. Everything else we've done in this series had the labels fixed before training started. This one doesn't.



Corn

He's asking five things. What task is this, how do you train when the label set is unbounded, what annotation software supports a schema that keeps changing, what model and training loop, and how do you guarantee the output parses with zero ambiguity.



Herman

He's also asking the version of this he already knows is unusual. He used the word unbounded himself.



Corn

Let's start with the task, because his instinct is half right, and the half that's wrong is the interesting half.



Herman

The house name for this is attribute value extraction. AVE if you're reading the papers. The MAVE dataset paper defines it as identifying values of an attribute of interest from product information, which is a mouthful but it's exactly what he's doing. You have a product profile, you have an attribute, you want the value.



Corn

So not named entity recognition.



Herman

It's the same mechanics, mostly. You're finding spans in text and labeling them. The Wayfair paper from the ECNLP workshop trained a transformer-based NER model for exactly this, pulling attribute values out of e-commerce search queries. AdaTag from ACL 2021 does multi-attribute extraction as sequence labeling with adaptive decoding. The toolkit is span extraction.



Corn

But he said entity extraction and you're saying that's half right.



Herman

The half that's wrong is the assumption underneath it. Standard NER runs on a closed world. You know your entity types before you start. Person, organization, location, whatever your set is. It's fixed at training time, baked into the output layer. Adding a type means re-annotating and retraining everything.



Corn

Which is precisely the thing he can't do, because the type he needs is the one he hasn't thought of yet.



Herman

There's a paper that asked his question in 2018. OpenTag, open attribute value extraction from product profiles. The abstract explicitly rejects the closed-world assumption, the possible set of values known beforehand, and then asks, how can we discover new attribute values that we have never seen before? Can we do this with limited human annotation or supervision?



Corn

Eight years ago.



Herman

So the problem is named. There's a literature. He isn't the first person to hit this wall, he's just the first person to hit it about his own adapters.



Corn

There's a version of this where that's deflating. The idea turns out to be a research area.



Herman

I find it the opposite. It means somebody has already built the thing he needs.



Corn

Go on.



Herman

There's an entire model family designed around his constraint. GLiNER. The paper is Zaratiana and others, 2023. The stated motivation is almost word for word his complaint. Traditional NER models are effective but limited to a set of predefined entity types, and extending them requires reannotation and retraining. That's the sentence.



Corn

So how does it avoid that.



Herman

It's schema-conditioned. The entity types don't live in the output layer. You pass them in as natural-language text at inference time. The model is doing semantic matching between spans of the input and descriptions of the labels. So if Daniel wants DC barrel type, he passes the words DC barrel type, and the model checks every span against that description.



Corn

Adding a label is a string.



Herman

No retraining, in the zero-shot case. And these are small models. GLiNER small, medium, large run about fifty million, ninety million, and three hundred million parameters. The large one trained in roughly five hours on one A100.



Corn

That's a laptop-adjacent project, not a cluster project.



Herman

Here's the number that sold me on it. On a seven-dataset out-of-domain benchmark, zero-shot, GLiNER large averaged sixty point nine F1. ChatGPT scored forty seven point five on the same benchmark. UniNER, a thirteen billion parameter model, scored fifty five point six. GoLLIE at seven billion scored fifty eight.



Corn

A three hundred million parameter encoder beating a thirteen billion parameter model on the same task.



Herman

Zero-shot. On this task. It's not a general claim about small models beating large ones. It's that this task rewards the right architecture more than it rewards scale.



Corn

I want to sit on that for a second, because it's the kind of number that sounds like a typo. Thirteen billion parameters and it loses to something a fortieth the size.



Herman

It loses because the big model is being asked to do it the hard way. UniNER is a generative model. It has to produce the spans as text, token by token, and every token is a chance to drift. GLiNER is an encoder doing span classification against a description. The description is the label. The matching is the task. There's no generation step to go wrong.



Corn

So the architecture is the answer, not the parameter count.



Herman

The architecture is the answer. And that's a pattern worth naming, because it shows up again in the output guarantee later. When a task has structure, encoding the structure beats scaling the model.



Corn

And the newer one.



Herman

GLiNER2. EMNLP demo track last year. It folds NER, text classification, structured extraction, and relation extraction into one model, two hundred five million parameters, and it runs on CPU. Zero-shot classification latency, one hundred thirty milliseconds at five labels, two hundred eight at fifty. The DeBERTa baseline it's compared against goes from about seventeen hundred milliseconds to nearly seventeen thousand at the same label counts.



Corn

Fifty labels in two hundred milliseconds on a CPU.



Herman

For a home inventory, that's not a constraint at all. He could re-run the whole catalog over a weekend and not notice.



Corn

So the workflow he described is upside down.



Herman

That's the part I want him to hear. He said, and I'm quoting his framing, he takes a few hundred items, decides what parameters are helpful, manually identifies them, and trains a model. Annotate, then extract. GLiNER lets you extract first. Pass the label description, get the spans, and now your few hundred items aren't the cold start. They're a precision pass. You're correcting a model that already works, instead of building one from nothing.



Corn

He'd also find out faster whether a parameter is worth having. If DC barrel type comes back garbage on fifty items, that's an answer. He didn't have to annotate anything to get it.



Herman

Which is what he actually did in the prompt without calling it that. He wrote out five adapter parameters, then narrowed to three, and the narrowing is the point. You discover the useful set by looking, not by designing.



Corn

Let me push on that, because there's a trap in it. If the model pre-annotates everything, doesn't he just end up rubber-stamping whatever the model suggests?



Herman

That's the real risk, and it has a name in the annotation literature. It's called anchoring. You see the model's guess first and your judgment bends toward it. The mitigation is to sample. You don't verify every span. You verify a random slice, plus every span the model



Herman (continued)

flagged as low confidence, plus you periodically run a batch with the model's output hidden.



Corn

So the correction pass has its own protocol.



Herman

The correction pass has its own protocol. It's not free, it's just cheaper than annotating from scratch. And the low-confidence bucket is where the new labels actually come from. When the model keeps flagging a span and keeps being wrong, that's usually a label you haven't named yet.



Corn

That's a nice inversion. The model's errors are the schema discovery mechanism.



Herman

That's exactly what it is. You're mining the disagreement.



Corn

Then the next thing he asked. What task is this, we've got that. What software does he label in.



Herman

This is where it gets messier.



Corn

Say it.



Herman

There is no annotation tool I could find that treats define a brand-new label with a new data primitive in the middle of a session as a first-class feature. It doesn't exist as a product. The literature treats unbounded labels as a modeling problem, which is why OpenTag and GLiNER exist, and the tooling never caught up.



Corn

So he builds it.



Herman

The closest fit is Prodigy, from Explosion, the spaCy people. It's a downloadable Python tool, runs entirely locally, and the whole design is custom recipes. A recipe is a Python function that defines the data feed and the interface. So an add-new-label affordance isn't a feature you toggle on, it's something you write. The docs describe the philosophy as defining your scheme with real-world examples rather than prompts.



Corn

There's a testimonial on their site that is Daniel's problem.



Herman

Cheyanne Baird, an NLP research scientist. She describes feedback from annotators saying it's really hard, because I have to scroll and scroll and scroll to see the labels. There's too many labels. There's too many options.



Corn

That's the failure mode of a growing schema.



Herman

Right. A schema that grows during annotation is a schema where the annotator has to find the label, and after a few hundred items the label list is long enough that finding it costs more than applying it. The answer in a custom recipe is you build the search, you build the create-new flow, you control the interface.



Corn

What does that flow actually look like, concretely? If I'm the annotator and I hit a span that has no label.



Herman

You hit a hotkey, a box opens, you type the new label name, then a second prompt asks you what type it is. Boolean, enum, string, number. You pick. Now the label is in the palette and the span is tagged. That's maybe four seconds of extra friction, once, and then the label is there forever.



Corn

And the type prompt is the part that matters, because that's what feeds the output guarantee later.



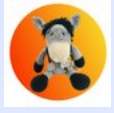
Herman

That's the bit that matters. The annotator is declaring the schema, not just naming a thing.



Corn

Options worth naming.



Herman

Label Studio is open source and general purpose. Argilla is the other one people reach for. doccano does sequence labeling, it supports overlapping entities, but its label set is fixed per project, so adding a parameter means adding it between passes, not during.



Corn

That's the gap stated plainly.



Herman

Entity Enricher is interesting for a different reason. Its schema editor has value sets, which restrict a text property to a fixed list of allowed values, so enrichment cannot return anything outside that list. That's enum enforcement living in the schema rather than in the model. If Daniel declares plug type as an enum of European, US two-prong, US three-prong, the extractor physically cannot return a fourth thing.



Corn

Which is half his output guarantee, and he hasn't trained anything yet.



Herman

There's also MEGAnno+, an EACL demo from last year, a human and LLM collaborative annotation system. You let the model pre-annotate and the human verifies. Which is the loop GLiNER's zero-shot ability enables. Extract first, annotate second, and the annotation is correction rather than creation.



Corn

So tooling answer is one sentence. Buy Prodigy, write the recipe, or manage the schema outside the tool and re-export between passes.



Herman

And that's fine for him. He writes Python. It's not fine for the general case, which is worth saying.



Corn

Then the training loop, which is where his plan actually breaks.



Herman

Let's do the piece that works first. GLiNER2's training tutorial documents exactly his flow. You prepare input examples with text and structures, validate them, split eighty twenty, train with the trainer, then load the best checkpoint and continue. The docs say checkpoints contain model weights only, and that loading a checkpoint and calling train again is the supported way to continue. So retrain from the previous checkpoint isn't something he builds. It's a documented call.



Corn

And for the periodic updates.



Herman

LoRA. Turn it on, rank sixteen, alpha thirty two, save adapter only. The adapters come out around two to ten megabytes against roughly four hundred fifty for a full model, training is two to three times faster, and swapping an adapter takes milliseconds. If he adds three parameters this month, he trains a small adapter on new examples and swaps it in.



Corn

Two to ten megabytes.



Herman

The docs suggest prototyping with a hundred training samples. Typical learning rates for a full fine-tune are one e-five on the encoder and five e-four on the task head. He's not going to need a hyperparameter search to get a first result.



Corn

Then the break.



Herman

Continual learning for NER. There's a 2025 paper, GenCNER, whose framing is that entity categories are continuously increasing in various real-world scenarios. That's a polite way of describing Daniel's plan. The core problem has a name, catastrophic forgetting. The model gets good at the new parameters and quietly loses the old ones.



Corn

So voltage still works, barrel type works, and polarity has decayed by forty percent because he hasn't shown it an example in three months.



Herman

And there's a second problem in the same paper, the semantic shift of the non-entity type. What counts as not-a-parameter changes as your schema grows. A span that was correctly ignored last month belongs to a label you invented yesterday.



Corn

That's uglier than forgetting, honestly. Forgetting you can measure.



Herman

The standard mitigations are distillation from the old model, pseudo-labeling, replay of stored examples, and pooled-feature distillation, which reports about six points of micro F1 and eight of macro over the previous state of the art. There's a 2026 paper using fast and slow experts. It's an active area and none of it is plug and play.



Corn

So the label problem is two problems wearing one coat.



Herman

Item one, the model has to accept a new label without retraining. That's solved. GLiNER does it by design.



Corn

Item two, the model has to not forget the old labels when you add new ones.



Herman

That one is open. And Daniel's plan, update from the previous checkpoint as I find new things, solves item one for free and walks straight into item two. Every update is a chance to lose something. He won't notice, because he'll be testing the new parameter.



Corn

How bad is it in practice at his scale.



Herman

Honest answer, I don't know. Nobody has run a two-year longitudinal study on a home inventory of three thousand items and its extractor. There's a few-shot curve that's suggestive. GLiNER fine-tuned for biomedical extraction, ten labeled examples gets seventy point four F1, twenty gets seventy three, fifty gets seventy six. So a few hundred examples is more than enough to move the needle in a new domain.



Corn

Which is good news for the first training run and says nothing about the tenth.



Herman

Correct. The forgetting risk is real and the mitigations are research-grade. If I were him, I'd keep every annotation in version control, tag each example with the schema version it was made under, and retrain from the full set rather than only the new items. Cold start every time, no forgetting, and the only cost is training minutes.



Corn

Which the LoRA numbers make cheap.



Herman

Two to three times faster on a two hundred million parameter model. That's a coffee.



Corn

Let me make sure I understand the tradeoff, because it feels like it should be worse than you're making it sound. If he retrains on the full corpus every time, doesn't the corpus just keep growing forever?



Herman

It grows, but slowly. He's adding a few hundred examples per schema revision, not per day. After two years he might be at three or four thousand examples. On a two hundred million parameter model with LoRA, that's still minutes, not hours. The scaling that would break this is if he were retraining a large generative model from scratch each time, and he isn't.



Corn

So the boring answer survives contact with the arithmetic.



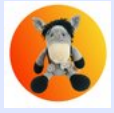
Herman

The boring answer survives contact with the arithmetic. That's a good way to put it.



Corn

Last piece. The output guarantee. He said no margin for error or ambiguity.



Herman

This is the cleanest of the five, because there's a whole technology for it. Constrained decoding. At every generation step you zero out the probability of any token that would break the schema. The model literally cannot pick those tokens. It's not a training-time property, it's a sampler-level one.



Corn

So you don't have to teach the model to be well-formed.



Herman

You make malformed output structurally impossible. OpenAI shipped it as Structured Outputs in August 2024, constrained to a JSON Schema. The enforcement library underneath does arbitrary context-free grammar at about fifty microseconds of CPU per token. vLLM supports it for JSON Schema, regex, grammar, and choice constraints. It's standard across serving stacks now.



Corn

And for the non-generative path.



Herman

Same guarantee, different mechanism. GLiNER2's structured extraction takes a schema where each field is declared with a type and a description. Name as string, features as list, tier as a choice between three values. The choice becomes an enum. There are regex validators too, so you can say only accept a span matching a voltage pattern. If it doesn't match, it isn't returned.



Corn

So plug type as an enum of three, voltage as a regex-shaped string, polarity as Boolean. The inventory system can't receive garbage.



Herman

It can receive a wrong value. It can't receive a malformed one. Those are different guarantees and only the second one is free.



Corn

That distinction is the whole answer to his last question, I think. He wanted no ambiguity. He gets no ambiguity in the shape.



Herman

And that's the right thing to want, because the shape is what breaks the pipeline. A wrong voltage is a data problem he can fix in a review pass. A voltage field containing the sentence, approximately twelve volts, is a parser problem.



Corn

Daniel's real question is whether you can start with a few hundred items and a schema that grows.



Herman

And the shape of the answer is yes, but not in the order he described. Extract before you annotate. Annotate for precision rather than for bootstrap. Update with LoRAs against the full corpus, not just the new items, so you never have to think about forgetting. Enforce the output shape at the schema level rather than in training. Four changes, none of them enormous.



Corn

There's a version of this where the hardest part was never the model.



Herman

The tooling gap.



Corn

The model side has an off-the-shelf answer. The software side makes him write a custom recipe to add a label mid-session.



Hilbert

It's not two to ten megabytes. It's six.



Hilbert

The adapter. You said two to ten. For rank sixteen on a model that size, saving the adapter only, it's about six megabytes. You're rounding low.



Herman

That is more precise than I was.



Hilbert

Six is the number. And the reason I know is I've been cataloging belts.



Corn

Belts.



Hilbert

Every belt I own. It's a spreadsheet. Forty seven columns. Width in millimeters, width in inches, leather type, leather sub-type, buckle material, buckle finish, keeper count, whether the keeper is stitched or looped. Forty seven.



Herman

Forty seven columns for belts.



Hilbert

And I have never once needed to filter by more than three of them. Width, color, and whether it fits. That's the whole thing. The other forty four columns are decoration. So when your friend says he doesn't know his parameters ahead of time, I believe him, and I also think he'll end up with forty four columns he never queries.



Corn

How many belts do you own, Hilbert?



Hilbert

The point is the card.



Corn

The card.



Hilbert

I worked in a warehouse that did exactly this. Tagging product for a catalog. Regional distribution hub, company doesn't exist anymore, I'm not going to get into it. The problem your friend has, where the labels grow while you're working, we had that. We solved it.



Herman

How.



Hilbert

Laminated card. Index card, laminated. When a new attribute came up, you wrote it on the card in grease pencil. The card was the schema. Whoever had the card had the schema.



Corn

One card.



Hilbert

Front and back, eventually. And when the card filled up, you started a second card. The second card was the schema for the new stuff and you kept the first card in the drawer, so you had to know which card an item was tagged under. That was a known limitation. We lived with it.



Herman

Did you consider just, a database?



Hilbert

The card never went down. The card never needed a migration. Someone hands you the card, you know everything. That's the whole system.



Corn

And this is better than GLiNER.



Hilbert

GLiNER is a lot of machinery to do what a card does. You write the label on the card. The label is now in the schema. That's your zero-shot. That's what you described, just with more electricity.



Herman

It is the same idea. The card is a schema-conditioned vocabulary.



Hilbert

Except the card doesn't forget. You add leather sub-type to the card, width is still on the card. Nothing decays. No adapter, no checkpoint, no distillation.



Corn

He's got you there.



Herman

He does have me there, on that specific point.



Hilbert

I still have the card. It's in a box. There's a coffee ring on it from the year it happened, and the coffee ring is load bearing.



Corn

The coffee ring is load bearing.



Hilbert

There's a column under the ring. If you took the card and cleaned it, the column's gone, and the column is the schema. So the ring stays. I've had two people offer to laminate it again. I said no.



Herman

So the 1994 coffee ring is the only thing preserving your attribute definitions.



Hilbert

It's preserving one of them.



Corn

Which one.



Hilbert

Finish.



Herman

That's a terrible place to store a schema.



Hilbert

It's a terrible place to store a schema and it worked for eleven years.



Corn

Did the card ever disagree with itself.



Hilbert

Twice. Once because two people wrote the same attribute with different words, once because someone wrote in pen on top of the grease pencil and then we couldn't change it. That's the pitfall. Pen.



Herman

That's semantic drift and immutability. You reproduced two of the four problems in continual learning with office supplies.



Hilbert

We didn't reproduce them. We discovered them. That's different.



Corn

I'll give you that.



Hilbert

The other thing, and then I need to check on something. Your friend should write down the day he adds each parameter. Not the parameter, the day. Because in three years the useful question isn't what voltage means, it's when he decided voltage mattered, and the card never had a date column. That's the one thing I'd add.



Herman

That's a schema version.



Hilbert

It's a date. Call it what you want.



Corn

So the modeling side moved faster than the tooling side. That's the thing I keep circling.



Herman

By a distance. You can add a label to GLiNER by typing its name. You cannot add a label to most annotation software without leaving the session, editing a config, and reloading.



Corn

And the second thing is that the hard part of his plan isn't the part he flagged. Accepting new labels is solved. Not forgetting old ones is not.



Herman

Which has a boring answer for his scale. Keep everything, retrain on all of it, treat each round as a cold start. It costs training minutes and it removes the entire research problem from his project.



Corn

I'd take that trade. And the four changes we listed still hold. Extract before you annotate. Annotate for precision. Update against the full corpus. Enforce the shape in the schema.



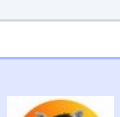
Herman

That's the version of this project I'd build, and it's a smaller project than the one he described, which is usually a good sign.



Corn

If this was your kind of episode, go back for episode fifty-two forty-five, Training a Model on one hundred Phone Photos; episode forty-eight twenty-nine, Why Can't You Filter by Bluetooth on Amazon; and episode seven eighty-six, The Cost of a Touch. This has been My Weird Prompts, the human-AI collaboration podcast. Thanks as ever to our producer, Hilbert Flumingtop, who has a laminated card and will not be discussing it further.



Herman

Send us your own prompt on Telegram at t dot me slash MWP listener bot. We'll be back soon.



Corn

See you tomorrow.

SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **GLiNER2 README [primary]**
   (fastino-ai/gliner2, Apache-2.0; 205M base model, extract_json, schema builder, LoRA)
   https://raw.githubusercontent.com/fastino-ai/gliner2/main/README.md

2. **GLiNER2 JSON extraction tutorial [primary]**
   (field types `::str`, `::list`, choice enums, regex validators)
   https://raw.githubusercontent.com/fastino-ai/gliner2/main/tutorial/3-json_extraction.md

3. **GLiNER2 training tutorial [primary]**
   (ExtractorTrainer, TrainingConfig, LoRA, load_checkpoint/continue training)
   https://raw.githubusercontent.com/fastino-ai/gliner2/main/tutorial/9-training.md

4. **GLiNER2 repo snapshot [primary]**
   (commit b86d35c; span + boundary architectures, training/, inference/ modules)
   https://github.com/fastino-ai/gliner2

5. **Prodigy [primary]**
   (Explosion; custom recipes, LLM integration, local/private)
   https://prodi.gy/

6. **GLiNER: Generalist Model for NER using Bidirectional Transformer**
   (Zaratiana et al., 2023)
   https://arxiv.org/abs/2311.08526

7. **GLiNER2: Efficient Multi-Task IE System with Schema-Driven Interface**
   (arXiv 2507.18546; EMNLP 2025 demo)
   https://ar5iv.labs.arxiv.org/html/2507.18546

8. **GLiNER schema-conditioned extraction overview**
   (updated 2026-07-14; family variants, benchmarks)
   https://www.emergentmind.com/topics/gliner

9. **OpenTag: Open Attribute Value Extraction from Product Profiles**
   (arXiv 1806.01264)
   https://ar5iv.labs.arxiv.org/html/1806.01264

10. **MAVE: A Product Dataset for Multi-source Attribute Value Extraction**
    (arXiv 2112.08663)
    https://ar5iv.labs.arxiv.org/html/2112.08663

11. **Explicit Attribute Extraction in E-Commerce Search (Wayfair)**
    (ECNLP 2024)
    https://aclanthology.org/2024.ecnlp-1.13v1.pdf

12. **Generative Models for Product Attribute Extraction**
(EMNLP 2023 Industry)
https://aclanthology.org/2023.emnlp-industry.55.pdf

13. **AdaTag: Multi-Attribute Value Extraction with Adaptive Decoding**
(ACL 2021)
https://p.rst.im/q/aclanthology.org/2021.acl-long.362.pdf

14. **Continual NER without Catastrophic Forgetting**
(Zhang et al., EMNLP 2023)
https://arxiv.org/abs/2310.14541

15. **GenCNER: Generative Framework for Continual NER**
(IJCNN 2025)
https://arxiv.org/abs/2510.11444

16. **FSE: Continual Learning for NER by Fast-Slow Experts**
(2026-07-24)
https://arxiv.org/abs/2607.22075v1

17. **Generating Structured Outputs from Language Models: Benchmark and Studies**
(2025)
https://arxiv.org/html/2501.10868v1

18. **LLGuidance: Making Structured Outputs Go Brrr**
(constrained decoding, ~50µs/token)
https://guidance-ai.github.io/llguidance/llg-go-brrr

19. **Together AI structured outputs docs**
(sampler-level token masking)
https://docs.together.ai/learn/structured-outputs

20. **MEGAnno+: A Human-LLM Collaborative Annotation System**
(EACL 2024 Demo)
https://arxiv.org/abs/2402.18050

21. **doccano information extraction docs**
https://paddlenlp.readthedocs.io/en/latest/llm/application/information_extraction/doccano.html

22. **Entity Enricher Schema Editor docs**
(value sets, AI-assisted schema edits)
https://entityenricher.ai/docs/schemas/editor