Matrix-Level Documentation of `gline-rs` Processing Steps

Frédérik Bilhaut

This documents aims at providing a matrix-level description of the pipeline needed for GLiNER inferences, as implemented by `gline-rs`.

Concrete examples are provided for each step, all of which build on the input given in the first one.

Contents

1. Pre-Processing (Common)

1.1. Text Input

This is the “user” input for the whole processing.

1.1.1. Source Code

- Related struct: `gliner::model::input::text::TextInput`

1.1.2. Format

- n: number of input texts
- k: number of entity class labels
- I: sequence of input texts matrix of type `string` and size n
- E: entity class labels matrix, of type `string` and size k

$$I = \begin{bmatrix} \text{text}_1 \\ \text{text}_2 \\ \vdots \\ \text{text}_n \end{bmatrix}$$

$$E = \begin{bmatrix} \text{label}_1 \\ \text{label}_2 \\ \vdots \\ \text{label}_k \end{bmatrix}$$

1.1.3. Example

$$I = \begin{bmatrix} \text{"My name is James Bond"} \\ \text{"I like to drive my Aston Martin"} \end{bmatrix}$$

$$E = \begin{bmatrix} \text{"movie character"} \\ \text{"vehicle"} \end{bmatrix}$$

1.2. Word-Level Tokenization

1.2.1. Transformation

$(I, E) \to (T, E)$

1.2.2. Source Code

- Struct: `gliner::model::input::tokenized::TokenizedInput`
- Transformation: `gliner::model::input::prompt::RawToTokenized`

1.2.3. Format

- n, k: same as before
- T: sequence of sequence of tokenized input texts, of type `string` and size n
- E: same as before

$$T = \begin{bmatrix} \begin{bmatrix} \text{token}_{1,1} & \text{token}_{1,2} & \text{...} \end{bmatrix} \\ \begin{bmatrix} \text{token}_{2,1} & \text{token}_{2,2} & \text{...} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \text{token}_{n,1} & \text{token}_{n,2} & \text{...} \end{bmatrix} \end{bmatrix}$$

1.2.4. Example

$$T = \begin{bmatrix} \begin{bmatrix} \text{"My"} & \text{"name"} & \text{"is"} & \text{"James"} & \text{"Bond"} \end{bmatrix} \\ \begin{bmatrix} \text{"I"} & \text{"like"} & \text{"to"} & \text{"drive"} & \text{"my"} & \text{"Aston"} & \text{"Martin"} \end{bmatrix} \end{bmatrix}$$

1.3. Prompt Preparation

Prepared prompts, appending entity and text tokens.

1.3.1. Transformation

$(T, E) \to P$

1.3.2. Source Code

- Struct: `gliner::model::input::prompt::PromptInput`
- Transformation from `TokenizedInput`: `gliner::model::input::prompt::TokenizedToPrompt`

1.3.3. Format

$$P = \begin{bmatrix} [\text{<<ENT>>} \ \ \text{label}_{1,1} \ \ \text{<<ENT>>} \ \ \text{label}_{1,2} \ \ \ldots \ \ \text{<<SEP>>} \ \ \text{token}_{1,1} \ \ \text{token}_{1,2} \ \ \ldots] \\ [\text{<<ENT>>} \ \ \text{label}_{2,1} \ \ \text{<<ENT>>} \ \ \text{label}_{2,2} \ \ \ldots \ \ \text{<<SEP>>} \ \ \text{token}_{2,1} \ \ \text{token}_{2,2} \ \ \ldots] \\ \vdots \\ [\text{<<ENT>>} \ \ \text{label}_{n,1} \ \ \text{<<ENT>>} \ \ \text{label}_{n,2} \ \ \ldots \ \ \text{<<SEP>>} \ \ \text{token}_{n,1} \ \ \text{token}_{n,2} \ \ \ldots] \end{bmatrix}$$

1.3.4. Example

$$P = \begin{bmatrix} [\text{<<ENT>>} \ \ \text{"movie character"} \ \ \text{<<ENT>>} \ \ \text{"vehicle"} \ \ \ldots \ \ \text{<<SEP>>} \ \ \text{"My"} \ \ \text{"name"} \ \ \text{"is"} \ \ \text{"James"} \ \ \text{"Bond"}] \\ [\text{<<ENT>>} \ \ \text{"movie character"} \ \ \text{<<ENT>>} \ \ \text{"vehicle"} \ \ \ldots \ \ \text{<<SEP>>} \ \ \text{"I"} \ \ \text{"like"} \ \ \text{"to"} \ \ \text{"drive"} \ \ \text{"my"} \ \ \text{"Austin"} \ \ \text{"Martin"}] \end{bmatrix}$$

1.4. Prompt Encoding (Sub-Word Tokenization)

1.4.1. Transformation

$P \to (I, A, W, L)$

1.4.2. Source Code

- Struct: `gliner::model::input::encoded::EncodedPrompt`
- Transformation: `gliner::model::input::encoded::PromptsToEncoded`

1.4.3. Format

- k: maximum number of sub-word tokens within a sequence, adding start (1) and end (2) tokens
- I: encoded prompts of type `i64` and shape $(n * k)$
- A: attention masks of type `i64` and shape $(n * k)$
- W: word masks of type `i64` and shape $(n * k)$
- L: text lengths of type `i64` and shape $(n * 1)$

$$I = \begin{pmatrix} \text{token\_id}_{1,1} & \text{token\_id}_{1,2} & \ldots & \text{token\_id}_{1,k} \\ \text{token\_id}_{2,1} & \text{token\_id}_{2,2} & \ldots & \text{token\_id}_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ \text{token\_id}_{n,1} & \text{token\_id}_{n,2} & \ldots & \text{token\_id}_{n,k} \end{pmatrix}$$

$$A = \begin{pmatrix} \text{attn\_mask}_{1,1} & \text{attn\_mask}_{1,2} & \ldots & \text{attn\_mask}_{1,k} \\ \text{attn\_mask}_{2,1} & \text{attn\_mask}_{2,2} & \ldots & \text{attn\_mask}_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ \text{attn\_mask}_{n,1} & \text{attn\_mask}_{n,2} & \ldots & \text{attn\_mask}_{n,k} \end{pmatrix}$$

$$W = \begin{pmatrix} \text{word\_mask}_{1,1} & \text{word\_mask}_{1,2} & \ldots & \text{word\_mask}_{1,k} \\ \text{word\_mask}_{2,1} & \text{word\_mask}_{2,2} & \ldots & \text{word\_mask}_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ \text{word\_mask}_{n,1} & \text{word\_mask}_{n,2} & \ldots & \text{word\_mask}_{n,k} \end{pmatrix}$$

$$L = \begin{pmatrix} l_1 \\ \vdots \\ l_n \end{pmatrix}$$

1.4.4. Example

$$I = \begin{pmatrix} 1 & 128002 & 1421 & 1470 & 128002 & 1508 & 128003 & 573 & 601 & 269 & 1749 & 8728 & 2 & 0 & 0 \\ 1 & 128002 & 1421 & 1470 & 128002 & 1508 & 128003 & 273 & 334 & 264 & 1168 & 312 & 20844 & 2963 & 2 \end{pmatrix}$$

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$W = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 & 4 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 0 \end{pmatrix}$$

$$L = \begin{pmatrix} 5 \\ 7 \end{pmatrix}$$

2. Pre-Processing (Span Mode)

Downstream of the aforementioned steps.

2.1. Span Preparation

2.1.1. Transformation

$(I, A, W, L) \rightarrow (I, A, W, L, S_I, S_M)$

2.1.2. Format

- n, k, I, A, W, L: same as before.
- s: maximum possible number of spans for one sequence
- S_I: span offsets, of type `i64` and shape $(n * s * 2)$
- S_M: span masks, of type `bool` and shape $(n * s)$

$$S_I = \begin{pmatrix} (\mathrm{start}_{1,1} \;\; \mathrm{end}_{1,1}) & (\mathrm{start}_{1,2} \;\; \mathrm{end}_{1,2}) & \dots & (\mathrm{start}_{1,s} \;\; \mathrm{end}_{1,s}) \\ (\mathrm{start}_{2,1} \;\; \mathrm{end}_{2,1}) & (\mathrm{start}_{2,2} \;\; \mathrm{end}_{2,2}) & \dots & (\mathrm{start}_{2,s} \;\; \mathrm{end}_{2,s}) \\ \vdots & \vdots & \ddots & \vdots \\ (\mathrm{start}_{n,1} \;\; \mathrm{end}_{n,1}) & (\mathrm{start}_{n,2} \;\; \mathrm{end}_{n,2}) & \dots & (\mathrm{start}_{n,s} \;\; \mathrm{end}_{n,s}) \end{pmatrix}$$

$$S_M = \begin{pmatrix} \mathrm{span\_mask}_{1,1} & \mathrm{span\_mask}_{1,2} & \dots & \mathrm{span\_mask}_{1,s} \\ \mathrm{span\_mask}_{2,1} & \mathrm{span\_mask}_{2,2} & \dots & \mathrm{span\_mask}_{2,s} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{span\_mask}_{n,1} & \mathrm{span\_mask}_{n,2} & \dots & \mathrm{span\_mask}_{n,s} \end{pmatrix}$$

2.1.3. Example

Note: for readability purposes, inside matrices are split into rows (one per token) but they are actually in one dimension s (see format above).

$$S_I = \begin{pmatrix} \begin{pmatrix} (0\;0) & (0\;1) & (0\;2) & (0\;3) & (0\;4) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & \hookleftarrow \\ (1\;1) & (1\;2) & (1\;3) & (1\;4) & (0\;0) & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \hookleftarrow \\ (2\;2) & (2\;3) & (2\;4) & (0\;0) & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \hookleftarrow \\ (3\;3) & (3\;4) & (0\;0) & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \hookleftarrow \\ (4\;4) & (0\;0) & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \hookleftarrow \\ (0\;0) & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \hookleftarrow \\ (0\;0) & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \end{pmatrix} \\ \begin{pmatrix} (0\;0) & (0\;1) & (0\;2) & (0\;3) & (0\;4) & (0\;5) & (0\;6) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & \hookleftarrow \\ (1\;1) & (1\;2) & (1\;3) & (1\;4) & (1\;5) & (1\;6) & (0\;0) & \dots & \dots & \dots & \dots & \dots & \hookleftarrow \\ (2\;2) & (2\;3) & (2\;4) & (2\;5) & (2\;6) & (0\;0) & \dots & \dots & \dots & \dots & \dots & \dots & \hookleftarrow \\ (3\;3) & (3\;4) & (3\;5) & (3\;6) & (0\;0) & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \hookleftarrow \\ (4\;4) & (4\;5) & (4\;6) & (0\;0) & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \hookleftarrow \\ (5\;5) & (5\;6) & (0\;0) & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \hookleftarrow \\ (6\;6) & (0\;0) & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \end{pmatrix} \end{pmatrix}$$

$$S_M = \begin{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \text{⏎} \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \text{⏎} \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \text{⏎} \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \text{⏎} \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \text{⏎} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \text{⏎} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \end{pmatrix} \\ \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & \text{⏎} \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \text{⏎} \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \text{⏎} \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \text{⏎} \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \text{⏎} \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \text{⏎} \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \end{pmatrix} \end{pmatrix}$$

3. Pre-Processing (Token Mode)

Nothing to be done beside the common steps.

4. Post-Processing (Span Mode)

4.1. Logits Output

4.1.1. Source Code

- Struct: `gliner::model::output::TensorOutput`

4.1.2. Format

- n: number of text sequences
- w: maximum number of tokens in one sequence
- s: maximum number of possible spans for one token (seee above)
- k: number of entity labels
- O: logits output, of type `f32` and shape $(n * w * s * k)$
- $v_{\mathrm{n,w,s,k}}$: raw model output for sequence n, token w, span s and label k.

$$O = \begin{pmatrix} \begin{pmatrix} \begin{pmatrix} (v_{1,1,1,1} & \cdots & v_{1,1,1,\mathrm{k}}) \\ & \vdots & \\ (v_{1,1,\mathrm{s},1} & \cdots & v_{1,1,\mathrm{s,k}}) \end{pmatrix} & \cdots & \begin{pmatrix} (v_{1,\mathrm{w},1,1} & \cdots & v_{1,\mathrm{w},1,\mathrm{k}}) \\ & \vdots & \\ (v_{1,\mathrm{w,s},1} & \cdots & v_{1,\mathrm{w,s,k}}) \end{pmatrix} \end{pmatrix} \\ \vdots \\ \begin{pmatrix} \begin{pmatrix} (v_{\mathrm{n},1,1,1} & \cdots & v_{\mathrm{n},1,1,\mathrm{k}}) \\ & \vdots & \\ (v_{\mathrm{n},1,\mathrm{s},1} & \cdots & v_{\mathrm{n},1,\mathrm{s,k}}) \end{pmatrix} & \cdots & \begin{pmatrix} (v_{\mathrm{n,w},1,1} & \cdots & v_{\mathrm{n,w},1,\mathrm{k}}) \\ & \vdots & \\ (v_{\mathrm{n,w,s},1} & \cdots & v_{\mathrm{n,w,s,k}}) \end{pmatrix} \end{pmatrix} \end{pmatrix}$$

4.1.3. Example

In this case $s = 12$. For readability purposes, the raw values are “sigmoided” ($S(x) = \frac{1}{1+e^{-x}}$) and then “ReLUed” with a threshold $t = 0.5$.

$$O_{\mathrm{S,t}} = \begin{pmatrix} \begin{pmatrix} (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) \\ (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) \\ (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) \\ (0\;0) & (\mathbf{0.89}\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) \\ (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) \\ (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) \\ (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) \end{pmatrix} \\ \begin{pmatrix} (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) \\ (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) \\ (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) \\ (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) \\ (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) \\ (0\;0) & (0\;\mathbf{0.96}) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) \\ (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) & (0\;0) \end{pmatrix} \end{pmatrix}$$

Which means:

- In the 1st sequence, the span starting with the 4th token and ending with the 5th one has a probability of 0.89 to match the 1st entity class.
- In the 2nd sequence, the span starting with the 6th token and ending with the 7th one has a probability of 0.96 to match the second 2nd class.

4.2. Span Decoding

4.2.1. Transformation

$(O, L) \to S$

4.2.2. Source Code

- Struct: `gliner::model::output::decoded::SpanOutput`
- Transformation: `gliner::model::output::decoded::span::TensorsToDecoded`

4.2.3. Format

- t: threshold
- n: number of input sequences
- L: text lengths as defined before
- S: sequence of spans (i, j, k, p) where:
  - i is the index of the first token of sequence m with $i < j$ and $i < L(m)$
  - j is the index of the last token with the same constraints as i
  - k is the entity class,
  - p is the probability for class k with $p \geq t$

$$S = \begin{bmatrix} [(i_{1,1}, j_{1,1}, k_{1,1}, p_{1,1}) \quad (i_{1,2}, j_{1,2}, k_{1,2}, p_{1,2}) \quad \dots] \\ \vdots \\ [(i_{n,1}, j_{n,1}, k_{n,1}, p_{n,1}) \quad (i_{n,2}, j_{n,2}, k_{n,2}, p_{n,2}) \quad \dots] \end{bmatrix}$$

4.2.4. Example

$$S = \begin{bmatrix} [(4, 5, 1, 0.89)] \\ [(6, 7, 2, 0.96)] \end{bmatrix}$$

5. Post-Processing (Token Mode)

5.1. Logits Output

5.1.1. Source Code

- Struct: `gliner::model::output::TensorOutput`

5.1.2. Format

- n: number of text sequences
- w: maximum number of tokens in one sequence
- k: number of entity labels
- O: logits output, of type `f32` and shape $(3 * n * w * k)$ with:
  - $s_{\mathrm{n,w,k}}$: raw model output for a start token w in sequence n and label k.
  - $e_{\mathrm{n,w,k}}$: raw model output for an end token w in sequence n and label k.
  - $i_{\mathrm{n,w,k}}$: raw model output for an inside token w in sequence n and label k.

$$O = \begin{pmatrix} \begin{pmatrix} \begin{pmatrix} s_{1,1,1} & \cdots & s_{1,1,\mathrm{k}} \\ \vdots & \ddots & \vdots \\ s_{1,\mathrm{w},1} & \cdots & s_{1,\mathrm{w},\mathrm{k}} \end{pmatrix} & \dots & \begin{pmatrix} s_{\mathrm{n},1,1} & \cdots & s_{\mathrm{n},1,\mathrm{k}} \\ \vdots & \ddots & \vdots \\ s_{\mathrm{n},\mathrm{w},1} & \cdots & s_{\mathrm{n},\mathrm{w},\mathrm{k}} \end{pmatrix} \end{pmatrix} \\ \begin{pmatrix} \begin{pmatrix} e_{1,1,1} & \cdots & e_{1,1,\mathrm{k}} \\ \vdots & \ddots & \vdots \\ e_{1,\mathrm{w},1} & \cdots & e_{1,\mathrm{w},\mathrm{k}} \end{pmatrix} & \dots & \begin{pmatrix} e_{\mathrm{n},1,1} & \cdots & e_{\mathrm{n},1,\mathrm{k}} \\ \vdots & \ddots & \vdots \\ e_{\mathrm{n},\mathrm{w},1} & \cdots & e_{\mathrm{n},\mathrm{w},\mathrm{k}} \end{pmatrix} \end{pmatrix} \\ \begin{pmatrix} \begin{pmatrix} i_{1,1,1} & \cdots & i_{1,1,\mathrm{k}} \\ \vdots & \ddots & \vdots \\ i_{1,\mathrm{w},1} & \cdots & i_{1,\mathrm{w},\mathrm{k}} \end{pmatrix} & \dots & \begin{pmatrix} i_{\mathrm{n},1,1} & \cdots & i_{\mathrm{n},1,\mathrm{k}} \\ \vdots & \ddots & \vdots \\ i_{\mathrm{n},\mathrm{w},1} & \cdots & i_{\mathrm{n},\mathrm{w},\mathrm{k}} \end{pmatrix} \end{pmatrix} \end{pmatrix}$$

5.1.3. Example

For readability purposes, the raw values are “sigmoided” ($S(x) = \frac{1}{1+e^{-x}}$) and then “ReLUed” with a threshold $t = 0.5$.

$$O_{\mathrm{S,t}} = \begin{pmatrix} \begin{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \mathbf{0.97} & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & \mathbf{0.99} \\ 0 & 0 \end{pmatrix} \end{pmatrix} \\ \begin{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \mathbf{0.96} & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & \mathbf{0.97} \end{pmatrix} \end{pmatrix} \\ \begin{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \mathbf{0.98} & 0 \\ \mathbf{0.98} & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & \mathbf{0.99} \\ 0 & \mathbf{0.99} \end{pmatrix} \end{pmatrix} \end{pmatrix}$$

5.2. Span Decoding

5.2.1. Transformation

$O \to S$

5.2.2. Source Code

- Struct: `gliner::model::output::decoded::SpanOutput`
- Transformation: `gliner::model::output::decoded::token::TensorsToDecoded`

5.2.3. Format

Same format as in span-mode.

6. Post-Processing (Common)

6.1. Span Filtering (Greedy Search)

6.1.1. Transformation

$S \to S'$

6.1.2. Source Code

- Struct: `gliner::model::output::decoded::SpanOutput`
- Transformation: `gliner::model::output::decoded::greedy::GreedySearch`

6.1.3. Format

Same as span output.