Semantic Role Labeling with Conditional Random Field and SpanBERT

Slides originally by Tuur Leeuwenberg and Quynh Do, adapted by Elias Moons, Sumam Francis, Ruiqi Li, Nathan Cornille, Ruben Cartuyvels, Marie-Francine Moens

LIIR lab, Department of Computer Science, KU Leuven

2025

Semantic Role Labeling (SRL)

Who does **what** to **whom** and **how**?

He$_?$ loves$_{predicate}$ dogs$_?$ tremendously$_?$

He$_?$ ran$_{predicate}$ fast$_?$

John$_?$ dances$_{predicate_1}$ while Mary$_?$ sings$_{predicate_2}$

Semantic Role Labeling (SRL)

Who does **what** to **whom** and **how**?

He$_{A0}$ loves$_{predicate}$ dogs$_{A1}$ tremendously$_{AM-MNR}$

He$_{A0}$ ran$_{predicate}$ fast$_{AM-MNR}$

John$_{A0_1}$ dances$_{predicate_1}$ while Mary$_{A0_2}$ sings$_{predicate_2}$

$A0$ = agent, $A1$ = patient, $AM - MNR$ = manner.

SRL as Sequence Prediction

Input sentence and predicate index (s, j):

He$_?$ loves$_{predicate}$ dogs$_?$ tremendously$_?$

Output label sequence:

$$L = [A0, A1, AM - MNR]$$

Model:

$$P(L|s, j)$$

Sequence Prediction with a Linear Chain CRF

$P(L|s,j)$ is calculated using weights $\lambda_k \in \lambda$ and features $f_k \in F$ by first scoring the label configuration L.

$$score(L|s,j) = \sum_{i=1}^{C(s,j)} \sum_{k=1}^{K} \lambda_k f_k(s,i,j,l_i,l_{i-1})$$

with $C(s,j)$ is the total number of arguments of the given predicate.

Sequence Prediction with a Linear Chain CRF

$P(L|s,j)$ is calculated using weights $\lambda_k \in \lambda$ and features $f_k \in F$ by first scoring the label configuration L.

$$score(L|s,j) = \sum_{i=1}^{C(s,j)} \sum_{k=1}^{K} \lambda_k f_k(s,i,j,l_i,l_{i-1})$$

with $C(s,j)$ is the total number of arguments of the given predicate.

The score is made a probability by normalizing over all possible labelings for the sequence (softmax):

$$P(L|s,j) = \frac{exp(score(L|s,j))}{\sum_{L'} exp(score(L'|s,j))}$$

Feature Function

Each feature function $f_k(s, i, j, l_i, l_{i-1})$ is a function taking as input:

1. Sentence s
2. Position i of an argument in the sentence
3. Position j of the predicate
4. Label l_i of the current word
5. Label l_{i-1} of the previous argument.

and often outputs binary values (0 or 1).

Example Feature: Argument word-form

For each possible (argument word, label) pair (w, y) in the training data, we have one feature function:

$$f_{w,y}(s, i, j, l_i, l_{i-1}) = \begin{cases} 1 & \text{if } a(s,j)[i] = w \text{ and } l_i = y \\ 0 & \text{otherwise} \end{cases}$$

Example Feature: Argument word-form

For each possible (argument word, label) pair (w, y) in the training data, we have one feature function:

$$f_{w,y}(s, i, j, l_i, l_{i-1}) = \begin{cases} 1 & \text{if } a(s,j)[i] = w \text{ and } l_i = y \\ 0 & \text{otherwise} \end{cases}$$

Example Revisited:

He$_{A0}$ loves$_{predicate}$ dogs$_{A1}$ tremendously$_{AM-MNR}$

He$_{A0}$ ran$_{predicate}$ fast$_{AM-MNR}$

12 extracted features (4 word types, 3 label types):

(he,A0),(he,A1), (he,AM-MNR),(dogs,A0),(dogs,A1),(dogs,AM-MNR), (fast,A0), (fast,A1), (fast,AM-MNR), (tremendously,A0), (tremendously, A1), (tremendously,AM-MNR)

Feature: Predicate argument-roles

For each possible (predicate word (*p*), label (*y*)) pair in the training data, we have one feature function:

$$f_{p,y}(s,i,j,l_i,l_{i-1}) = \begin{cases} 1 & \text{if } s[j] = p \text{ and } l_i = y \\ 0 & \text{otherwise} \end{cases}$$

Example Revisited:

He$_{A0}$ loves$_{predicate}$ dogs$_{A1}$ tremendously$_{AM-MNR}$

He$_{A0}$ ran$_{predicate}$ fast$_{AM-MNR}$

6 extracted features (2 predicate types, 3 label types):

(ran,A0),(ran,A1),(ran,AM-MNR),(loves, A0), (loves,A1), (loves,AM-MNR)

Feature: Label-transitions

For each possible (previous label (l_{i-1}), current label (l_i)) pair in the training data, we have one feature function:

$$f_{l_1,l_2}(s,i,j,l_i,l_{i-1}) = \begin{cases} 1 & \text{if } l_i = l_2 \text{ and } l_{i-1} = l_1 \\ 0 & \text{otherwise} \end{cases}$$

Example Revisited:

He$_{A0}$ loves$_{predicate}$ dogs$_{A1}$ tremendously$_{AM-MNR}$

He$_{A0}$ ran$_{predicate}$ fast$_{AM-MNR}$

Extracted features: $(Null, A0), (Null, A1), (Null, AM-MNR), (A0, A1), (A1, A0), (A0, AM-MNR), (AM-MNR, A0), (A1, AM-MNR), (AM-MNR, A1)$

Feature Engineering

1. The combination of labels and instances' traditional features (e.g., argument word form, predicate word form) may result in a large number of parameters.
2. But, many of these features never occur in the training data e.g., (he,AM-MNR).
3. Perhaps surprisingly, these features can be useful: they can be given a negative weight that prevents the spurious label from being assigned high probability.
4. Including these features typically results in slight improvements in accuracy, at the cost of greatly increasing the number of parameters in the model.

Training: Objective function

Our objective is to find a set of parameters λ so that the resulting distribution $P(L|s,j,\lambda)$ best fits the set of training examples.
A standard way of training CRFs is by **maximum log-likelihood**:

$$l(\lambda) = log(\prod_{s,j} P(L|s,j)) = \sum_{s,j} log(P(L|s,j)) =$$

$$\sum_{s,j} log(\frac{exp(\sum_{i=1}^{C(s,j)} \sum_{k=1}^{K} \lambda_k f_k(s,i,j,l_i,l_{i-1}))}{\sum_{L'} exp(\sum_{i=1}^{C(s,j)} \sum_{k=1}^{K} \lambda_k f_k(s,i,j,y'_i,y'_{i-1}))})$$

Training: Stochastic Gradient Descent

For each sentence s, predicate position j, and the gold labeling L, we compute the gradient of the local likelihood $l^{s,j}(\lambda) = log(P(L|s,j))$ with respect to each feature weight λ_q:

$$\frac{\partial}{\partial \lambda_q} logP(L|s,j) =$$

$$\sum_{i=1}^{C(s,j)} f_q(s,i,j,l_i,l_{i-1}) - \sum_{L'} [P(L'|s,j) * \sum_{i=1}^{C(s,j)} f_q(s,i,j,y'_i,y'_{i-1})]$$

Deriving the gradient

$$
\begin{aligned}
&\frac{\partial}{\partial \lambda_q} \log P(L|s,j) \\
&= \frac{\partial}{\partial \lambda_q} \log\left(\frac{exp(\sum_{i=1}^{C(s,j)} \sum_{k=1}^{K} \lambda_k f_k(s,i,j,l_i,l_{i-1}))}{\sum_{L'} exp(\sum_{i=1}^{C(s,j)} \sum_{k=1}^{K} \lambda_k f_k(s,i,j,l'_i,l'_{i-1}))}\right) \\
&= \frac{\partial}{\partial \lambda_q} \log(exp(\sum_{i=1}^{C(s,j)} \sum_{k=1}^{K} \lambda_k f_k(s,i,j,l_i,l_{i-1}))) - \frac{\partial}{\partial \lambda_q} \log(\sum_{L'} exp(\sum_{i=1}^{C(s,j)} \sum_{k=1}^{K} \lambda_k f_k(s,i,j,l'_i,l'_{i-1}))) \\
&= \frac{1}{exp(\sum_{i=1}^{C(s,j)} \sum_{k=1}^{K} \lambda_k f_k(s,i,j,l_i,l_{i-1}))} exp(\sum_{i=1}^{C(s,j)} \sum_{k=1}^{K} \lambda_k f_k(s,i,j,l_i,l_{i-1})) * \sum_{i=1}^{C(s,j)} f_q(s,i,j,l_i,l_{i-1}) \\
&\quad - \frac{1}{\sum_{L''} exp(\sum_{i=1}^{C(s,j)} \sum_{k=1}^{K} \lambda_k f_k(s,i,j,l''_i,l''_{i-1}))} \sum_{L'} [exp(\sum_{i=1}^{C(s,j)} \sum_{k=1}^{K} \lambda_k f_k(s,i,j,l'_i,l'_{i-1})) * \sum_{i=1}^{C(s,j)} f_q(s,i,j,l'_i,l'_{i-1})] \\
&= \sum_{i=1}^{C(s,j)} f_q(s,i,j,l_i,l_{i-1}) - \sum_{L'} [\frac{exp(\sum_{i=1}^{C(s,j)} \sum_{k=1}^{K} \lambda_k f_k(s,i,j,l'_i,l'_{i-1}))}{\sum_{L''} exp(\sum_{i=1}^{C(s,j)} \sum_{k=1}^{K} \lambda_k f_k(s,i,j,l''_i,l''_{i-1}))} * \sum_{i=1}^{C(s,j)} f_q(s,i,j,l'_i,l'_{i-1})] \\
&= \sum_{i=1}^{C(s,j)} f_q(s,i,j,l_i,l_{i-1}) - \sum_{L'} P(L'|s,j) * \sum_{i=1}^{C(s,j)} f_q(s,i,j,l'_i,l'_{i-1})
\end{aligned}
$$

Stochastic Gradient Descent

Update the weights using the computed gradient:

$$\lambda_q = \lambda_q + \alpha * \frac{\partial}{\partial \lambda_q} log P(L|s, j)$$

with α a learning rate.

SRL - Training[1] [Exercise*]

We consider only the following sentence as all training data:

He$_{A0}$ ran$_{predicate}$ fast$_{AM-MNR}$

Using this training data, the feature set has 6 observed features:

- Argument word-forms: $(he, A0), (fast, AM-MNR)$
- Predicate argument-roles: $(ran, A0), (ran, AM-MNR)$
- Label transitions: $(Null, A0), (A0, AM-MNR)$

So a feature vector is constructed by evaluating the following features for each (argument, predicate) pair:

[$f_{he,A0}$, $f_{fast,AM-MNR}$, $f_{ran,A0}$, $f_{ran,AM-MNR}$, $f_{Null,A0}$, $f_{A0,AM-MNR}$]

We have two (argument, predicate) pairs:

1. (ran,he) with label A0 results in $[1, 0, 1, 0, 1, 0]$
2. (ran,fast) with label AM-MNR gives $[0, 1, 0, 1, 0, 1]$

[1] to make it simpler for the exercise, we only consider features that are observed in the training data (at least one training instance has value of 1).

SRL - Training

At the start of training we set our feature weights λ to zero.

Initialized weights: $\lambda = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]$

Iteration 1

He$_{A0}$ ran$_{predicate}$ fast$_{AM-MNR}$

From the above training example we obtain gold labeling L

$L = [A0, AM - MNR]$

For the example we have (only 2) possible label assignments L'

$L' \in \{[A0, AM - MNR], [AM - MNR, A0]\}$.

To update λ with SGD we set learning rate $\alpha = 0.1$ and calculate gradients using the gold labeling.

$\lambda_{new} = \lambda_{old} + 0.1 * \frac{\partial}{\partial\lambda} logP([A0, AM - MNR]|s, 2)$

Calculating gradients $logP([A0, AM\text{-}MNR]|s, 2)$

$$\frac{\partial}{\partial \lambda_q} logP([A0, AM\text{-}MNR]|s, j) =$$

$$\sum_{i=1}^{C(s,j)} f_q(s, i, j, l_i, l_{i-1}) - \sum_{L'} [P(L'|s, j) * \sum_{i=1}^{C(s,j)} f_q(s, i, j, y'_i, y'_{i-1})]$$

$$= f(s, 1, 2, A0, Null) + f(s, 3, 2, AM\text{-}MNR, A0) -$$
$$P([A0, AM\text{-}MNR]|s, 2) * (f(s, 1, 2, A0, Null) + f(s, 3, 2, AM - MNR, A0)) -$$
$$P([AM\text{-}MNR, A0]|s, 2) * (f(s, 1, 2, AM\text{-}MNR, Null) + f(s, 3, 2, A0, AM\text{-}MNR))$$
$$= [1, 0, 1, 0, 1, 0] + [0, 1, 0, 1, 0, 1] -$$
$$1/2 * ([1, 0, 1, 0, 1, 0] + [0, 1, 0, 1, 0, 1]) -$$
$$1/2 * ([0, 0, 0, 1, 0, 0] + [0, 0, 1, 0, 0, 0])$$
$$= [0.5, 0.5, 0, 0, 0.5, 0.5]$$

where

$$P([A0, AM\text{-}MNR]|s, 2) = \frac{exp(score([A0, AM\text{-}MNR]|s, 2))}{exp(score([A0, AM\text{-}MNR]|s, 2)) + exp(score([AM\text{-}MNR, A0]|s, 2))}$$
$$= \frac{exp(0+0)}{exp(0+0) + exp(0+0)} = 1/2$$

where

$$\begin{aligned} score([A0, AM\text{-}MNR]|s, 2) &= \sum_{i=1}^{C(s,j)} \sum_{k=1}^{K} \lambda_k f_k(s, i, j, l_i, l_{i-1}) \\ &= \lambda * f(s, 1, 2, A0, Null) + \lambda * f(s, 3, 2, AM\text{-}AMNR, A0) = 0 \\ &= ([0,0,0,0,0,0] * [1,0,1,0,1,0]) + ([0,0,0,0,0,0] * [0,1,0,1,0,1]) \\ &= 0 \end{aligned}$$

and

$$score([AM\text{-}MNR, A0]|s, 2) = \lambda * f(s, 1, 2, AM\text{-}MNR, Null) + \lambda * f(s, 3, 2, A0, AM\text{-}MNR) = 0$$

Similarly, we also obtain $P([AM\text{-}MNR, A0]|s, 2) = 1/2$

Update weights:

$$\lambda_{new} = \lambda_{old} + 0.1 * [0.5, 0.5, 0, 0, 0.5, 0.5] = [0.05, 0.05, 0, 0, 0.05, 0.05]$$

Iteration 2 (Same procedure but with a different λ)

Exactly the same except that $P([A0, AM\text{-}MNR]|s, 2)$ and $P([A0, AM\text{-}MNR]|s, 2)$ are different because λ is different.

$$\frac{\partial}{\partial\lambda} logP((A0, AM - MNR)|s, 2) =$$

$$= f(s, 1, 2, A0, Null) + f(s, 3, 2, AM\text{-}MNR, A0) -$$

$$P([A0, AM\text{-}MNR]|s, 2) * (f(s, 1, 2, A0, Null) + f(s, 3, 2, AM - MNR, A0)) -$$

$$P([AM\text{-}MNR, A0]|s, 2) * (f(s, 1, 2, AM\text{-}MNR, Null) + f(s, 3, 2, A0, AM\text{-}MNR))$$

$$= [1, 0, 1, 0, 1, 0] + [0, 1, 0, 1, 0, 1] -$$

$$.55 * ([1, 0, 1, 0, 1, 0] + [0, 1, 0, 1, 0, 1]) -$$

$$.45 * ([0, 0, 0, 1, 0, 0] + [0, 0, 1, 0, 0, 0])$$

$$= [0.45, 0.45, 0, 0, 0.45, 0.45]$$

Update weights:

$\lambda_{new} = \lambda_{old} + 0.1 * [0.45, 0.45, 0, 0, 0.45, 0.45] = [0.05, 0.05, 0, 0, 0.05, 0.05] + [0.045, 0.045, 0, 0, 0.045, 0.045] = [0.095, 0.095, 0, 0, 0.095, 0.095]$

where

$$P([A0, AM\text{-}MNR]|s, 2) = \frac{exp(score([A0, AM\text{-}MNR]|s, 2))}{exp(score([A0, AM\text{-}MNR]|s, 2)) + exp(score([AM\text{-}MNR, A0]|s, 2))}$$
$$= \frac{exp(0.2)}{exp(0.2) + exp(0)} \approx .55$$

where

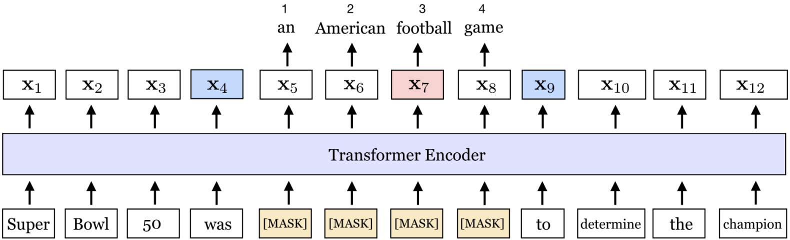
$$\begin{aligned}
score([A0, AM\text{-}MNR]|s, 2) &= \sum_{i=1}^{C(s,j)} \sum_{k=1}^{K} \lambda_k f_k(s, i, j, l_i, l_{i-1}) \\
&= \lambda * f(s, 1, 2, A0, Null) + \lambda * f(s, 3, 2, AM\text{-}AMNR, A0) \\
&= ([0.05, 0.05, 0, 0, 0.05, 0.05] * [1, 0, 1, 0, 1, 0]) \\
&\quad + ([0.05, 0.05, 0, 0, 0.05, 0.05] * [0, 1, 0, 1, 0, 1]) \\
&\approx .2
\end{aligned}$$

and

$$\begin{aligned}
score([AM\text{-}MNR, A0]|s, 2) &= \lambda * f(s, 1, 2, AM\text{-}MNR, Null) + \lambda * f(s, 3, 2, A0, AM\text{-}MNR) \\
&= ([0.05, 0.05, 0, 0, 0.05, 0.05] * [0, 0, 0, 1, 0, 0]) \\
&\quad + ([0.05, 0.05, 0, 0, 0.05, 0.05] * [0, 0, 1, 0, 0, 0]) \\
&= 0
\end{aligned}$$

SpanBERT: pre-training

$$\mathcal{L}(\text{football}) = \mathcal{L}_{\text{MLM}}(\text{football}) + \mathcal{L}_{\text{SBO}}(\text{football})$$
$$= -\log P(\text{football} \mid \mathbf{x}_7) - \log P(\text{football} \mid \mathbf{x}_4, \mathbf{x}_9, \mathbf{p}_3)$$



The span an *American football game* is masked.

x_4 and x_9 are the output representations of the boundary tokens.

SBO predicts each token in the masked span.

p_3 is the position embedding of the third token of the span.

SpanBERT pre-training: Spans to mask

- Random sub-word tokens
- Random whole words
- Noun Phrases
- Named Entities

SpanBERT: downstream SRL

Does this make sense?

- Arguments usually cover a span of words, so yes.
- SpanBERT might be overly complex for SRL?
- Is there an advantage of using these span representations instead of predicting per individual word what SRL argument it belongs too?
- How do you determine span boundaries, or do you test all possible spans (there are $\mathcal{O}(T^2)$ possible spans in a length T sentence)?

SpanBERT: Objective for downstream SRL

Sentences $X^n, n \in 1 \dots N$ as sequences of T^n words $x_1^n \dots x_{T(n)}^n$.
Each sentence has a set of S^n masked spans. Each span is defined by its length K_i^n and start position c_i^n.
Each word x_j^n has a position embedding p_j^n and an SRL label z_j^n.
Θ_c are the common model parameters involved in both objectives, Θ_{SBO} the parameters only involved in the SBO loss, and Θ_{CAT} those only involved in the CAT loss.
Then:

$$\mathcal{L} = \sum_{n=1}^{N} \mathcal{L}^n$$

$$\mathcal{L}^n =$$

SpanBERT: Objective for SRL

Sentences $X^n, n \in 1 \dots N$ as sequences of T^n words $x_1^n \dots x_{T(n)}^n$.
Each sentence has a set of S^n masked spans. Each span is defined by its length K_i^n and start position c_i^n.
Each word x_j^n has a position embedding p_j^n and an SRL label z_j^n.
Θ_c are the common model parameters involved in both objectives, Θ_{SBO} the parameters only involved in the SBO loss, and Θ_{CAT} those only involved in the CAT loss.
Then:

$$
\begin{aligned}
\mathcal{L} &= \sum_{n=1}^{N} \mathcal{L}^n \\
\mathcal{L}^n &= \sum_{i=1}^{S^n} \sum_{k=0}^{K_i^n-1} \mathcal{L}_{SBO}(x^n_{c_i^n+k}) + \sum_{t=1}^{T^n} \mathcal{L}_{CAT}(z_t^n) \\
&= \sum_{i=1}^{S^n} \sum_{k=0}^{K_i^n-1} \Big[-\log P(x^n_{c_i^n+k} | x^n_{c_i^n-1}, x^n_{c_i^n+K_i^n}, p^n_{c_i^n+k}; \varTheta_c, \varTheta_{SBO}) \Big] \\
&\quad + \sum_{t=1}^{T^n} -\log P(z_t^n | X^n; \varTheta_c, \varTheta_{CAT}))
\end{aligned}
$$