KU Leuven - CSAI

Exercise Session 3 - Solutions
SRL

For questions contact:
Toledo Forum for Questions or
nlp@ls.kuleuven.be

Quynh Do, Victor Milewski and Marie-Francine Moens

1 Semantic Role Labeling with CRF

Question 1.A: designing feature functions

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

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

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

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

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

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

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

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

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

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

  This results in 10 Extracted features (3 label types + *Null*):

$$(Null, A0), (Null, A1), (Null, AM-MNR), (A0, A1), (A1, A0), (A0, AM-MNR),$$
$$(AM-MNR, A0), (A1, AM-MNR), (AM-MNR, A1)$$

This results in 28 feature functions but many of these features never occur in the training data e.g., (he,AM-MNR). Perhaps surprisingly, these features can be useful: they can be given a negative weight that prevents the spurious label from being assigned high probability. Including these features typically results in slight improvements in accuracy, at the cost of greatly increasing the number of parameters in the model.

Question 1.B: training the model

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 $[0, 1, 0, 1, 0, 1]$

At the start of training we set our feature weights $\boldsymbol{\lambda}$ to zero. Initialized weights:

$$\boldsymbol{\lambda} = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]$$

Iteration 1:
From the training example we obtain gold labeling L:

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

This training example has (only 2) possible label assignments L':

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

To update $\boldsymbol{\lambda}$ with SGD (learning rate $\alpha = 0.1$) we calculate gradients using the gold labeling:

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

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

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

and

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

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

Update weights:

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

Iteration 2 (Same procedure but with a different $\boldsymbol{\lambda}$):
Exactly the same except that $P([A0, AM\text{-}MNR]|s, 2)$ and $P([A0, AM\text{-}MNR]|s, 2)$ are different because $\boldsymbol{\lambda}$ is different.

$$\frac{\partial}{\partial \lambda_q} logP([A0, AM\text{-}MNR]|s, j) =$$
$$\sum_{i=1}^{C(s,j)} f_q(s, i, j, l_i, l_{i-1}) - \sum_{L'}[P(L'|s, j) * \sum_{i=1}^{C(s,j)} f_q(s, i, j, y'_i, y'_{i-1})]$$

$$= f(s, 1, 2, A0, Null) + f(s, 3, 2, AM\text{-}MNR, A0) -$$
$$P([A0, AM\text{-}MNR]|s, 2) * (f(s, 1, 2, A0, Null) + f(s, 3, 2, AM - MNR, A0)) -$$
$$P([AM\text{-}MNR, A0]|s, 2) * (f(s, 1, 2, AM\text{-}MNR, Null) + f(s, 3, 2, A0, AM\text{-}MNR))$$
$$= [1, 0, 1, 0, 1, 0] + [0, 1, 0, 1, 0, 1] -$$
$$.55 * ([1, 0, 1, 0, 1, 0] + [0, 1, 0, 1, 0, 1]) -$$
$$.45 * ([0, 0, 0, 1, 0, 0] + [0, 0, 1, 0, 0, 0])$$
$$= [0.45, 0.45, 0, 0, 0.45, 0.45]$$

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

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

and

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

Similarly, we also obtain $P([AM\text{-}MNR, A0]|s, 2) \approx .45$

Update weights:

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

2 SpanBERT

In this question we will take a look at SpanBERT and start thinking about how to use the model. Figure 1 shows the architecture of the model and how it is trained (in a self-supervised manner).

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

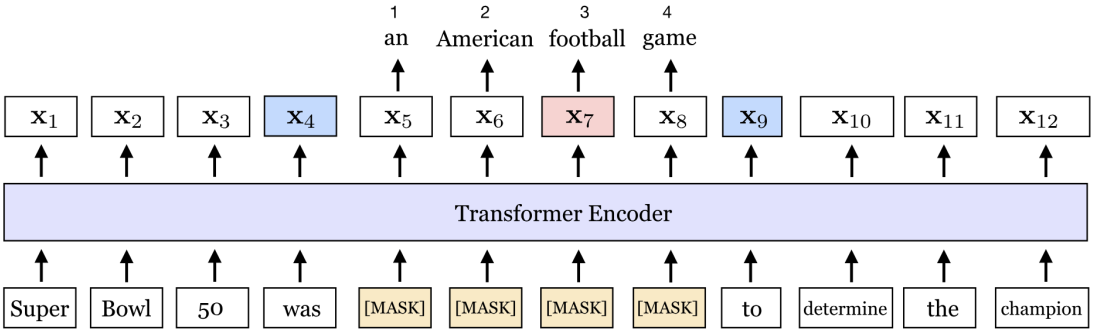

Figure 1: An illustration of SpanBERT training. The span an *American football game* is masked. The SBO uses the output representations of the boundary tokens, $\boldsymbol{x}_4$ and $\boldsymbol{x}_9$ (in blue), to predict each token in the masked span. The equation shows the MLM and SBO loss terms for predicting the token, *football* (in pink), which as marked by the position embedding $\boldsymbol{p}_3$, is the third token from x_4.

3.A: Self-supervised pre-training with masked spans

See also cited SpanBERT paper [1].

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

3.B: SpanBERT finetuned for downstream SRL

1. Does it 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 $\mathcal{O}(T^2)$ spans in a length T sentence)? Would have to be tested empirically (i.e., implemented and compared).

2. What is the objective?

   Assume N sentences, where each sentence $X^n, n \in 1 \dots N$ is a sequence 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.
   For each word x_j^n, there is also a position embedding p_j^n, and an SRL label z_j^n.
   Finally, the model parameters are denoted with Θ. Θ_c are the common 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:

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

References

[1] Mandar Joshi et al. “Spanbert: Improving pre-training by representing and predicting spans”. In: *Transactions of the association for computational linguistics* 8 (2020), pp. 64–77.