Capturing Argument Interaction in Semantic Role Labeling with Capsule Networks

Xinchi Chen\textsuperscript{1}  Chunchuan Lyu\textsuperscript{1}  Ivan Titov\textsuperscript{1,2}

xchen13@exseed.ed.ac.uk  chunchuan.lv@gmail.com  ititov@inf.ed.ac.uk
\textsuperscript{1} ILCC, School of Informatics, University of Edinburgh
\textsuperscript{2} ILLC, University of Amsterdam

Abstract
Semantic role labeling (SRL) involves extracting propositions (i.e., predicates and their typed arguments) from natural language sentences. State-of-the-art SRL models rely on powerful encoders (e.g., LSTMs) and do not model non-local interaction between arguments. We propose a new approach to modeling these interactions while maintaining efficient inference. Specifically, we use Capsule Networks (Sabour et al., 2017): each proposition is encoded as a tuple of capsules, one capsule per argument type (i.e. role). These tuples serve as embeddings of entire propositions. In every network layer, the capsules interact with each other and with representations of words in the sentence. Each iteration results in updated proposition embeddings and updated predictions about the SRL structure. Our model substantially outperforms the non-refinement baseline model on all 7 CoNLL-2009 languages and achieves state-of-the-art results on 5 languages (including English) for dependency SRL. We analyze the types of mistakes corrected by the refinement procedure. For example, each role is typically (but not always) filled with at most one argument. Whereas enforcing this approximate constraint is not useful with the modern SRL system, iterative procedure corrects the mistakes by capturing this intuition in a flexible and context-sensitive way.\textsuperscript{1}

1 Introduction
The task of semantic role labeling (SRL) involves the prediction of predicate-argument structure, i.e., both the identification of arguments and labeling them with underlying semantic roles. The shallow semantic structures have been shown beneficial in many natural language processing (NLP) applications, including information extraction (Christensen et al., 2011), question answering (Eckert and Neves, 2018) and machine translation (Marchegiani et al., 2018).

In this work, we focus on the dependency version of the SRL task (Surdeanu et al., 2008). An example of a dependency semantic-role structure is shown in Figure 1. Edges in the graph are marked with semantic roles, whereas predicates (typically verbs or nominalized verbs) are annotated with their senses.

Intuitively, there are many restrictions on potential predicate-argument structures. Consider the ‘role uniqueness’ constraint: each role is typically, but not always, realized at most once. For example, predicates have at most one agent. Similarly, depending on a verb class, only certain subcategorization patterns are licensed. Nevertheless, rather than modeling the interaction between argument labeling decisions, state-of-the-art semantic role labeling models (Marchegiani and Titov, 2017; Cai et al., 2018; Li et al., 2019b) rely on powerful sentence encoders (e.g., multi-layer Bi-LSTMs (Zhou and Xu, 2015; Qian et al., 2017; Tan et al., 2018)). This contrasts with earlier work on SRL, which hinged on global declarative constraints on the labeling decisions (FitzGerald et al., 2015; Tércio et al., 2015; Das et al., 2012). Modern SRL systems are much more accurate and hence enforcing such hard and often approximate constraints is unlikely to be as beneficial (see our experiments in Section 7.2).

\textsuperscript{1}Code: https://github.com/DalstonChen/CapNetSRL.

Figure 1: An example predicate-argument structure: green and red arcs denote correct and wrong predictions, respectively.
Instead of using hard constraints, we propose to use a simple iterative structure-refinement procedure. It starts with independent predictions and refines them in every subsequent iteration. When refining a role prediction for a given candidate argument, information about the assignment of roles to other arguments gets integrated. Our intuition is that modeling interactions through the output spaces rather than hoping that the encoder somehow learns to capture them, provides a useful inductive bias to the model. In other words, capturing this interaction explicitly should mitigate overfitting. This may be especially useful in a lower-resource setting but, as we will see in experiments, it appears beneficial even in a high-resource setting.

We think of semantic role labeling as extracting propositions, i.e. predicates and argument-role tuples. In our example, the proposition is sign.01(Arg0:signer = Hiti, Arg1:document = contract). Across the iterations, we maintain and refine not only predictions about propositions but also their embeddings. Each ‘slot’ (e.g., correspond to role Arg0:signer) is represented with a vector, encoding information about arguments assumed to be filling this slot. For example, if Hiti is predicted to fill slot Arg0 in the first iteration then the slot representation will be computed based on the contextualized representation of that word and hence reflect this information. The combination of all slot embeddings (one per semantic role) constitutes the embedding of the proposition. The proposition embedding, along with the prediction about the semantic role structure, gets refined in every iteration of the algorithm.

Note that, in practice, the predictions in every iteration are soft, and hence proposition embeddings will encode current beliefs about the predicate-argument structure. The distributed representation of propositions provides an alternative ("dense-embedding") view on the currently considered semantic structure, i.e. information extracted from the sentence. Intuitively, this representation can be readily tested to see how well current predictions satisfy selection restrictions (e.g., contract is a very natural filler for Arg1:document) and check if the arguments are compatible.

To get an intuition how the refinement mechanism may work, imagine that both Hiti and injury are labeled as Arg0 for sign.01 in the first iteration. Hence, the representation of the slot Arg0 will encode information about both predicted arguments. At the second iteration, the word injury will be aware that there is a much better candidate for filling the signer role, and the probability of assigning injury to this role will drop. As we will see in our experiments, whereas enforcing the hard uniqueness constraint is not useful, our iterative procedure corrects the mistakes of the base model by capturing interrelations between arguments in a flexible and context-sensitive way.

In order to operationalize the above idea, we take inspiration from the Capsule Networks (CNs) (Sabour et al., 2017). Note that we are not simply replacing Bi-LSTMs with generic CNs. Instead, we use CN to encode the structure of the refinement procedure sketched above. Each slot embedding is now a capsule and each proposition embedding is a tuple of capsules, one capsule per role. In every iteration (i.e. CN network layer), the capsules interact with each other and with representations of words in the sentence.

We experiment with our model on standard benchmarks for 7 languages from CoNLL-2009. Compared with the non-refinement baseline model, we observe substantial improvements from using the iterative procedure. The model achieves state-of-the-art performance in 5 languages, including English.

2 Base Dependency SRL Model

In dependency SRL, for each predicate p of a given sentence \( x = \{x_1, x_2, \ldots, x_n\} \) with \( n \) words, the model needs to predict roles \( y = \{y_1, y_2, \ldots, y_n\} \) for every word. The role can be none, signifying that the corresponding word is not an argument of the predicate.

We start with describing the factorized baseline model which is similar to that of Marcheggiani et al. (2017). It consists of three components: (1) an embedding layer; (2) an encoding layer and (3) an inference layer.

2.1 Embedding Layer

The first step is to map symbolic sentence \( x \) and predicate \( p \) into continuous embedded space:

\[ e_i = \text{Lookup}(x_i), \] \hspace{1cm} (1)
\[ p = \text{Lookup}(p), \] \hspace{1cm} (2)

where \( e_i \in \mathcal{R}^{d_e} \) and \( p \in \mathcal{R}^{d_e} \).
2.2 Encoding Layer

The encoding layer extracts features from input sentence \( x \) and predicate \( p \). We extract features from input sentence \( x \) using stacked bidirectional LSTMs (Hochreiter and Schmidhuber, 1997):

\[
x_i = \text{Bi-LSTMs}(e_i),
\]

where \( x_i \in \mathcal{R}^{d_l} \). Then we represent each role logit of each word by a bi-linear operation with the target predicate:

\[
b_{j|i} = x_i W p,
\]

where \( W \in \mathcal{R}^{d_l \times d_c} \) is a trainable parameter and \( b_{j|i} \in \mathcal{R} \) is a scalar representing the logit of role \( j \) for word \( x_i \).

2.3 Inference Layer

The probability \( P(y|x, p) \) is then computed as

\[
P(y|x, p) = \prod_i P(y_i|x, i, p),
\]

\[
= \prod_i \text{softmax}(b_{j|i}y_i),
\]

where \( b_{j|i} = \{b_{1|i}, b_{2|i}, \ldots, b_{|\mathcal{T}|i}\} \) and \( |\mathcal{T}| \) denotes the number of role types.

3 Dependency based Semantic Role Labeling using Capsule Networks

Inspired by capsule networks, we use capsule structure for each role state to maintain information across iterations and employ the dynamic routing mechanism to derive the role logits \( b_{j|i} \) iteratively. Figure 2 illustrates the architecture of the proposed model.

3.1 Capsule Structure

We start by introducing two capsule layers: (1) the word capsule layer and (2) the role capsule layer.

3.1.1 Word Capsule Layer

The word capsule layer is comprised of capsules representing the roles of each word. Given sentence representation \( x_i \) and predicate embedding \( p \), the word capsule layer is derived as:

\[
u_{j|i}^k = x_i W_{j}^k p; \quad k = 1, 2, \ldots, K,
\]

where \( W_{j}^k \in \mathcal{R}^{d_l \times d_c} \) and \( u_{j|i} \in \mathcal{R}^K \) is the capsule vector for role \( j \) of word \( x_i \). \( K \) denotes the capsule size. Intuitively, the capsule encodes the argument-specific information relevant to deciding if role \( j \) is suitable for the word. These capsules do not get iteratively updated.

3.1.2 Role Capsule Layer

As discussed in the introduction, the role capsule layer could be viewed as an embedding of a proposition. The capsule network generates the capsules in the layer using “routing-by-agreement”. This process can be regarded as a pooling operation. Capsules in the role capsule layer at \( t \)-th iteration are derived as:

\[
v_{j}^{(t)} = \text{Squash}(s_{j}^{(t)}) = \frac{||s_{j}^{(t)}||^2}{1 + ||s_{j}^{(t)}||^2} \frac{s_{j}^{(t)}}{||s_{j}^{(t)}||},
\]

where intermediate capsule \( s_{j}^{(t)} \) is generated with the linear combination of capsules in the word capsule layer with weights \( c_{ij}^{(t)} \):

\[
s_{j}^{(t)} = \sum_i c_{ij}^{(t)} u_{j|i}.
\]

Here, \( c_{ij}^{(t)} \) are coupling coefficients, calculated by “softmax” function over role logits \( b_{j|i}^{(t)} \). The coefficients \( c_{ij}^{(t)} \) can be interpreted as the probability that word \( x_i \) is assigned role \( j \):

\[
c_{i}^{(t)} = \text{softmax}(b_{j|i}^{(t)}).
\]

The role logits \( b_{j|i}^{(t)} \) are decided by the iterative dynamic routing process. The Squash operation will deactivate capsules receiving small input \( s_{j}^{(t)} \) (i.e. roles not predicted in the sentence) by pushing them further to the 0 vector.
Input: # of iterations $T$, all capsules in $l_w u_{ji}$
Result: semantic role logits $b_{ji}^{(t)}$
Initialization: $b_{ji}^{(0)} = 0$;
for $t \in [0, 1, \cdots, T-1]$ do
\hspace{1cm} for all capsules in $l_w$: $c_{j}^{(t)} = \text{softmax}(b_{ji}^{(t)})$;
\hspace{1cm} for all capsules in $l_s$: $s_{ji}^{(t)} = \sum_i c_{ij}^{(t)} u_{ji}$;
\hspace{1cm} for all capsules in $l_r$: $v_{ji}^{(t)} = 8(\text{tanh}(s_{ji}^{(t)}))$;
\hspace{1cm} for all capsules in $l_w$, $l_r$: $b_{ji}^{(t+1)} = b_{ji}^{(t)} + v_{ji}^{(t)} W_{u_{ji}}$;
end

Algorithm 1: Dynamic routing algorithm. $l_w$ and $l_r$ denote word capsule layer and role capsule layer, respectively.

3.2 Dynamic Routing
The dynamic routing process involves $T$ iterations. The role logits $b_{ji}^{(0)}$ before first iteration are all set to zeros: $b_{ji}^{(0)} = 0$. Then, the dynamic routing process updates the role logits $b_{ji}^{(t)}$ by modeling agreement between capsules in two layers:

$$b_{ji}^{(t+1)} = b_{ji}^{(t)} + v_{ji}^{(t)} W_{u_{ji}}$$  \hspace{1cm} (11)

where $W \in \mathbb{R}^{K \times K}$. The whole dynamic routing process is shown in Algorithm 1. The dynamic routing process can be regarded as the role refinement procedure (see Section 5 for details).

4 Incorporating Global Information
When computing the $j$-th role capsule representation (Eq 9), the information originating from an $i$-th word (i.e. $u_{ji}$) is weighted by the probability of assigning role $j$ to word $x_i$ (i.e. $c_{ij}$). In other words, the role capsule receives messages only from words assigned to its role. This implies that the only interaction the capsule network can model is competition between arguments for a role.\footnote{In principle, it can model the opposite, i.e. collaboration \textit{not} attraction but it is unlikely to be useful in SRL.} Note that this is different from imposing the hard role-uniqueness constraint, as the network does this dynamically in a specific context. Still, this is a strong restriction.

In order to make the model more expressive, we further introduce a global role $g^{(t)}$ to incorporate global information about all arguments at the current iterations. The global node is a compressed representation of the entire proposition, and used in the prediction of all arguments, thus permitting arbitrary interaction across arguments. The global node $g^{(t)}$ at $t$-th iteration is derived as:

$$g^{(t)} = W v^{(t)}$$  \hspace{1cm} (12)

where $W \in \mathbb{R}^{K \times (K \cdot |T|)}$. Here, $v^{(t)} = v^{(t)}_1 \oplus v^{(t)}_2 \oplus \cdots \oplus v^{(t)}_{|T|}$ is the concatenation of all capsules in the role capsule layer.

We append an additional term for the role logits update in Eq (11):

$$b_{ji}^{(t+1)} = b_{ji}^{(t)} + v_{ji}^{(t)} W_{u_{ji}} + g^{(t)} W_{g} u_{ji}$$  \hspace{1cm} (13)

where $W \in \mathbb{R}^{K \times K}$ and $W_{g} \in \mathbb{R}^{K \times K}$.

5 Refinement
The dynamic routing process can be seen as iterative role refinement. Concretely, the coupling coefficients $c_{ij}^{(t)}$ in Eq (10) can be interpreted as the predicted distribution of the semantic roles for word $x_i$ in Eq (5) at $t$-th iteration:

$$c_{ij}^{(t)} = P^{(t)}(\cdot|x, i, p).$$  \hspace{1cm} (14)

Since dynamic routing is an iterative process, semantic role distribution $c_{ij}^{(t)}$ in iteration $t$ will affect the semantic role distribution $c_{ij}^{(t+1)}$ in next iteration $t+1$:

$$c_{ij}^{(t+1)} = f(c_{ij}^{(t)}, x, i, p),$$  \hspace{1cm} (15)

where $f(\cdot)$ denotes the refinement function defined by the operations in each dynamic routing iteration.

6 Training
We minimize the following loss function $\mathcal{L}(\theta)$:

$$\mathcal{L}(\theta) = -\frac{1}{n} \sum_i \log P^{(T)}(y_i|x, i, p; \theta) + \lambda ||\theta||^2_2,$$  \hspace{1cm} (16)

where $\lambda$ is a hyper-parameter for the regularization term and $P^{(T)}(y_i|x, i, p; \theta) = c_{ij}^{(T)}$. Unlike standard refinement methods (Belanger et al., 2017; Lee et al., 2018) which sum losses across all refinement iterations, our loss is only based on the prediction made at the last iteration. This encourages the model to rely on the refinement process rather than do it in one shot.

Our baseline model is trained analogously, but using the cross-entropy for the independent classifiers (Eq (5)).
Uniqueness Constraint Assumption As we discussed earlier, for a given target predicate, each semantic role will typically appear at most once. To encode this intuition, we propose another loss term \( L_u(\theta) \):

\[
L_u(\theta) = \frac{1}{|T|} \sum_j \log \text{softmax}(b_j^{(T)}), \tag{17}
\]

where \( b_j^{(T)} \) are the semantic role logits in \( T \)-th iteration. Thus, the final loss \( L^*(\theta) \) is the combination of the two losses:

\[
L^*(\theta) = L(\theta) + \eta L_u(\theta), \tag{18}
\]

where \( \eta \) is a discount coefficient.

7 Experiments

7.1 Datasets & Configuration

We conduct experiments on CoNLL-2009 (Hajič et al., 2009) for all languages, including Catalan (Ca), Chinese (Zh), Czech (Cz), English (En), German (De), Japanese (Jp) and Spanish (Es). As standard we do not consider predicate identification: the predicate position is provided as an input feature. We use the labeled `semantic` F1 which jointly scores sense prediction and role labeling.

We use ELMo (Peters et al., 2018) (dimension \( d_e \) as 1024) for English, and FastText embeddings (Grave et al., 2018) (dimension \( d_e \) as 300) for all other languages on both the baseline model and the proposed CapsuleNet. LSTM state dimension \( d_l \) is 500. Capsule size \( K \) is 16. Batch size is 32. The coefficient for the regularisation term \( \lambda \) is 0.0004. We employ Adam (Kingma and Ba, 2015) as the optimizer and the initial learning rate \( \alpha \) is set to 0.0001. Syntactic information is not utilized in our model.

7.2 Model Selection

Table 1 shows the performance of our model trained with loss \( L^*(\theta) \) for different values of discount coefficient \( \eta \) on the English development set. The model achieves the best performance when \( \eta \) equals to 0. It implies that adding uniqueness constraint on loss actually hurts the performance. Thus, we use the loss \( L^*(\theta) \) with \( \eta \) equals to 0 in the rest of the experiments, which is equivalent to the loss \( L(\theta) \) in Eq (16). We also observe that the model with 2 refinement iterations performs the best (89.92% F1).

7.3 Overall Results

Table 2 compares our model with previous state-of-the-art SRL systems on English. Some of the systems (Lei et al., 2015; Li et al., 2018; Cai et al., 2018) only use local features, whereas others (Swayamdipta et al., 2016) incorporate global information at the expense of greedy decoding. Additionally, a number of systems exploit syntactic information (Roth and Lapata, 2016; Marcheggiani and Titov, 2017; He et al., 2018). Some of the results are obtained with ensemble systems (FitzGerald et al., 2015; Roth and Lapata, 2016).

As we observe, the baseline model (see Section 2) is quite strong, and outperforms the previous state-of-the-art systems on both in-domain and out-of-domain sets on English. The proposed CapsuleNet outperforms the baseline model on English (e.g. obtaining 91.06% F1 on the English test set), which shows the effectiveness of the cap-
7.4 Ablation

Table 3 gives the performance of models with ablation on some key components, which shows the contribution of each component in our model.

- The model without the global node is described in Section 3.
- The model that further removes the role capsule layer takes the mean of capsules $u_{j|i}$ in Eq (7) of word capsule layer as the semantic role logits $b_{j|i}$:
  \[
  b_{j|i} = \frac{1}{K} \sum_{k} u_{j|ki}^k,
  \]
  where $K$ denotes the capsule size.
- The model that additionally removes the word capsule layer is exactly equivalent to the baseline model described in Section 2.

As we observe, CapsuleNet with all components performs the best on both development and test sets on English. The model without using the global node performs well too. It obtains 91.05% F1 on the English test set, almost the same performance as full CapsuleNet. But on the English out-of-domain set, without the global node, the performance drops from 82.72% F1 to 82.36% F1. It implies that the global node helps in model generalization. Further, once the role capsule layer is removed, the performance drops sharply. Note that the model without the role capsule layer does not use refinements and hence does not model argument interaction. It takes the mean of capsules $u_{j|i}$ in the word capsule layer as the semantic role logits $b_{j|i}$ (see Eq (19)), and hence could be viewed as an enhanced ('ensembled') version of the baseline model. Note that we only introduced a very limited number of parameters for the dynamic routing mechanism (see Eq (11-13)). This suggests that the dynamic routing mechanism does genuinely captures interactions between argument labeling decisions and the performance benefits from the refinement process.

7.5 Error Analysis

The performance of our model while varying the sentence length and the number of arguments per proposition is shown in Figure 3. The statistics of the subsets are in Table 4. Our model consistently outperforms the baseline model, except on sentences of between 50 and 59 words. Note that the subset is small (only 391 sentences), so the effect may be random.

7.6 Effect of Refinement

7.6.1 Refinement Performance

We take a CapsuleNet trained to perform refinements in two iterations and vary the number of iterations at test time (Figure 4). Capsule Nets with and without the global node perform very differently. CapsuleNet without the global node reduces the recall and increases the precision with every iteration. This corresponds to our expectations: the version of the network is only capable of modeling competition and hence continues with this filtering process. In contrast, CapsuleNet with the global
node reduces the precision as the number of iteration grows. This model is less transparent, so it harder to see why it chooses this refinement strategy. As expected, the F1 scores for both models peak at the second iteration, the iteration number used in the training phase. The trend for the exact match score is consistent with the F1 score.

7.6.2 Duplicate Arguments

We measure the degree to which the role uniqueness constraint is violated by both models. We plot the number of violations as the function of the number of iterations (Figure 5). Recall that the violations do not imply errors in predictions as even the gold-standard data does not always satisfy the constraint (see the orange line in the figure). As expected, CapsuleNet without the global node captures competition and focuses on enforcing this constraint. Interestingly, it converges to keeping the same fraction of violations as the one observed in the gold standard data. In contrast, the full model increasingly ignores the constraint in later iterations. This is consistent with the overgenerating trend evident from the precision and recall plots discussed above.

Figure 6 illustrates how many roles get changed between consecutive iterations of CapsuleNet, with and without the global node. Green indicates how many correct changes have been made, while red shows how many errors have been introduced. Since the number of changes is very large, we represent the non-zero number \( q \) in the
log space $\tilde{q} = \text{sign}(q) \log(|q|)$.

As we expected, for both models, the majority of changes are correct, leading to better overall performance. We can again see the same trends. CapsuleNet without the global node tends to filter our arguments by changing them to “None”. The reverse is true for the full model.

### 7.7 Multilingual Results

Table 5 gives the results of the proposed CapsuleNet SRL (with global node) on the in-domain test sets of all languages from CoNLL-2009. As shown in Table 5, the proposed model consistently outperforms the non-refinement baseline model and achieves state-of-the-art performance on Catalan (Ca), Czech (Cz), English (En), Japanese (Ja) and Spanish (Es). Interestingly, the effectiveness of the refinement method does not seem to be dependent on the dataset size: the improvements on the smallest (Japanese) and the largest datasets (English) are among the largest.

### 8 Additional Related Work

The capsule networks have been recently applied to a number of NLP tasks (Xiao et al., 2018; Gong et al., 2018). In particular, Yang et al. (2018) represent text classification labels by a layer of capsules, and take the capsule actions as the classification probability. Using a similar method, Xia et al. (2018) perform intent detection with the capsule networks. Wang et al. (2018) and Li et al. (2019a) use capsule networks to capture rich features for machine translation. More closely to our work, Zhang et al. (2018) and Zhang et al. (2019) adopt the capsule networks for relation extraction. The previous models apply the capsule networks to problems that have a fixed number of components in the output. Their approach cannot be directly applied to our setting.

### 9 Conclusions & Future Work

State-of-the-art dependency SRL methods do not account for any interaction between role labeling decisions. We propose an iterative approach to...
SRL. In each iteration, we refine both predictions of the semantic structure (i.e., a discrete structure) and also a distributed representation of the proposition (i.e., the predicate and its predicted arguments). The iterative refinement process lets the model capture interactions between the decisions. We relied on the capsule networks to operationalize this intuition. We demonstrate that our model is effective, and results in improvements over a strong factorized baseline and state-of-the-art results on standard benchmarks for 5 languages (Catalan, Czech, English, Japanese and Spanish) from CoNLL-2009. In future work, we would like to extend the approach to modeling interaction between multiple predicate-argument structures in a sentence as well as to other semantic formalisms (e.g., abstract meaning representations (Banarescu et al., 2013)).

Acknowledgments

We thank Diego Marcheggiani, Jonathan Mallinson and Philip Williams for constructive feedback and suggestions, as well as anonymous reviewers for their comments. The project was supported by the European Research Council (ERC StG BroadSem 678254) and the Dutch National Science Foundation (NWO VIDI 639.022.518).

References


