Semantic Role Labelling

Final project for Natural Language Processing by Prof. Roberto Navigli

Son Tung Nguyen

September 5, 2018

1 Introduction

Semantic Role Labelling (SRL), first proposed by [1], is the task of identifying the arguments for one specific predicate among all the words in the sentence. Answering the question "who did what to whom?", solving SRL task was proved to be beneficial to many NLP applications. Figure 1 shows an example of a SRL task, that is to find out the roles of arguments "I" and "the computer" are "A0" and "A1" of the predicate "repair".

Figure 1: A semantic role labelling example.

2 State of the art

Long short term memory (LSTM) was showed to be very effective in tackling SRL problem. One of the first system to use LSTM outperformed previous methods and was a huge success [2]. Since then, many improvements have been made to increase the performance further. [3] exploited other information of the sentence (POS, dependency relation) and found them to be beneficial when solving SRL. Another more recent system used Highway connection and constrained A* decoding [4].

3 Proposed method

3.1 Model architecture

Used model architecture was closely followed by [2]. The input to the model was in shape of [$batch\ size, sequence\ length, 251$]. That means for each word in a sentence, the following information was passed:

- Currently processed word (in form of a 50d vector).
- Lemma of currently processed predicate (in form of a 50d vector).
- Predicate context (in form of a 50d vector).
- A binary number indicating that current word was in predicate context.
- POS index indicating the POS tag of current word.
- DEP index indicating the dependency of current word to predicate.

All the above information were concatenated together to produce a single vector of 251 entries. Then, a LSTM of state size 64 was used to capture the sequential relationship of the sentence. The output of this LSTM layer was then connected to a fully-connected layer before forwarded to Conditional Random Fields (CRF) [5] layer to compute the loss. This simple model alone could achieve a F1 score of 0.55 on the development set without using any additional information from POS tagging or dependency (DEP).

3.2 Improvements

Stack more LSTM layers The network was modified from LSTMCell to MultiRNNCell to support multi LSTM layers. Table 1 shows that when using MultiRNNCell with 2 layer F1 score went up to 0.59 but decreased to 0.55 for 3 layers.

Bigger predicate window Predicate window size was experimented to see if bigger size would lead to better F1 score. The result was not as expected since bigger predicate window size did not have a big positive effect on F1 score.

POS and DEP information POS and DEP information were also integrated into the input vector to see if there would be any improvement. There were 48 POS types and 69 DEP types. Each type was encoded by a unique number from 0 to 47 (for POS) and 0 to 68 (for DEP). We can see using POS information helped quite well (F1 score increased from 0.55 to 0.58). This can be further enhanced with DEP information (0.59).

Bi LSTM architecture I tried to apply Bi-LSTM architecture using [6]. The network architecture was basically from the input layer to the first linear layer which was activated be a tanh function. The output of this layer then was forwarded to LSTM layers to be processed in both directions. After this, the concatenated h vectors went to another tanh-linear layer and a linear layer. Finally, the CRF layer took over to compute the loss and make inference. This new architecture did not work for me as it only achieved a F1 score of 0.57.

4 Experiments

Datasets Dataset using in experiments was CONLL 2009 [7]. This dataset was split into train set and development set for training and testing the model.

Word vectors Pretrained GLOVE vectors [8] was used. Each word corresponded to one vector of size 50d. If there was not any match for a word, this word would get a fixed random vector (UNK vector). About 0.6 percent of the words were unknown.

Training settings A learning rate $lr_1 = 0.1$ was used to train the model. The loss was optimized by a Gradient Descent optimizer [9] for 5 epochs.

5 Evaluation

All models and methods were evaluated on the development set by F1 metrics. F1 score was calculated by two additional metrics: precision and recall. However, in order to compute precision and recall, I first counted the number of true positives (TP), false positives (FP) and false negatives (FN). The rule for counting was basically summarized as the following code.

```
if gold[h] == pred[h] and gold[h] != CLASSES["_"]:
    TP += 1
elif gold[h] != pred[h] and pred[h] != CLASSES["_"]:
    FP += 1
elif gold[h] != pred[h] and pred[h] == CLASSES["_"]:
    FN += 1
```

After we have TP, FP and FN, precision, recall and F1 score are computed as below:

$$precision = \frac{TP}{TP+FP}$$

$$recall = \frac{TP}{TP+FN}$$

$$F1 = 2 \cdot \frac{precision \times recall}{precision + recall}$$

Based on this evaluation scheme, my best model had F1 score, precision and recall of

0.63, 0.67 and 0.59, respectively. Recall was lower than precision score. This meaned the model mislabelled more words as negatives (null label) resulting in a higher FN. Figure 2 and Figure 3 represents confusion matrix on the ten most popular classes, which give us a closer look on how the model performed.

Table 1: Summarisation of multiple methods.

Model	**Pred window size**	**Number of LSTM layers**	**Using POS**	**Using DEP**	**F1 score**
LSTMCell	1	1	No	No	0.558
LSTMCell	2	1	No	No	0.574
LSTMCell	3	1	No	No	0.54
LSTMCell	3	1	Yes	No	0.583
LSTMCell	3	1	No	Yes	0.558
LSTMCell	3	1	Yes	Yes	0.59
MultiRNNCell	1	1	No	No	0.55
MultiRNNCell	1	2	No	No	0.59
MultiRNNCell	1	3	No	No	0.55
MultiRNNCell	3	2	Yes	Yes	0.63
Bi-LSTM	1	1	No	No	0.57

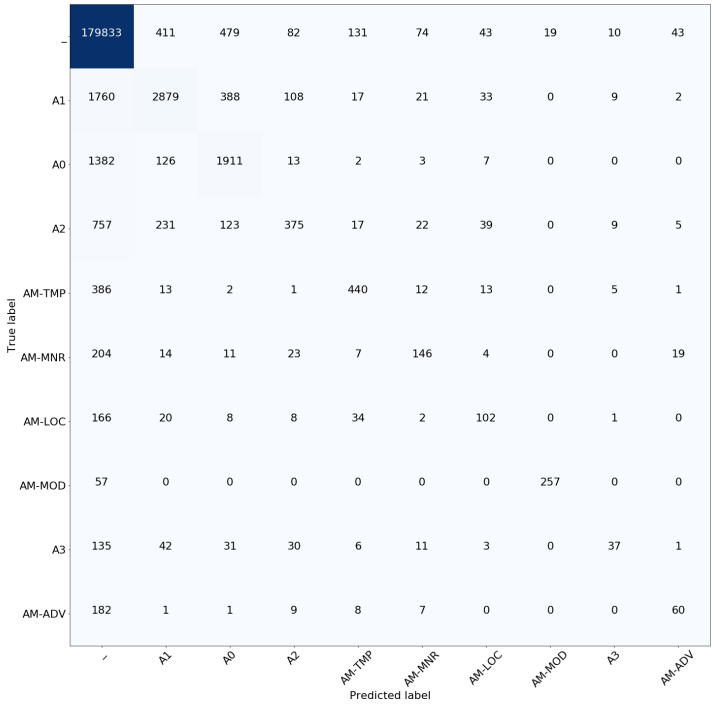

Figure 2: Confusion matrix of 10 most popular classes in the development set.

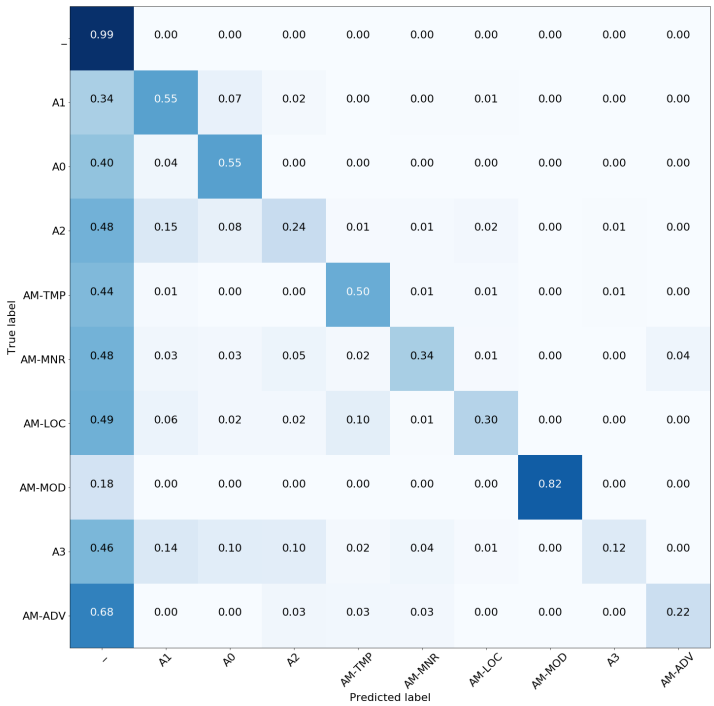

Figure 3: Normalized confusion matrix of 10 most popular classes in the development set.

References

[1] D. Gildea and D. Jurafsky, "Automatic labeling of semantic roles," *Association for Computational Linguistics*, 2002.

[2] J. Zhou and W. Xu, "End-to-end learning of semantic role labeling using recurrent neural networks," *Association for Computational Linguistics*, 2015.

[3] D. Marcheggiani, A. Frolov, and I. Titov, "A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling," *Association for Computational Linguistics*, 2017.

[4] L. He, K. Lee, M. Lewis, and L. Zettlemoyer, "Deep semantic role labeling: What works and whats next," 2018.

[5] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, (San Francisco, CA, USA), pp. 282–289, Morgan Kaufmann Publishers Inc., 2001.

[6] Z. Wang, T. Jiang, B. Chang, and Z. Sui, "Chinese semantic role labeling with bidirectional recurrent neural networks," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1626–1631, Association for Computational Linguistics, 2015.

[7] J. Hajič, M. Ciaramita, R. Johansson, D. Kawahara, M. A. Martí, L. Màrquez, A. Meyers, J. Nivre, S. Padó, J. Štěpánek, P. Straňák, M. Surdeanu, N. Xue, and Y. Zhang, "The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages," in *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, CoNLL '09, (Stroudsburg, PA, USA), pp. 1–18, Association for Computational Linguistics, 2009.

[8] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[9] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010* (Y. Lechevallier and G. Saporta, eds.), (Heidelberg), pp. 177–186, Physica-Verlag HD, 2010.