Semantic role labeling as translation:
A sequence to sequence encoder-decoder model with LSTMs

Max Boholm (gusbohom)

1. Introduction

Given the predicate–argument structure of sentences, semantic roles can be defined as the semantic interpretation of arguments relative the meaning of its predicate, for example, *Agent*, *Patient*, *Location*, *Experiencer*, and *Stimulus*. Note that there is no consensus on what set of semantic roles to use in analysis (for overviews, see, e.g., Jurafsky & Martin 2020). For an example, consider the following "interpretation" of the sentence *John broke the window with a rock*:

$$[\text{John}]_{\text{Agent}}\ [\text{broke}]_{\text{Pred.}}\ [\text{the window}]_{\text{Patient}}\ [\text{with a rock}]_{\text{Instrument}}$$

Semantic Role Labeling (SRL) is the process of automatically label semantic roles of the arguments of a sentence. Since semantic roles and related ideas are considered essential to semantic theory (e.g., Fillmore 1976), SRL is an interesting challenge for its own sake, as part of exploring if and how computers can learn and represent meaning. In addition, there are several downstream applications of SRL in Natural Language Processing (NLP), for example, information extraction, machine translations, and question-answering.

Traditionally, linear approaches (e.g., Support Vector Machines) with multiple features have often been used for SRL. In such approaches, syntactic parsing has been a key component of the SRL: semantic roles are assigned to the arguments of a sentence, by iterating over the arguments (sentence constituents, verb dependents) and based on their features, for example, predicate, phrase type, headword, the headword's Part of Speech, and voice, decide what semantic role they have (Gildea & Jurafsky 2002). More recently, like many other NLP tasks, state-of-the-art of SRL has been advanced by deep learning. While early attempts at SRL was made with Convolutional Neural Networks, Recurrent Neural Networks (RNNs) is most commonly seen SRL models today. Several varieties of these models can be distinguished based on the method of encoding sentences, the type of algorithm used for label classification, and aspects of the SRL problem addressed (e.g., segmentation, or overlapping role labeling). One main division can be drawn between models that use syntactic structure of sentences from parsing in the SRL pipeline, and models that do not (He, S. et al. 2018). The latter type of models is sometimes called "syntax agnostic" approaches to SRL (in contrast to "syntax aware" ones). One main reason for a syntax-agnostic approach is that parsing is itself a challenging task; any problem with the parsing step will affect the success of the semantic parsing (if dependent thereof). For example, He et al. 2017 present a model which reaches state-of-the-art performance on SRL by a rather simple model architecture of stacked LSTMs, which uses no other syntactic information of sentences other than which words being predicates.

2. The present work

The present paper reports work which aimed to develop and experiment with a syntax agnostic semantic role labeler, namely a sequence to sequence (seq2seq) encoder–decoder model, with LSTMs. The approach reported here treats SRL as a type of translation from a source sequence *s* (a sentence) to a target sequence *t* (the sequence of semantic roles assigned to *s*). While the proposed work was not expected to advance state-of-the art, it could hopefully provide some insights on the success and efficiency of seq2seq models for SRL. As we will see, the success of the approach is, unfortunately, very limited. *The **general problem** of the proposed model is that it converge to a behavior of always predicting the same label (the "outside", "O", which is the most common one in the training data), for every input.* This "strategy" of the model yields

rather high accuracy, but poor F1 on average. I have not been able to overcome this problem with experimenting with hyperparameters, suggesting that more thorough modification of the model and/or another preparation of the data is required for improved performance. More detailed discussion of the problem and suggestions for improvement are presented below.

3. Data

3.1. Dataset

Often SRL is trained and tested on the CoNLL-2005 dataset (Carreras & Màrquez 2005). These resources are not free of charge and was therefore not considered here. Instead, the present work uses a sub-sample of the ukWaC corpus (Ferraresi etal 2008), being automatically labeled with semantic roles (Sayeed et al 2018), using He, L. et al.'s (2018) model for SRL. The sample used for the present work was retrieved personally from Asad Sayeed, course coordinator and one of the collaborator of developing the dataset used (Sayeed et al 2018). In total, the dataset contains 21,187 sentences, whereof 208 are missing annotations of semantic roles, and therefore ignored. The annotations of semantic roles follow the labels of "proto-roles" used in PropBank (Kingsbury & Palmer 2002), for example, *Argument 0* (being agent-like), *Argument 1* (being patient-like), etc.

3.2 Another look at the classification problem and data preparation

At first glance, the classification problem addressed in this work can be represented as taking sentences as input to a model which outputs a sequence of semantic roles. However, this initial conception of the problem was revised, because sentences can (and often *do*) contain multiple predicates. For example, consider the sentence *I saw the sloth climbing*, which for the predicate *saw* can be interpreted as:

$[\text{I}]_{\text{Arg0}}$ [saw]$_{\text{Prd}}$ [the sloth climbing]$_{\text{Arg1}}$,

while for the predicate *climbing*, be interpreted as:

I saw [the sloth]$_{\text{Arg0}}$ [climbing]$_{\text{Prd}}$.

In order to simplify the classification problem (by, e.g., avoiding elements of the output sequence having combined semantic roles), the problem is here defined as taking *sentence–predicate pairs* as input to a model that predicts a sequence of role labels. In total, the data contain 76,757 such sentence-predicate pairs. Using Inside-Outside-Begining (IOB) tagging, the data specifically prepared for present purposes have three fields (of equal length): *sentence text*, *predicate vector*, and *semantic role sequence*. For the construed example *I saw the sloth climbing* there would be two instances in the data, since it has to predicates, as shown below:

Instance 1:

```
[i, saw, the, sloth, climbing],          (sentence)
[0, 1, 0, 0, 0],                         (predicate)
[B-ARG0, PRD, B-ARG1, I-ARG1, I-ARG1]    (semantic role sequence)
```

Instance 2:

```
[i, saw, the, sloth, climbing],          (sentence)
[0, 0, 0, 0, 1],                         (predicate)
[O, O, B-ARG0, I-ARG0, PRD]              (semantic role sequence)
```

Note that considering the predicate as part of the input of the model disqualifies it as being truly syntax agnostic, as the concept of predicate is a syntactic category. I ignore the predicate classification problem and use existing annotations for this.

4. Model

This work implements an RNN Encoder-Decoder Seq2Seq model (Sutskever, et al 2014). Figure 1 illustrates the model implemented in this work. The basic structure of this models is that a the words of a sentence is (in mini-batches) sent to an embedding layer (from size of vocabulary *V* to some dimension *E*). For each word of the sequence, its status as predicate, or not, is encoded as an additional dimension of the word, by concatenating the predicate vector to the [batch, sequence, embedding_dim] vector. The resulting vector is, in turn, input to the (stacked) LSTM encoder. The number of LSTM layers is a hyperparameter. The cell state and hidden state of the encoder (sometimes together referred to as the *context vector*) is feed into an LSTM decoder together with a start-of-sequence token to output the first token of the output sequence. Thereafter, the output sequence produced so far, the final cell and hidden state of the decoder are then feed into the model to output the next token, and so on.

In addition, the model implemented here uses teacher forcing in decoding, during training. In teacher forcing, the input to the next element of the sequence is either the previously predicted element of the model or the true label of the sequence at that index. The teacher forcing ratio (TFR), determines the ratio at which the true label or predicted label is used in sequence generation. A TFR of 1.0 means that every element is taken from the true sequence, while a TFR of 0.0 means that only predicted elements are used; a TFR of 0.5 means that true labels are used half of the time, etc.

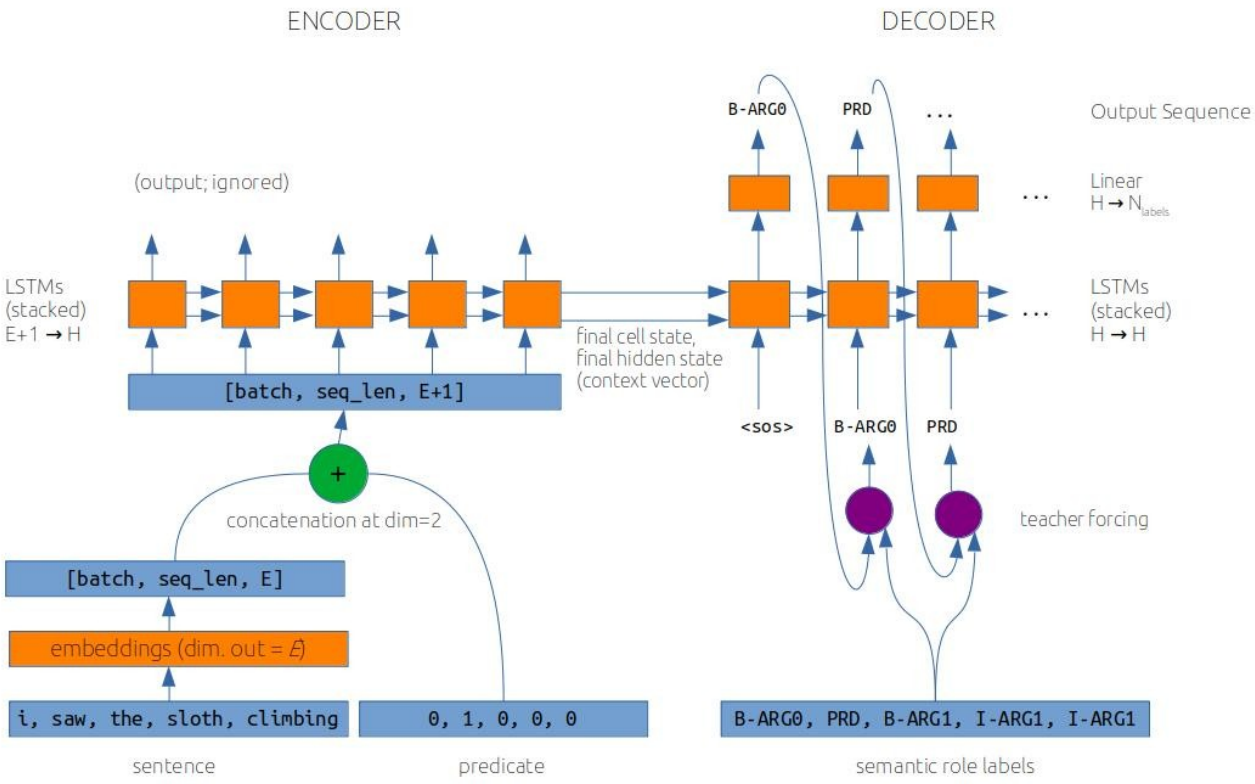

Figure 1: Encoder-Decoder Model

The model has been defined and trained in PyTorch. The algorithm Adam has been used has been used as optimizer (https://pytorch.org/docs/stable/generated/torch.optim.Adam.html) and Cross Entropy Loss is the loss function used (https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html). The cross entropy loss function expects an input of dimensions of [N, C] and a target dimension of [N], where C is the number of classes. Usually, N is the size of the minibatch, but note that here *N = size of mini-batch × sequence length*. As default, the index of the pad token is ignored when computing the loss of every batch. Exploding gradients has been a problem during training. To handle this, the PyTorch clipping gradient utility function has been used (https://pytorch.org/docs/stable/generated/torch.nn.utils.clip_grad_norm_.html), with *max norm of the gradients* = 1.

5. General testing and hyperparameters

The model has been trained with and evaluated with several different settings of hyperparameters, but none of the tested settings has overcome the general problem with this model (mentioned in sect. 2):

- Batch sizes of: 16, 32, 64, 128
- Epochs of: 10, 20, 40
- Encoder Embedding dimensions of: 256, 512
- Decoder Embeddings dimensions of: 43, 86
- The LSTM's hidden dimensions of: 256. 512, 1024
- Learning rates of: 0.01, 0.005, 0.001
- Number of layers of LSTM: 1, 2, 4
- Dropouts (in LSTM) of: 0.1, 0.2, 0.5
- TFRs of: 0, 0.2, 0.5

The combinations of these values for hyperparameters has not systematically been tested.

6. Evaluation, baseline and further experimenting

Several forms of evaluation have been considered. These are listed and discussed below. Given the general problem mentioned above, there is limited value to discuss the performance of models given particular hyperparameters in any detail. The proposed model obviously performs in similarily and poorly for *all* hyperparameters considered. Below, evaluation measures are discussed nevertheless.

6.1. Measures of accuracy and F1 and baseline

Accuracy is the number of correct predictions divided by all predictions. Accuracy can be calculated locally for a sentence or for multiple (all) sentences as a "pooled" sequence (defined here as *Pooled Accuracy*). Another generalization of the accuracy of the model is the mean of sentence-local accuracy measures (*Mean Accuracy*).

F1 is a balanced measure considering the relation between a model's precision and its recall. It can be defined as:

$$F1 = \frac{TP}{TP + 0.5 \times (FP + FN)}$$

where *TP* stands for true positive, *FP* for false positive and *FN* for false negative. As such, F1 presupposes a binary classification. However, it can be used for multi-class classification, with a F1 score for every label: $F1_{Label1}$, $F1_{Label2}$, etc. To generalize from this set of F1 scores, we can (among other things) consider the mean of $F1_{Label1}$, …, $F1_{LabelN}$, i.e. *F1 Macro*, or the weighed mean of $F1_{Label1}$, …, $F1_{LabelN}$, i.e. the mean of $F1_{Label1}$, …, $F1_{LabelN}$, proportionally adjusted to the frequency of Label 1, …, Label N. This is here called *F1 Weigthed*

(see https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html). Like accuracy, F1 Macro and F1 Weigthed can be calculated once on the test data as a pooled sequences (*Pooled F1 Macro* and *Pooled F1 Weighted*), or as the mean of each sentence considered first locally (*Mean F1 Macro* and *Mean F1 Weighted*).

With some configurations of hyperparameters, the model developed here has a Pooled Accuracy of 0.75, but this only reflects the behavior of always predicting the most common label of the train and test data, namely "O". In 75% of the training data as well as the test data, the label is "O", suggesting that the definition of the classification problem, and consequently the preparation of data, leaves room for improvement. The data is very unbalanced.

Scores for F1, which better reflect the desired performance of the model, is substantially lower than for accuracy. Pooled F1 Macro seldom reaches scores over 0.03. For the "best" models Mean F1 Macro is around 0.25, but this again reflects the problem already discussed. For many instances, F1 Macro of an-always-predict-O model will be quite high due to structure of the data.

6.2 Confusion matrix

Confusion matrices confirms the general problem already discussed. No matter the configuration of hyperparameters, the model strongly converges towards predicting "O" in every instance of prediction.

6.3 Qualitative evaluation

Again, qualitative inspection of labeled sentences, reveals the same pattern: "O" is the label predicted throughout.

6.4 Length

As length of sequences has been identified as a problem for Seq2Seq models in previous work, length of sentences was considered as part of evaluation. The accuracy of sentences is positively correlated with length (in general, long sentences tend to correspond with semantic role sequences with many "outside" labels). The more relevant F1 scores are not positively correlated with sequence length.

6.5 Ignorant models

In order to better understand the models performance in relation to the general problem, *ignorant models* has been considered, i.e., models which during training ignore the outside label "O" when computing the loss (since the default value of the training function is to ignore the pad token, `"<pad>"`, this parameters (`ignore_label`) has been set to `"O"` when training the ignorant models). Below is the comparison of a model trained with 4 layers of LSTMs and its ignorant counterpart;

Table 1. *Performance of Models and their ignorant counterparts*

	L4Model	L4Model Ignorant
Pooled Accuracy	0.745	0.107
Pooled F1_macro	0.023	0.028
Pooled F1_weighted	0.852	0.174
Mean Accuracy	0.644	0.149
Mean F1_macro	0.256	0.203
Mean F1_weighted	0.731	0.195
Correlation sentence length and accuracy	0.449	-0.297
Correlation sentence length and F1_macro	-0.0227	-0.056
Correlation sentence length and F1_weighted	0.405	-0.286

Comment: other hyperparameters being: *epochs* = 10, *learning rate* = 0.001, *dropout* = 0.5, TFR = 0.5, *batch size* = 32, *embedding dimension encoder* = 256, *embedding dimension decoder* = 43, *hidden dimension of LSTM layers* = 512.

It is clear from Table 1, that accuracy is worse for the ignorant model, while its performance in terms of F1 Macro is slightly (but not likely significantly) better. The difference between the models becomes clearer when qualitatively comparing output of the models. In terms of F1 Macro, one of the best sentences for the ignorant model is (sentence: *Next stop was Crosshouse Hospital in Kilmarnock, followed by Ayr Hospital.*):

```
<sos>/B-ARG1 next/B-PRD stop/B-PRD WAS/I-ARG1 crosshouse/I-ARG1
hospital/I-ARG1 in/I-ARG1 kilmarnock/I-ARG1 ,/I-ARG1 followed/I-ARG1 by/I-
ARG1 ayr/I-ARG1 hospital/I-ARG1 ./<eos>
```

which remotely looks like the desired output. The same sentence analyzed by the non-ignorant model is:

```
<sos>/O next/O stop/O WAS/O crosshouse/O hospital/O in/O kilmarnock/O ,/O
followed/O by/O ayr/O hospital/O ./<eos>
```

Concluding remarks

The present work has not succeeded in developing a successful model for SRL. Possible improvements of the existing Seq2Seq encoder-decoder architecture include: implementing attention to help the decoder focus on the relevant aspects of the encoded sentence. Another possibility is to use pre-trained word-embeddings. An alternative to the Seq2Seq encoder-decoder architecture would be to implement a model with a stacked LSTM encoder followed by some other classification algorithm (e.g., He L et al. 2017, 2018). Perhaps, there is little hope in successfully implementing a Seq2Seq encoder-decoder architecture with LSTMs for SRL, but more work is needed before drawing that conclusion. In particular, the proposed model architecture did not perform well on the data used – or perhaps rather due to *how* the data was used. Further work should thoroughly re-consider the link between model and data. The overflow of the outside label (“O”) in the data set as defined here has resulted in massively imbalanced data, in turn, learning the model to always predict “O”, with no discriminatory power for other, much less common, labels. Given these comments, a better balanced dataset and re-thinking of the machine learning problem, including the preparation of the data and the classification task, are needed for further experimenting with Seq2Seq encoder decoder networks for SRL.

References

Carreras, X., & Màrquez, L. (2005). Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the ninth conference on computational natural language learning (CoNLL-2005)*, 152-164.

Ferraresi, A., Zanchetta, E., Baroni, M., and Bernardini, S. (2008). Introducing and evaluating ukWaC, a very large web-derived corpus of english. *Proceedings of the 4th Web as Corpus Workshop (WAC-4) Can we beat Google*, 47–54.

Fillmore, C. J. (1976). Frame semantics and the nature of language. *Annals of the New York Academy of Sciences: Conference on the origin and development of language and speech,* 280: 20-32.

Gildea, D., & Jurafsky, D. (2002). Automatic labeling of semantic roles. *Computational linguistics*, *28*(3): 245-288.

He, L., Lee, K., Lewis, M., & Zettlemoyer, L. (2017). Deep semantic role labeling: What works and what’s next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 473-483.

He, L., Lee, K., Levy, O., & Zettlemoyer, L. (2018). Jointly predicting predicates and arguments in neural semantic role labeling. *arXiv preprint arXiv:1805.04787*.

He, S., Li, Z., Zhao, H., & Bai, H. (2018). Syntax for semantic role labeling, to be, or not to be. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2061-2071.

Jurafsky, D., & Martin, J. H. (2020). *Speech and Language Processing*. Pearson Prentice Hall.

Kingsbury, P. R., & Palmer, M. (2002). From TreeBank to PropBank. *LREC*, 1989-1993.

Sayeed, A., Shkadzko, P., & Demberg, V. (2018). Rollenwechsel-English: a large-scale semantic role corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104-3112.