Semantic Role Labeling

Ahmed ElSheikh - 1873337

1 Introduction

Semantic Role Labeling (SRL) is the process of assigning labels to words in a given sentence indicating their semantic roles. It provides the answer to ”Who did what to whom at where?”
e.g. The police officer detained the criminal at the crime scene
This report aims to show approaches and results obtained for the tasks {arguments identification, and classification}, associated with a predicate (verb). Formally, given a sentence with T tokens, for every token $x \in T$, an argument is identified and assigned to one of 35 possible argument tags.

2 Dataset Preparation

No specific preprocessing was involved with given data. Training dataset was created from ”lemmas” of the sentences batched and fed to the model padded per batch, to avoid any loss of information in truncated sentences. The tokens were in their basic form (lemmas) which makes it easier for the model to learn, due to the reduced vocabulary size e.g. instead of having ”eats”, ”eating”, ”ate”, ”eaten” the vocabulary will only have ”eat”. Further insights can be found in Table 1

3 Network Architectures

3.1 Base Model

Given that SRL task is sequence-to-sequence classification task, hence RNN model was used, an LSTM (Long-Short-Term-Memory) captures long range dependencies, without being biased towards recent inputs. a variant of it was used, Bidirectional LSTM (BiLSTM), which operates in both directions (forward, backward) recording information of both the past and future, allowing model to have a better understanding of the inputs’ context. Or stacking LSTMs allowing our model greater complexity, added to, being able to create a hierarchical feature representation of the input data, resulting in an improved model generalization capabilities.

3.2 Variants

Different approaches were employed aiming to improve model’s performace.

3.2.1 Pretrained Word Embeddings

Instead of training an Embeddings layer from scratch, it was decided to initialize the embedding layer with weights that have been already precomputed on larger corpora, which will allow the model to pick up semantic signals, making it easier for the model to train, and enhance its performance. Fasttext was used with 300D word embeddings dimension, not only for the previously mentioned reasons, but also because its ability to deal with OOVs, which will be broken into n-grams and fetch its embeddings i.e. decomposes a word into n-gram characters.

3.2.2 Multi Input Model

In order to inform the model with the predicate position in the sentence, allowing the model to correctly identify & classify the arguments, every sentence was duplicated n times, where n is the number of predicates per sentence, each predicate was replaced by its VerbAtlas group, while other tokens were replaced by a null-tag ”_”
For example ”Marry ate the apples, while going to school.”, “ate“ lemmatized form is “eat“ which belongs to “EAT_BITE“ VerbAtlas group, same applies for “going“ which belongs to “Leave-Depart-Run Away“ group. So our sentence will be represented by 2 sentences “_ EAT_BITE _ _ _ _ _ _ “ and “_ _ _ _ _ LEAVE_DEPART_Run_Away _ _ “.

3.2.3 Incorporate POS tags

Semantic Role labeling can be improved incorporating POS tags. Part-Of-Speech (POS) tagging is an auxiliary task which often helps improving models' performance in multiple downstream NLP tasks. This hypothesis is proved in Resutls section.

4 Experiments

This section details all the experiments conducted in our effort to study the effect of combining all the aforementioned concepts and intuitions.
All models were trained using Adam optimizer with learning rate of 0.001 minimizing Categorical Cross Entropy loss function. All experiments were conducted in batches of 128 (input was padded per batch) using Colab GPU. F1 Score was used as a metric measure models' performance. Gradient clipping was used to avoid the problem of exploding gradients that LSTMs might face during training which reduces their performance. Models were set to train for 50 epochs each but to prevent overfitting, Early Stopping callback which monitors val_loss whilst model training, as well as, Model Checkpoints callbacks to save the best model weights. Training losses were logged using TensorBoard.

Test 1: First expirement done was feeding model inputs with every predicate in the sentence is replaced with $< PREDICATE >$, duplicating sentence n times, n is num of predicates

Test 2: Adding dropout layers between embeddings layer and LSTM layer, which prevents model dependency on certain tokens while learning; as embedding dropout drop words randomly from input sequences while training (6)

Test 3: Using Fasttext instead of training Embeddings layer from scratch, aiming to yield better results.

Test 4: Increasing model's complexity, by stacking another BiLSTM, which might increase model's performance as it allow the model greater complexity, at the cost of increased overfitting risk.

Test 5: Instead of having very abstract $< PREDICATE >$ flag in input sequences, replicate input sequences but with null tags instead of all tokens but the predicates, and replace those predicates with their verb atlas group, and utilize another embeddings layer solely for predicates(verbs) in the replicated input sequences, refer to section 3.2.2

Test 6: POS tags provide a linguistic signal on how a word is used in the sentence scope, which might facilitate the process of learning

5 Results

First approach to identify and classify arguments was to duplicate sentences as per number of predicates -in the sentence, and replace each predicate with a flag denoting its position. This approach yielded lowest results due to its very high abstract level, which led to high loss of information, resulting in a very low performing model. The previous approach was replaced with having another sequence of predicates, with Verb Atlas group replacing its predicate (total of 458 predicate groups) and null-tags anywhere else, this sequence had its own embeddings layer, then its output was concatenated to words embeddings layer before being fed to BiLSTM, this allowed the model to learn a richer representation of input sequences. The model converged to a better solution in comparison with other experiments. Refer to test 1 & test 5 table 4. for loss plot of exp 1 refer to fig 2 and for exp 5 loss plot refer to fig 3

To encourage the model to depend on different words of the same input sequences, dropout training was deployed which yielded better results, as expected; as the model gets to learn without depending on certain tokens. Refer to test 2 table 4.

Using pretrained word embeddings[1] helped the model pick up more semantic signals, improving overall model's generalization abilities. Same intuition with incorporating of POS tags providing the model with more linguistic information about how word can be utilized in a given sentence. Both increased overall model's performance as the model takes into account the syntactic and semantic structure of the English sentences, check Test 3 & test 6 table 4.

Stacking of BiLSTM helped model achieve better results, due to, its better ability to represent the given sequences, as well as, allowing model a better chance for hierarchical re-composition of abstract features learned. However, model was more at risk of overfitting and model started to overfit before all other models. Added to, number of model's parameters increased, requiring more time per epoch. Refer to figure 2

[1] This was integrated with dropout training

A Data Insights

Dataset	Size
Training Set	$89K$
Validation Set	$3K$
Testing Set	$4K$

Table 1: Train-Val-Test split used in experiments

Vocabulary	Size
Input	27,349
Predicates	458
POS	50
Tags	35

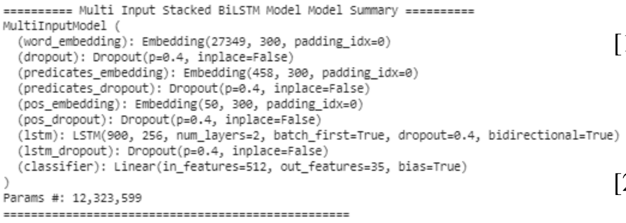
Table 2: Vocabularies' Size

B Hyperparameters

Hyper-parameters	Value
Hidden Neurons	256
Bidirectional	True
Num of Layer	2
Embeddings Dim	300
PreTrained Embeddings	Fasttext
Dropout	0.4
Batch Size	128

Table 3: Model Hyper-parameters

C Summary



```
========== Multi Input Stacked BiLSTM Model Model Summary ==========
MultiInputModel (
  (word_embedding): Embedding(27349, 300, padding_idx=0)
  (dropout): Dropout(p=0.4, inplace=False)
  (predicates_embedding): Embedding(458, 300, padding_idx=0)
  (predicates_dropout): Dropout(p=0.4, inplace=False)
  (pos_embedding): Embedding(50, 300, padding_idx=0)
  (pos_dropout): Dropout(p=0.4, inplace=False)
  (lstm): LSTM(900, 256, num_layers=2, batch_first=True, dropout=0.4, bidirectional=True)
  (lstm_dropout): Dropout(p=0.4, inplace=False)
  (classifier): Linear(in_features=512, out_features=35, bias=True)
)
Params #: 12,323,599
==================================================
```

Figure 1: Best Model's Architecture

Word embeddings are concatenated with Predicate & POS embeddings before being passed to Stacked BiLSTM

D Performance

Test #	F1 Score	
_	Arg Identification	Arg Classification
01	81.55%	61.47%
02	84.55%	64.58%
03	88.64%	66.06%
04	86.64%	67.22%
05	89.67%	84.95%
06	91.55%	87.07%

Table 4: Experiments' Results

Experiments' were implemented in an incremental approach, with experiment 1 approach is then integrated with exp 2 approach. e.g. Dropout training (exp2) was used when expirementing use of Fasttext embeddings (exp3)

E Losses

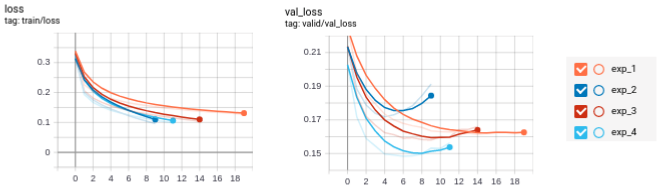


Figure 2: Experiments' Losses

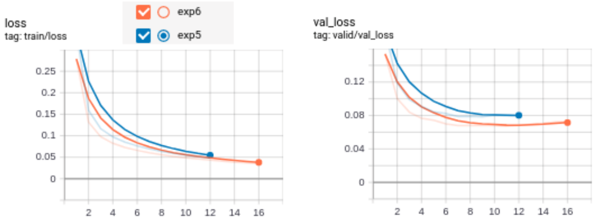


Figure 3: Experiments' Losses II

References

[1] VerbAtlas: a Novel Large-Scale Verbal Semantic Resource and Its Application to Semantic Role Labeling
Andrea Di Fabio, Simone Conia, Roberto Navigli

[2] Framewise phoneme classification with bidirectional lstm and other neural network architectures.
Alex Graves and Jürgen Schmidhuber.

[3] Adam: A method for stochastic optimization.
Diederik P Kingma and Jimmy Ba. 2014.

[4] Neural sequence learning models for word sense disambiguation

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017.

[5] Fasttext Enriching Word Vectors with Subword Information P. Bojanowski, E. Grave, A. Joulin, T. Mikolov

[6] A Theoretically Grounded Application of Dropout in Recurrent Neural Networks
Yarin Gal, Zoubin Ghahramani

[7] Deep Semantic Role Labeling: What Works and What's Next
Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer