CS 272 Homework 3

Junayed Naushad
Univ. of California, Irvine
jnaushad@uci.edu

Abstract

In this paper I explore sequence tagging, specifically named entity recognition (NER) and part of speech (POS) tagging for Twitter data. I compare different configurations of neural tagger architectures to determine what works best for the aforementioned tasks.

1 Improving Simple Tagger

The two main components of a neural tagger are the embedder and the encoder. The embedder creates a dense vector representation of the input token and the encoder uses the embedding to generate another representation which can be used to determine the output label.

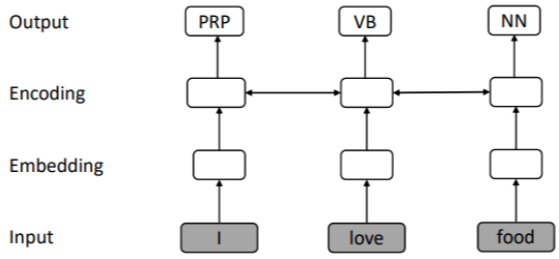

Figure 1: Neural tagger architecture

In its most basic form, the neural tagger uses a randomly initialized 50-dimensional embedding with no encoder. We can improve upon this configuration by using pre-trained embeddings (e.g. Glove) and a recurrent network (e.g. bidirectional LSTM) for the encoder. The following subsections present the results for the different configurations; a batch size of 32, 20 epochs for NER and 40 epochs for POS, and the Adam optimizer with a learning rate of 0.001 were used for all tests.

1.1 Results for NER

Model	Dev Accuracy
Simple Tagger	0.9444
+ Glove	0.9347
+ RNN	0.9450
+ GRU	0.9483
+ LSTM	0.9465
+ Glove + RNN	**0.9544**
+ Glove + GRU	0.9537
+ Glove + LSTM	0.9505

Table 1: Glove embedding is 50-dim and RNN, GRU, LSTM are all bidirectional and have 1 layer with 25-dim hidden state

The results above show that adding just the Glove embeddings does not improve the simple tagger but adding just an encoder does improve performance and adding both the embeddings and an encoder yields the best results. We can try to improve upon these results by stacking layers and adding dropout.

Model	2 Layers	3 Layers
+ Glove + RNN	0.9537	0.9537
+ Glove + GRU	0.9533	0.9471
+ Glove + LSTM	0.9469	0.9461

Table 2: Dev set accuracy using multiple layers and dropout of 0.1

We can see that adding additional layers with dropout does not improve the performance of the model, in fact, as we add more layers the dev accuracy of the model drops.

Model	Test Accuracy
Simple Tagger	0.8985
+ Glove + RNN	**0.9064**
+ Glove + GRU	0.9055
+ Glove + LSTM	0.9032

Table 3: Results for the test set using the Glove embeddings and single layer recurrent networks

The results on the test set show that using Glove embeddings with an RNN yields the best performance. We can examine individual label accuracies to determine which labels improved the most.

Label	Simple	RNN	GRU	LSTM
B-company	.0805	**.3205**	.2786	.2012
B-geo-loc	.1156	**.4705**	.4649	.4354
B-person	.0832	**.5468**	.5177	.4964
I-person	.0100	**.3746**	.2341	.0167
O	**.9900**	.9827	.9830	.9854

Table 4: Showing label accuracies that had the most improvement on the test set; RNN, GRU, and LSTM are using Glove embeddings

The overall accuracy clearly does not tell the whole story since there is a significant improvement over the simple tagger across most labels. The simple tagger performs best for the "O" label, which is used for tokens that are not named entities, and since this is the most common label it causes the simple taggers overall accuracy to appear higher than it really is. Using a bidirectional RNN, GRU, or LSTM as the encoder provides a significant improvement over the simple tagger because it incorporates context. Context is crucial for NER tagging because the label for a token is ambiguous when previous and future tokens are not taken into consideration. The results show a significant improvement for the B-company and B-person labels which makes sense because it is very difficult to differentiate between the name of a company and the name of a person without context.

1.2 Results for POS

Model	Dev Accuracy
Simple Tagger	0.6414
+ Glove	0.7835
+ RNN	0.7907
+ GRU	0.7830
+ LSTM	0.7854
+ Glove + RNN	0.8474
+ Glove + GRU	0.8421
+ Glove + LSTM	**0.8479**

Table 5: Dev set results for POS tagging

Unlike for NER tagging, the Glove embedding by itself provides a significant improvement over the simple tagger for POS tagging. The combination of Glove embedding and bidirectional LSTM encoder yields the best performance on the dev set.

Model	2 Layers	3 Layers
+ Glove + RNN	0.8364	0.8326
+ Glove + GRU	0.8374	0.8393
+ Glove + LSTM	0.8416	0.8116

Table 6: Dev set accuracy using multiple layers and dropout of 0.1

Once again, adding layers and dropout only decreases the performance of the models.

Model	Test Accuracy
Simple Tagger	0.6329
+ Glove + RNN	0.8541
+ Glove + GRU	0.8499
+ Glove + LSTM	**0.8557**

Table 7: Results for the test set using the Glove embeddings and single layer recurrent networks

The neural tagger with Glove embeddings and bidirectional LSTM has the highest test set accuracy. For NER tagging the RNN performed best but for POS tagging the LSTM performs marginally better than the RNN. This may be attributed to the fact that POS tagging is a more challenging task and thus benefits from having a more complex encoder like the LSTM which offers finer control over the amount of context to use.

Label	Simple	RNN	GRU	LSTM
ADJ	.4078	**.6702**	.6915	.6312
NOUN	.3616	.7877	.7597	**.7968**
NUM	.2609	**.6087**	.5978	**.6087**
VERB	.6759	.8586	.8655	**.8701**
X	.2483	.8207	.8190	**.8379**

Table 8: Label accuracies that had the most improvement on the test set

We notice that adding an encoder provides a significant improvement over the simple tagger across most of the POS labels. Having input context greatly reduces the ambiguity of labelling tokens.

2 Viterbi Implementation

The Viterbi algorithm is a dynamic program used to speed up CRF decoding. The algorithm takes as input emission, transition, start, and end scores and returns the best sequence of labels along with its score. My implementation uses 2 data structures: one array to keep track of sequence scores and another array to keep track of back pointers. As the algorithm iterates through all tokens and labels, it updates scores and back pointers like this:

$$v[t, l] = \max_{l'} v[t-1, l'] * em[t, l'] * tr[l', l]$$

$$bp[t, l] = \operatorname*{argmax}_{l'} v[t-1, l'] * em[t, l'] * tr[l', l]$$

where v is a 2d-array storing the sequence scores, bp is a 2d-array storing the back pointers, t iterates through the tokens, and l iterates through the labels.

3 Improve & Compare CRF to Simple Tagger

Adding a CRF to our neural tagger architecture will allow the model to make predictions based on both the input context and label context. Since the stacked encoders did not provide any additional performance to the simple tagger and due to time/computation constraints, I did not test them with the CRF tagger.

3.1 Results for NER

Model	Dev Acc	Test Acc
CRF	0.9489	0.9032
+ Glove	0.9446	0.8848
+ RNN	0.9415	0.8819
+ GRU	0.9449	0.8847
+ LSTM	0.9475	0.9032
+ Glove + RNN	**0.9550**	0.9048
+ Glove + GRU	**0.9550**	**0.9057**
+ Glove + LSTM	0.9503	0.8977

Table 9: Using 50-dim embedding, single layer network with 25-dim hidden unit, and same hyperparameters as the simple tagger experiments

The CRF tagger by itself (0.9032) has higher test set accuracy than the default simple tagger (0.8985) but does not perform as well as the best simple tagger (0.9064). The best variant of the CRF tagger (0.9057) also does not perform as well as the best simple tagger.

Label	CRF	RNN	GRU	LSTM
B-company	.0741	**.3237**	.2931	.2593
B-geo-loc	.0737	.4512	**.4649**	.3946
B-person	.0832	**.5759**	.5468	.5301
I-person	.0100	**.4247**	.4047	.3645
O	**.9962**	.9796	.9808	.9762

Table 10: CRF is the default CRF tagger, showing label accuracies that had the most improvement on the test set

If we compare the individual label accuracies between the simple tagger and the CRF tagger for the same configurations, we notice that the CRF taggers perform better on most of the labels. For example, the Glove + RNN configuration of the CRF tagger performs better on "B-company", "B-person", and "I-person" compared to its corresponding simple tagger variant but has a marginally lower overall test accuracy. The CRF taggers show a significant improvement for the "I-person" label and "I" labels in general because "I" tokens fall inside a span and thus are easier to predict when you know the surrounding labels.

3.2 Results for POS

Model	Dev Acc	Test Acc
CRF	0.7463	0.7427
+ Glove	0.7973	0.8058
+ RNN	0.7783	0.7877
+ GRU	0.7983	0.7946
+ LSTM	0.7759	0.7780
+ Glove + RNN	0.8488	**0.8547**
+ Glove + GRU	0.8379	0.8506
+ Glove + LSTM	**0.8522**	0.8541

Table 11

The default CRF tagger (0.7427) has significantly higher test set accuracy than the default simple tagger (0.6329) but it does not perform as well as the best simple tagger (0.8557). The best simple tagger still outperforms the best CRF tagger (0.8547).

Label	CRF	RNN	GRU	LSTM
ADJ	0.3830	0.6596	**0.6986**	0.6277
NOUN	0.6914	**0.7915**	0.7627	0.7900
NUM	0.25	0.6087	**0.6304**	0.5978
VERB	0.7057	0.8609	0.8678	**0.8736**
X	0.5310	0.8259	0.8172	**0.8362**

Table 12: Label accuracies that had the most improvement on the test set

We saw that for NER tagging although the best CRF tagger did not have higher overall accuracy than the best simple tagger, it still performed better on most of the individual categories. This is not the case for POS tagging; the CRF taggers and simple taggers have very similar performance across different labels. I believe this is the case because POS tagging relies heavily on input context, hence the large improvement in performance found by adding an encoder, but it does not depend as much on label context.

4 Statement of Collaboration

I did not collaborate with anyone on this assignment nor did I use any outside resources.