Multilingual Natural Language Inference using Deep Learning Techniques: A Recurrent Neural Network Approach with Long Short-Term Memory Units

Arita Berisha | University of Prishtina

Introduction

The exponential growth of data and the increasing need for understanding and processing different languages have led to the advancement of Natural Language Processing (NLP). This report explores the utilization of Deep Learning techniques, specifically, Recurrent Neural Networks (RNNs) with Long Short-Term Memory (LSTM) units for the task of Natural Language Inference (NLI) across multiple languages.

Problem Description

Natural Language Inference is a fundamental task in NLP that involves determining the relationship between a pair of sentences. This relationship could be entailment, contradiction, or neutrality. When dealing with multiple languages (Fig.1), the complexity of the task increases, as it involves understanding context, semantics, and syntax across different languages. The primary challenge tackled in this assignment is to design a model capable of performing NLI in multiple languages with high accuracy. The dataset consists of 12121 sentences in total.

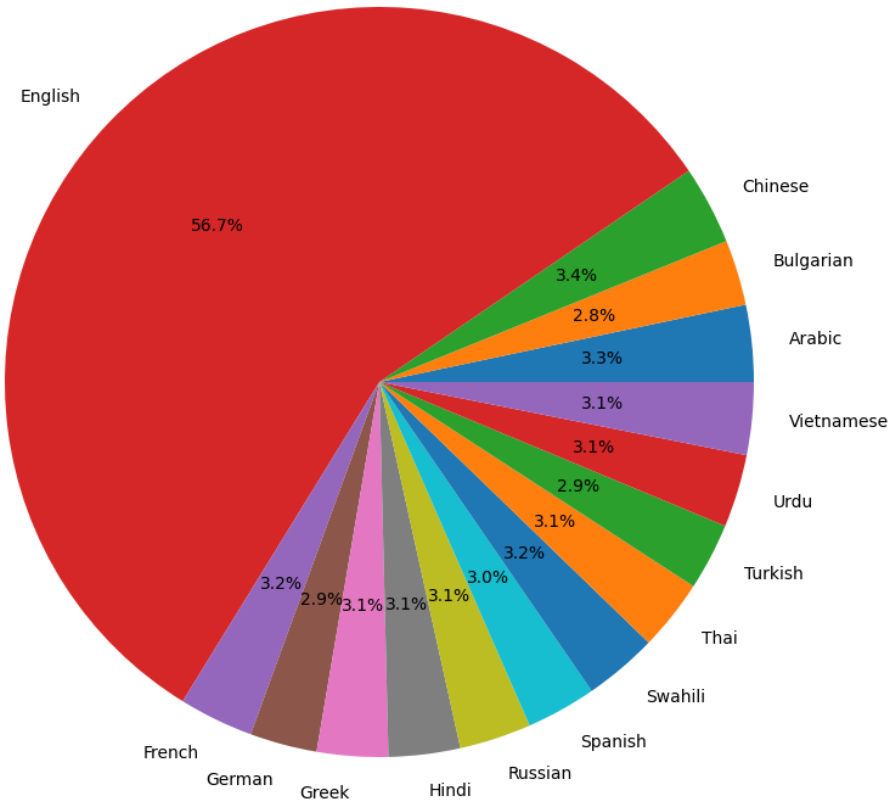

Figure. 1. Multilingual Distribution

Recurrent Neural Networks and LSTM

RNNs are a type of artificial neural network designed to recognize patterns in sequences of data, such as text, genomes, handwriting, or the spoken word. However, vanilla RNNs suffer from problems like vanishing and exploding gradients, making them ineffective in learning long-range dependencies. LSTM, a variant of RNN, is designed to remember information for long periods, which makes it ideal for such tasks. We employ LSTMs for our task due to their capability to deal with sequence data effectively.

Model Building and Parameters

Our model architecture is built using a pre-trained BERT model for encoding sentences and an LSTM layer for sequence learning. Parameters tuned during the training process include:

1. **lstm_dim**: This parameter refers to the dimensionality of the output space in the LSTM layer, i.e., the number of neurons or units. Each LSTM unit contributes to the model's learning ability by capturing and retaining information over arbitrary lengths of time in their internal state. Larger values provide the LSTM the capacity to learn more complex patterns and long-term dependencies between elements in the sequence data. However, they also increase the computational requirements and may lead to overfitting if the model complexity surpasses what is warranted by the training data.
2. **learning_rate**: This is a crucial hyperparameter for any gradient-based optimization algorithm, which includes those typically used in training neural networks. The learning rate determines the size of the steps the optimizer takes down the gradient of the loss function towards a (local) minimum. A smaller learning rate may cause the training process to converge slowly, but it can provide a more refined adjustment of the weights and potentially reach a better minima. Conversely, a larger learning rate may expedite convergence but also cause it to overshoot the optimal point.
3. **dropout_rate**: Dropout is a type of regularization technique that aims to prevent overfitting in neural networks. During training, a neuron is "dropped-out" or deactivated with a probability equal to the dropout rate, reducing the interdependent learning between neurons. This effectively creates a version of the model that is less complex and thus less likely to overfit to the training data. The dropout rate determines the proportion of neurons to drop, with higher values increasing the amount of regularization applied.
4. **clipnorm**: Gradient clipping is a technique to prevent exploding gradients, a problem where large gradient values result in an excessively large step in the gradient descent optimization process, potentially causing instability during training. The clipnorm parameter is a threshold value for the magnitude of the

gradients. If a gradient's magnitude exceeds this threshold, it's rescaled to have a magnitude equal to the clipnorm value.

5. **optimizer_type**: This defines the specific algorithm used to update the model's weights based on the calculated gradients. Different optimizers have different ways of adjusting the weights, with algorithms like 'Adam' and 'RMSProp' commonly used due to their adaptive learning rate properties.
6. **regularizer_type**: This defines the type of regularization applied to the model's weights. Regularization helps prevent overfitting by adding a penalty term to the loss function, which increases as the magnitude of the weights increases. 'L1' and 'L2' are types of regularization, referring to the first and second order norms, respectively.
7. **regularization_rate**: This is the scalar multiplier for the regularization term in the loss function. Higher values apply a more substantial penalty to larger weights, encouraging the model to learn smaller, more distributed weight values and thus reducing overfitting.
8. **batch_size**: The batch size is the number of examples the model is shown to make predictions and update the weights per iteration of training. Larger batch sizes allow the model to complete training faster as it processes more data at once, but it may also result in less accurate gradient estimates.
9. **epochs**: An epoch is one complete pass through the entire training dataset. The number of epochs is the number of times the learning algorithm will work through the entire training dataset. Too few epochs can result in underfitting of the model, whereas too many can lead to overfitting.

Results

The models were trained and tested using a systematic approach. Several configurations of the model parameters were tried, and the corresponding results were recorded (Table. 1).

lstm_dim	learning_rate	dropout_rate	clipnorm	optimizer_type	regularizer_type	regularization_rate	batch_size	epochs	Result
64	0.001	0.1	1.0	'Adam'	'L1'	0.01	32	5	38.7%
128	0.002	0.2	1.0	'RMSprop'	'L2'	0.02	64	10	42.1%
256	0.001	0.2	1.0	'Adam'	'L1'	0.01	64	7	44.9%
128	0.005	0.2	1.0	'Adam'	'L2'	0.05	32	10	40.6%
64	0.001	0.3	1.0	'RMSprop'	None	0	64	5	34.8%
128	0.002	0.2	0.5	'Adam'	'L1'	0.02	32	7	42.4%
512	0.0005	0.2	1.0	'Adam'	'L1'	0.01	32	3	33.2%
256	0.0001	0.3	1.0	'RMSprop'	'L2'	0.02	64	5	31.4%

512	0.0003	0.1	1.0	'Adam'	'L2'	0.01	32	8	39.9%
256	0.0002	0.2	0.5	'RMSprop'	'L1'	0.01	64	7	35.8%
64	0.0005	0.3	0.5	'Adam'	'L2'	0.02	32	5	32.3%

Table 1. Results

Conclusion

This report presents a deep learning approach to tackle the challenge of Natural Language Inference in multiple languages. We showcased the use of Recurrent Neural Networks with LSTM units and fine-tuned various parameters to optimize the performance. Despite the complexity of the task, promising results were achieved, validating the effectiveness of LSTM networks in handling sequence data and their potential in multilingual NLP tasks. Future work could explore other architectures and training strategies to further improve performance.