Classification: Was it Machine Translated?

Amol Sharma, Akshay Chalana

15 March 2019

For this project, our datasets consist of sentences in English and German, and a label telling us whether it was machine-translated or human translated. Our task is to be able to predict given two sentences whether they were machine translated. We decided to use a biattentive classification model to solve this problem.

1 Introduction: Model Selection

For this task, we are essentially building a binary classifier. The complication is that we are classifying based on not one, but two input sentences, which happen to be related (same sentence in different languages). Therefore, our first thought was to use a classifier similar to the academic paper example in the AllenNLP docs (AllenNLP). However, while that example looks at title and abstract to choose a label, we needed to look at the connection between the German and English sentences to choose our label. Therefore, we did a little research and came across a Biattentive Classification Network (McCann et al.), which is what we chose to use.

2 Method

We tested 3 separate models after learning about the BCN. Our final optimal solution was a BCN using BERT Cased Multilingual Encodings where our input was a concatenation of the source and candidate sentences (concatenated by a modified dataset reader). A variation we tried was changing the encoder and using character embeddings alongside BERT. The second model was a modification of the BCN which instead utilized 2 embedders, and thus separate pre-encode feed-forward layers, encoders, and integrators where we tried to pass in the source and candidate inputs separately without concatenating them. For reference, we also tested the model based on the sample Academic Paper Classifier provided in the AllenNLP classifier. Furthermore, we experimented with adding small ELMO embeddings to the Academic Paper Classifier model.

We had also heard of other networks such as a Siamese network, but from our understanding, a BCN fit our requirements because it had layers (e.g. the encoder) which could be trained on multiple languages thus capturing the connection/features between German and English and then use that for other tasks such as classification. We did not completely understand biattention, but it seemed that it also helped capture relational information between 2 inputs. For these reasons, we stuck with our model.

3 BCN with BERT

BCN was already implemented in AllenNLP but we attempt to describe how it works. The model fed the input into duplicate feed-forward networks, then duplicate encoders (LSTMs which themselves needed to be trained with German and English embeddings), then computed biattention, and integrated and pooled the weights, and fed it into a maxout network to obtain final class probabilities.

We had to make a couple modifications to make it work for our case: for the purpose of the “tokens” input, we had to modify the Dataset Reader to output a single “tokens” output instead of separate “source” and “candidate” outputs. We also needed to add a Metadata input/parameter for forward in the model to allow us to pass back the source and candidate values for the predictor. In the config, we use pre-trained

BERT (Bidirectional Encoder Representations from Transformers) as a token indexer, specifically using the bert-base-multilingual-cased pre-training, which has been trained on German as well as English, in opposition to the ELMo embeddings, which were only trained for English (separate ones exist for German, but we did not have easy access to these). We later experimented with using token character indexing and embedding without pre-training. Importantly, we provide offsets for BERT to allow for embedding of tokens beyond the granularity of words. We specify a dropout of .25 for these embeddings. For our pre-encode feed-forward network, we specify an input size of 768, a single hidden layer, and a ReLU activation to the next layer. Our encoder itself is a Bidirectional LSTM. Next comes the integrator, which is once again a biLSTM. Finally, our Output Layer is a Maxout Network with 3 layers. Our pooling size before this is 4, and dropout is .2, .3, and 0. We use a bucket iterator and vary batch size from 64 to 128, and train for 30 epochs with a patience of 10, gradient normalization of 5.0, and either the Adam or Adagrad optimizer with a learning rate of .0005.

4 Things that did not Work as Well

4.1 Simple LSTM Classifier with ELMo Embeddings

This model based on the Academic Paper Classifier uses a single text field embedder, which uses pre-trained 100d GLoVe embeddings. We tried to introduce multiple embedders to additionally allow usage of German FastText embeddings. We then put the separate encodings of source and candidate inputs each through 100-dimensional, single hidden layer of size 100 Bidirectional LSTMs. Finally, we concat both these outputs and put them through a 2-layer feed-forward classifier with ReLU and Linear activations and an output size of 2 for the binary classification. To add ELMo or BERT, we simply add an additional token embedder and augment the input sizes of the encoders by the appropriate amount: 256. This model was passing some of the baseline tests, but we felt that it did not capture all the information we wanted it to.

4.2 Duplicating everything in the model

As mentioned before, we tried to feed our separate source and candidate inputs into the BCN. This required us to heavily edit the model code as the authors had written it assuming a case of self-attention. We tried to write duplicate encoders, feed forward layers, integrators, and incorporated padding (since the sequences in English and German were mostly of different length). However, the model never seemed to learn and performed with close to random accuracy. We were unsure of what the bug was and decided to concatenate the inputs instead. Perhaps with more work, this model may have performed better.

5 Accuracy Report

Table 1: Grid Search for Hyperparameters

Grad. Normalization	Batch Sz	Optimizer	LR	Out. Lyr. Dropout	Train. Acc	Val. Acc
5	128	adagrad	0.0005	[0.2, 0.3, 0.0]	0.892	0.755
5	100	adam	0.0005	[0.2, 0.3, 0.0]	0.912	0.794
5	100	adagrad	0.001	[0.2, 0.4, 0.0]	0.924	0.764
3	64	adam	0.001	[0.2, 0.5, 0.0]	0.887	0.759
2	64	adagrad	0.01	[0.2, 0.3, 0.0]	0.874	0.742

6 Thoughts for the Future

We think that this model performed quite well. We did notice in later epochs that the training accuracy was increasing while validation accuracy was not/was decreasing. This suggests to us that our model may

be overfitting. Another thing we would like to try is to incorporate a German/English dictionary or perhaps part of speech tagging.

Works Cited

AllenNLP. "AllenNLP Tutorials". allennlp.org/tutorials.

McCann, Bryan, et al. "Learned in Translation: Contextualized Word Vectors". *CoRR*, vol. abs/1708.00107, 2017. arXiv: 1708.00107, arxiv.org/abs/1708.00107.