Shared Metrics Task - TM-IMS21

Text Mining

Nova IMS
Members:
Frederico Santos: 20200604,
Ivan Kisialiou: 20200998,
Tiago Ramos, 20200613

Introduction

We present results and findings related to our proposals of adopted metrics to the Shared Metrics Task, as proposed for the Text Mining Project course.

Methodology

We initially considered and implemented a variety of simpler lexical and embedding based metrics, as well as different preprocessing pipelines (e.g. removal of punctuation, lowercasing)).

For the test corpora, we chose for each language pair the metric and preprocessing pipeline with the best results from the corresponding training corpus.

The applied metrics are:

- Basic BLEU metric. This metric was quickly discarded as it was virtually the same result as the Rouge-L metric;
- ROUGE metric, more specifically, the f score from Rouge-1, Rouge-2, and Rouge-L;
- Crhf metric, with a range of different parameters. (Beta of 1 or 3, minimum character length and maximum character length;
- Cosine Similarity of LASER-embeddings, between the reference-translation embedding pairs.

The outputs were standardized and the correlations of these z-scores were computed in regards to the annotator's scores.

Additionally, we applied trainable metrics on both numerical outputs of these metrics and the LASER embeddings provided. For the latter we applied several neural network architectures, regressing on the z-scores:

- Multivariate Linear Regression, using the z-scores of the cosine similarities as input. Two versions were tested: one which used all three computed cosine similarities, and one which used only two (of source-reference and source-hypothesis pairs);
- Fully connected neural network, with a concatenated 2048-sized vector of the LASER embeddings of the reference and the translation. It was built and compiled with four hidden layers of size 1024,512,128, and 64, with *relu* activations. To account for overfitting, two dropout layers (0.5 and 0.25) were added after the second and third layers. The final node was built with a tanh activation function.
- Convolutional neural networks:
    - Version 1: takes each embedding of 1024, reshapes it to 32x32, creates three channels (source, reference, hypothesis)
    - Version 2: takes each embedding and concatenates it to create a 3x1024 tensor.
- Decoder with fully connected layers with 8.8 M trainable params (appx)
    - Replaced Self-Attention Layer [1] with a Fast Fourier Transform in place of the self-attention layer inspired by the FNet paper [2] where the Self-Attention in a Transformer Encoder Layer was replaced by Fast Fourier Transform. When comparing training times to the same network architecture with Self-Attention Layers, it is up to 7 times faster due to the time complexity of processing self-attention. This is possible because the FFT layer is a simple linear "mixing" of inputs.
    - To help the network converge, we used the Rectified Adam optimizer which greatly improves training time while softening the need to hypertune the learning rate hyperparameter.
    - We also leverage information from the source segments, regressing on source, reference and hypothesis to estimate the human assessment score.

Results and Discussion

Regarding the convolutional neural networks, we could not find a way to make them converge and quickly stopped iterating on them.

Below we present a table of most relevant metrics and the correlation to the target z-scores that we obtained in the training corpus. Regarding chrf, bX indicates a beta of X, and nYZ a minimum and maximum character n-grams of correspondingly Y and Z. The best scores are bolded for readability.

Language Pair	cs-en		de-en		en-fi		en-zh		ru-en		zh-en	
Metric	Kendall Tau	Pearson	Kendall Tau	Pearson	Kendall Tau	Pearson	Kendall Tau	Pearson	Kendall Tau	Pearson	Kendall Tau	Pearson
FNet	**0.363**	**0.524**	**0.255**	**0.380**	0.333	0.488	0.277	0.407	0.231	0.323	**0.287**	**0.417**
Linear Regression	0.304	0.446	0.235	0.331	0.417	0.596	0.289	0.417	**0.2489**	**0.35433**	0.233	0.338
Chrf (best parameters)	*b3_n210* 0.304	*b3_n210* 0.451	b1_n110 0.233	b1_n110 0.34	**b1_n110 0.407**	**b1_n110 0.608**	**b1_n210 0.311**	**b1_n210 0.432**	b1_n16 0.245	b1_n16 0.358	b1_n210 0.222	b1_n210 0.3363
Fully Connected NN	-	0.4257	-	0.357	-	0.500	-	0.386	-	0.281	-	0.3936
ROUGE (best parameter)	L 0.295	L 0.437	L 0.225	L 0.322	1 0.350	1 0.532	-	-	L 0.103	L 0.146	L 0.139	L 0.209

In regards to the test set, we chose to apply the metric that returned the best results for each language pair.

In general, it is relevant to highlight the transversal positive performance of the chrf metric, especially when considering its computational efficiency when compared to more complex and time consuming metrics, such as ones based on embedding. In fact, Chrf was observed to perform better than any other lexical or non-trainable metric (ROUGE and BLEU in this project's case). We attribute this to several factors. One, its parametric nature allows it to be fine tuned to most language's nuances and specificities in training (e.g.character-based languages such as chinese). Two, due to this parametric aspect, it can capture the same information as other word-based metrics such as ROUGE and BLEU that have a bag-of-words and word n-gram approach to classification. Third, it can capture sequences of the same grammatical family, that despite being different words, are still able to convey the same meaning, and thus be a sufficient translation.

Chrf also performed universally better in an unprocessed version of the corpus, which is a similar behavior to other metrics. ROUGE was the only applied metric which returned better results in the processed version of the corpus, which removed punctuation and lowercase words.

The FNet was able to achieve the best results in half of the corpus, for some of the most complicated pairs, interestingly scoring significantly lower in english-to-pairs.

Conclusion

Our results and work in this project highlight the challenges inherent to natural language processing. There is still no universal metric that can objectively and reliably assess the quality of a translation; even the human translator assessments are prone to being subjective and susceptible to error, further difficulting this task.

In the context of this task, these issues could be mitigated by enriching the corpus with multiple references per translation, as well as a higher number of human annotations. None of these solutions are frugal (in terms of both cost and data), but sparseness of data is precisely one of the reasons why some supposedly powerful metrics (such as the trainable ones) are much less efficient than the others - they scale very well with the amount of data provided.

Future work:

- Using Ranger21[3] instead of RAdam to see how much faster the networks converge
- We would like to be able to feed LaBSE embeddings to the neural networks. LaBSE are considered superior to the LASER embeddings we used. We hypothesise using LaBSE would greatly improve the performance of our model.

References

- Attention Is All You Need, Google Brain & University of Toronto, arXiv:1706.03762 [cs.CL]
- FNet: Mixing Tokens with Fourier Transforms, James Lee-Thorp et al, arXiv:2105.03824v1 [cs.CL]
- Ranger21 - https://github.com/lessw2020/Ranger21

Appendix

1. FNet Architecture:

```
Model(
  (decoder): FNet(
    (layers): ModuleList(
      (0): DecoderLayer(
        (ff): Sequential(
          (0): Linear(in_features=3072, out_features=1024, bias=True)
          (1): GELU()
          (2): Dropout(p=0.1, inplace=False)
          (3): Linear(in_features=1024, out_features=3072, bias=True)
          (4): Dropout(p=0.1, inplace=False)
        )
        (norm1): LayerNorm((3072,), eps=1e-05, elementwise_affine=True)
        (dropout1): Dropout(p=0.1, inplace=False)
        (norm2): LayerNorm((3072,), eps=1e-05, elementwise_affine=True)
      )
    )
  )
  (fc_bloc): Sequential(
    (0): Linear(in_features=3072, out_features=768, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.1, inplace=False)
    (3): Linear(in_features=768, out_features=192, bias=True)
    (4): ReLU()
    (5): Dropout(p=0.1, inplace=False)
    (6): Linear(in_features=192, out_features=1, bias=True)
  )
)
```