Project 1 Report: How Much is 131 Million dollars? Putting Numbers in Perspective with Compositional Descriptions

Soham Parikh
CIS, University of Pennsylvania
sohamp@seas.upenn.edu

Abstract

This is a report on the implementation of the model in the paper (Chaganty and Liang, 2016). Specifically, it describes the model details, the dataset, the experimental details, an analysis on the results and the issues faced during reproduction.

1 Problem Description

The paper aims to put numbers mentioned in news articles into perspective. Given a mention of a number in an article *e.g., "he spent 131 million dollars"*, the task is to put this quantity into perspective by generating a relevant description *e.g., "the cost to employ everyone in Texas over a lunch period"*. To obtain the different statistics using which the quantity is put into perspective, a knowledge base consisting of a quantity (numerical value and unit) and corresponding description is used. Using the knowledge base, for each mention of a quantity, a formula is generated of the form *"10 × 50 $/yr × 20 yr"*, where each quantity contained in the formula has a relevant description attached. This formula is then used to generate a natural language description. The focus of this project is to implement and reproduce the RNN-based sequence to sequence model which generates the natural language description.

2 Model Details

The model is adapted from the model proposed in (Jia and Liang, 2016). The tokens/words from the formulae are mapped to a vector using word embeddings. These vectors are passed as inputs to the encoder bidirectional-LSTM and the description is generated using a decoder LSTM, which uses the final output from the encoder LSTM as the hidden state for the first time step. At every time step, the decoder LSTM generates a token, which can either be one of the source tokens or a token from the vocabulary. The input for each time step of the decoder LSTM is the previously predicted token concatenated with a bilinear attention weighted representation of the encoder LSTM outputs. Specifically, the equations for the model are given as

$$
\begin{aligned}
e_{ji} &= s_j^T W^{(a)} b_i \\
\alpha_{ji} &= \frac{exp(e_{ji})}{\sum_i exp(e_{ji})} \\
c_j &= \sum_i \alpha ji b_i \\
P(y_j = \text{Vocab}[i]) &\propto exp(U_w[s_j, c_j]) \\
P(y_j = \text{Formula}[i]) &\propto exp(e_{ji}) \\
s_{j+1} &= LSTM([y_j, c_j], s_j)
\end{aligned}
$$

where s_j is the decoder hidden state at time step j, b_i is the encoder output at time step i and y_j is the vector representation of the token predicted at the time step j.

3 Dataset Details

The dataset used for training the model consists of 19496 training pairs, while the test dataset consists of 433 pairs and there can be multiple descriptions corresponding to each formula, both in the training and the testing dataset. There are about 1678 unique tokens in from all instances in both training and testing sets. Each pair consists of a formula (which is treated as the input to the model) and a corresponding description (treated as the output of the model). Some examples from the dataset are given in Table 1

Formula	Description
100 * 10 liters per minute (rate of flow of water from tap)	100 times the rate of water flow from a tap
1/20 * 4 pounds per day per person (trash generated in the US) * 60 minutes (time taken for a basketball game) * 27 million people (population of Texas)	1/20th the amount of trash generated by Texans during the time it takes to watch a basketball game.

Table 1: Examples from the test set

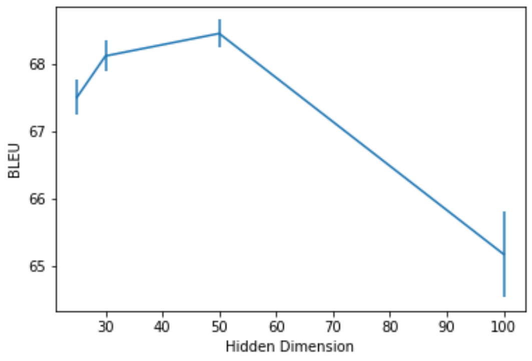

Figure 1: Error bar for the average BLEU score vs the hidden dimension. For each configuration, the other hyperparameters are kept the same and the results from 4 different runs for each value hidden dimension is averaged.

4 Implementation and Experiments

The model is implemented in Python 3 with the help of PyTorch, using AllenNLP for wrapper and util functions. The training dataset was initially divided using an 80-20 split into training and validation sets, using which the hyperparameters (number of epochs, hidden sizes of the LSTMs, batch size and learning rate) were chosen. A patience parameter of 5 epochs is used and the model is allowed to train for up to 20 epochs, since in all of the cases, the model performance did not improve after less than 15 epochs. Pre-trained embeddings of dimension 50 were used as provided in the worksheet by the original paper. The extensive list of all the hyperparameters tried is listed in Table 2. The error bar corresponding to different hidden dimensions tried is shown in Figure 1

After various runs, I settled on using 5×10^{-4} for the learning rate, a batch size of 16, 50 for the dimension of the outputs of LSTMs and the maximum number of epochs as 20.

Hyperparameter	Values
Hidden size	$[25, 30, 50, 100]$
Learning Rate	$[0.01, 5 \times 10^{-3}, 3 \times 10^{-3}, 10^{-3}, 5 \times 10^{-4}]$
Max. Epochs	$[10, 20, 30]$
Batch Sizes	$[8, 16, 32]$

Table 2: Hyperparameter settings used

Metric	Original	
	Baseline	Model
BLEU-1	57.32	69.79

Table 3: BLEU scores on the test set as reported by (Chaganty and Liang, 2016)

5 Results

The evaluation metric used for measuring performance of the model is **BLEU** (Papineni et al., 2001). (Chaganty and Liang, 2016) also propose a baseline method in which they concatenate the descriptions with functional words like *"for"*, *"of"* and *"times"*. While they don't mention about the ngram weights they use for computing their BLEU score, I tried to reproduce their BLEU scores using the baseline method using different combination of weights and figured out that they have reported the results considering only unigrams. Table 3 shows the original results while Table 4 shows the reproduced results. A few predicted examples from the trained model are shown in Table 5

5.1 Additional Experiments

I tried multiple ablation studies to see what works better:

- Setting all the words to lower case gives a BLEU score improvement of about 3 points for unigrams.

- The CopyNet (Gu et al., 2016) implementation uses special *"COPY"* tokens to indicate

Metric	Reproduced	
	Baseline	**Model**
BLEU-1	50.65	68.67
BLEU-2	29.83	43.47
BLEU-3	19.31	31.34
BLEU-4	12.47	22.61
BLEU-ALL	24.56	37.83

Table 4: BLEU scores obtained on the test set using my implementation

if the previous predicted token was copied from the source. I experimented with using these special tokens but I did not see any noticeable increase in performance

- (Gu et al., 2016) also use a selective read along with the attentive read (c_j) and uses a different equation for obtaining the copy weights. I experimented with the CopyNet model on this dataset and obtained an increase in the performance by 0.8 BLEU points.

5.2 Error Analysis

Looking at some samples from the test dataset, I observed that a lot of examples that were of low quality were from longer formulae *i.e.,* formulae with 2 or more constituent formulae. This could be because of inadequate unique training examples (there were multiple instances of the same formula with slightly different descriptions). Moreover, language generation for longer sequences is still not a solved problem so maybe a better method is needed for generation the descriptions from longer formulae. A simple trick to augment the dataset could be to break down the longer formulae into shorter formulae and break down (roughly) the corresponding descriptions as well into shorter descriptions.

The baseline model underperforms as compared to the result reported in the original paper. I hypothesize that this is the result of using different preprocessing of the formulae and description, as the method used is exactly the same as that given by the authors. The actual model performs a bit worse than the results reported in the original paper. I credit this mainly due to a different choice of hyperparameters and pre-processing since the gap in the performance is not huge.

Further, I believe that the evaluation is noisy for this task, since the evaluation is done against a single gold description. There are multiple correct descriptions possible for a formula. A better evaluation metric would be to use the multiple descriptions for a single formula and use a metric like ROUGE (Lin, 2004) which takes multiple gold summaries into account and takes recall for n-grams into account while evaluating the prediction.

6 Problems faced

There were multiple challenges faced while implementing model in (Jia and Liang, 2016), even though a lot of it was made simpler by the available implementation of CopyNet in AllenNLP.

- As is the case with learning any new library or framework, it took a bit of time to understand how AllenNLP API works and how the internal structure of each object and function is. A lot of times, I had to refer to the source code to understand what exactly a function/object/wrapper does since there is not a lot of available document.

- There is no way in AllenNLP yet to use a random seed if torch is imported in the same Python project. This is key for reproducing exact results. However, luckily, there is not a huge variance in the results obtained from different runs of the same hyper-parameter settings.

- Since I was modifying the copynet code, a lot of time was spent in removing or modifying the components that were not needed for implementing this model. Moreover, it took a bit of time to figure out what exaclty each module of copynet was expecting

- Since I had not looked at the original code by (Chaganty and Liang, 2016), I was not sure whether they used the special *"COPY"* tokens and hence, I had to experiment with both the using the *"COPY"* tokens and using the previous prediction token to determine which worked better.

- Last but definitely not the least, the most amount of time was spent in trying to get

Formula	Description
20*0.001 sq-meters (area of the face of a credit card)	20 times the area of a face of a credit card
1/40 * 22 gallons per person per year (coffee consumption) * 1 days (a day) * 17 thousand people (number of employees at mckinsey)	1/40 the amount of coffee by the number of employees at mckinsey.

Table 5: Examples generated by the model

the BLEU score anywhere near the one reported in the paper. I was using the default BLEU score initially (which equally assigns weight to 1,2,3 and 4-grams). However, I kept getting BLEU scores of about 37-38 points, which is way below the BLEU score reported in the original paper. I spent days trying to figure out what was wrong with my code and I could not find anything. It was only later that I ran their baseline code and experimented with different ngram weights for the BLEU score that I found that the metric reported in the paper is BLEU score based on only unigrams and there was nothing wrong with my implementation. I feel that the authors (for any NLG paper) should mention in detail the metric they are using, whether it is BLEU or ROUGE or METEOR or any other metric.

7 Conclusion

The general task of putting numbers and quantities into perspective is very interesting. However, I personally believe that most of the contribution of the paper lies in constructing relevant formulae (according to proximity, similarity, familiarity and compactibility) from the knowledge base. The perspective generation task is modeled as a sequence to sequence task and there is no novelty in the model as such. The implemented model performs almost as well as the performance reported in the original paper and I believe that it is a matter of different pre-processing and hyperparameter settings that gives the better result in the latter.

Having said that, implementing a deep learning model, or even reproducing the results is an effort in itself and is a lengthy and time consuming process. The main aspects I learnt that we need to take care of while implementing any model are the following:

- Pre-processing and formatting the input and output data
- Experimenting with multiple hyper parameter settings to select a decent setting
- Using the correct metric to evaluate the task, since many metrics do not give the correct idea

References

Arun Tejasvi Chaganty and Percy Liang. 2016. How much is 131 million dollars? putting numbers in perspective with compositional descriptions. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *CoRR*, abs/1603.06393.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *ACL 2004*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. In *ACL*.