Sentence Simplification

Vishwesh S

July 2019

Definitions

Simplification consists of modifying the content and structure of a text in order to make it easier to read and understand while preserving its main idea and approximating its original meaning. Simplifying a text automatically could improve performance on NLP tasks, such as parsing, summarization, information extraction, semantic role labelling, and machine translation.

Problem Statement

The initial aim of the project was to split a complex sentence to two/more simple sentences and then rephrase the obtained simple sentences to meaningful sentences. This will syntactically simplify the sentence. Later also apply lexical simplification which includes replacing complex or uncommon vocabulary words with simpler synonyms which are easier to understand. The main constraint incorporated is that the meaning of the complex sentence has to be completely retained in the set of simple sentences obtained by the simplification process.

Literature Review

In this section, the various sentence simplification methods and their limitations in the past literature are discussed.

Rule-Based Systems

Handcrafted rules were designed to simplify sentences. Usually, in these methods, first, a structural representation of the sentence is obtained. The common methods for this task are to use a Finite State Grammar (FSG) to produce noun and verb groups and the other method is to use a Supertagging model to produce dependency linkages. After obtaining structural representation, appropriate rules are applied to simplify the sentence. This method involves a lot of manual work and fails to generalize.

Later, researchers felt that this task could be treated as a translation problem in which the input would be a complex sentence and the output would be a set of simple sentences. By using this idea, all the proposed machine translation methods, like phrase-based, syntax-based, deep learning methods were applied. As expected deep learning methods outperformed the other methods.

Deep Learning Approaches

In deep learning methods, sequence to sequence model is used as a base and additional techniques are incorporated with includes copy mechanism, deep reinforcement learning framework, entailment and paraphrase generation to improve performance. The dataset used was Original English Wikipedia pages as the source for complex sentences and Simple English Wikipedia pages as the source for simple sentences.

Error Analysis

The major errors that occur in the sequence to sequence model are:

1. Information gets deleted in the output of the model
2. Very few input sentences are split into two/more simple sentences
3. Grammatical mistakes in the output

By the incorporation of deep reinforcement learning framework and entailment generation techniques, grammatical mistakes were eliminated to a great extent. But still, even in these methods, there were instances in which information gets deleted in the output and it also fails to split sentences.

On a closer look, it was found out that the problem was with the dataset. In the dataset, there were many instances in which the complete information was not retained in the reference sentences and only 6.1% of the complex sentences were split into two/more simple sentences.

With these drawbacks with the dataset, the models were not able to learn to completely retain information and split sentences which are essential according to the problem statement.

Datasets

Due to the drawbacks in the Original-Simple Wikipedia English dataset as explained in the previous section, researchers developed two datasets in which each complex sentence is split into two/more simple sentences and complete information is retained in the reference sentences.

WEBSPLIT

This dataset was derived from the WEBNLG corpus. In WEBNLG dataset each item consists of a set of RDF triples and one or more texts verbalising those triples. The WEBSPLIT was derived by matching up sentences in the WEBNLG corpus according to partitions of their underlying meaning representations (RDF triples).

WIKISPLIT

Wikipedia maintains snapshots of entire documents at different timestamps, which makes it possible to reconstruct edit histories for documents. The WIKISPLIT corpus was constructed by identifying edits that involved sentences being split. WIKISPLIT contains one million naturally occurring sentence rewrites, providing sixty times more distinct split examples and a ninety times larger vocabulary than the WEBSPLIT corpus.

WEBSPLIT provides a basis for measuring progress on splitting and rephrasing sentences. However, its small size, inherent repetitiveness, and synthetic nature limit its broader applicability. In particular, it can be used as a viable benchmark for evaluating models, but not for training them. For training, WIKISPLIT is a better choice as it is diverse and contains some noise.

Implementation

The state of art method (Sequence to Sequence model with copy mechanism) was implemented. WIKISPLIT dataset was used for training.

Training Details

A single layer LSTM encoder-decoder model, with encoder as bi-directional rnn, was used. The size of embedding vectors and the LSTM units used was 512. The batch-size used was 64. For optimization, SGD was used with an initial learning rate of 1.0 and with a learning rate decay of 0.5 every 10000 steps after 50000 training steps.

Results

The model was trained for 200000 steps and the final validation accuracy and perplexity were 87.1118% and 2.22825 respectively.

Errors

Three types of errors in the output were identified in manual analysis.

1. Unsupported facts
2. Repeated facts
3. Missing facts

Experiments

Experiments were conducted to find a way to eliminate the three types of errors listed in the previous section.

Entailment Checking

The idea here was to incorporate an auxiliary entailment generation task in the splitting and rephrasing task. By this, the output of the model would be entailed to the reference sentence. But when entailment checking was done on the outputs with unsupported/missing facts and corresponding reference sentences, the entailment generation task classified them as entailed which on an ideal case should be classified as non-entailed. Based on this observation it was concluded that there won't any significant improvements by incorporating entailment generation task.

Sentence Similarity using Universal Sentence Encoder (USE)

Sentence embeddings were obtained by using the USE and then the cosine similarity was calculated between the reference and the output. It was observed that when there were unsupported/missing facts in the output, the cosine similarity scores were less. Therefore, the idea here was to use the cosine similarity value as a reward value to train a sequence to sequence model in a RL framework. If the model initially produces outputs with unsupported/missing facts, the reward value would be less and the model would try to produce outputs which have high reward value in the process of maximizing reward function. This would eventually make the output of the model close to the reference sentence.

Sentence-level BLEU Score

In this, the idea was to use the sentence-level BLEU score as the reward value to train a sequence to sequence model in a RL framework. This would improve the performance of the model as explained above.

Both cosine similarity values and BLEU scores lie between 0 to 1. When these two values were calculated for an output with unsupported/missing facts and the corresponding reference sentence, the BLEU score came to be less than the cosine similarity value. Less score is equivalent to more penalization and hence, it was concluded that using BLEU score as reward value would be more appropriate.

There were very few instances where outputs were produced with repeated facts and a common method to eliminate repetitions produced in the output by a sequence to sequence model is to incorporate coverage mechanism.

Model Proposed

Based on the experiments as explained in the previous section, it was concluded to use a RL framework with BLEU score between the output and the reference as reward value.

On a detailed study on RL algorithms, it was found that direct policy optimization faces significant challenges: unlike ML, a stochastic gradient given a mini-batch of training examples is extremely noisy and has a high variance; gradients need to be estimated via sampling from the model, which is a non-stationary distribution; the reward is often sparse in a high-dimensional output space, which makes it difficult to find any high-value predictions, preventing learning from getting off the ground; and, finally, maximizing reward does not explicitly consider the supervised labels, which seems inefficient.

Instead, using a Reward Augmented Maximum Likelihood (RML) was found out to be more efficient. This algorithm simply adds a sampling step on top of the typical likelihood objective. Instead of optimizing conditional log-likelihood on training input-output pairs, given each training input, the algorithm first samples an output proportionally to its exponentiated scaled reward. Then, it optimizes log-likelihood on such auxiliary output samples given corresponding inputs. When the reward for output is defined as its similarity to a ground truth output, then the output sampling distribution is peaked at the ground truth output, and its concentration is controlled by a temperature hyper-parameter. The advantage of the RML framework is its computational efficiency at training time. The framework only requires sampling from a fixed exponentiated payoff distribution, which can be thought of as a form of input pre-processing. This pre-processing can be parallelized by model training by having a thread handling loading the data and augmentation. This sampling is the price that has to be paid to learn with rewards.

Sequence to Sequence model with reward augmented maximum likelihood (RML) objective was implemented. WIKISPLIT dataset was used for training.

Training Details

A single layer LSTM encoder-decoder model, with encoder as bi-directional rnn, was used. The size of embedding vectors and the LSTM units used was 256. The batch-size used was 8. For optimization, adam optimizer was used with an initial learning rate of 0.001 and with a learning rate decay of 0.5 when there was no improvement on the validation set. The temperature hyperparameter was set to 0.6. To study the influence of using RML objective, a sequence to sequence model and a sequence to sequence model with copy mechanism were trained with the same training setting. All the three models were trained with early stopping (patience value was set to 5).

Results

The Seq2Seq model with RML objective outperformed both the Seq2Seq and Seq2Seq with copy mechanism.

Model	Validation Perplexity
Seq2Seq	3.3084
Seq2Seq with copy mechanism	2.8636
Seq2Seq with RML objective	2.7840

Future Work

1. To incorporate copy mechanism in Seq2Seq model with RML objective, which not only improves the model's performance but will also help to deal with out of vocabulary words in the input.
2. To train the Seq2Seq model with RML objective on a larger batch-size and hidden states size. By increasing the batch-size and the hidden states size, better performance was observed in the state of the art model.

Code

All code and pre-trained models available at:
https://github.com/VishweshS/Sentence-Simplification

Improvements to be done

1. For the Seq2Seq model with RML objective, only validation perplexity is reported because if bleu/word_acc/sent_acc is used as the validation metric it takes a lot of time to compute the values.
2. For the Seq2Seq model with RML objective, training was done only on batch-size of 8 and hidden states size of 256, because if these values are increased, the GPU memory needed for training exceeds 12GB (memory capacity of a standard GPU).