Fine-Tuning a CodeT5 Model for SQL-To-Text Translations

Chris Lewis
University of Massachusetts Lowell
Christopher_Lewis1@student.uml.edu

Abstract

In recent years, transformer-based sequence-to-sequence (seq2seq) models have become increasingly popular in natural language processing (NLP) and have been applied to various machine translation tasks. However, such tasks require large amounts of training data and computational resources to achieve acceptable results. As a result, it has become standard practice to fine-tune pre-trained seq2seq models on specific datasets for a downstream task. In this work, the pre-trained CodeT5-base model is fine-tuned on a SQL dataset to perform SQL-to-text translations. It is shown that this fine-tuned model achieves a higher BLEU score than baseline models. Additionally, human evaluation is performed on the model's predictions to further demonstrate its viability in performing SQL-to-text translation.

1 Introduction

The relatively new Transformer architecture (Vaswani et al., 2017) and its adaptations have shown to achieve strong performance on many sequence-to-sequence (seq2seq) problems, such as machine translation. This success is largely due to the global self-attention mechanism implemented by the Transformer architecture.

In order for Transformer-based sequence-to-sequence models to achieve competitive results, it is necessary that they are trained on large amounts of data. However, training such a model is very computationally expensive. Pre-trained models help address this problem, because they have already been trained on large amounts of data that is related to the target task. As a result, pre-trained models can be fine-tuned using a relatively small number of examples from the dataset of choice, to achieve strong results.

In this paper, a T5 model (Raffel et al., 2020), specifically the pre-trained CodeT5-base model checkpoint (Wang et al., 2021), is fine-tuned using the WikiSQL dataset (Zhong et al., 2017) to perform SQL-to-text translations. This model accepts a valid SQL query as an input sequence and produces a corresponding natural language question/statement that is satisfied by the SQL query.

This paper empirically shows that the fine-tuned CodeT5 model outperforms several baseline models in the task of SQL-to-text translation, as well as the CodeT5 model trained from-scratch. Human evaluation is also performed to gain further insight into the metrics achieved by this model.

1.1 Related Work

CodeT5

CodeT5 is a unified encoder-decoder Transformer model that has been pre-trained on code-based datasets that span a wide variety of programming languages. In total, around 8.35 million examples were used during pre-training. This model takes advantage of the code semantics conveyed from developer-assigned identifiers and it lends itself to tasks such as code generation and code understanding. Because it has been pre-trained on a wide variety of programming languages, it has been shown to better capture semantic information, which is beneficial when fine-tuning it on a code-based dataset such as WikiSQL.

Attention-based Seq2Seq Models

Attention-based seq2seq models have been created largely to perform machine translation-related tasks. Originally, these models encoded a source sentence into a fixed-length vector, from which a decoder generates a translation. However, the encoder-decoder architecture was extended to automatically soft-search for parts of a source sentence that are relevant to predicting a target word, without having to explicitly form them as a fixed segment (Bahdanau et al., 2014). At the time, this extension of the encoder-decoder model demon-

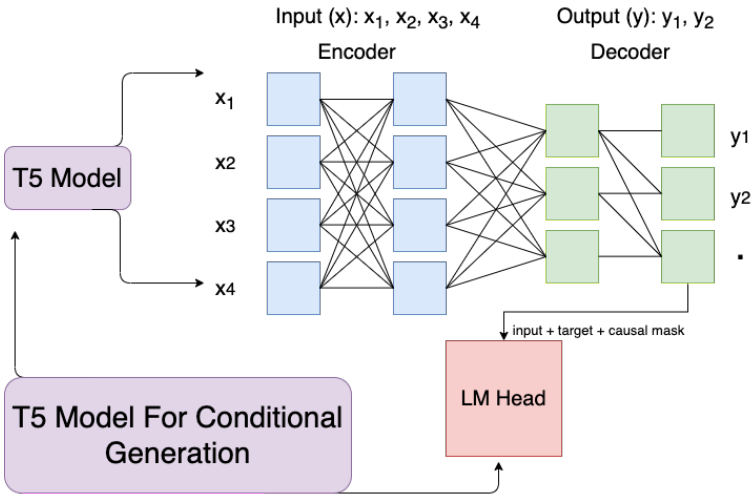

Figure 1: Fine-tuned T5 model diagram

strated state-of-the-art translation capabilities. This model, introduced by Bahdanau et al. (2014), will be referred to as "Seq2Seq" and does not refer to seq2seq models in general.

CopyNet (Gu et al., 2016) has improved upon the Seq2Seq model, by implementing a copy-mechanism in the decoder. This copy mechanism chooses subsequences from the input sequence and puts them in appropriate places in the output sequence. This model will be referred to as "CopyNet".

Attention-based Tree2Seq

Eriguchi et al. (2016) introduced an end-to-end syntactic NMT model, acting as an extension of seq2seq models with source-side phrase structure. It has an attention mechanism that allows the decoder to generate a translated word while aligning it with phrases and words from the source sentence. Because this model leverages syntactic information, it is able to effectively capture important features in code during the encoding process. This paper's model will be referred to as "Tree2Seq".

2 Methods

2.1 Dataset

The dataset used to fine-tune the CodeT5 model is Salesforce's WikiSQL dataset. This dataset consists of 87,726 hand-annotated SQL query and natural language question pairs. These pairs are divided into 61,297 training, 9,145 validation, and 17,284 testing examples.

2.2 Model Overview

Figure 1 shows a digram of the fine-tuned model's architecture. The CodeT5 model is represented by the encoder/decoder portion of the diagram. Each block in the encoder/decoder represents an element in the input/target sequences. The "▪" in the final decoder block corresponds to the end-of-sequence token that represents the end of a prediction. Fully-visible masking is used in the internal encoder and encoder-decoder connections and causal masking is used in the internal decoder connections.

The fine-tuned CodeT5 model (T5 model for conditional generation) includes a language modeling head on top of the original T5 model's decoder. For the T5 model specifically, the language modeling head is a linear layer, applying a linear transformation with input dimensions equal to the number of expected features in the decoder outputs (embedding size), and output dimensions equal to the vocabulary size.

Tokenization

The CodeT5 model itself was trained using a code-specific Byte-Pair-Encoding (BPE). Therefore, a pre-trained RoBERTa tokenizer, which uses byte-level BPE, was loaded using the CodeT5-base model checkpoint.

Training

The T5 architecture formats all tasks as text-to-text tasks. In doing so, it can always train using standard maximum likelihood, using teacher forcing and a cross-entropy loss.

Input sequences are fed to the model using

the tokenized input IDs, and target sequences are "shifted" to the right, meaning they are prepended by the start-sequence token (padding token in this case), before being fed to the decoder. In teacher forcing, the target sequence is also appended by the end-of-sentence token and corresponds to the labels.

Inference

At inference time, the input sequence is encoded and the encoded hidden states are fed via cross-attention layers to the decoder, where the decoder output is generated auto-regressively. Beam search is used as the model sampling method.

2.3 Experimental Setup and Evaluation

Baselines

The baseline models used for evaluation are the attention-based sequence-to-sequence models, Seq2Seq and CopyNet, and the tree-to-sequence model, Tree2Seq, introduced in Section 1.1. These models were also used as baselines to evaluate IBM's Graph2Seq model (Xu et al., 2018), where they were evaluated against the WikiSQL dataset for SQL-to-text translations. The baseline results presented in IBM's paper are also used as baselines in this paper to evaluate the fine-tuned CodeT5 model.

Automated Metrics

This paper's model and all baselines are evaluated using the BLEU scoring metric (Papineni et al., 2002), which is an automated metric used to evaluate machine translations. Specifically, the BLEU-4 score is calculated, meaning 1-gram through 4-gram scores are computed and given equal weights to provide a final score.

It is worth noting that in IBM's code, the generated and actual output sequences were both converted to lowercase before computing the BLEU scores, while in this paper they were not. It was not done in this paper, because of the significance of case-sensitivity in SQL queries. For example, the two questions: "*Select all user IDs where the username contains* ***gHsA***" and "*Select all user IDs where the username contains* ***ghsa***" may be interpreted as two different SQL queries that match on strings "***gHsA***" and "***ghsa***", which will yield different results for case-sensitive database fields. However, not converting the generated and actual sequences to lowercase, which normalizes them, will naturally lower the BLEU score.

This paper's model was also evaluated against a CodeT5 model trained from scratch. The BLEU score was computed for the pre-trained and from-scratch models after 2,500 training steps to evaluate the effectiveness of using a pre-trained model.

Human Evaluation

Human evaluation was performed on 110 unseen test examples, and follows an approach similar to Papineni et al. (2002). Each generated sequence was given a score between 1-5, based on pre-defined criteria, with as score of 1 being very bad and a score of 5 being very good. The first criteria was readability, i.e. how easy it was to understand the generated text. The second criteria was coverage, i.e. if all of the query clauses/requirements were mentioned in the generated text or not. The third criteria was exactness, i.e. whether the values in the generated text were exactly the same as the query values or not.

After the human-generated scores were made, the BLEU scores were calculated for the same set of examples. Then, the Pearson correlation was calculated between each pair of scores to assess how viable the BLEU score is for this task and to identify relationships between the two scores.

Hyperparameters

CodeT5 was fine-tuned using the AdaFactor optimizer, using the original pre-trained model's parameters and an external learning rate of $3e^{-4}$. A linear learning rate scheduler was used to decrease the learning rate during training. A batch size of 8 was used and during inference a beam size of 10 was used. During pre-training, this model used a constant learning rate of 0.01 for the first 10^4 steps and exponentially decayed the learning rate until pre-training was finished. During pre-training the AdaFactor optimizer was also used.

It is worth noting that the AdaFactor optimizer's learning rate scaling parameter, time-dependent learning rate parameter, and warmup initialization parameter were all disabled.

3 Results

Baseline Metrics

Table 1 shows the BLEU score results of the fine-tuned CodeT5 model and the baseline models. The fine-tuned CodeT5 model was evaluated using case-sensitive BLEU scores and and the baseline models used case-insensitive BLEU scores.

Model	BLEU-4
Seq2Seq	20.91
CopyNet	24.12
Tree2Seq	26.67
CodeT5-Base (Fine-Tuned)	**28.047**

Table 1: Baseline comparison results on WikiSQL

Table 2 shows the BLEU score results after 2,500 training steps for the fine-tuned CodeT5 model and the from-scratch CodeT5 model. It is worth noting that the from-scratch CodeT5 model also used the AdaFactor optimizer with the same external learning rate and model configuration, but with no model weights.

Model	BLEU-4
CodeT5-Base (From-Scratch)	1.297
CodeT5-Base (Fine-Tuned)	**28.047**

Table 2: From scratch comparison results on WikiSQL

Optimizers

Figure 2 shows the BLEU score results for the fine-tuned CodeT5 model when using both the AdaFactor and the AdamW optimizers.

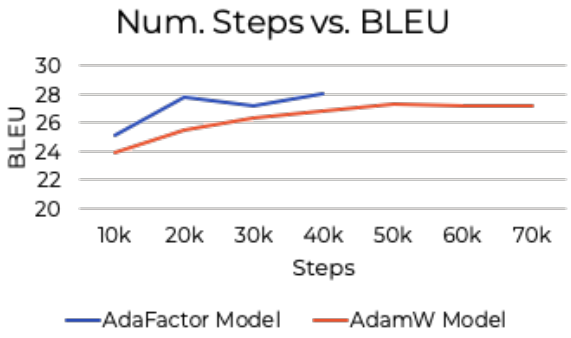

Figure 2: Fine-tuned CodeT5 model results

It is worth noting that in Figure 2, the parameters used for both model runs were the same and only different optimizers were used. The AdaFactor model was trained further after 40,000 steps, but there were no improvements in BLEU score.

Human Evaluation

Figure 3 shows the results of the human evaluation and BLEU score experiment described in section 2.3.

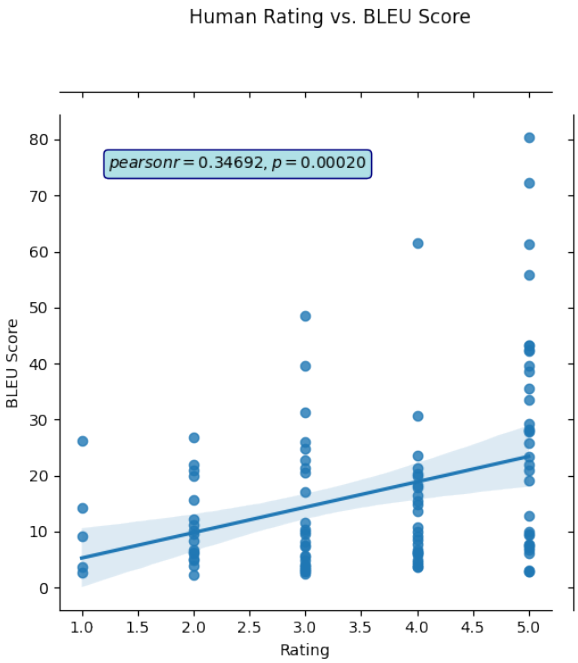

Figure 3: Human evaluation results

The ratings and BLEU scores in Figure 3 were calculated using 110 model generations on unseen test examples. A linear regression model was fit to the data to visualize correlations between the human ratings and the BLEU scores. The Pearson correlation coefficient and two-tailed p-values are given as well.

4 Conclusion

Overall, this paper demonstrates the capabilities and effectiveness of pre-training. This is done by showing that the pre-trained CodeT5 model outperforms the baseline models, as well as a CodeT5 model that has no pre-trained components. This paper also shows that the AdaFactor optimizer is preferred over the AdamW optimizer, particularly when when training T5 models. Finally, human evaluation was performed to find the correlations between human-rated scores and BLEU scores and to understand how significant these correlations are.

4.1 Discussion

Baseline Results

From Table 1, we can see that the fine-tuned CodeT5 model outperforms the baseline models, even when using case-sensitive BLEU scoring. This is expected, because it has been pre-trained on large amounts of code/natural language pairs,

allowing it to learn the SQL-to-text task more effectively during training. We also see that the CopyNet model outperforms the standard Seq2Seq model, due to the added copy mechanism. For SQL-to-text translations it is very likely that the same subsequences in the query will appear in the description, particularly for column names and field values. Finally, the Tree2Seq model outperforms Seq2Seq and CopyNet, because its tree-based encoder considers the syntactic structure of the SQL query.

From Table 2, we see that the CodeT5 model performs significantly better after 2,500 training steps when it has been pre-trained instead of trained from scratch. This demonstrates the advantage that pre-training gives to models in terms of training time and results.

Optimizer Differences

As shown in Figure 2, the AdaFactor optimizer offers better performance than the AdamW optimizer. The AdaFactor optimizer was preferred, especially for T5 models, because the T5 authors use it themselves and it was developed specifically with Transformers/T5 in mind. Additionally, AdamW wastes a large amount of memory in general, while AdaFactor does not.

Significance of Human Evaluation

From Figure 3 we can see the correlation between the human rating scores and the BLEU scores. Based on the Pearson correlation coefficient and the regression line, we can observe a weak, positive linear relationship between the two scores. Similarly, the p-value of $0.0002 < 0.05$ indicates the relationship is statistically significant. It is worth noting that when case-insensitive BLEU scores are used, the Pearson correlation coefficient becomes 0.41930 and the p-value becomes 0.00001, indicating a stronger positive linear relationship that is more statistically significant. However, it was common during human evaluation to see that, when the target and generated sequences did not match due to case-sensitivity, the generated sequence often matched the query's capitalization patterns better than the target sequence.

In general, the correlations are not as strong as those reported by Papineni et al. (2002), largely due to the nature of SQL-to-text translations. SQL tokens, such as keywords and column names, can be ambiguous in terms of how they are represented in a natural language question. For example, the words "*distinct*", "*unique*", and "*different*" can all be appear in a natural language question to signify that the *DISTINCT* keyword should be used. Therefore, while the BLEU score may be low between sequences such as: "*Get the number of distinct ids*", and "*Count all of the different id records*", they are both valid translations.

4.2 Future Work

Curriculum learning would likely be useful to improve this paper's results. As previously mentioned, SQL tokens are often ambiguous in terms of how they are represented in natural language. As the size and complexity of the SQL queries grow, so too does the risk of ambiguity. Using curriculum learning to fine-tune the CodeT5 model, starting with simple queries and progressively increasing their complexity, would benefit the model.

Another improvement to this work would be to fine-tune a larger model, specifically the CodeT5-large model, provided enough computing power is available. As previously mentioned, fine-tuning a model that has been pre-trained on a wide variety of code examples improves the performance of SQL-to-text translations. Because a larger model has seen more code examples, it will likely perform better during the fine-tuning process.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 823–833, Berlin, Germany. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Yue Wang, Weishi Wang, Shafiq Joty, and Steven C. H. Hoi. 2021. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation.

Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, Michael Witbrock, and Vadim Sheinin. 2018. Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.