Pre-training for Semantic Parsing using encoder-decoder architecture and BERT

Dheeraj Kumar Kondaparthi
Department of Computer Science
University of Illinois at Chicago
IL, USA
dkonda2@uic.edu

Abstract

Semantic parsing is a core Natural Language Processing(NLP) task which is used to convert natural language occurrences into logical form. It is a unique technique used to extract meaning from an input sentence. The applications of semantic parsing are varied in NLP and they cover from question answering, machine translation, instruction following or regular expression generation. As part of this project, I investigate use of encode-decoder architecture models for semantic parsing task. I also compare the results of a simple transformer based encoder-decoder model built by[1] with new encoder – decoder model with encoder as BERT [2], a transformer which is pre-trained to learn bi-directional model language model on large corpus of text. These models have been trained on semantic parsing data using various data recombination techniques presented by[3]. Encoder-decoder transformer architectures and in particular, using BERT as encoder has significant benefits for semantic parsing across different evaluations techniques ranging from 4% to 20% improvement in accuracy on Geoquery dataset[4]. Fine tuning of BERT as encoder model is found to be difficult because of limited data.

KEYWORDS

Semantic parsing, BERT, Transformer, RNN, Encoder, Decoder

1. Introduction

A significant amount of information in the world is structured and semi-structured knowledge bases. Efficient and simple methods to query them are essential and must not be restricted only those who have expertise in formal query languages. So, the field of semantic parsing deals with converting natural language utterances to logical forms that can be easily executed on a knowledge base. The logical forms which are generated, can be executed in a variety of environments in order to enable tasks such as understanding and executing commands for robotic navigation and query parsing in conversational agents such as Siri and Alexa.

However, a challenge in semantic parsing is building logical properties that model conditional independence. In many cases, part of the meaning of a sentence(consider *what states border Illinois?*) is indeed independent of some words in the sentence(here if *Illinois* is replaced by any other state, the type of the question remains the same). Expanding data augmenting techniques such data recombination provides a framework to model these invariances and conditional independences by injecting prior knowledge. Namely, this setting introduces high precision generative model from the training data and within no time, model samples from it to generate new training samples.

Setting: Given a training set *D* of *(x,y)* pairs defining a $\hat{p}(x,y)$ distribution, we fit a generative model $\tilde{p}(x,y)$(recombinant examples) to $\hat{p}$. The training of the actual model $p_\theta(y|x)$ will be executed by maximizing $E(p_\theta(y|x))$ where x and y are drawn from $\tilde{p}$.

In[3], authors proposed data recombination, a innovative technique to overcome the limited amount of data. Data recombination is a specific form a data augmentation which focuses on creating new training examples by recombination of existing examples and the syntax of natural language questions and queries. In this paper authors, train a Recurrent Neural Network(RNN) based Encoder-Decoder model. In this project, I investigate more recent neural network architectures for natural language processing : Transformers. RNN based architectures are good at learning directional representations of natural language forms, while Transformers are also able to learn bidirectional representations. Bidirectional representations

look particularly relevant to queries in the dataset that is used here and motivates the approach in this project. Moreover, lack of enough available data makes use of pre-trained natural language model appealing, so, decision has been made to study benefits from using BERT in the models.

In this project, two models are trained, a simple Transformer based encoder-decoder model and Transformer based encoder-decoder with a pre-trained BERT Transformer as encoder. Comparative study of the performances across different architecture choices, training and fine-tuning procedures with data recombination and hyper parameters is presented. Eventually large scale encoder-decoder model with BERT as encoder showed performance improvement of 50% via strict evaluation metric, 60% improvement via knowledge based evaluation metric and 4.5% improvement on Jaccard evaluation metric, thereby demonstrating that BERT based encoder - decoder model performs better than simple transformer based encoder-decoder model.

2. Related Work

Data recombination pre-processing [3] and the introduction of the Transformer deep learning model in [1] followed by the state-of-the-art BERT technique [2] for sentence encoding offer powerful tools to improve performances on semantic parsing tasks.

Data Recombination: It leverages synchronous context-free grammars(SCFG) which are sets of production rules $X \rightarrow (\alpha, \beta)$ where X is category(non-terminal) and α and β are sequences of terminal and non-terminal symbols. Using a composition of the rules presented in [3], Jial et al. achieved state of the art test accuracy of 89.3 on GeoQuery Dataset.

Transformers: The Transformer architecture, introduced in [1] offers a model exclusively relying on a new self-attention mechanism, called multi-head attention, in order to draw dependencies between the input and the output. This allows for significantly more parallelization and achieves state-of-the-art results in translation tasks. Originally proposed as a sequence-to-sequence model, the whole architecture was composed of two Transformer blocks, an Encoder and a Decoder, with auto-regressive in the Decoder, as previously generated symbols become additional input when generating the next. The structure can be seen in the Figure1.

BERT: As introduced in [2], BERT is a language representation model that can be applied to a variety of natural language processing tasks. It offers a language model pre-training technique by first training a model on text data and then using the language representations learned by the model to solve a downstream natural language processing task with fine-tuning of hyperparameters. In this setting, the language representations are obtained using a multi-layer, bidirectional Transformer encoder. The Datasets used are the Books Corpus (800M words) and English Wikipedia (2,500M words). In addition to the usage of transformers, the main contribution of BERT is the tasks defined to learn language representations during pre-training:

- *Masked Language Model*: this task is essential to allow the Transformer encoder to produce bidirectional language representations of tokens in any input sentence. During pre-training, the model is fed with sequences of words from which some are masked. 80% of the time the masked word is replaced by a [MASK] token, 10% of the time by a random word (as [MASK] token won't be observed during the downstream task), 10% of the time it is left unchanged (resulting in representations biased towards the actually observed word).
- *Next sentence prediction*: this task is used to improve the model performance on downstream tasks which require to understand the relationship between two text sentences. During pre-training, the model encoder is fed with pairs of sentences and trained to classify if two sentences are consecutive or not.

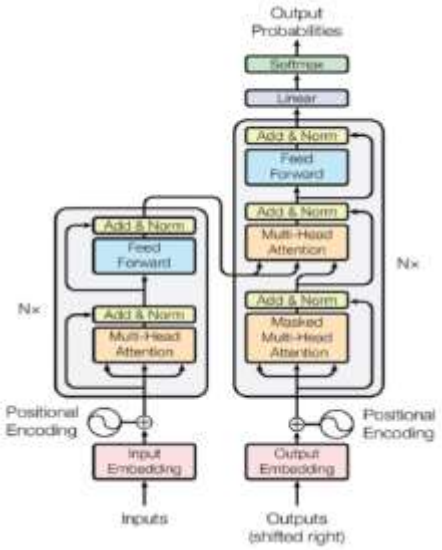


Figure 1: The Transformer architecture – figure from [1]

3. Problem Statement

In this project, I intend to apply transformer based encoder-decoder model to the task of semantic parsing using GeoQuery dataset[4]. Due to recent advances in state of the art language model, BERT[2], I also intend to used BERT as encoder in transformer based encoder decoder model and compare the performance using different evaluation techniques such as strict evaluation, knowledge based evaluation, Jaccard similarity evaluation and strict Jaccard evaluation. Upon verifying performances, I intend to make case for BERT based encoder-model for semantic parsing.

4. Technical Approach

In this project, two Transformer based encoder-decoder models are implemented, trained and evaluated on semantic parsing tasks.

4.1 Semantic Parsing inputs

In semantic parsing tasks, the inputs correspond to natural language sentences and target outputs are represented by the logical statements corresponding to their input. To feed input natural language sentences to our architectures, following steps are taken:

- Input sentence is tokenized using WordPiece[5]: sentence broken down into its tokens which correspond to its different words and some sub-word units ("wordpieces"). This approach limits the likelihood of encountering unknown tokens in semantic parsing datasets.
- Each input is thus a list of WordPiece tokens, and we use d_{model} dimensional trainable look-up embeddings of those tokens to feed them to our Transformer encoders.

After having encoded the dataset D as an initial grammar with rules $ROOT \rightarrow (x, y)$, each grammar induction is defined as a mapping from an input grammar G_{in} to G_{out}. As in [3], following three grammar rules are considered:

1. **Entity**: abstracting entities with their types, based on predicates in the logical form (such as stateid). For each $X \rightarrow (\alpha, \beta)$ in G_{in}, two rules are added: (i) both occurrences are replaced by the type of the entity (e.g. state) and (ii) a new rule maps the type to the entity (e.g. $Stateid \rightarrow ("illinois", Illinois)$. The entity rule swaps one instance of a logicalform (e.g. texas instance of stateid) with any other instance of the same form (ex: virginia).
2. **Nesting**: abstracting both entities and whole phrases with their types. The first rule added is the same as previously. The second one maps the whole expression β (the logical answer) to a particular type. In this setting, the expression of what states border A' is mapped to the type what is the highest mountain in A in order to produce what is the highest mountain in states border A'.
3. **Concatenate-k:** combining k sentences into one single rule ROOT.

4.2 Transformer Semantic Parser(TSP)

Transformer Semantic Parser is simple encoder-decoder semantic parser and different components associated with it are:

Encoder: The encoder is composed of N stacked Encoder Transformer layers, as described in [1]. Each Encoder Transformer layer is built as follows:

- Each input sequence in the batch is a tensor whose length is the number of tokens in the input sequence and where each element is a d_{model} dimensional WordPiece token embedding.
- The input is then recursively fed to N transformer sub-layers, a sequence of multi-head attention and feed-forward layers (whose intermediary dimension is dint, after being mapped back into d_{model}). Between each sublayer, residual connections, layer normalization and dropout are applied.

Input sequences are fed to the first Transformer layer. The outputs of each Transformer layer is the input of the next Transformer layer.

Positional Encoding of inputs: The Transformer architecture does not use recurrence or convolutions. Therefore, to make sure that the representation of each token in the input tensor depends on its position in the sequence, a positional encoding is added to the input tensor. Consider an input tensor of length L (number of tokens in the input sequence) and dimension d_{model} (dimension of the token embeddings). The positional encoding added to any such input tensor is a tensor with the same sizes and whose entries are sinusoid functions that will take different values depending on the dimension $1 \leq i \leq d_{model}$ and the position in the sequence $1 \leq pos \leq L$:

$$PE_{(pos,2i)} = \sin\left(pos/1000^{2i/d_{model}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(pos/1000^{2i/d_{model}}\right)$$

Positional encodings are added to input sentence in Encoder and to target sentence in decoder.

Attention mechanisms: Attention models link queries Q to key value pairs K; V . The output of this model is a weighted sum of the values, where the weights depend on the compatibility of the queries with the keys. The model attends to the values V depending on how relevant their keys K are to the queries Q. Multi-head, self-attention mechanisms with scaled dot-product attention are described in more details in [1].

Decoder: The decoder is composed of N stacked Transformer layers, each adding a Masked Multi- Head Attention sublayer to the Transformer layers as described for the Encoder. Each Decoder Transformer layer is built as follows:

- The positionally-encoded target sequence is fed to a Masked Multi-Head Attention sublayer. Here, the target output of the Decoder is fed, and the self-attention mechanism should respect the auto-regressive nature of the decoding procedure.
- This first embedding of the target becomes the query Q of a multi-head attention sublayer with keys K and values V equal to the output of the encoder.
- As before, a Feed-Forward layers completes the block. Residual connections, layer normalization and dropout are still applied between sublayers.

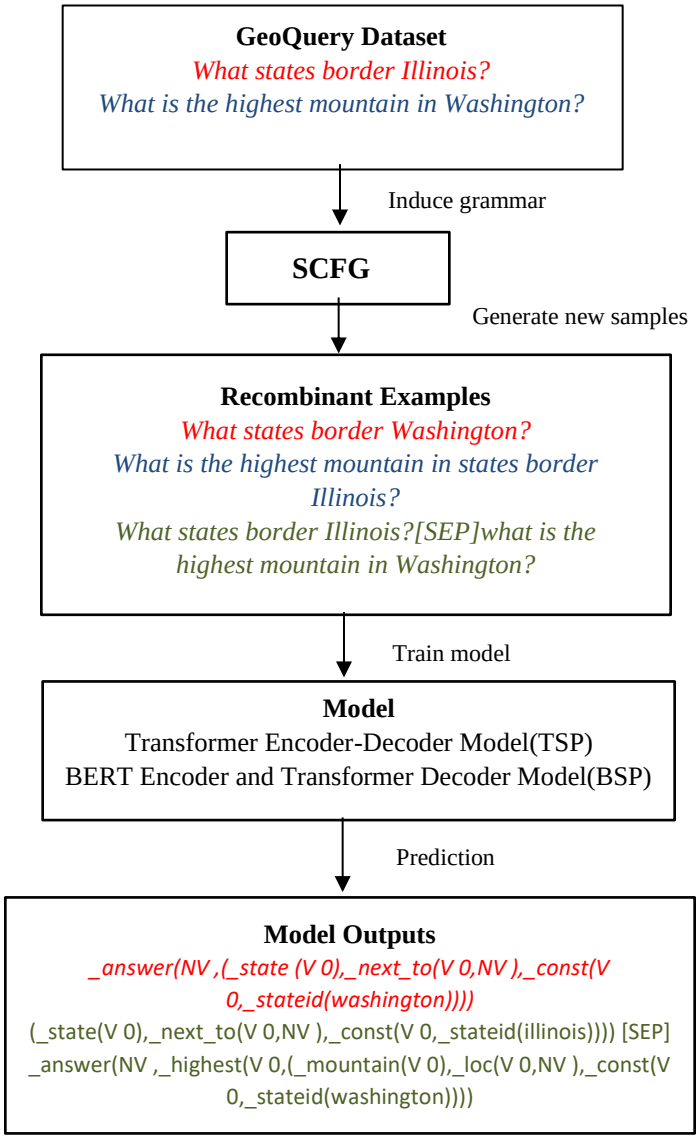
Model output: The output the of N Transformer Decoder layers is fed to a linear layer followed by a Softmax to obtain a probability distribution over the tokens in the WordPiece vocabulary.

4.3 BERT Semantic Parser(BSP)

BSP is encoder decoder semantic parser where replace the simple Transformer Encoder of TSP is replaced by BERT, a Transformer sentence encoder, trained to learn a bidirectional language model. In this project, analysis is performed whether BERT Encoder can improve TSP's performance. The two main assets of the BERT Encoder seem to be the large amount of data and bidirectional language modelling task from which it learns embeddings of natural language sentences.

4.4 Overall Model

Given the GeoQuery dataset, a high-precision synchronous context-free grammar is induced first. Then sample from this grammar is used to generate a new training and development set, which is used to train TSP or BSP model.



4.5 Implementation using PyTorch

Implementations are based on the following sources:

- HuggingFace's pre-trained BERT GitHub repository[6]: BERT tokenizer is sued to split othe inputs into tokens, and their BERT pre-trained model is used as the Encoder of BSP model.
- Harvard NLP's blogpost about transformers[7]: Code which they expose for positional encodings

and the masking of the Transformer Decoder inputs is directly used.

5. Experimental Setup

In this section experimental approach and evaluation techniques are discussed.

5.1 Dataset

In this project, Geoquery dataset [4] is used, which is the standard for semantic parsing tasks. It basically contains at each line a single question (input) (*what is the highest point in washington?*) and its logical utterance (output) *(_answer(A,_highest(A,(_place(A), loc(A,B),_const(B,_stateid(washington))))))*.

Dataset is split into three different files: one used as training set containing 600 data points, one for development set containing 100 data points and one for testing containing 280 data points. Then the three recombination techniques presented in the approach section are applied to the training and development sets to augment and resample the data.

5.2 Training

To perform the training of model, for each training example*(x,y)*, likelihood is maximized given the problem parameters and model is updated using stochastic gradient descent.

$$L_\theta(x,y) = \frac{1}{T}\sum_{t=1}^{T} p_\theta(y_t|x, y_1, \dots, y_{t-1})$$

5.3 Evaluation Metrics

When evaluating the performances of models, comparison is made between the queries they output to the target query given an input natural language question. To compare output and target queries, the following two metrics, strict accuracy and knowledge-based evaluation are used. The latter is employed in [3] and in the majority of the literature on GeoQuery. Two simpler and softer metrics are also useful for model selection and development: the Jaccard similarity and the strict Jaccard similarity.

Strict Evaluation: The share of output queries strings that perfectly match the target strings.

$$Strict = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}(y_i = \hat{y}_i)$$

Knowledge-based Evaluation: As discussed in[8], knowledge-based evaluation consists in interpreting the outputs of model as SQL queries. The metric computes the share of model outputs that both

- Correspond to a correct query to the GeoQuery geographical database
- Yield to an identical output to the target query y_i

Similar to [3], beam search is used to produce 5 model outputs per test example and retain most likely hypothesis that corresponds to correct query. Part of Scala and Java code implemented as part of [3] is used to parse and execute queries. Adapting this knowledge-based evaluation code to models is proved to be challenging. One reason for this could be that, models are initially trained to output WordPiece tokens will eventually detokenize. It is found out that some (model output, target query) tuples would not be parsed correctly by the knowledge-based evaluator, resulting in errors.

Jaccard Similarity metric: A much less strict evaluation metric, useful for development. It consists in computing the Jaccard similarity between the sets of characters of a model output string and the corresponding target query string:

$$\frac{1}{n}\sum_i Jac(y_i, \hat{y}_i)$$

Where, $Jac(y_i, \hat{y}_i) = \frac{\#(Y_i \cap \hat{y}_i)}{\#(Y_i \cup \hat{y}_i)}$, $Y_i = set(y_i)$ and $\hat{Y}_i = set(\hat{y}_i)$

Strict Jaccard metric: An evaluation metric more severe than the Jaccard similarity and closer to the strict evaluation metric, corresponding to the share of (model output, target query) tuples where the set of their characters perfectly match:

$$\frac{1}{n}\sum_i Jac\_strict(y_i, \hat{y}_i)$$

Where, $Jac\_strict(y_i, \hat{y}_i) = \mathbb{I}(Jac(y_i, \hat{y}_i) = 1)$

5.4 Experimental details

Throughout the experiments, the following hyperparameters are fixed:

- The batch size is set to 32 natural language questions - logical queries tuples
- The dropout rate between sublayers of our Transformers is set to 0.1
- Decoding method is BEAM search with 5 output hypotheses.

6. Results and Discussion

6.1 Baseline: Shallow TSP

First, a baseline model is trained, this is a Transformer Encoder-Decoder model with reduced number of parameters and as per below hyperparameter settings:

Hyperparameter	Value
d_{model}	128
d_{int}	128
N	2
Learning rate	0.0001

Table1: Baseline Hyperparameters

In Table2, results obtained by this model are reported after comparing output strings(once detokenized). This baseline model is without data recombination technique and due to format mismatch in output, knowledge base evaluation technique proved to be very hard to compute. Hence it is not reported in results.

Model	Strict	Jaccard	Jaccard Strict
Baseline TSP	0.461	0.940	0.569

Table2: Baseline TSP model results.

6.2 Data Recombination with fine-tuning:

Further tests were carried out with larger TSP architecture and different data recombination strategies. Here the goal was to evaluate benefits of data recombination techniques and larger models in GeoQuery Dataset. Following training technique is employed here:

- Training the model on the GeoQuery train and development sets where the number of examples is doubled by selecting a data recombination technique. Training is limited to 200 epochs and learning rate is fixed.
- Then model is finetuned on original GeoQuery train set with learning rate divided by 10. Training epochs is limited to 75.

Hyperparameters of this model are specified in Table3 and results of this model are specified in Table4.

Hyperparameter	Value
d_{model}	512
d_{int}	512
N	6
Learning rate	0.00001

Table3: Large TSP model Hyperparameters

Discussion: From this series of experiments, it is evident that poor performance of Large TSP models on GeoQuery dataset because of limited number of training examples available. Data recombination technique seems to be limiting the number of recombined examples to twice the size of original dataset is also proving to be restrictive. Nesting and Concat-2 recombination techniques have early stopped model below 200 epochs of training.

Recombination	Strict	Knowledge-based	Jaccard	Jaccard strict
No recombination	0.126	0.150	0.881	0.215
Entity	0.175	0.245	0.900	0.272
Nesting	0.175	0.241	0.890	0.272
Concat-2	0.150	0.180	0.888	0.200

Table4: Results for Large TSP model with data recombination

6.3 Large BSP and TSP Model

In this experiment, performance and benefits of using BERT as the encoder are studied and TSP and BSP are compared for GeoQuery dataset.

BERT encodings are fixed to an embedding size of 768 as per [2]. Adaptive learning rate is employed using Adam optimizer as per [1], on Transformers: learning rate increases for first 4000 optimization steps and decreases afterwards.

Since the learning rate is adaptive for BSP model, a new TSP model with fixed learning rate is also exposed since it performed better in previous experiment. Models are trained using training + fine-tuning procedure described in previous section. Only *entity* data recombination technique is used since the other two techniques have cause overfitting in Large TSP model.

Basis hyperparameters are not changed as per Table4 and results are specified in Table6 as per below:

Model	Strict	Knowledge Based	Jaccard	Jaccard Strict
TSP Fixed	0.175	0.245	0.900	0.272
TSP Adaptive	0.080	0.140	0.840	0.110

BSP Adaptive	0.262	0.379	0.930	0.435
Relative improvement	50%	55%	3%	60%

Table5: Results for Large BSP and TSP model with fixed and adaptive learning rate.

Discussion: Results presented in Table5 show significant improvements brought by BSP model as result of using BERT as encoder. There is 50% improvement in strict accuracy compared to TSP model. Since BERT is pre-trained architecture, the limited amount of data in GeoQuery dataset is less of a problem to fine-tune the model. However, these results are still below the shallow model baseline as per experiment in 6.1 section. This is because, a much larger Transformer decoder must be trained from scratch on limited available data. Finally changing the learning rate for TSP from fixed to adaptive did not help here.

Several BSP models with varying embedding size for decoder Transformer {64, 128, 256} have been tried with fixed and adaptive learning rates for the optimizer but none of those models have been able to learn properly and produce satisfying results. This might be due to significant loss of information through linear projection that is introduced between fixed BERT encoder outputs of size 768 and decoder input which is significantly less.

7. Conclusion and Future work

This project has illustrated how transformer based encoder-decoder model can be applied to semantic parsing task using GeoQuery dataset. By training a larger model on GeoQuery, significant improvements we obtained through BERT encoder. However, simple linear projection was not sufficient to properly link BERT to a shallow Transformer Decoder, so shallow network architectures can be tried to improve the results. So as future work following steps can be explored to improve the models and thereby results,

- Experiment with shallow TSP model by combining all data recombination techniques to increase the training set size to improve results from baseline model.
- Experiment with shallow but wide transformer decoder with BERT as encoder by keeping dimension of decoder close to BERT dimension.
- Try even more data recombination techniques to generate more training samples, thereby improving baseline and larger models results even further.
- Further research to made to explore ways to pre-train the decoder network rather than training from scratch.
- Test the models on other semantic parsing datasets such as ATIS and Overnight datasets for generalization.

By working on this project, I have learned a lot about transformer implementations and BERT model in general. I realized how sensible model implementation can be and learned a lot about model building and hyper parameter tuning for deep learning models. Semantic parsing task is interesting challenge to work on, but however there is lot of scope to increase the amount of data and design new data recombination techniques.

Acknowledgements

I would like to thank all the authors who have generously uploaded their literature in internet without which it would have never possible for me implement this project and finally my course instructor Cornelia Caragea for all the excellent lecture videos and presentations which have helped me very much to learn about Deep Learning architectures in Natural Language Processing.

References

[1] Ashish Vaswani et al. “Attention Is All You Need”. In: CoRR abs/1706.03762 (2017). arXiv:1706.03762. URL: http://arxiv.org/abs/1706.03762.

[2] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: CoRR abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[3] Robin Jia and Percy Liang. “Data Recombination for Neural Semantic Parsing”. In: CoRRabs/1606.03622 (2016). arXiv: 1606.03622. URL: http://arxiv.org/abs/1606.03622.

[4]Luke S. Zettlemoyer and Michael Collins. “Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars”. In: (2005), pp. 658–666.

[5]Yonghui Wu and al. “Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation”. In: CoRR abs/1609.08144 (2016). arXiv: 1609.08144. URL: http://arxiv.org/abs/1609.08144.

[6]https://github.com/huggingface/pytorch-pretrained-RTdoc

[7] http://nlp.seas.harvard.edu/2018/04/03/attention.html

[8] Percy Liang, Michael I. Jordan, and Dan Klein. “Learning Dependency-Based Compositional Semantics”.

In: (2011), pp. 590–599. URL: http://www.aclweb.org/anthology/P11-1060.
[9]https://www.cs.utexas.edu/users/ml/nldata/geoquery.html
[10]Arnaud Autef, Simon Hagege. “Compositonal Pre-training for Semantic Parsing using BERT”. URL
[11]Junhui Li, Muhua Zhu, Wei Lu, Guodong Zhou. “Improving Semantic Parsing with Enriched Synchronous Context-Free Grammar” URL: https://www.aclweb.org/anthology/D15-1170.pdf