# CS 224N - Project Proposal
# Compositional Pre-Training for Semantic Parsing with BERT

**Arnaud Autef**
Stanford Unversity
arnaud15@stanford.edu

**Simon Hagege**
Stanford Unversity
hagege@stanford.edu

## Abstract

Semantic parsing - the conversion of natural language utterances to logical forms - is a typical natural language processing technique for extracting the meaning of an input sentence. Its applications cover a wide variety of tasks, such as question answering, machine translation, instruction following or regular expression generation. In our project, we will implement a new sequence-to-sequence model for semantic parsing built on a pre-trained bidirectional Transformer BERT (Devlin et al., 2018) encoder and a recurrent neural network (RNN) or Transformer (Vaswani et al., 2017) decoder. The encoder-decoder architecture will be fine-tuned on the semantic parsing data using data recombination techniques described in (Jia and Liang, 2016). The main contribution of our work will be the use of BERT and transformer-based structure in the semantic parsing setting and the analysis of its potential benefits on such tasks. We will be mentored by Robin Jia from Stanford University's Computer Science Department.

## 1 Related work

In this section, we review the research papers from which our project is built. First, we present the data recombination technique introduced in (Jia and Liang, 2016) to improve the performances of their encoder-decoder architecture on semantic parsing. Second, we review the Transformers deep learning model, proposed first in (Vaswani et al., 2017). Finally, we describe the state-of-the-art BERT technique (Devlin et al., 2018) for sentence encoding.

### 1.1 Data recombination and semantic parsing

Data recombination, or compositionally-generated data, is a novel data augmentation technique proposed in (Jia and Liang, 2016) to improve the performance of neural encoder-decoder architectures on semantic parsing tasks. This paper provides a useful state-of-the art approach to build a pre-training generative model and produce training data according to logical rules given a priori.

**Motivation and problem statement:** One challenge of semantic parsing is building logical properties that model conditional independence. In many cases, part of the meaning of a sentence (consider *what states border Texas?*) is indeed independent of some words in the sentence (here *Texas* is replaced by any other state, the type of question remains the same). Expanding data augmentation techniques, data recombination provides a framework to model these invariances and conditional independence by injecting prior knowledge. Namely, this setting introduces a high-precision generative model from the training data, and in a second time, sample from it to generate new training examples.

**Setting:** Given a training set $\mathcal{D}$ 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 $\mathbb{E}(p_\theta(y|x))$ where $(x, y)$ is drawn from $\tilde{p}$.

**SCFGs for semantic parsing:** The generative model is based on synchronous context-free grammar (SCFG), *i.e* a set of production rules $X \rightarrow \langle \alpha, \beta \rangle$ where $X$ is a category (non-terminal) and $\alpha$ and $\beta$ are sequences of terminal and non-terminal symbols. The SCFG will be used to represent joint derivation of utterances $x$ and logical forms $y$. First, the dataset $\mathcal{D}$ is encoded as an initial

grammar with rules ROOT $\rightarrow \langle x, y \rangle$. Next, each grammar induction is defined as a mapping from an input grammar $G_{\text{in}}$ to $G_{\text{out}}$. Three strategies are proposed in the paper:

1. Abstracting Entities: abstracting entities with their types, based on predicates in the logical form (such as `stateid`). For each $X \rightarrow \langle \alpha, \beta \rangle$ in $G_{\text{in}}$, 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 $\rightarrow \langle$"$texas$", `texas`$\rangle$).

2. Abstracting Whole Phrases: abstracting both entities and whole phrases with their types. The first rule added is the same as previously. The second one maps, if possible, the whole expression $\beta$ (the logical answer) to a particular type.

3. Concatenation: combining $k$ sentences into one single rule ROOT. Unlike the two other methods, it does not introduce any additional information about compositionality.

Finally, each grammar induction strategies can be composed to produce more complex grammars, applying the mathematical operator $f_1 \circ f_2$.

**Performance:** The model was evaluated on the GEOQUERY (Zettlemoyer and Collins, 2005), ATIS (Zettlemoyer and Collins, 2007) and OVERNIGHT (Wang and Yang, 2015) datasets. Compellingly, the model proposed in (Jia and Liang, 2016) achieves state-of-the art results on GEO, with a test accuracy of **89.3**, only behind (Liang et al., 2011).

### 1.2 Transformers

We now describe an innovative neural architecture for natural language processing tasks which has received a lot of attention recently, based solely on attention mechanisms and avoiding recurrent and convolutional structure. It is presented in (Vaswani et al., 2017), this paper will help us understanding the BERT (Devlin et al., 2018) pre-training Encoder as well as developing a Transformer-based Decoder.

**Motivation:** RNNs, long short-term memory networks (LSTM) and gated recurrent units (GRU) are popular neural architecture for NLP tasks. However, their computations are sequential by nature and cannot be performed in parallel. This becomes critical for longer sequence lengths, as memory constraints limit batching across examples. Also, the addition of attention models had become very popular with RNNs to improve their performances. Thus, the idea behind the Transformer architecture is to come up with a model relying exclusively 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.

**Model architecture:** 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.

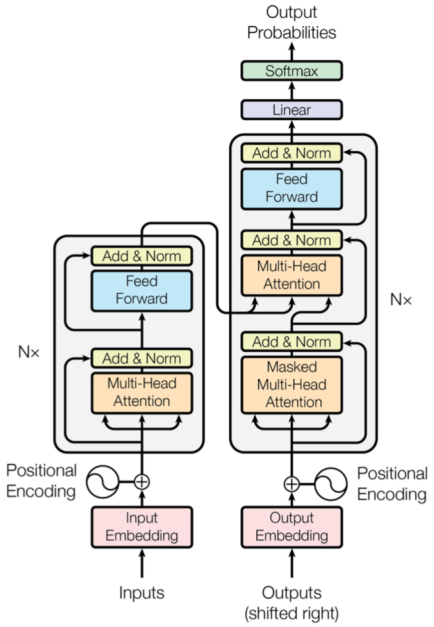


Figure 1: The Tranformer - architecture

- Encoder: stack of 6 identical layers, each divided in two sublayers: one multi-head attention and one fully-connected feed-forward network, both linked with residual connections followed by layer normalization

- Decoder: stack of 6 identical layers, same sublayers as the Encoder and a third sublayer performing multi-head attention on the output of the encoder stack. The first layer in

the stack is masked to make sure the attention prediction at position $i$ is based on positions before $i$.

**Multi-head attention:** Instead of performing a sequential dot-product-based attention, the paper proposes to perform linear projection of the queries, keys and values $h$ times with different, learned linear projection projections to $d_v$, $d_k$ and $d_k$. On each of the projection, the attention function is computed in parallel and then, all attention computations are calculated:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O$$
$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

where $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$ and $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$.
Multi-head attention allows the model to access to information from different representation subspaces at different positions.

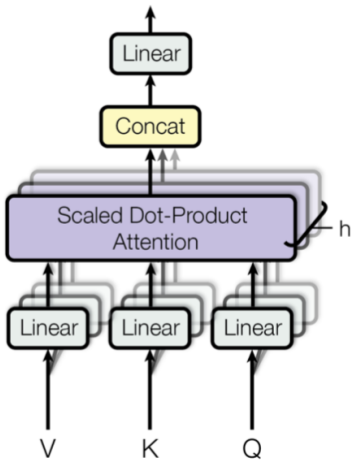


Figure 2: Multi-head attention

Considering a self-attention model to link input sequence to output sequence offers four main upsides:

- Despite a runtime quadratic in the length of the input, the total computational complexity per layer is linear in the hidden dimensions
- More computations can be done in parallel
- the maximum forward and backward path length that have to traverse the network to learn long-range dependencies is higher
- Attention distribution over output words improves interpretability.

**Training techniques:** Among others, the paper used several techniques to boost the training performance. In particular, an Adam Optimizer with hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$ was selected, with an increasing then decreasing learning rate over the course of the training. To reduce overfitting, dropout with dropout rate $P_{drop} = 0.1$ is applied to the output of each sub-layer as well as to the sums of the embeddings and the positional encodings in both the encoder and the decoder. To further regularize the model, label smoothing with $\epsilon_{ls} = 0.1$ has been performed on target one-hot word vector distributions.

### 1.3 BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

**Context and motivation:** BERT is a language representation model that can be applied to a variety of natural language processing tasks, its training and usage belong to *language model pre-training* techniques. The idea of language model pre-training is to first train a language model on text data and then use language representations learned by the model to solve a downstream natural language processing task. Two major strategies are applied by researchers in this context: *feature-based* and *fine-tuning*. BERT is a fine-tuning strategy: a minimal number of additional parameters are introduced to apply the pre-trained language model to the selected downstream task, those parameters and the pre-trained language model parameters are fine-tuned on the downstream task. The main contribution of BERT are the tasks defined to learn language representations during pre-training: *masked language model* and *next sentence prediction*. In BERT, language representations are obtained using a multi-layer, bidirectional *Transformer* encoder.

**Overview**

- **Language model architecture:** a multi-layer bidirectional Transformer encoder. For any sequence of input tokens - words - the model outputs vector representations of those tokens learned from their *bidirectional context*. Each Transformer layer corresponds to a *multi-head* self-attention and a *feed-forward* neural network block with residual connections. In the BERT paper, two architectures are proposed, a small architecture with L=12

Transformer layers, a H=768 hidden size and A=12 attention heads for total of 110M parameters; a large architecture with 24 Transformer layers, size 1024 hidden layers and 16 attention heads, for a total of 340M parameters.

- **Pre-training**: Datasets used are the BooksCorpus (800M words) and English Wikipedia (2,500M words) and the two tasks are:
    - · *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.

  Pre-training is carried out for 40 epochs over the data, using an Adam optimizer with learning rate warm-up and linear decay. A dropout probability of 0.1 is added to all model layers.

## 2 Project outline

The goal of our work is to evaluate the performance of an encoder-decoder model with a pre-trained BERT encoder on selected semantic parsing datasets. In particular, in (Jia and Liang, 2016), an encoder-decoder architecture using recurrent neural networks (RNN) with compositionally generated data obtains state-of-the-art results on the GeoQuery dataset and, during the project we will:

1. Replace the RNN encoder by a pre-trained BERT bidirectional Transformer.
2. Select an appropriate decoder architecture (possibly another Transformer, or a RNN architecture similar to (Jia and Liang, 2016) to pinpoint the benefits of the BERT encoder).
3. Define and implement a fine-tuning procedure to train the encoder and decoder models while leveraging compositionally generated data.
4. Study the results obtained.

### 2.1 Datasets

We will first work with the following semantinc parsing datasets described in (Jia and Liang, 2016):

- GeoQuery (GEO) contains natural language questions about US geography paired with corresponding Prolog database queries.
- ATIS (ATIS) contains natural language queries for a flights database paired with corresponding database queries written in lambda calculus.
- Overnight (OVERNIGHT) contains logical forms paired with natural language paraphrases across eight varied subdomains.

### 2.2 Experimental approach

1. Start from a BERT bidirectional Transformer encoder pre-trained on the BooksCorpus and English Wikipedia datasets.
2. Select (and, possibly, create new) SCFG rules to apply on each semantic parsing dataset.
3. Apply the training algorithm decribed in Figure 4. of (Jia and Liang, 2016) with selected SFCG rules to fine-tune the BERT encoder and learn the parameters of the decoder (Transformer, RNN) on the train set of each semantic parsing dataset.
4. Observe the accuracy of our encoder-decoder model on GeoQuery, ATIS and Overnight's test sets.
5. Discuss and analyze our results. An interesting idea would be to carry out appropriate ablation studies to identify the effects of the pre-trained BERT encoder.

### 2.3 Objectives and Questions:

At this point, our primary objective is to successfully implement our proposed encoder-decoder model on GeoQuery, ATIS and Overnight and obtain performances consistent with similar models in the litterature. Then, we will conduct additional experiments to try and explain those results. Many questions already come to our mind from this proposed workplan:

- The *masked language model* task during the BERT pre-training allows this Transformer encoder to learn bidirectional language representations. Is this new feature of our encoder going to improve performances significantly and on all our target semantic parsing datasets?

- The *next sentence prediction* task during the BERT pre-training does not seem to be relevant to our encoder-decoder architecture, as we intend to feed a single input sentence at a time to the BERT encoder. Is it going to be detrimental to our overall model's performances? It would also be interesting if we could find a way to change our encoder-decoder architecture to leverage this pre-training task.

- What kind of architecture should we select for the decoder? Implementing a Bidirectional LSTM with the same parameters as (Jia and Liang, 2016) looks interesting to identify the contribution of the BERT encoder to performances, but a Transformer decoder such as in (Vaswani et al., 2017), coupled with the BERT pre-training, may lead to better performances.

- The one-step fine tuning method we describe looks sensible, but other options seem relevant too. As suggested by Robin Jia, we could fine-tune in two step: first fine-tune the encoder-decoder model on compositionally-generated data, then fine-tune on the original data only.

## Acknowledgments

Throughout this project, we will be advised by Robin Jia from Stanford University's Computer Science department.

## References

[Devlin et al.2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

[Jia and Liang2016] Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. *CoRR*, abs/1606.03622.

[Liang et al.2011] Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. pages 590–599.

[Vaswani et al.2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

[Wang and Yang2015] William Yang Wang and Diyi Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using #petpeeve tweets. pages 2557–2563.

[Zettlemoyer and Collins2005] Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. pages 658–666.

[Zettlemoyer and Collins2007] Luke S. Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. pages 678–687.