Encoder-Decoder Models

Advanced ML with TensorFlow on GCP

End-to-End Lab on Structured Data ML

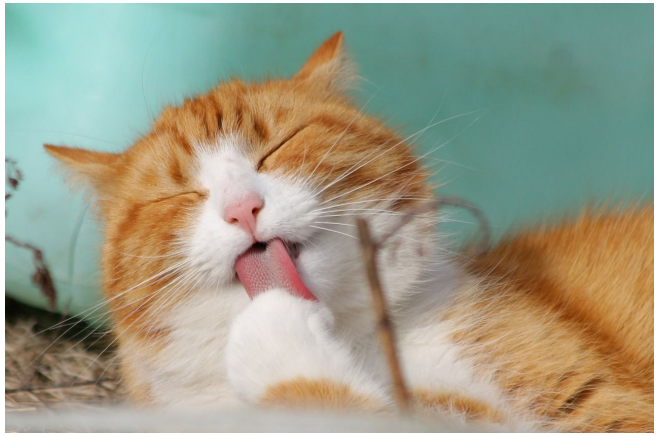
Production ML Systems

Image Classification Models

Sequence Models

Recommendation Systems

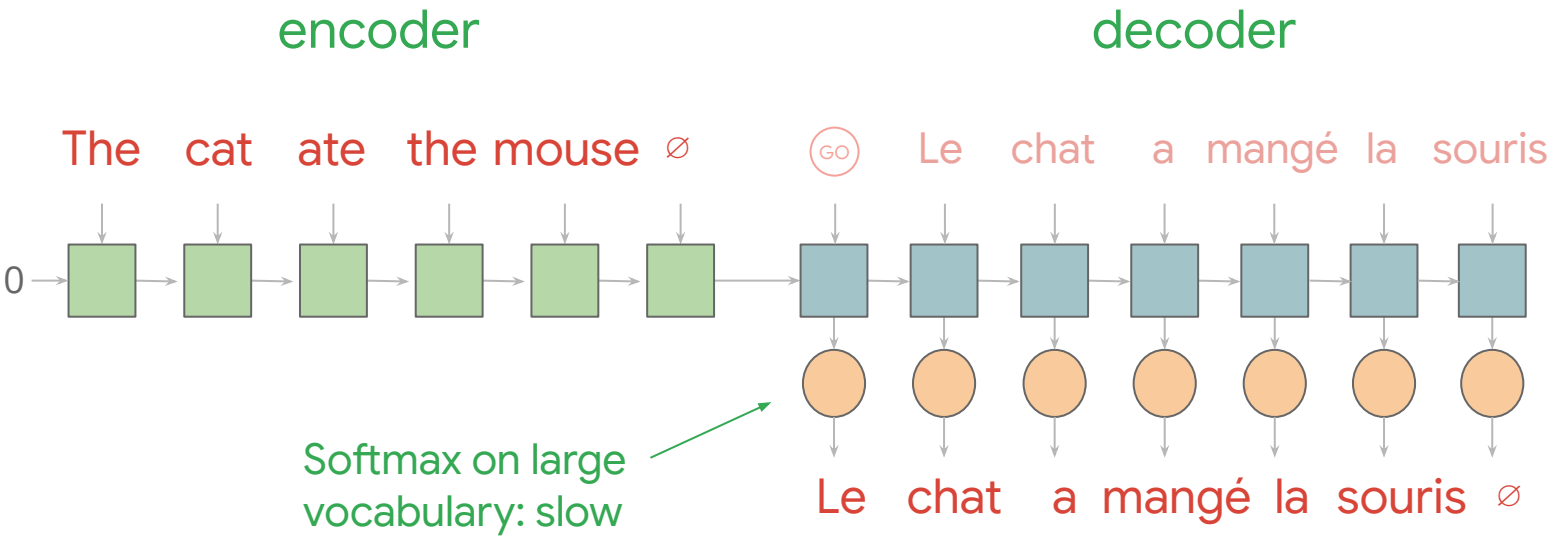
Translate English sentence to French



The cat ate the mouse ∅

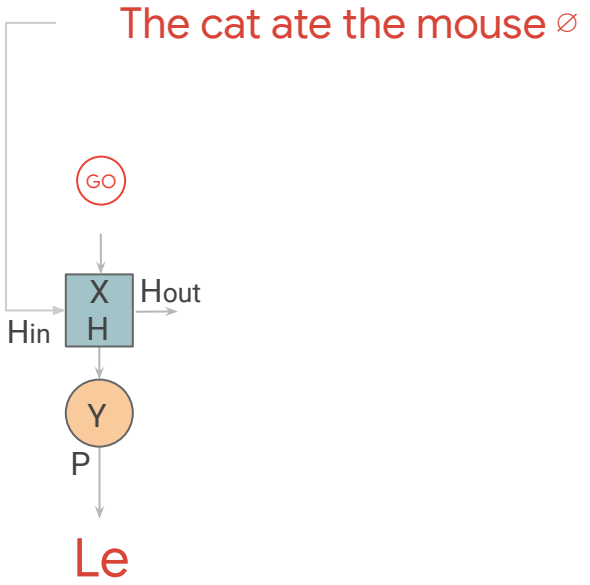
Le chat a mangé la souris ∅

Encoder decoder networks



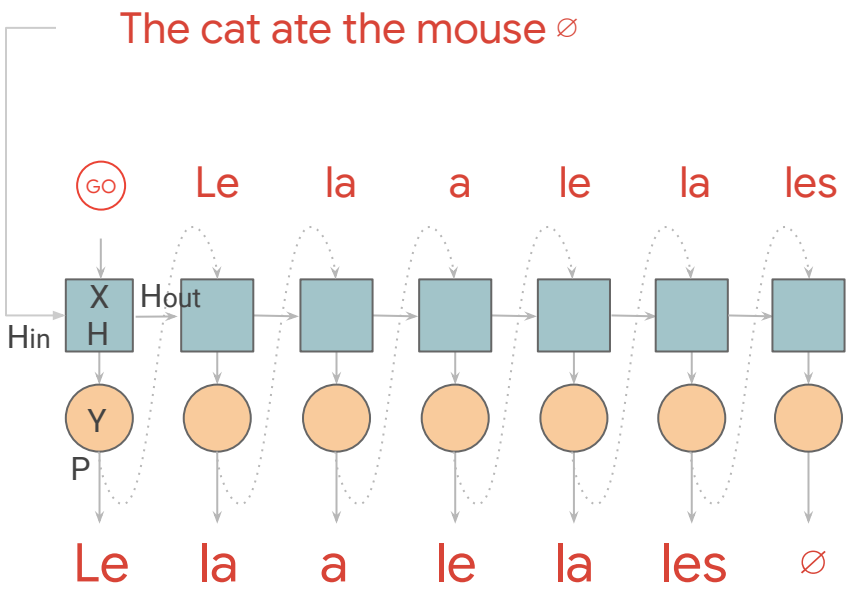
fast → `tf.nn.sampled_softmax_loss(...)`

Greedy search versus Beam search




```
X = tf.nn.embedding_lookup(embeddings, inWord)

H, Hout =
  tf.nn.dynamic_rnn(cell, X, initial_state=Hin)

Y = tf.layers.dense(H, vocab_size)

P = tf.nn.softmax(Y)

outWord = tf.nn.argmax(P)
```

Greedy search versus Beam search



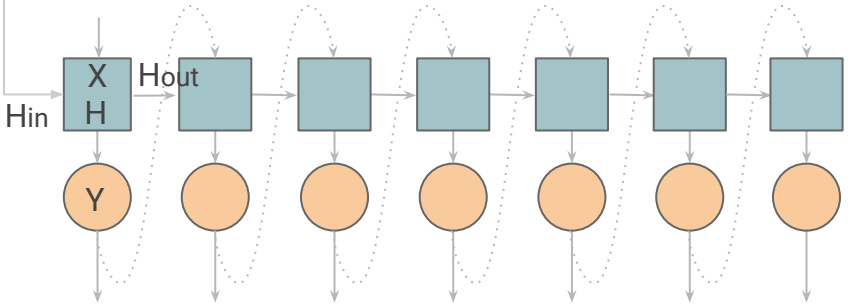

```
X = tf.nn.embedding_lookup(embeddings, inWord)

H, Hout =
  tf.nn.dynamic_rnn(cell, X, initial_state=Hin)

Y = tf.layers.dense(H, vocab_size)

P = tf.nn.softmax(Y)

outWord = tf.nn.argmax(P)
```

Bad idea

Greedy search versus Beam search

The cat ate the mouse ∅

GO Le chat a mangé la souris



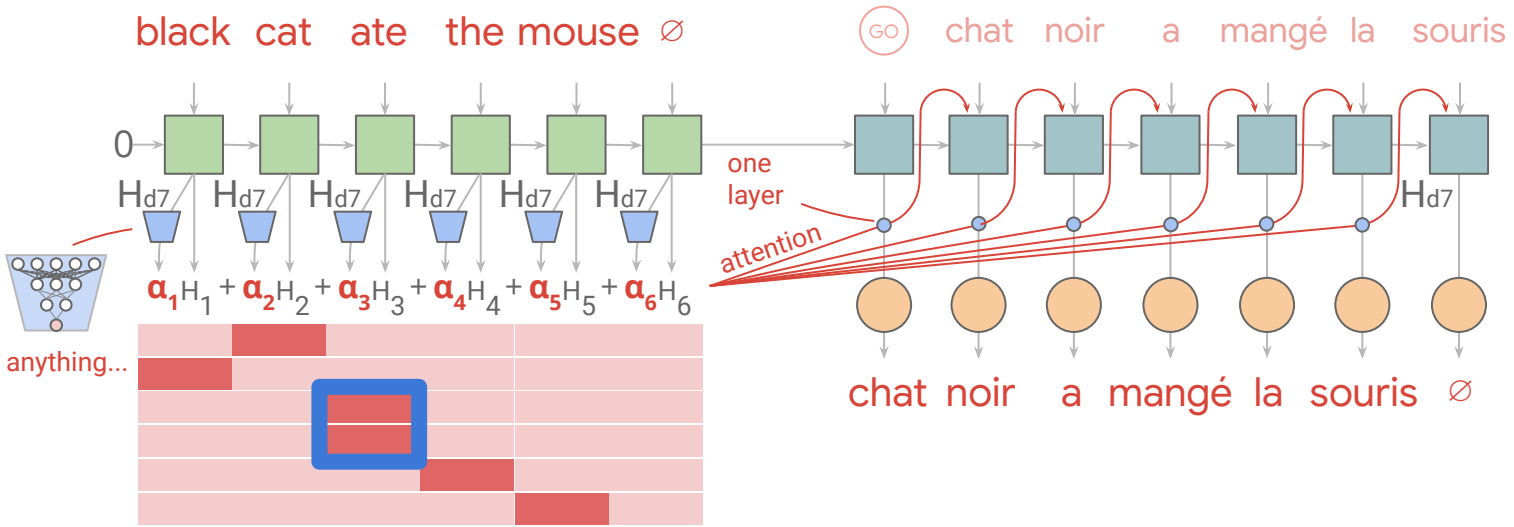
Le chat a mangé la souris ∅

```
X = tf.nn.embedding_lookup(embeddings, inWord)

H, Hout =
  tf.nn.dynamic_rnn(cell, X, initial_state=Hin)

Y = tf.layers.dense(H, vocab_size)

P = tf.nn.softmax(Y)
```

~~outWord = tf.nn.argmax(P)~~ Bad idea

→ Use tf.contrib.seq2seq.BeamSearchDecoder

Improve the translation with attention network



Four ML steps to train an encoder-decoder model

1 Embed

Turn words into numeric tensors.

Pretrained embeddings as a starting point, then continue to train.

Save vocabularies and embeddings for prediction.

2 Encode

Use multilayer GRUs to save memory and state.

Tune the GRUs carefully.

Make sure that input sequences are rich enough for the model to learn long-term rules.

3 Attend

Dynamically weight the outputs based on the current input word.

The number of attention heads is an important hyperparameter.

4 Predict

NEXT!

TensorFlow provides APIs for many of these steps

```
x = tf.nn.embedding_lookup(embeddings, sentences)
```

```
encoder_cell = tf.rnn.GRUCell(encoding_dimension)
wrapped_cell = tf.rnn.DropoutWrapper(encoder_cell, output_keep_prob=p)
encoded_sentences, encoder_state = tf.nn.dynamic_rnn(wrapped_cell, x)
```

```
decoder_cell = tf.rnn.GRUCell(encoding_dimension)
decoder = tf.seq2seq.BeamSearchDecoder(decoder_cell, embeddings,
                                       sos_tokens, eos_token, encoder_state, beam_width)
outputs, final_state, _ = tf.seq2seq.dynamic_decode(decoder, maximum_iterations=max_length)
```

Embed

Encode

Attend

Predict

Translation with attention

Embed

Encode

Attend

Predict

```
inattentive_decoder_cell = tf.rnn.GRUCell(encoding_dimension)

attention_mechanism = tf.seq2seq.LuongAttention(encoding_dimension, encoded_sentences)

decoder_cell = tf.seq2seq.AttentionWrapper(inattentive_decoder_cell, attention_mechanism)
```

Replace decoder_cell

```
x = tf.nn.embedding_lookup(embeddings, sentences)
```

```
encoder_cell = tf.rnn.GRUCell(encoding_dimension)
wrapped_cell = tf.rnn.DropoutWrapper(encoder_cell, output_keep_prob=p)
encoded_sentences, encoder_state = tf.nn.dynamic_rnn(wrapped_cell, x)
```

```
decoder = tf.seq2seq.BeamSearchDecoder(decoder_cell, embeddings,
                                       sos_tokens, eos_token, encoder_state, beam_width)
outputs, final_state, _ = tf.seq2seq.dynamic_decode(decoder, maximum_iterations=max_length)
```

Embed

Encode

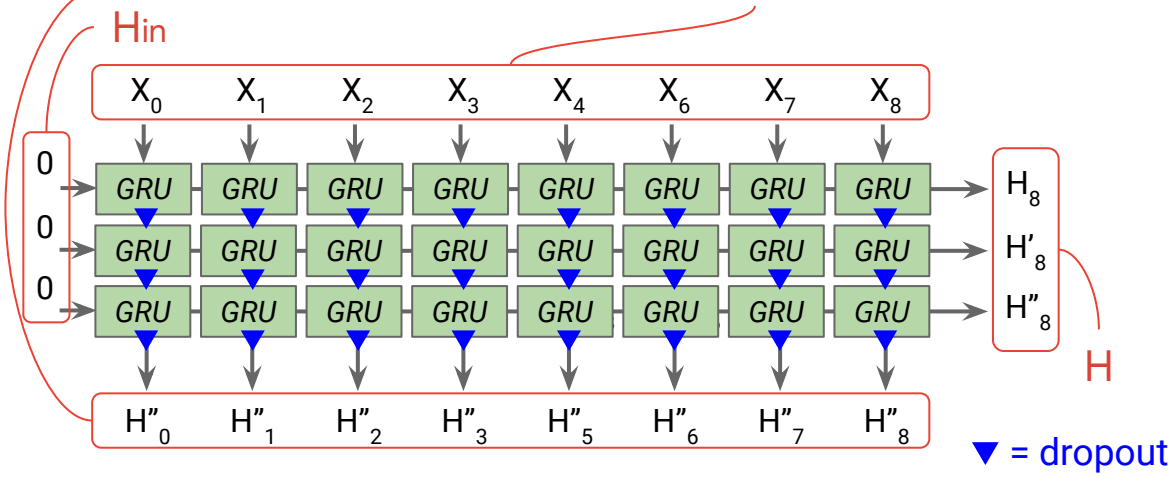
Attend

Predict

Dropout for multi-layer GRUs

```
cells = [tf.nn.rnn_cell.GRUCell(CELLSIZE) for _ in range(NLAYERS)]

cells = [tf.nn.rnn_cell.DropoutWrapper(cell, output_keep_prob=pkeep) for cell in cells]

mcell = tf.nn.rnn_cell.MultiRNNCell(cells, state_is_tuple=False)

Hr, H = tf.nn.dynamic_rnn(mcell, X, initial_state=Hin)
```

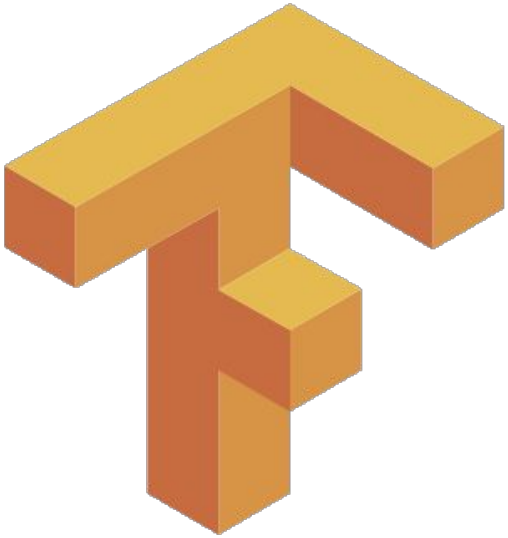
Dropout on outputs



Typical params for character-based language model handling ~100 characters.

```
CELLSIZE = 512
NLAYERS =   3
SEQLEN =  30
```

tensor2tensor comes to the rescue



github.com/tensorflow/tensor2tensor

Use tensor2tensor to complete lines of poetry

I wandered lonely as a cloud

Model

That floats on high o'er vales and hills,

A **Problem** ties together all the pieces that make up an ML system

```
@registry.register_problem
class PoetryLineProblem(translate.Text2TextProblem):
  @property
  def targeted_vocab_size(self):
    return 2**12  # 4096

  def generate_samples(self, data_dir, tmp_dir, dataset_split):
    ...
            yield {
                "inputs": prev_line,
                "targets": curr_line
            }
        prev_line = curr_line
```

Reuse Text2Text Problem except what we explicitly override.

Most common 4096 words form vocabulary.

Yield inputs, targets.

Generate training data after the preprocessing

```
PROBLEM=poetry_line_problem
t2t-datagen \
  --t2t_usr_dir=./poetry/trainer \
  --problem=$PROBLEM \
  --data_dir=$DATA_DIR \
  --tmp_dir=$TMP_DIR
```

T2t-datagen comes with tensor2tensor and creates tf.train.Example protobufs for fast training with TensorFlow.

Customize the sequence-to-sequence model

```
@registry.register_hparams
def transformer_poetry():
  hparams = transformer.transformer_base()
  hparams.num_hidden_layers = 2
  hparams.hidden_size = 128
  hparams.filter_size = 512
  hparams.num_heads = 4
  hparams.attention_dropout = 0.6
  hparams.layer_prepostprocess_dropout = 0.6
  hparams.learning_rate = 0.05
  return hparams
```

Making the model smaller.

Increasing dropout regularization.

Lower training rate.

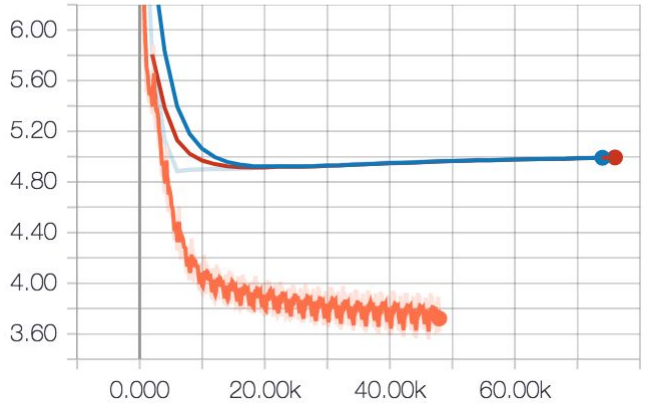
Train the model locally and on Cloud ML Engine

```
PROBLEM=poetry_line_problem
t2t-trainer \
  --data_dir=gs://${BUCKET}/poetry/subset \
  --t2t_usr_dir=./poetry/trainer \
  --problem=$PROBLEM \
  --model=transformer \
  --hparams_set=transformer_poetry \
  --output_dir=$OUTDIR \
  --train_steps=7500 --cloud_mlengine --worker_gpu=1
```

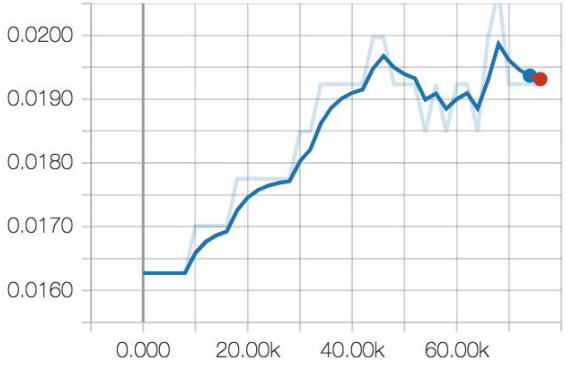
We don't have to use gcloud. The tensor2tensor library makes a REST API call to Cloud ML Engine to submit the training job.

Monitor training on TensorBoard

loss



metrics-
poetry_line_problem/accuracy_per_sequence



To hyperparameter tune on Cloud ML Engine, specify the hyperparameter ranges within the Problem definition

```
# hyperparameter tuning ranges
@registry.register_ranged_hparams
def transformer_poetry_range(rhp):
  rhp.set_float("learning_rate", 0.01, 0.2, scale=rhp.LOG_SCALE)
  rhp.set_int("num_hidden_layers", 2, 4)
  rhp.set_discrete("hidden_size", [128, 256, 512])
  rhp.set_float("attention_dropout", 0.4, 0.7)
```

Register the ranges within which to explore the hyperparameters.

Launch off the hyperparameter tuning job on Cloud ML Engine

```
t2t-trainer \
  --data_dir=gs://${BUCKET}/poetry/subset \
  --t2t_usr_dir=./poetry/trainer \
  --problem=$PROBLEM \
  --model=transformer \
  --hparams_set=transformer_poetry \
  --output_dir=$OUTDIR \
  --hparams_range=transformer_poetry_range \
  --autotune_objective='metrics-poetry_line_problem/accuracy_per_sequence' \
  --autotune_maximize \
  --autotune_max_trials=40 \
  --autotune_parallel_trials=4 \
  --train_steps=7500 --cloud_mlengine --worker_gpu=4
```

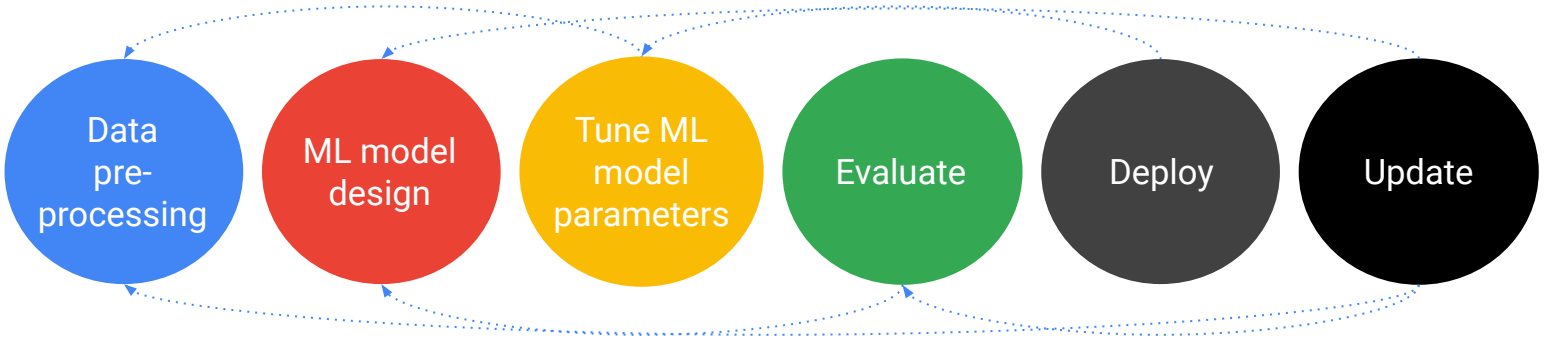
Lab

Cloud poetry: training and hyperparameter tuning custom text models on Cloud ML Engine

Lab Steps

1. Define a Problem with tensor2tensor.
2. Generate training and evaluation datasets.
3. Customize the sequence to sequence model.
4. Train the model locally and on Cloud ML Engine.
5. Fine-tune hyperparameters.
6. Decode lines of poetry.
7. Deploy the model.

Custom model building can be complex and time-intensive



Large computational resource

Machine learning expertise

Manual data labeling

Use AutoML Translation[BETA] as a pre-built model

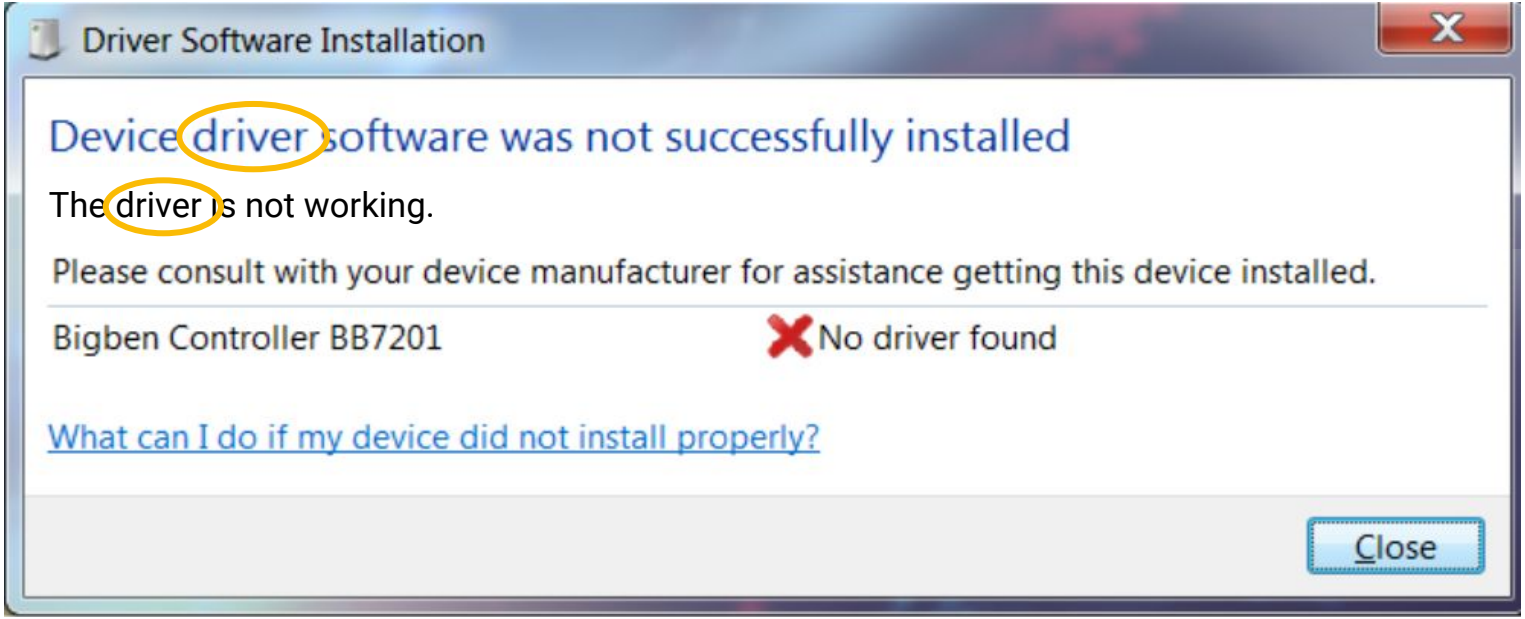
文-A AutoML Translation

1. Prepare train and test sets.
2. **Use pre-built model.**
3. Evaluate results.

Custom Models

1. Prepare train and test sets.
2. **Build model.**
3. Evaluate results.

Why is domain-specific translation important?

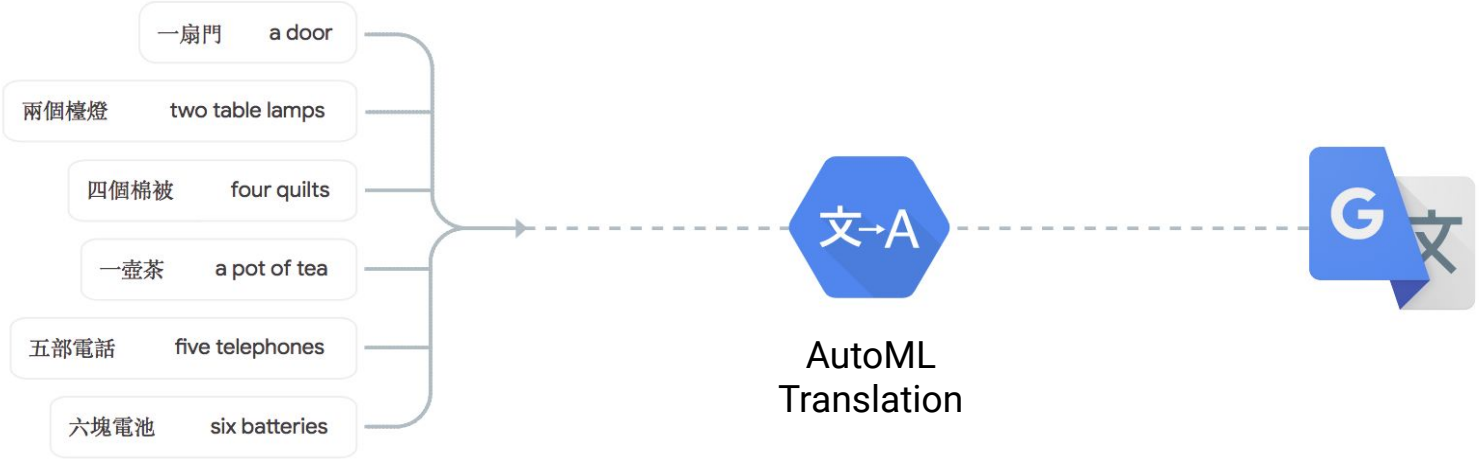


What is the correct translation?

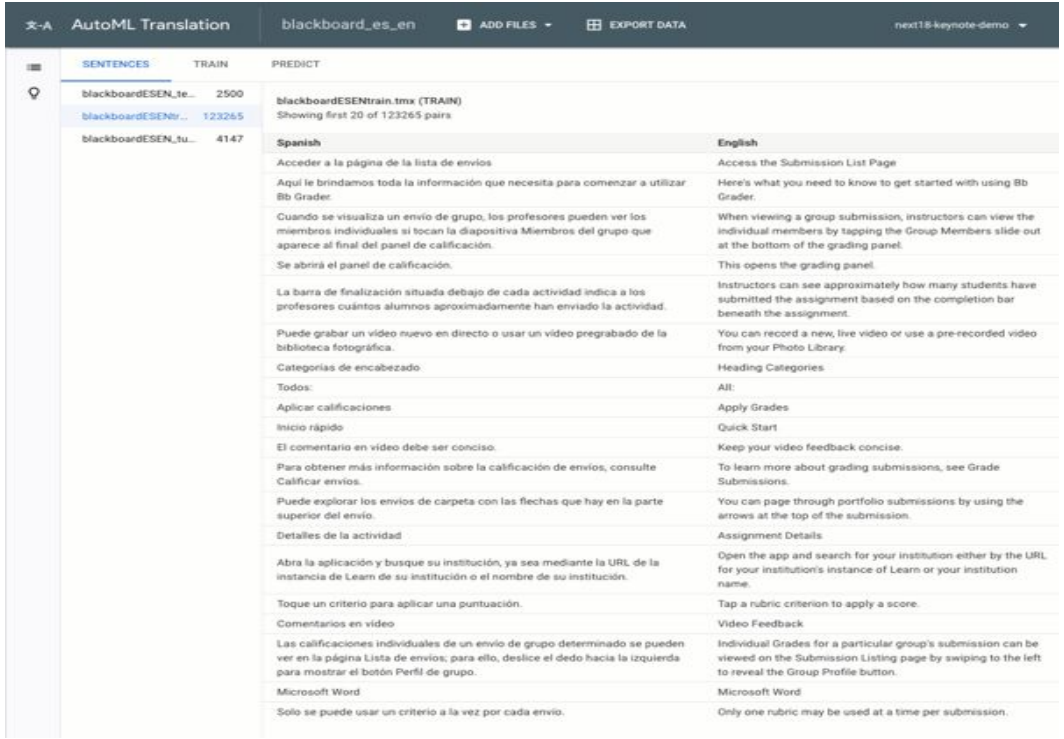
The **driver**
is not working.

How AutoML Translation[BETA] works?

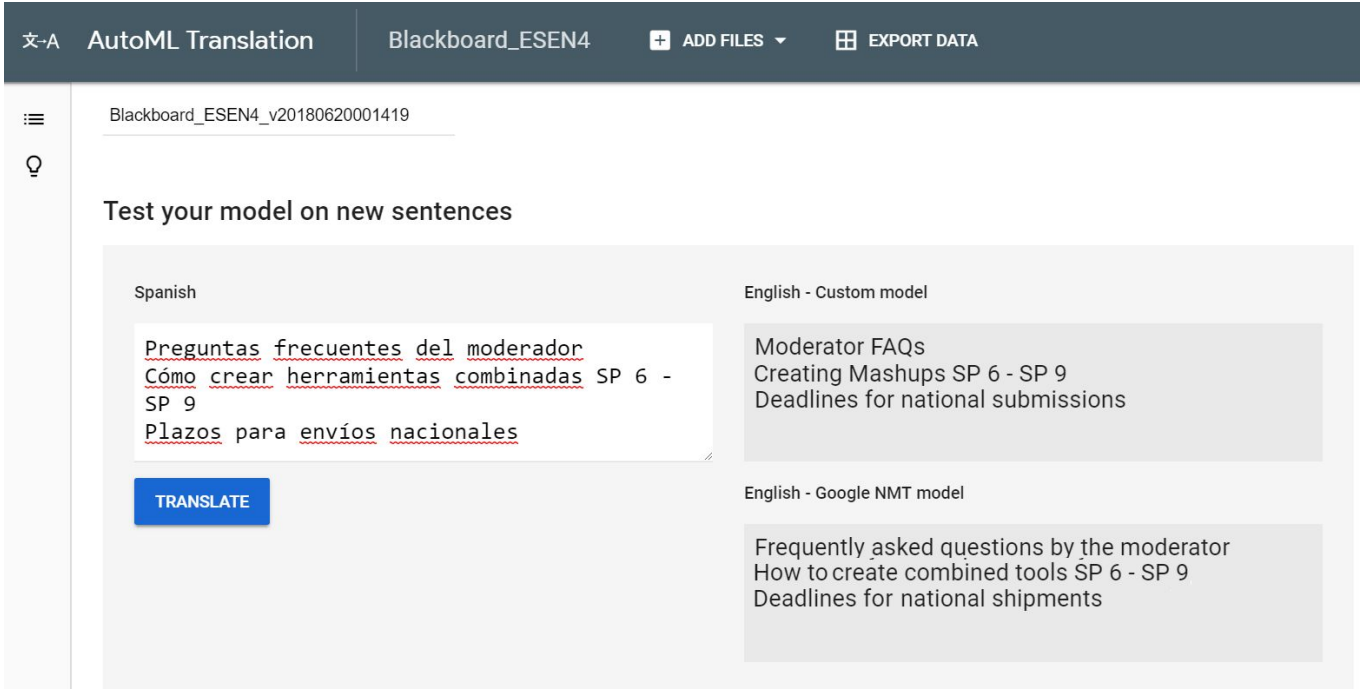
1. Upload translated language pairs
2. Train your model
3. Evaluate



How AutoML Translation^{BETA} works?

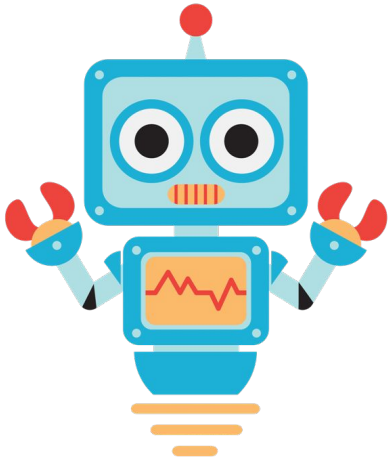


Education domain



Natural language conversations

Dialogflow



An end-to-end developer platform for building natural and rich conversational experiences.

There are three key components in Dialogflow

1. Intents
2. Entities
3. Contexts

Intents are the actions a user wants to take

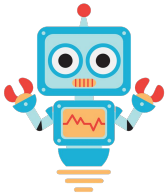
2 lattes please
Intent: Order coffee

Entities are the nouns in your dialog

I'd like an **Americano**!

Context helps the chatbot keep track

What is the weather in **San Francisco**, **today**?



Beautiful and sunny! 70 degrees.

What about **tomorrow**?

Lab

Getting started with Dialogflow

In this lab, we'll create a chatbox using Dialogflow.

Lab Steps

1. Familiarize yourself with Dialogflow console.
2. Learn to create intents.
3. Extract data with entities.
4. Manage state with contexts.
5. Test the chatbot.

cloud.google.com