

# Application of Transformers Models

MAIN NLP TASKS

- **Summary Main NLP Tasks**
- **Peer Review Baseline Models**
- **Evaluation Metrics**

# MAIN NLP TASKS

- **Token classification**
- **Fine-tuning a masked language model**
- **Translation**
- **Summarization**
- **Training a causal language model from scratch**
- **Question answering**

# TOKEN CLASSIFICATION

- **Named entity recognition (NER)**
- **Part-of-speech tagging (POS)**
- **Chunking**

# PREPROCESSING

- Tell a tokenizer to ignore a word („0-Label“) by giving it a value of „-100“
- **DataCollatorForTokenClassification():**
  - Applies dynamic padding using the specified tokenizer
  - Default is to provide a value of -100 for each padded input
  - In comparison to DataCollatorForPadding() also pads the labels

**Tokens assigned with the label “-100” are ignored when calculating the loss.**

# TRAINING THE MODEL

```
num_epochs = 3  
num_train_steps = len(tf_train_dataset) * num_epochs  
  
optimizer, schedule = create_optimizer(  
    init_lr=2e-5,  
    num_warmup_steps=0,  
    num_train_steps=num_train_steps,  
    weight_decay_rate=0.01,  
)
```

# **FINE-TUNING A MASKED LANGUAGE MODEL**

- **Main purpose:** Domain adaption
- **Unsupervised Training on a large corpus from the more special domain**
- **Typically, training for 1 epoch**

# PREPROCESSING

- A common step is to concatenate all examples (using a special EOS token between the actual samples) and then split the whole corpus into chunks of equal size.
- The size of the chunks depend on the amount of GPU memory available and on the model's maximum context size. (As, e.g., given by `tokenizer.model_max_length()`).
- After chunking `DataCollatorForLanguageModeling()` can be used to create randomly selected [MASKED] tokens
- For, e.g., whole word masking you have to implement your own data collator.

# TRAINING THE MODEL

```
num_train_steps = len(tf_train_dataset)

optimizer, schedule = create_optimizer(
    init_lr=2e-5,
    num_warmup_steps=1_000,
    num_train_steps=num_train_steps,
    weight_decay_rate=0.01,
)
```

# TRANSLATION

**A sequence-to-sequence task, similar to:**

- **Summarization**
- **Style transfer (e.g., formal to casual or Shakespearean English to modern English)**
- **Generative question answering (Generates answers to questions, given a context)**

**Used metric is typically the BLUE score (or sacreBLUE)**

# PREPROCESSING

- **Tokenization of input and targets using a Python context manager:**

```
en_sentence = split_datasets["train"][1]["translation"]["en"]
fr_sentence = split_datasets["train"][1]["translation"]["fr"]
```

```
inputs = tokenizer(en_sentence)
with tokenizer.as_target_tokenizer():
    targets = tokenizer(fr_sentence)
```

- **Using DataCollatorForSeq2Seq():**
  - **Padding the encoder input sequence as usual**
  - **Padding the labels with “-100”**
  - **Setting the decoder input sequence equal to the labels but shifted one position to the left**

# TRAINING THE MODEL

```
num_epochs = 3  
num_train_steps = len(tf_train_dataset) * num_epochs  
  
optimizer, schedule = create_optimizer(  
    init_lr=5e-5,  
    num_warmup_steps=0,  
    num_train_steps=num_train_steps,  
    weight_decay_rate=0.01,  
)
```

# SUMMARIZATION

- Preprocessing is equal to translation
- Seems to need fine-tuning for quite a few epochs (8 in the example)
- Used metric is typically the ROUGE score

# TRAINING THE MODEL

```
num_train_epochs = 8  
num_train_steps = len(tf_train_dataset) * num_train_epochs  
  
optimizer, schedule = create_optimizer(  
    init_lr=5.6e-5,  
    num_warmup_steps=0,  
    num_train_steps=num_train_steps,  
    weight_decay_rate=0.01,  
)
```

# TRAINING A CAUSAL LANGUAGE MODEL FROM SCRATCH

- Any plain text dataset with a tokenizer can be used for training
- Common metrics are:
  - Perplexity
  - Cross Entropy

# PREPROCESSING

- **Concatenate all examples and then split them into chunks of equal size (see “Fine-Tuning a Masked language Model”).**
- **Remove the last chunk if is shorter than max length to avoid padding issues**
- **After chunking DataCollatorForLanguageModeling() can be used with argument mlm=False**

# TRAINING THE MODEL

```
num_train_steps = len(tf_train_dataset)
optimizer, schedule = create_optimizer(
    init_lr=5e-5,
    num_warmup_steps=1_000,
    num_train_steps=num_train_steps,
    weight_decay_rate=0.01,
)
```

# QUESTION ANSWERING

- **Encoder models are good for extractive question answering**
- **Alternative is generative question answering using:**
  - **a decoder model or**
  - **an encoder-decoder model (more similar to text summarization)**
- **Common metrics are:**
  - **Exact Match**
  - **F1-Score**

# PREPROCESSING

- **For training only one correct answer is defined.**  
**(For evaluation multiple corrects answers are provided.)**
- **Long contexts are split with about 50% overlap, the questions remain equal for all splits.**
- **Labels for the splits have to be provided via a custom function; thereafter the DefaultDataCollator() is used.**

# TRAINING THE MODEL

```
num_train_epochs = 3  
num_train_steps = len(tf_train_dataset) * num_train_epochs  
  
optimizer, schedule = create_optimizer(  
    init_lr=2e-5,  
    num_warmup_steps=0,  
    num_train_steps=num_train_steps,  
    weight_decay_rate=0.01,  
)
```

# PROJECT MILESTONES

- 16.11. Form project groups**
- 23.11. Literature review**
- 30.11. Dataset characteristics**
- 04.01. Baseline model**
- 11.01. Project presentations**

# **BASELINE MODEL RESULTS**

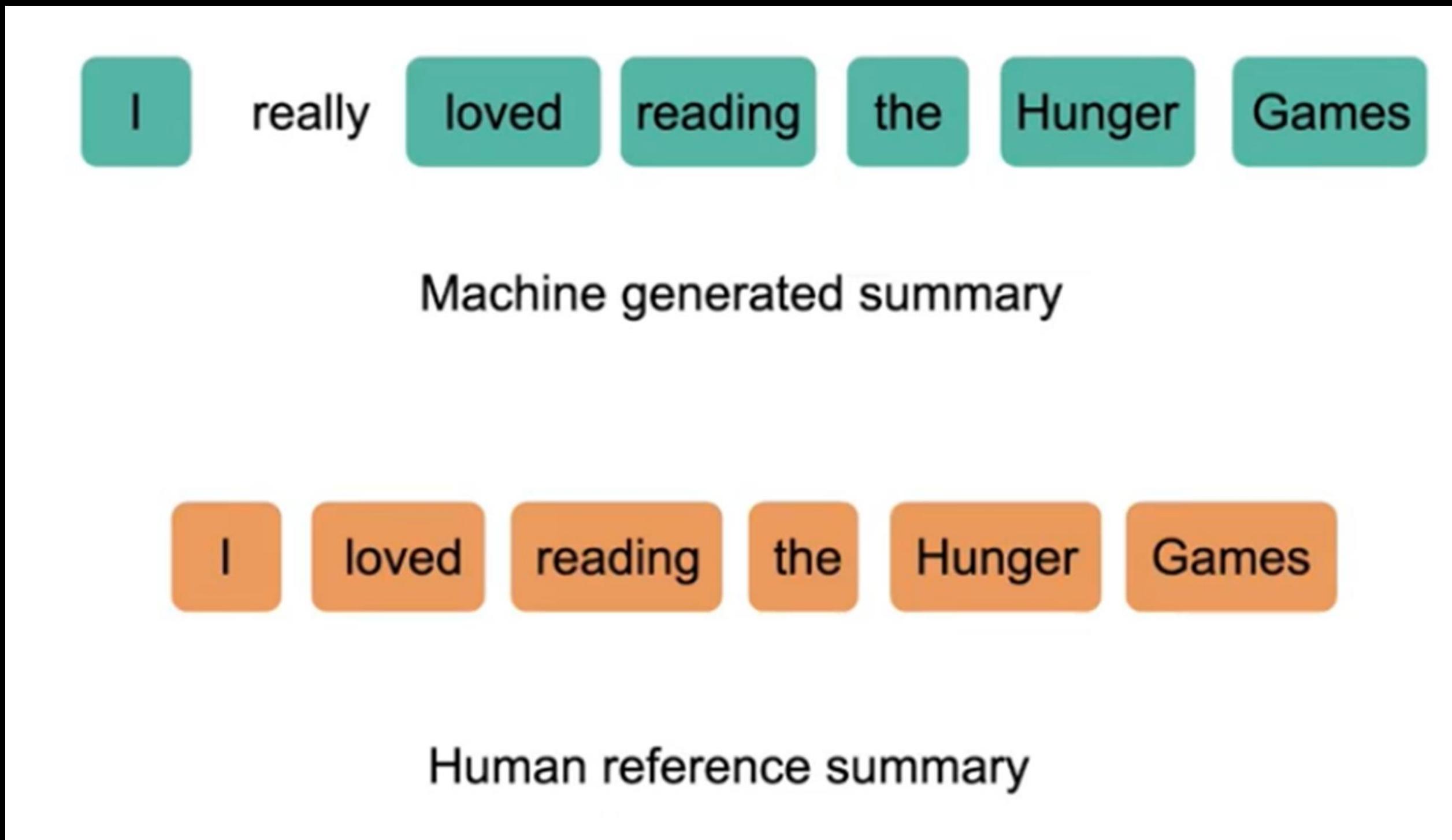
**Help you understand your data:**

- Which classes are harder to separate?**
- What type of signal picks your model up on?  
How is your model making decisions?**
- What signal is your model missing?  
Is it possible to engineer an additional feature?**

# EVALUATION METRICS FOR LANGUAGE MODELS

- ROUGE Score
- BLUE Score
- (Log-) Perplexity

# ROUGE-1 SCORE

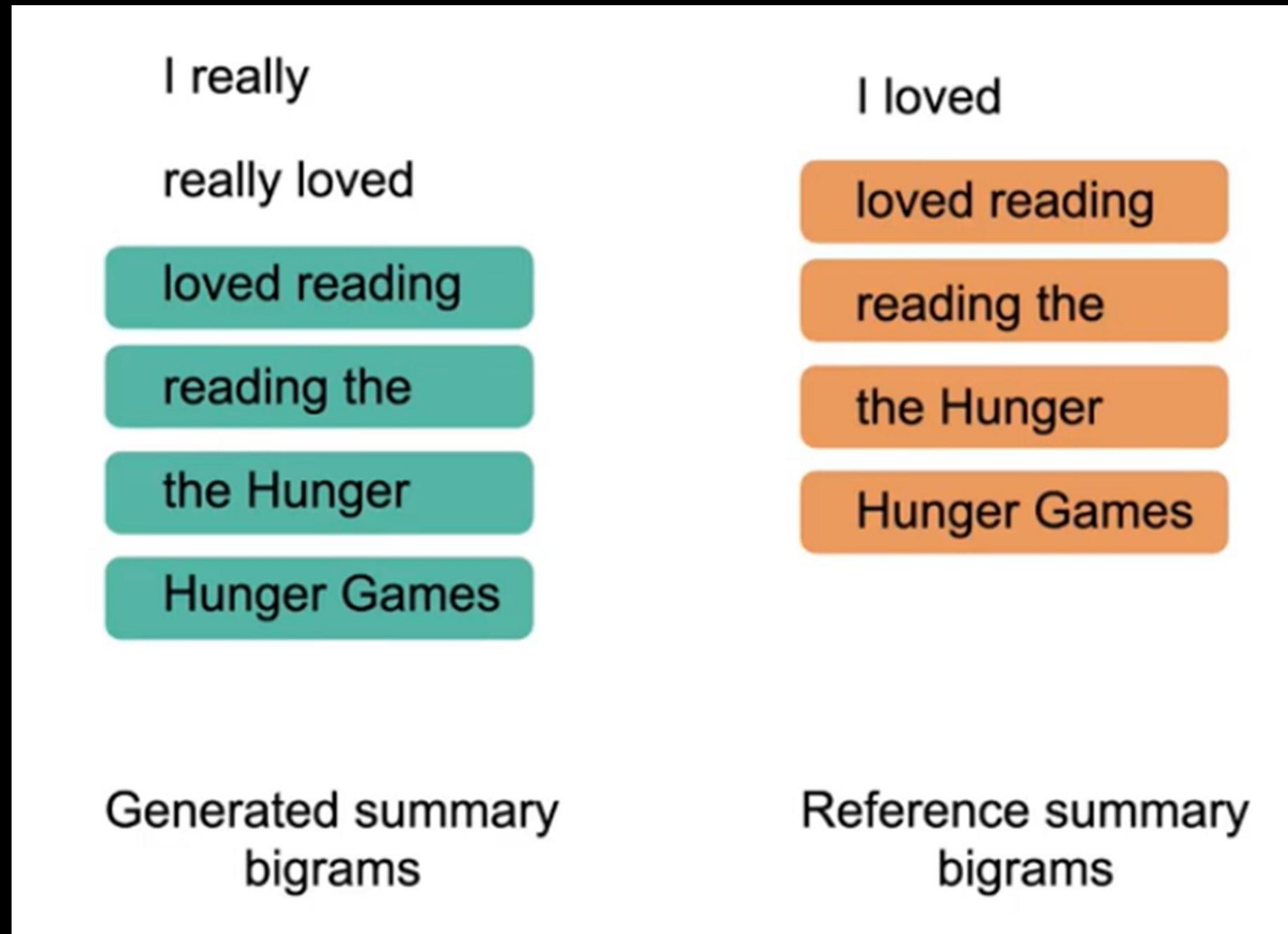


$$\text{ROUGE-1 recall} = \frac{\text{Num word matches}}{\text{Num words in reference}} = \frac{6}{6}$$

$$\text{ROUGE-1 precision} = \frac{\text{Num word matches}}{\text{Num words in summary}} = \frac{6}{7}$$

$$\text{ROUGE-1 F1-score} = \sqrt{2} \left( \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \right)$$

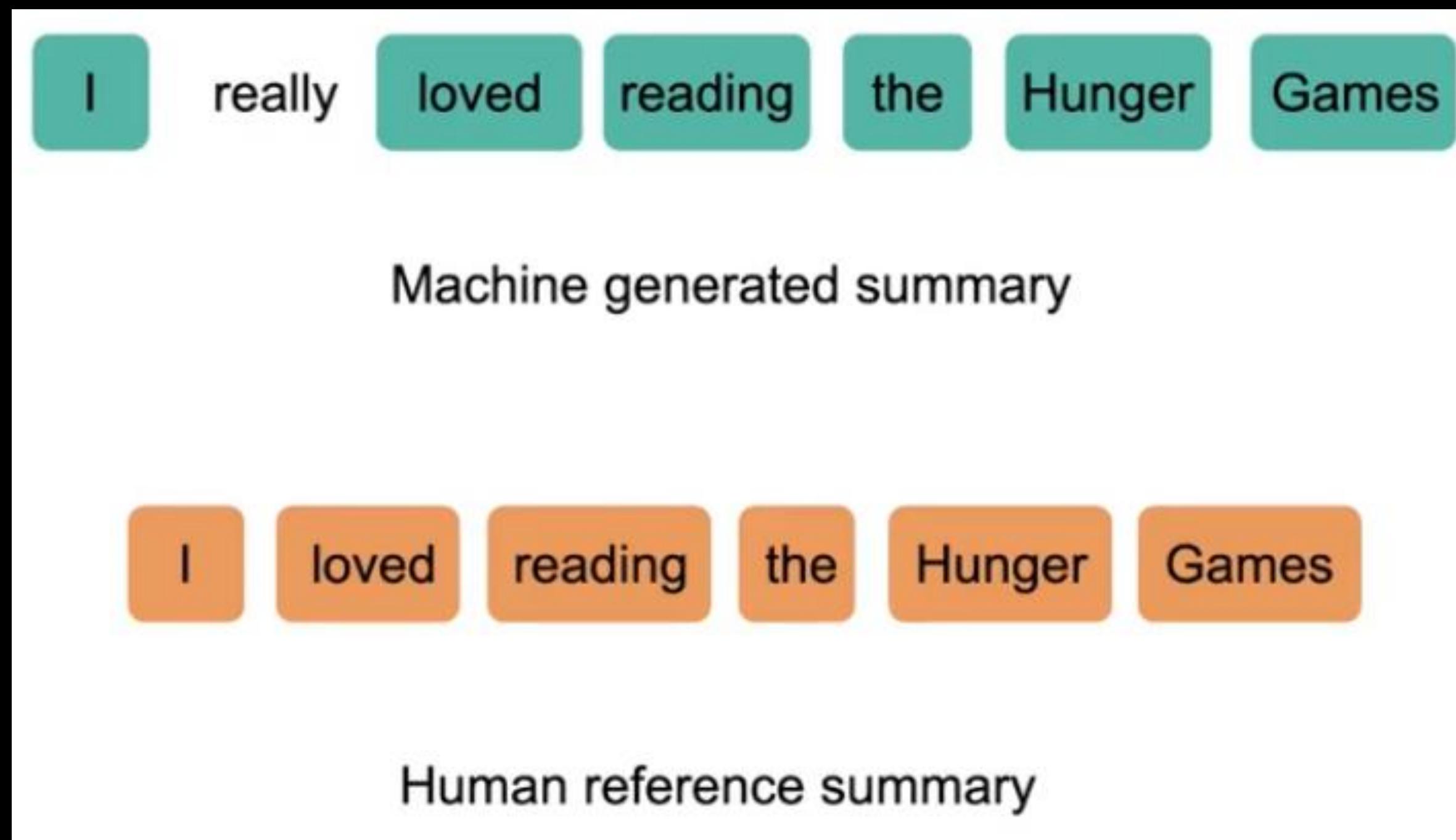
# ROUGE-2 SCORE



$$\text{ROUGE-2 recall} = \frac{\text{Num bigram matches}}{\text{Num bigrams in reference}} = \frac{4}{5}$$

$$\text{ROUGE-2 precision} = \frac{\text{Num bigram matches}}{\text{Num bigram in summary}} = \frac{4}{6}$$

# ROUGE-L SCORE



$$\text{ROUGE-L recall} = \frac{\text{LCS}(\text{gen}, \text{ref})}{\text{Num words in reference}} = \frac{6}{6}$$

$$\text{ROUGE-L precision} = \frac{\text{LCS}(\text{gen}, \text{ref})}{\text{Num words in summary}} = \frac{6}{7}$$

# What is the BLEU metric?



with Lewis



# LIKELIHOOD OF A SEQUENCE

$$P(X) = \prod_{i=0}^t p(x_i \mid x_{<i})$$

Hugging Face is a startup based in New York City and Paris

p(word|context)

# CROSS-ENTROPY

$$CE(X) = -\frac{1}{t} \log P(X)$$

# LOG-PERPLEXITY

$$\begin{aligned} PPL(X) &= e^{CE(X)} \\ &= e^{-\frac{1}{t} \sum_{i=0}^t \log p(x_i | x_{<i})} \end{aligned}$$

# MIXED PRECISION TRAINING

- **Use of both 16-bit and 32-bit floating-point types to train faster and use less memory.**
- **NVIDIA GPUs (20xx/V100 or newer) and TPUs are faster in float16. However, variables and a few computations should still be in float32 for numeric reasons so that the model trains to the same quality.**
- **If you're using a Colab GPU or other GPU that does not have accelerated float16 support, you should rather not use it.**

```
# Train in mixed-precision float16
tf.keras.mixed_precision.set_global_policy("mixed_float16")
```

# PROJECT PRESENTATIONS

- **About 15 Minutes for each project**

**Content should be roughly equivalent to that of a “Model Card” (see [this section](#) for more details):**

- **Model description**
- **Intended uses & limitations**
- **How to use**
- **Limitations and bias**
- **Training data**
- **Training procedure**
- **Evaluation results**