Link for reference: https://www.datacamp.com/tutorial/how-transformers-work

GROUP 1: I031, I033

## STEP 1: Define the Objective and Problem

- **Goal Specification**: Determine whether the model will be used for tasks like text generation, language translation, or question answering.
- **Task Understanding**: If the model is focused on a specific domain (e.g., medical or legal language), narrow the dataset and design to suit that field.
- **Evaluation Metrics**: Identify how success will be measured (accuracy, perplexity, BLEU score, etc.).

**STEP 2:** Data Collection and Preparation

- **Data Sources**: Gather a large and diverse dataset, such as web scrapes, public text datasets (e.g., Common Crawl, Wikipedia), books, and news articles.
- **Data Cleaning**: Eliminate unnecessary content like HTML tags, special symbols, and duplicated text.
- **Special Tokens**: Include tokens like <PAD>, <EOS>, <CLS>, <SEP> for padding, end-of-sentence, and classification/segmentation tasks.

GROUP 2: I032, I034, I036, I037, I038

## STEP 3: **Tokenization and Vocabulary Creation (Transformer)**

- **Tokenizer Selection**: Use sub word tokenization algorithms like BPE or Sentence Piece to handle large vocabulary sizes.
- **Vocabulary Building**: Construct a vocabulary of tokens (sub words/words). LLMs often have vocabularies in the range of 30,000–50,000 tokens.
- Embeddings

## **STEP 4: Model Architecture Design (Transformer and N-gram)**

- **Embedding Layer**: Convert tokens into dense vectors.
- **Multi-Head Self-Attention**: Capture long-range dependencies and relationships between words by using attention mechanisms.
- **Feedforward Layers**: Further process the output from the attention layer.
- **Positional Encoding**: Since transformers do not have a built-in sequence model, add positional encodings to retain the word order information.

GROUP 3: I030, I039, I040, I041, I042

- **Encoder-Decoder (e.g., BERT-like) or Decoder-Only (e.g., GPT-like):**
- Encoder models are good for understanding tasks (classification, QA).
- Decoder-only models are suitable for generation tasks.

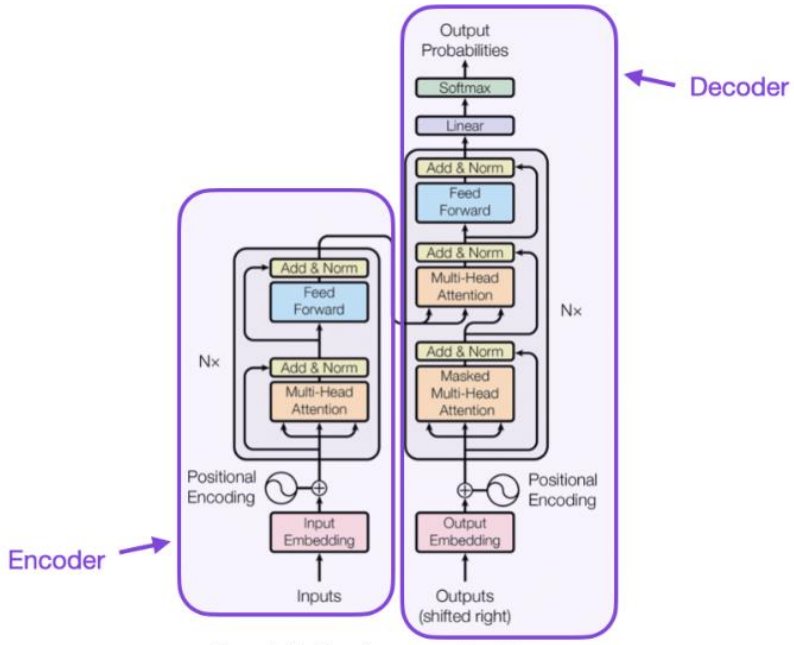


Figure 1: The Transformer - model architecture.

## STEP 5: Hyperparameter Selection

- **Embedding Dimension**: Typically, 512 or 1024.
- **Number of Layers**: The depth of the model (12, 24, or more layers depending on the model size).
- **Attention Heads**: Typically, 12 or 16 heads to capture multiple relationships between tokens.
- **Batch Size and Sequence Length**: Determine the input sequence length (often 512 tokens) and batch size (adjustable based on

memory constraints).

### STEP 6: Model Initialization

- Initialize model weights with methods like **Xavier Initialization** or **He Initialization** to set the starting point for training.
- Initialize token embeddings randomly or use pre-trained embeddings like **Word2Vec** or **GloVe** for a head start.

GROUP 4: I029, I043, I044, I046, I047

### STEP 7: Training the Model

- **Forward Propagation**:
- Feed tokenized text into the model, passing through attention layers, feed-forward layers, and positional encodings.
- The model outputs predictions (next token, sentence classification, etc.).

#### Loss Function:

- For generation tasks: Use **Cross-Entropy Loss** to minimize the difference between the predicted tokens and the ground truth.
- For masked language models (e.g., BERT), mask certain words and predict them.
- **Backpropagation**: Use **Stochastic Gradient Descent (SGD)** or **Adam Optimizer** to adjust weights based on the computed loss.
- **Training Setup**: Train using large-scale distributed computing (GPUs or TPUs) for several days or weeks. Apply gradient clipping to prevent exploding gradients.

## STEP 8: Evaluation

- **Validation Set**: Regularly evaluate the model on a validation set to monitor performance (e.g., loss, perplexity).
- **Early Stopping**: Stop training if the model stops improving on the validation set.
- **Evaluation Metrics**: Depending on the task, use metrics such as:
- Perplexity for language modeling.
- BLEU score for translation.
- F1-score or accuracy for classification.

GROUP 5: I048, I051, I052, I053

## STEP 9: Fine-Tuning

- **Pre-training**: After training on a broad dataset, fine-tune the model on specific tasks using task-specific datasets (e.g., sentiment analysis, summarization).
- **Transfer Learning**: Pre-trained models can be fine-tuned on small, domain-specific datasets to specialize the model.
- **Hyperparameter Tuning**: Adjust hyperparameters like learning rate, batch size, and attention heads for optimal task-specific performance.

## STEP 10: Deploying the Model

- **Model Compression**: Use techniques like **quantization**, **distillation**, or **pruning** to reduce the model size without losing much accuracy.
- **Inference Optimization**: Use inference libraries like **ONNX** or **TensorRT** to improve speed and memory efficiency during deployment.
- **Serving**: Deploy the model via RESTful APIs or integrate it into applications for real- time usage.