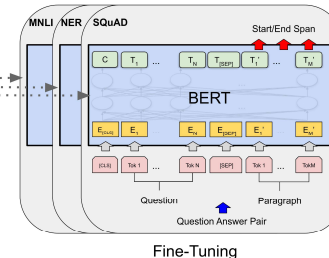
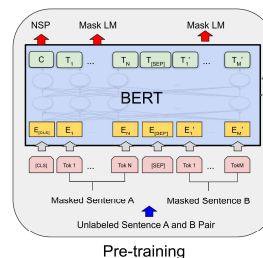


# Transfer Learning – Fine Tuning HF BERT using PyTorch Lightning for Text Classification



# Transfer Learning

- A technique in Machine Learning (ML) in which knowledge learned from a task is re-used in order to boost performance on a related task

# How Does Natural Language Processing work?

- Representing Language with Numbers

Why?

- Computers excel at handling numbers
- Reducing Dimensions of Language
- Feature Extraction
- Semantic Similar (Not lexical similarity)

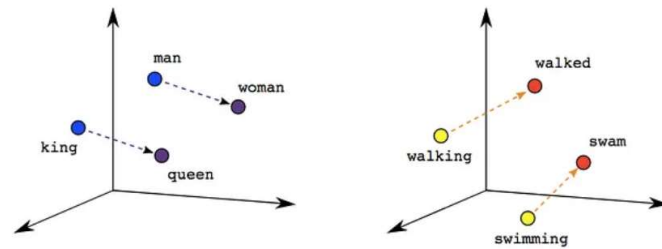
# Numeric Representation of Language

- Term-Document-Matrix
- Term Frequency – Inverse Document Frequency
- Latent Semantic Analysis
- Latent Dirichlet Analysis

# Neural Networks

- Good at handling numeric inputs and predicting numeric labels
- Early NLP NNs were Recurrent NNs - 1997
- RNNs are good at handling the sequence aspect of text data and the fact that text had long-range dependencies. LSTM 1997, GRU 2014
- NNs that learned embeddings came out in 2013, from Google,- Skip-Gram, CBOW
- What are Word Embeddings

# What are Word Embeddings?



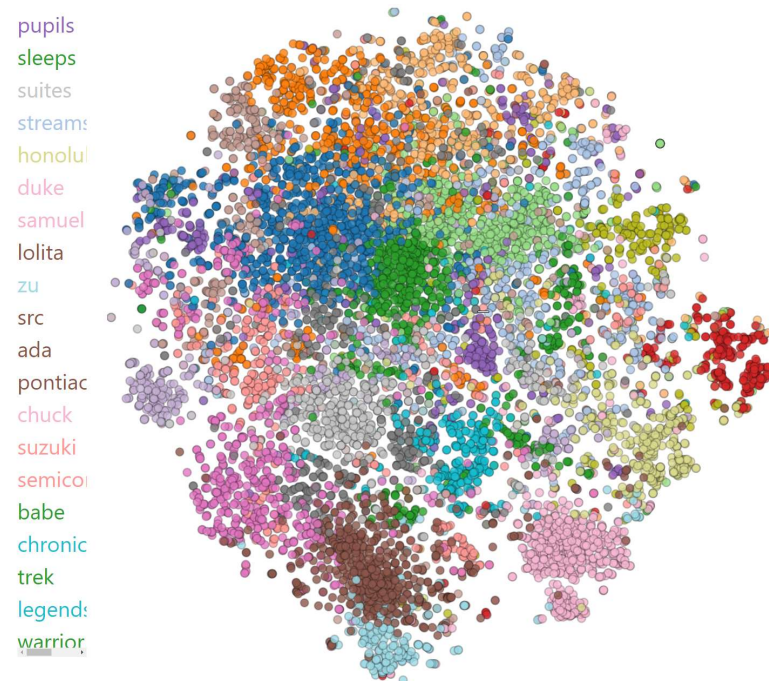
KING - MAN + WOMAN = QUEEN

Edwin Chen

Surge AI CEO: data labeling and  
RLHF, designed for the next  
generation of AI.

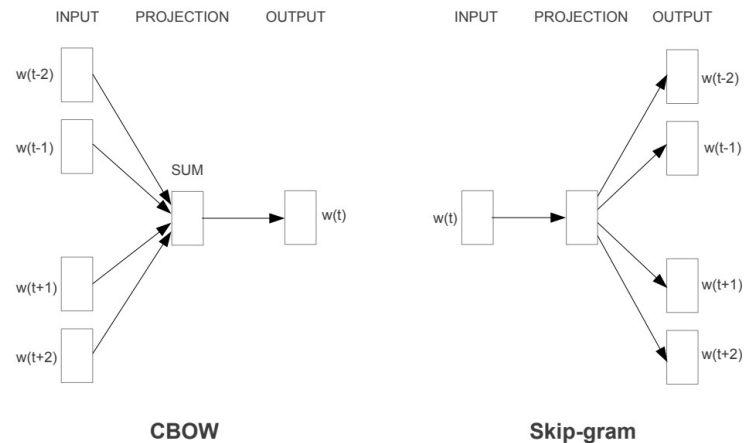
# Visualizing More Embeddings

- <http://blog.echen.me/embedding-explorer/>



# Limitations of Word2Vec

- Single Embedding per word, but words have multiple meanings
- Word level embeddings, but need sentence-level context
- Each word is represented by same vector regardless of context





# Transformers!

- Embeddings
- Big Breakthrough in Natural Language Processing
- Attention is all you Need White Paper
- Attention mechanism gives context to sentences

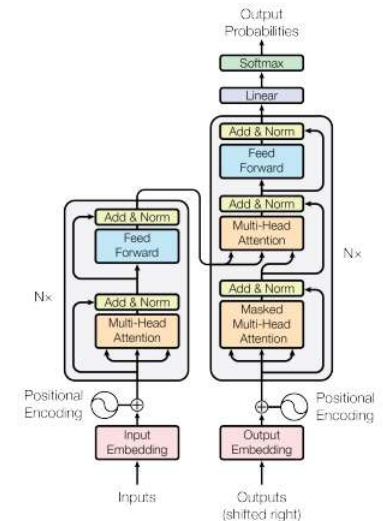


Figure 1: The Transformer - model architecture.

# Attention

- Model uses weights to determine how much focus is given to every word of sentence
- It considers other input words to determine focus to that word(self attention)
- Other input words, before and after the focused on word (bidirection)
- Model can process the text in parallel and focus on different parts, both in encoder and decoder (multi-headed)

- Encoder

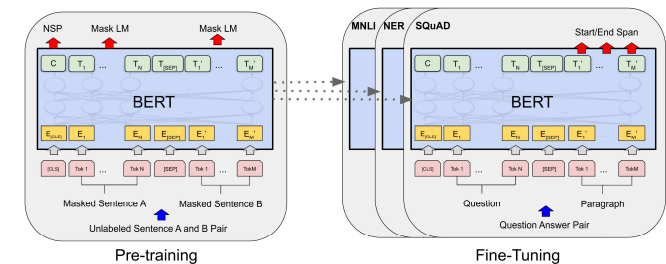
- Encoder focuses on processing input, giving contextualized embeddings

- Decoder

- Decoder focuses on output sequence, also has a cross attention component not present in encoder, it basically gets you from those contextualized embeddings to your output

# Why BERT, as opposed to GPT

- GPT is specifically for Text Generation
- Encoder Only, No need for Decoder
- Predates GPT
- Open-Source
- Understands Context
- BERT is great at text classification
- Bidirectional MLM



# Why HuggingFace



- One of the leaders in Transfer Learning and OpenSource models
- Easy to use
- The Unofficial Leaderboard for a lot of members within community
- [https://huggingface.co/spaces/HuggingFaceH4/open\\_llm\\_leaderboard](https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard)

Import transformers

```
bert_model = BertModel.from_pretrained(BERT_MODEL_NAME,  
return_dict=True)
```

# Why Pytorch

- Good framework for Neural Networks
  - Pytorch handles abstraction, FeedForward, Backpropagation
  - Setup and Execute from Checkpoints, Inspecting bottlenecks
- Works well with NVIDIA CUDA



# Why Lightning AI

- Lightning AI's Pytorch Lightning offers more Flexibility for Pytorch and allows it to be Scaled without sacrificing performance



# Why Google Translate?



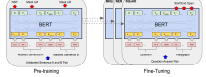


- Data Augmentation
- Data Augmentation is common in Deep Learning
- We use it to augment the class we want to pay attention to
- Translate English to French and back to English
- Generates new sentences that have different word makeup same meaning

```
# Create a Translator object  
backtranslator = GoogleTranslator(source='en', target='fr')  
forthtranslator = GoogleTranslator(source='fr', target='en')
```



# Deep Learning Building Steps

- 1) Problem Identification
- 2) Data Collection, Augmentation 
- 3) Data Preparation, Feature Engineering   

- 4) Model Selection
- 5) Model Training/ 6)Hyperparameter Tuning
- 7) Model Evaluation

# DL Problem Identification

- Build a model to accurately classify Toxic comments
- Multilabel Classification
- Toxic, Severely Toxic, Identity-Hate, Insult, Obscene, Threat
- Data Collection: <https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data>

# Data Preparation, Model Selection and Training

- Data Prep: Augmentation using methods common in language
- Feature Engineering: BERT is used as tokenizer, gives us contextualized word embeddings
- Model Selection: Our classifier is a simple Feed Forward Linear NN
- Simple NN uses Activation function to ensure output is between 0 and 1
- Training, Tuning is done over the course of multiple epochs
- Data is split up into batches and fed through tokenizer and linear NN

# Hyperparameter Tuning and Validation

- After forward pass, BCE loss is computed
- Backpropagation updates the weights of the network
- BERT Weights and Biases get Updated, Updating our Feature set
- AdamW – Adjusts the learning rate, includes weight decay, large weights are penalized, this prevents overfitting produces a better general model

Let's go to the Python Code!

# Base Model

	precision	recall	f1-score	support
toxic	0.56	0.94	0.70	728
severe_toxic	0.46	0.46	0.46	67
obscene	0.72	0.82	0.77	408
threat	0.41	0.50	0.45	18
insult	0.70	0.75	0.73	386
identity_hate	0.48	0.50	0.49	66
micro avg	0.61	0.83	0.70	1673
macro avg	0.56	0.66	0.60	1673
weighted avg	0.62	0.83	0.70	1673
samples avg	0.08	0.08	0.08	1673

Accuracy: 0.88

Precision (macro average): 0.56

Recall (macro average): 0.66

F1 Score (macro average): 0.60

# French, German, and Spanish Augmentation

	precision	recall	f1-score	support
toxic	0.57	0.99	0.72	728
severe_toxic	0.99	0.99	0.99	67
obscene	0.86	0.98	0.92	408
threat	0.72	1.00	0.84	18
insult	0.83	0.98	0.90	386
identity_hate	0.89	0.97	0.93	66
micro avg	0.70	0.98	0.82	1673
macro avg	0.81	0.98	0.88	1673
weighted avg	0.73	0.98	0.83	1673
samples avg	0.09	0.10	0.09	1673

Accuracy: 0.92

Precision: 0.81

Recall: 0.98

F1: 0.88

# French Augmentation

	precision	recall	f1-score	support
toxic	0.60	0.99	0.75	728
severe_toxic	0.87	0.93	0.90	67
obscene	0.87	0.98	0.92	408
threat	0.77	0.94	0.85	18
insult	0.86	0.97	0.91	386
identity_hate	0.78	0.98	0.87	66
micro avg	0.72	0.98	0.83	1673
macro avg	0.79	0.96	0.87	1673
weighted avg	0.75	0.98	0.84	1673
samples avg	0.09	0.09	0.09	1673

Accuracy: 0.93

Precision (macro average): 0.79

Recall (macro average): 0.96

F1 Score (macro average): 0.87



# What do Transfer Learning and Crypto have in common?

- Heavily rely on open-source communities
  - Python, PyTorch, BERT, SKLearn, huggingface, many others that we will explore
  - Bitcoin, Ethereum, Cypherpunks
- Very Fast Growing Technology Sectors
- Heavily Rely on Distributed Computing

# How Transfer Learning and Crypto will be used together?

- Developers will rely on LLMs to learn Smart Contract Development, like Solidity
- LLMs can be used to understand existing smart contract code repositories in production by people with different skillsets
- Blockchains can ensure that the underlying training data these models are trained on is public, verifiable, and Audit-able