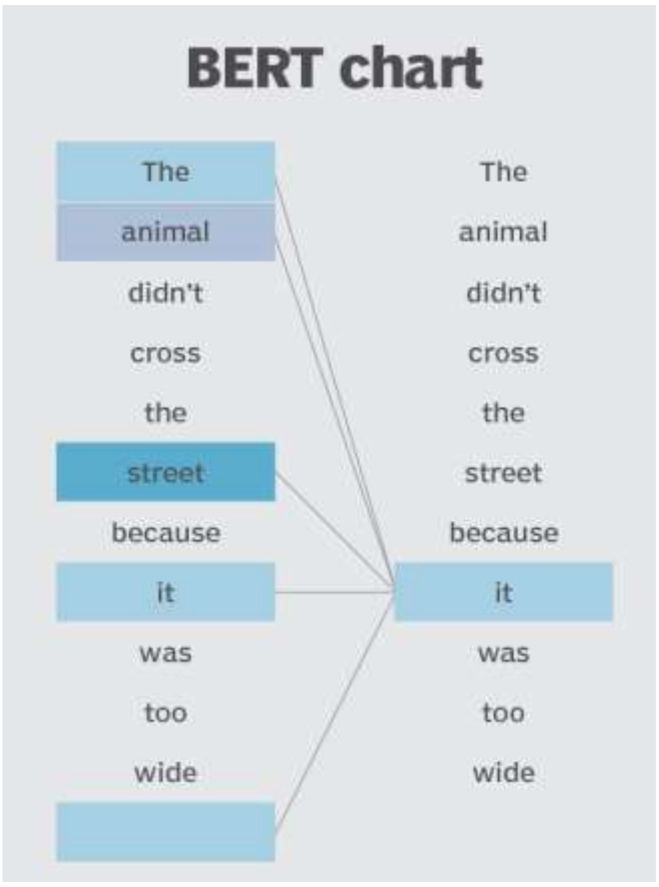
Open-Source AI Models and their differences

Some of the mainly used open source ai models are:

- **BERT (Bidirectional Encoder Representations from Transformers):**
  - BERT is an open-source machine learning framework for natural language processing (NLP). BERT is designed to help computers understand the meaning of ambiguous language in text by using surrounding text to establish context.
  - BERT, which stands for Bidirectional Encoder Representations from Transformers, is based on Transformers, a deep learning model in which every output element is connected to every input element, and the weightings between them are dynamically calculated based upon their connection.
  - Historically, language models could only read text input sequentially either left-to-right or right-to-left but couldn't do both at the same time. BERT is different because it is designed to read in both directions at once. This capability, enabled by the introduction of Transformers, is known as bidirectionality.
  - Using this bidirectional capability, BERT is pre-trained on two different, but related, NLP tasks: Masked Language Modelling and Next Sentence Prediction.
  - **Masked Language Model (MLM)** training is to hide a word in a sentence and then have the program predict what word has been hidden (masked) based on the hidden word's context.
    To do this, models typically need to train using a large repository of specialized, labelled training data. (Not in case of BERT)
  - **Next Sentence Prediction** training is to have the program predict whether two given sentences have a logical, sequential connection or whether their relationship is simply random.
  - BERT, however, was pre-trained using only an unlabelled, plain text corpus. It continues to learn unsupervised from the unlabelled text and improve even as its being used in practical applications (i.e. Google search).
    From there, BERT can adapt to the ever-growing body of searchable content and queries and be fine-tuned to a user's specifications. This process is known as transfer learning.

- BERT Chart



For example, in the image above, BERT is determining which prior word in the sentence the word "it" referring to, and then using its attention mechanism to weigh the options. The word with the highest calculated score is deemed the correct association. If this phrase was a search query, the results would reflect this subtler, more precise understanding the BERT reached.

- **Uses:**
  - Sequence-to-sequence based language generation tasks such as:
    - ❖ Question answering
    - ❖ Abstract summarization
    - ❖ Sentence prediction
    - ❖ Conversational response generation
  - Natural language understanding tasks such as:
    - ❖ Polysemy and Coreference (words that sound or look the same but have different meanings) resolution
    - ❖ Word sense disambiguation
    - ❖ Natural language inference
    - ❖ Sentiment classification

➢ **<u>GPT (Generative Pre-trained Transformer):</u>**

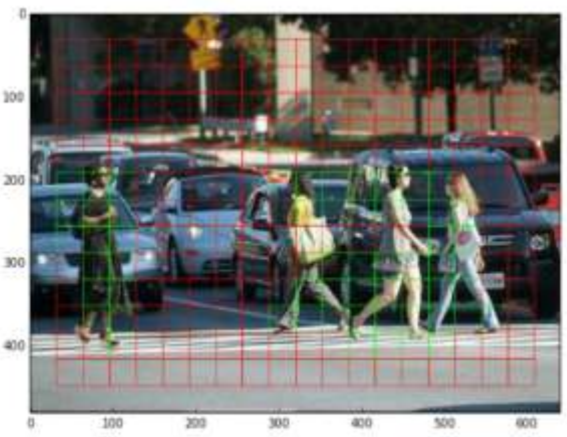
- The working of GPT (Generative Pre-trained Transformer) involves several key components and steps that enable it to understand and generate human-like text.
- **<u>Transformer Architecture:</u>** GPT is built upon the transformer architecture, which consists of an encoder-decoder structure. However, in the case of GPT, only the decoder part is utilized. The transformer architecture includes self-attention mechanisms that allow the model to consider the relationships between different words in a sentence.
- **<u>Pre-training:</u>** GPT undergoes a pre-training phase on a massive corpus of text data from the internet. During pre-training, the model learns to predict the next word in a sentence given the preceding words. It does so by considering the context of the words using self-attention mechanisms. The model learns to create embeddings (vectors) for words and their context, capturing semantic relationships and patterns in the text.
- 3. **<u>Contextualized Embeddings:</u>** GPT generates contextualized word embeddings by considering the entire sentence's context, rather than just the previous or next word. This contextualization allows the model to understand the nuanced meanings and dependencies between words.
- 4. **<u>Generating Text:</u>**
    - **<u>Input:</u>** To generate text, you provide a "prompt" to the GPT model. This could be a sentence, a question, or an incomplete text snippet.
    - **<u>Tokenization:</u>** The input text is tokenized, breaking it down into smaller units called tokens (words or subwords).
    - **<u>Positional Encodings:</u>** GPT adds positional encodings to the token embeddings, allowing the model to understand the order of the words in the input sequence.
    - **<u>Attention Mechanism:</u>** GPT employs self-attention mechanisms to weigh the importance of each token in relation to others, considering both local and global dependencies. This attention mechanism helps the model understand the context and relationships between words.
    - **<u>Generating Outputs:</u>** GPT generates text by predicting the next token in a sequence based on the preceding tokens and their contextual embeddings. It samples from the predicted probability distribution to select the next token. This process

is repeated iteratively to generate a sequence of tokens, forming coherent and contextually relevant text.

- **Fine-tuning:** After pre-training, GPT models can be fine-tuned on specific tasks using smaller datasets. During fine-tuning, the model's parameters are adjusted to perform well on the target task, such as text classification or language translation. Fine-tuning helps adapt the model's general language understanding to more specialized applications.
- **Generating Creative Text:** GPT's ability to generate creative and diverse text stems from its exposure to a wide range of language patterns during pre-training. It can generate human-like text that includes coherent sentences, accurate grammar, and contextually relevant content.
- It's important to note that while GPT is highly proficient at generating text, it doesn't possess true understanding or consciousness. It generates text based on learned patterns and associations, and its responses are influenced by the training data it has seen. GPT's performance and output quality are influenced by the size of the model, the quality of the training data, and the prompt given by the user.

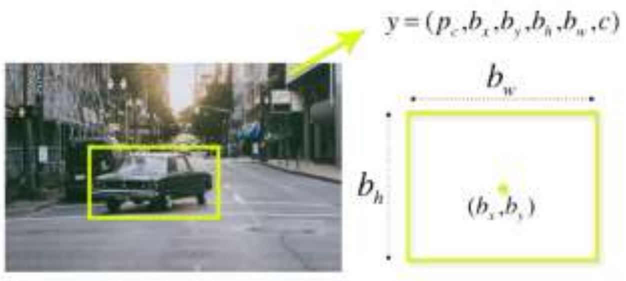
➢ <u>**YOLO (You Only Look Once):**</u>

- YOLO is an algorithm that uses neural networks to provide real-time object detection. This algorithm is popular because of its speed and accuracy. It has been used in various applications to detect traffic signals, people, parking meters, and animals.
- YOLO algorithm employs convolutional neural networks (CNN) to detect objects in real-time. As the name suggests, the algorithm requires only a single forward propagation through a neural network to detect objects.
- The YOLO algorithm consists of various variants. Some of the common ones include tiny YOLO and YOLOv3.
- YOLO algorithm works using the following three techniques:
    - Residual blocks
    - Bounding box regression
    - Intersection Over Union (IOU)
- <u>**Residual Blocks**</u>
    - First, the image is divided into various grids. Each grid has a dimension of S x S. The following image shows how an input image is divided into grids.

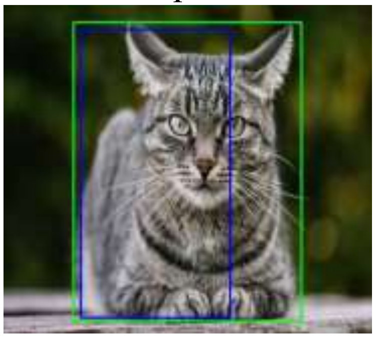


    - In the image above, there are many grid cells of equal dimension. Every grid cell will detect objects that appear within them. For example, if an object center appears within a certain grid cell, then this cell will be responsible for detecting it.

- **Bounding box regression:**
  - A bounding box is an outline that highlights an object in an image.



  - Every bounding box in the image consists of the following attributes:
    - ❖ Width (bw)
    - ❖ Height (bh)
    - ❖ Class (for example, person, car, traffic light, etc.)- This is represented by the letter c.
    - ❖ Bounding box center (bx,by)
  - A bounding box is an outline that highlights an object in an image.
- **Intersection over union (IOU):**
  - Intersection over union (IOU) is a phenomenon in object detection that describes how boxes overlap.
  - YOLO uses IOU to provide an output box that surrounds the objects perfectly.
  - Each grid cell is responsible for predicting the bounding boxes and their confidence scores.
  - The IOU is equal to 1 if the predicted bounding box is the same as the real box. This mechanism eliminates bounding boxes that are not equal to the real box.



        - In the image above, there are two bounding boxes, one in green and the other one in blue. The blue box is the predicted box while the green box is the real box. YOLO ensures that the two bounding boxes are equal.

- The process starts by dividing the image into grid cells. Each cell predicts multiple bounding boxes with confidence scores. It also estimates class probabilities for object recognition. All predictions are done together using one convolutional neural network.
- Using intersection over union, predicted boxes are matched with real object boxes, discarding unfitting boxes. This leads to accurate bounding boxes for each object. For instance, a pink box around the car, a yellow one around the bicycle, and a blue one around the dog.

- **DeepFace:**
  - DeepFace is built upon deep convolutional neural network (CNN) architecture, which excels at extracting intricate features from images.
  - Its primary objective is to accurately identify and verify individuals by analyzing their facial features. This makes DeepFace exceptionally useful for applications such as security systems, authentication mechanisms, and social media tagging.
  - DeepFace is the most lightweight face recognition and facial attribute analysis library for Python. The open-sourced DeepFace library includes all leading-edge AI models for face recognition and automatically handles all procedures for facial recognition in the background.
  - Components:
    - **Face Verification**: The task of face verification refers to comparing a face with another to verify if it is a match or not. Hence, face verification is commonly used to compare a candidate's face to another. This can be used to confirm that a physical face matches the one in an ID document.
    - **Face Recognition:** The task refers to finding a face in an image database. Performing face recognition requires running face verification many times.
    - **Facial Attribute Analysis:** The task of facial attribute analysis refers to describing the visual properties of face images. Accordingly, facial attributes analysis is used to extract attributes such as age, gender classification, emotion analysis, or race/ethnicity prediction.
    - **Real-Time Face Analysis:** This feature includes testing face recognition and facial attribute analysis with the real-time video feed of your webcam.

- **Tacotron and WaveGAN (for Text-to-Speech and Audio Generation):**
  - Tacotron and WaveGAN are two interconnected open-source AI models that together enable a comprehensive approach to speech synthesis.
  - Tacotron generates mel-spectrogram sequences from input text, while WaveGAN converts these spectrograms into realistic audio waveforms. This tandem of models has significantly improved the quality and naturalness of synthetic speech, enhancing applications such as text-to-speech (TTS) systems, voice assistants, and more.
  - **Tacotron: Generating Mel-Spectrogram Sequences:**
    - **Input Text:** Tacotron takes input text, typically represented as a sequence of phonemes or characters.
    - **Text Embedding:** The input text is converted into a fixed-dimensional text embedding using recurrent neural networks (RNNs) or other sequence-to-sequence models.
    - **Mel-Spectrogram Generation:** Tacotron employs attention mechanisms and RNNs to predict mel-spectrogram sequences. A mel-spectrogram is a visual representation of the frequency content of an audio signal over time. It captures the phonetic and prosodic characteristics of speech.
    - **Duration Modeling:** Tacotron also predicts the duration of each phoneme or character, ensuring that the generated spectrogram aligns with the timing of natural speech.
  - **WaveGAN: Transforming Spectrograms to Audio Waveforms**
    - **Input Spectrograms:** The mel-spectrogram sequences produced by Tacotron serve as input to WaveGAN.
    - **WaveGAN Architecture:** WaveGAN employs a generative adversarial network (GAN) architecture. The generator network takes mel-spectrogram sequences and produces corresponding audio waveforms.
    - **Discriminator Network:** The discriminator network distinguishes between real audio waveforms and synthetic waveforms generated by the generator.
    - **Training:** During training, WaveGAN's generator and discriminator networks compete, with the generator learning to produce more realistic audio waveforms over time.
    - **Audio Generation:** The trained WaveGAN generator takes mel-spectrogram sequences and generates high-quality, natural-sounding audio waveforms that match the input spectrograms.

Differences:

Model	Type	Application	Working Principle	Notable Features
BERT	NLP	Text Analysis	Bidirectional context in transformers	Contextual word embeddings
GPT	NLP	Text Generation	Unidirectional context in transformers	Generative language model
YOLO	Computer Vision	Object Detection	Real-time bounding box predictions	Speed and accuracy trade-off
DeepFace	Computer Vision	Facial Recognition	Deep CNN for feature extraction	Facial feature representation
Tacotron + WaveGAN	Speech Synthesis	Text-to-Speech	Sequence-to-spectrogram + GAN	Realistic speech generation

Sources:

https://www.techtarget.com/searchenterpriseai/definition/BERT-language-model

https://www.analyticsvidhya.com/blog/2022/10/generative-pre-training-gpt-for-natural-language-understanding/

https://www.theguardian.com/commentisfree/2020/sep/08/robot-wrote-this-article-gpt-3

https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/

https://viso.ai/computer-vision/deepface/#:~:text=DeepFace%20is%20the%20most%20lightweight,facial%20recognition%20in%20the%20background

https://towardsdatascience.com/text-to-speech-with-tacotron-2-and-fastspeech-using-espnet-3a711131e0fa

https://chat.openai.com/