Milestone 1 Instruction
**Milestone 1**

- Each team will present on either February 2nd or February 9th
- Prepare about 20 mins of presentation (and expect 3-5 mins of Q&A)
- All teams will submit a written Milestone 1 report on February 9th.

- Key elements in Milestone 1 presentation:
  - Task Goal:
    - What do you want to achieve?
    - Why is the research and development important?
    - Is this a new topic that considers challenges of:
      - Volume
      - Velocity
      - Variety
    - Does this topic try to incorporate multi-discipline knowledge?
  - (Optional) Is it related to A.I.?
Milestone 1

- Literature Survey:
  - What are the prior arts? What related works were done before?
  - Which research publications, tools and products may be utilized to build upon them to achieve the goal?

- Methodology:
  - What types of novel algorithms I shall try to invent and implement?
  - Where will you try to gather the data — Existing dataset? Self-collected dataset? Live dataset?
Milestone 1

- **System:**
  - What will your final system look like — potential backend components and interaction with front end?
  - How will you create visualization and user interface to help users consume your analysis outcome?

- **Timeline:**
  - What may you achieve in Milestones 2 and 3, and in the Final Project?

**Note** — it’s good to think about what you want to achieve as complete as possible and study what other people have done. But, like all projects, everything can change when you make progress. Please do not afraid of making bold assumptions and attempt!!
Big Data Analytics Basic Foundation
Key Open Source Big Data Foundations

Apache Hadoop

Apache Spark

Lightning-fast unified analytics engine

Apache Kafka

A distributed streaming platform
Other Important Foundations (visualization and web servers)
Where to store data?
How to get data in and out?
How to manage access of data?
How do I process the data?
How do I execute machine learning from the data?
How do I tell people my analytics results?
Spark

Spark SQL structured data
Spark Streaming real-time
MLib machine learning
GraphX graph processing

Spark Core
Standalone Scheduler
YARN
Mesos
Spark MLlib

Includes:

- ML Algorithms: common learning algorithms such as classification, regression, clustering, and collaborative filtering
- Featurization: feature extraction, transformation, dimensionality reduction, and selection
- Pipelines: tools for constructing, evaluating, and tuning ML Pipelines
- Persistence: saving and load algorithms, models, and Pipelines
- Utilities: linear algebra, statistics, data handling, etc.

MLlib: Main Guide

- Basic statistics
- Data sources
- Pipelines
- Extracting, transforming and selecting features
- Classification and Regression
- Clustering
- Collaborative filtering
- Frequent Pattern Mining
- Model selection and tuning
- Advanced topics
Spark MLlib Basic Statistics

Includes:

- Correlation
- Hypothesis testing
- Summarizer

Example of Calculating Correlation of Time Sequences:

```python
from pyspark.ml.linalg import Vectors
from pyspark.ml.stat import Correlation

data = [(Vectors.sparse(4, [(0, 1.0), (3, -2.0)]),),
        (Vectors.dense([4.0, 5.0, 0.0, 3.0]),),
        (Vectors.dense([6.0, 7.0, 0.0, 8.0]),),
        (Vectors.sparse(4, [(0, 9.0), (3, 1.0)]),)]
df = spark.createDataFrame(data, ["features"])

r1 = Correlation.corr(df, "features").head()
print("Pearson correlation matrix:
 + str(r1[0])")

r2 = Correlation.corr(df, "features", "spearman").head()
print("Spearman correlation matrix:
 + str(r2[0])")
```

<table>
<thead>
<tr>
<th>MLlib: Main Guide</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Basic statistics</td>
</tr>
<tr>
<td>- Data sources</td>
</tr>
<tr>
<td>- Pipelines</td>
</tr>
<tr>
<td>- Extracting, transforming and selecting features</td>
</tr>
<tr>
<td>- Classification and Regression</td>
</tr>
<tr>
<td>- Clustering</td>
</tr>
<tr>
<td>- Collaborative filtering</td>
</tr>
<tr>
<td>- Frequent Pattern Mining</td>
</tr>
<tr>
<td>- Model selection and tuning</td>
</tr>
<tr>
<td>- Advanced topics</td>
</tr>
</tbody>
</table>
Spark MLlib Features

Includes:
- Extraction: Extracting features from "raw" data
- Transformation: Scaling, converting, or modifying features
- Selection: Selecting a subset from a larger set of features
- Locality Sensitive Hashing (LSH): This class of algorithms combines aspects of feature transformation with other algorithms.

- Feature Extractors
  - TF-IDF
  - Word2Vec
  - CountVectorizer
  - FeatureHasher

- Feature Selectors
  - VectorSlicer
  - RFormula
  - ChiSqSelector

- Feature Transformers
  - Tokenizer
  - StopWordsRemover
  - n-gram
  - Binarizer
  - PCA
  - PolynomialExpansion
  - Discrete Cosine Transform (DCT)
  - StringIndexer
  - IndexToString
  - OneHotEncoder (Deprecated since 2.3.0)
  - OneHotEncoderEstimator
  - VectorIndexer
  - Interaction
  - Normalizer
  - StandardScaler
  - MinMaxScaler
  - MaxAbsScaler
  - Bucketizer
  - ElementwiseProduct
  - SQLTransformer
  - VectorAssembler
  - VectorSizeHint
  - QuantileDiscretizer
  - Imputer
Spark MLlib Supervised Machine Learning Algorithms

**Classification**
- Logistic regression
  - Binomial logistic regression
  - Multinomial logistic regression
- Decision tree classifier
- Random forest classifier
- Gradient-boosted tree classifier
- Multilayer perceptron classifier
- Linear Support Vector Machine
- One-vs-Rest classifier (a.k.a. One-vs-All)
- Naive Bayes

**Regression**
- Linear regression
- Generalized linear regression
  - Available families
- Decision tree regression
- Random forest regression
- Gradient-boosted tree regression
- Survival regression
- Isotonic regression
Spark MLlib UnSupervised Machine Learning & Recommendation Algorithms

Clustering:
• K-means
  ◦ Input Columns
  ◦ Output Columns
• Latent Dirichlet allocation (LDA)
• Bisecting k-means
• Gaussian Mixture Model (GMM)
  ◦ Input Columns
  ◦ Output Columns

Collaborative Filtering:
• Explicit vs. implicit feedback
• Scaling of the regularization parameter
• Cold-start strategy
Spark MLlib Model Selection and Tuning

- Model selection (a.k.a. hyperparameter tuning)
- Cross-Validation
- Train-Validation Split
Webservers

- Apache (http server) — the oldest and most popular web server exists in every linux machine, including MacOS machines.

- Display webpages: Flask
  
  ```python
  from flask import Flask, escape, request
  
  app = Flask(__name__)
  
  @app.route('/'
  def hello():
      name = request.args.get("name", "World")
      return f'Hello, {escape(name)}!'
  ```
D3 Visualization (via Javascript) Examples

Visual Index

Box Plots  |  Bubble Chart  |  Bullet Charts  |  Calendar View
---|---|---|---

Non-contiguous Cartogram  |  Chord Diagram  |  Dendrogram  |  Force-Directed Graph

Circle Packing  |  Population Pyramid 2000  |  Stacked Bars  |  Streamgraph
D3 Visualization (via Javascript) Examples

- Sunburst
- Node-Link Tree
- Treemap
- Voronoi Diagram

- Hierarchical Edge Bundling
- Voronoi Diagram
- Bubble Map
- Parallel Coordinates

- Scatterplot Matrix
- Zoomable Pack Layout
- Hierarchical Bars
- Epicyclical Gears
## D3 Visualization (via Javascript) Examples

<table>
<thead>
<tr>
<th>Visual Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collision Detection</td>
<td><img src="image" alt="Collision Detection" /></td>
</tr>
<tr>
<td>Collapsible Force Layout</td>
<td><img src="image" alt="Collapsible Force Layout" /></td>
</tr>
<tr>
<td>Force-Directed States</td>
<td><img src="image" alt="Force-Directed States" /></td>
</tr>
<tr>
<td>Versor Dragging</td>
<td><img src="image" alt="Versor Dragging" /></td>
</tr>
<tr>
<td>Choropleth</td>
<td><img src="image" alt="Choropleth" /></td>
</tr>
<tr>
<td>Collapsible Tree Layout</td>
<td><img src="image" alt="Collapsible Tree Layout" /></td>
</tr>
<tr>
<td>Zoomable Treemap</td>
<td><img src="image" alt="Zoomable Treemap" /></td>
</tr>
<tr>
<td>Zoomable Icicle</td>
<td><img src="image" alt="Zoomable Icicle" /></td>
</tr>
<tr>
<td>Zoomable Area Chart</td>
<td><img src="image" alt="Zoomable Area Chart" /></td>
</tr>
<tr>
<td>Drag and Drop Collapsible Tree Layout</td>
<td><img src="image" alt="Drag and Drop Collapsible Tree Layout" /></td>
</tr>
<tr>
<td>Radial Cluster Layout</td>
<td><img src="image" alt="Radial Cluster Layout" /></td>
</tr>
<tr>
<td>Sankey Diagram</td>
<td><img src="image" alt="Sankey Diagram" /></td>
</tr>
</tbody>
</table>
D3 Installation

Installing

For NPM, `npm install d3`. For Yarn, `yarn add d3`. Otherwise, download the latest release. The released bundle supports AMD, CommonJS, and vanilla environments. Create a custom bundle using Rollup or your preferred bundler. You can also load directly from d3js.org:

```html
<script src="https://d3js.org/d3.v5.js"></script>
```

For the minified version:

```html
<script src="https://d3js.org/d3.v5.min.js"></script>
```

You can also use the standalone D3 microlibraries. For example, `d3-selection`:

```html
<script src="https://d3js.org/d3-selection.v1.min.js"></script>
```
D3 Example Circles

https://bost.ocks.org/mike/circles/

```html
<svg width="720" height="120">
  <circle cx="40" cy="60" r="10"></circle>
  <circle cx="80" cy="60" r="10"></circle>
  <circle cx="120" cy="60" r="10"></circle>
</svg>
```

```javascript
var circle = d3.selectAll("circle");
circle.style("fill", "steelblue");
circle.attr("r", 30);

circle.attr("cx", function() { return Math.random() * 720; });
```
D3 Bar Chart Tutorial

https://bost.ocks.org/mike/bar/

```javascript
var data = [4, 8, 15, 16, 23, 42];
```

Selecting an Element

Javascript:

```javascript
var div = document.createElement("div");
div.innerHTML = "Hello, world!"
document.body.appendChild(div);
```

D3:

```javascript
var body = d3.select("body");
var div = body.append("div");
div.html("Hello, world!");
```
D3 Bar Chart Tutorial

Coding a Chart, Manually

```html
<!DOCTYPE html>
<style>
.chart div {
  font: 10px sans-serif;
  background-color: steelblue;
  text-align: right;
  padding: 3px;
  margin: 1px;
  color: white;
}
</style>

<div class="chart">
  <div style="width: 40px;">4</div>
  <div style="width: 80px;">8</div>
  <div style="width: 150px;">15</div>
  <div style="width: 160px;">16</div>
  <div style="width: 230px;">23</div>
  <div style="width: 420px;">42</div>
</div>
```
D3 Bar Chart Tutorial

```html
<!DOCTYPE html>
<style>
  .chart rect {
    fill: steelblue;
  }

  .chart text {
    fill: white;
    font: 10px sans-serif;
    text-anchor: end;
  }
</style>

<svg class="chart" width="420" height="120">
  <g transform="translate(0,0)">
    <rect width="40" height="19"/>
    <text x="37" y="9.5" dy="-.35em">4</text>
  </g>

  <g transform="translate(0,20)">
    <rect width="80" height="19"/>
    <text x="77" y="9.5" dy="-.35em">8</text>
  </g>

  <g transform="translate(0,40)">
    <rect width="120" height="19"/>
    <text x="117" y="9.5" dy="-.35em">15</text>
  </g>

  <g transform="translate(0,60)">
    <rect width="160" height="19"/>
    <text x="157" y="9.5" dy="-.35em">16</text>
  </g>

  <g transform="translate(0,80)">
    <rect width="200" height="19"/>
    <text x="197" y="9.5" dy="-.35em">23</text>
  </g>

  <g transform="translate(0,100)">
    <rect width="240" height="19"/>
    <text x="237" y="9.5" dy="-.35em">42</text>
  </g>
</svg>

Full code to do it manually
D3 Bar Chart Tutorial

```html
<!DOCTYPE html>
<meta charset="utf-8">
<style>
  .chart rect {
    fill: steelblue;
  }

  .chart text {
    fill: white;
    font: 10px sans-serif;
    text-anchor: end;
  }
</style>

<svg class="chart"></svg>
<script src="//d3js.org/d3.v3.min.js" charset="utf-8"></script>

Full code to do it automatically
```
D3 Bar Chart Tutorial

```javascript
<script>
var data = [4, 8, 15, 16, 23, 42];

var width = 420,
    barHeight = 20;

var x = d3.scale.linear()
    .domain([0, d3.max(data)])
    .range([0, width]);

var chart = d3.select(".chart")
    .attr("width", width)
    .attr("height", barHeight * data.length);

var bar = chart.selectAll("g")
    .data(data)
    .enter().append("g")
    .attr("transform", function(d, i) { return "translate(0," + i * barHeight + "); "});

bar.append("rect")
    .attr("width", x)
    .attr("height", barHeight - 1);

bar.append("text")
    .attr("x", function(d) { return x(d) - 3; })
    .attr("y", barHeight / 2)
    .attr("dy", ".35em")
    .text(function(d) { return d; });
</script>

Full code to do it automatically
```
D3 Bar Chart Tutorial

Load data

```javascript
// 1. Code here runs first, before the download starts.

d3.tsv("data.tsv", function(error, data) {
    // 3. Code here runs last, after the download finishes.
});

// 2. Code here runs second, while the file is downloading.
```

<table>
<thead>
<tr>
<th>name</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locke</td>
<td>4</td>
</tr>
<tr>
<td>Reyes</td>
<td>8</td>
</tr>
<tr>
<td>Ford</td>
<td>15</td>
</tr>
<tr>
<td>Jarrah</td>
<td>16</td>
</tr>
<tr>
<td>Shephard</td>
<td>23</td>
</tr>
<tr>
<td>Kwon</td>
<td>42</td>
</tr>
</tbody>
</table>

The equivalent of Javascript code:

```javascript
var data = [
    {name: "Locke", value: 4},
    {name: "Reyes", value: 8},
    {name: "Ford", value: 15},
    {name: "Jarrah", value: 16},
    {name: "Shephard", value: 23},
    {name: "Kwon", value: 42}
];
```
D3 Bar Chart Tutorial

```html
<!DOCTYPE html>
<meta charset="utf-8">
<style>
.chart rect {
  fill: steelblue;
}
.chart text {
  fill: white;
  font: 10px sans-serif;
  text-anchor: end;
}
</style>
<svg class="chart"></svg>
<script src="/d3js.org/d3.v3.min.js" charset="utf-8"></script>
```
D3 Bar Chart Tutorial

```javascript
var width = 420,
    barHeight = 20;

var x = d3.scale.linear()
    .range([0, width]);

var chart = d3.select(".chart")
    .attr("width", width);

d3.tsv("data.tsv", type, function(error, data) {
    x.domain([0, d3.max(data, function(d) { return d.value; })]);

    chart.attr("height", barHeight * data.length);

    var bar = chart.selectAll("g")
        .data(data)
        .enter().append("g")
        .attr("transform", function(d, i) { return "translate(0," + i * barHeight + ");"; });

    bar.append("rect")
        .attr("width", function(d) { return x(d.value); })
        .attr("height", barHeight - 1);

    bar.append("text")
        .attr("x", function(d) { return x(d.value) - 3; })
        .attr("y", barHeight / 2)
        .attr("dy", ".35em")
        .text(function(d) { return d.value; });

});

function type(d) {
    d.value = +d.value; // coerce to number
    return d;
}
```
Generative AI
### Levels of Artificial General Intelligence

<table>
<thead>
<tr>
<th>Performance (rows) x Generality (columns)</th>
<th>Narrow clearly scoped task or set of tasks</th>
<th>General wide range of non-physical tasks, including metacognitive abilities like learning new skills</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 0: No AI</td>
<td>Narrow Non-AI calculator software; compiler</td>
<td>General Non-AI human-in-the-loop computing, e.g., Amazon Mechanical Turk</td>
</tr>
<tr>
<td>Level 1: Emerging equal to or somewhat better than an unskilled human</td>
<td>Emerging Narrow AI GOFAI (Boden, 2014); simple rule-based systems, e.g., SHRDLU (Winograd, 1971)</td>
<td>Emerging AGI ChatGPT (OpenAI, 2023), Bard (Anil et al., 2023), Llama 2 (Touvron et al., 2023), Gemini (Pichai and Hassabis, 2023)</td>
</tr>
<tr>
<td>Level 2: Competent at least 50th percentile of skilled adults</td>
<td>Competent Narrow AI toxicity detectors such as Jigsaw (Das et al., 2022); Smart Speakers such as Siri (Apple), Alexa (Amazon), or Google Assistant (Google); VQA systems such as PaLI (Chen et al., 2023); Watson (IBM); SOTA LLMs for a subset of tasks (e.g., short essay writing, simple coding)</td>
<td>Competent AGI not yet achieved</td>
</tr>
<tr>
<td>Level 3: Expert at least 90th percentile of skilled adults</td>
<td>Expert Narrow AI spelling &amp; grammar checkers such as Grammarly (Grammarly, 2023); generative image models such as Imagen (Saharia et al., 2022) or Dall-E 2 (Ramesh et al., 2022)</td>
<td>Expert AGI not yet achieved</td>
</tr>
<tr>
<td>Level 4: Virtuoso at least 99th percentile of skilled adults</td>
<td>Virtuoso Narrow AI Deep Blue (Campbell et al., 2002), AlphaGo (Silver et al., 2016, 2017)</td>
<td>Virtuoso AGI not yet achieved</td>
</tr>
<tr>
<td>Level 5: Superhuman outperforms 100% of humans</td>
<td>Superhuman Narrow AI AlphaFold (Jumper et al., 2021; Varadi et al., 2021), AlphaZero (Silver et al., 2018), StockFish (Stockfish, 2023)</td>
<td>Artificial Superintelligence (ASI) not yet achieved</td>
</tr>
</tbody>
</table>

The Evolution of LLMs

1. In 2017, Google released the "Transformer Model", which can be used in question-answering systems, reading comprehension, sentiment analysis, instant translation of text or speech, and more.

2. In 2018, OpenAI proposed "GPT" and Google proposed the "BERT" model, widely used in search engines, speech recognition, machine translation, question-answering systems, and more.

3. From 2018 to 2022, most of the research focused on BERT-related algorithms, when GPT performance was inferior to BERT.

4. In 2023, ChatGPT (GPT3.5) was proposed by OpenAI, which significantly improves NLU’s ability to understand most texts and surpasses humans in some areas.

In NLU

- **CNN**: Local feature
- **RNN**: Front and Back Dependency Issues
- **Self-Attention**: One to all attention, more flexible and trainable

**Note:**

- **Diffusion** refers to a technique used in generative models, often in the context of image generation or language modeling.
- **Self-Attention** allows each element of the sequence to attend to all other elements, enabling the model to learn long-range dependencies.
- **Dependency Issues** highlight the challenges in accurately capturing the relationships between elements in sequences, which can affect the model's performance.

**Diagram:**

- The diagram shows a timeline from 2016 to 2023, highlighting key developments in text and image generation models.
- Key models and technologies, such as GPT, BERT, ChatGPT, and various diffusions, are visually represented along with their respective years of introduction.
The speed of development of Generative AI

The blessings of scale
AI training runs, estimated computing resources used
Floating-point operations, selected systems, by type, log scale

Quick learners
The speed at which artificial intelligence models master benchmarks and surpass human baselines is accelerating. But they often fall short in the real world.

Benchmarks
- MNIST (handwriting recognition)
- SQuAD 1.1 (reading comprehension)
- Switchboard (speech recognition)
- SQuAD 2.0 (reading comprehension)
- ImageNet (image recognition)
- GLUE (language understanding)

Human performance

Sources: “Compute trends across three eras of machine learning,” by J. Sevilla et al., arXiv, 2012; Our World in Data
Generative AI Application

- Image Generator
- Speech Generator
- text Generator
- Pose Generator
- Robot

Condition Model
- Conditional image Generator
- Audio Generator
- Chat Bot
- Summarization and Translation

Multi-Model
- NLU + Image Generator
- NLU + Robot

EECS E6895 BIG DATA ANALYTICS
What is Generative AI

Artificial intelligence systems that can produce high quality content, specifically **text, images, and audio**.

ChatGPT/OpenAI

Bard/Google

Bing Chat/Microsoft

1. "Shine bright, even if your eyes are digital! Introducing our latest sunglasses line for robots."
2. "Because even bots deserve to look cool in the sun! #RobotChic"
3. "Future’s so bright, our robots need shades! Check out our new eyewear collection now."

Prompts

- Write three captions for a social post about our new line of sunglasses for robots.
- Give me three ideas for t-shirt graphics that would appeal to llamas.
- Find me some recipes for snacks for hungry pirates.

Sure! Here are three ideas for t-shirt graphics that would appeal to llamas:

1. A cartoon llama with a funny expression, such as a surprised llama, a grumpy llama, or a silly llama.
2. A llama doing something that is typically associated with llamas, such as eating grass, spitting, or being majestic.

Sure! Here are some fun and tasty pirate-themed snacks that you can prepare for hungry pirates:

1. **Pirate Mix**: This snack includes pirate pegleg pretzel sticks, Gorilla Munch cannonballs, and “gold” “fish” whole grain Goldfish crackers.
Multimedia Generation

A beautiful, pastoral mountain scene. Landscape painting style (Midjourney)

Two cute kittens playing (DALL-E)
Generating Text using Large Language Models

Text generation process

I love eating

bagels with cream cheese
my mother’s meatloaf
out with friends

Prompt

AI output
How LLM works

LLMs are built by using supervised learning (A→B) to repeatedly predict the next word.

My favorite food is a bagel with cream cheese

<table>
<thead>
<tr>
<th>Input (A)</th>
<th>Output (B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>My favorite food is a</td>
<td>bagel</td>
</tr>
<tr>
<td>My favorite food is a bagel</td>
<td>with</td>
</tr>
<tr>
<td>My favorite food is a bagel with cream</td>
<td>cream</td>
</tr>
<tr>
<td>My favorite food is a bagel with cream</td>
<td>cheese</td>
</tr>
</tbody>
</table>

When we train a very large AI system on a lot of data (hundreds of billions of words), we get a Large Language Model like ChatGPT.
A New Way to Find Information

<table>
<thead>
<tr>
<th>What is the capital of South Africa?</th>
<th>What does LLM stand for?</th>
</tr>
</thead>
<tbody>
<tr>
<td>South Africa has three capital cities:</td>
<td>LL.M. stands for &quot;Legum Magister&quot; in Latin, which translates to &quot;Master of Laws&quot; in English.</td>
</tr>
<tr>
<td>1. Pretoria (administrative capital)</td>
<td>What about in the context of AI?</td>
</tr>
<tr>
<td>2. Cape Town (legislative capital)</td>
<td>In the context of AI, LLM refers to “Large Language Model”.</td>
</tr>
<tr>
<td>3. Bloemfontein (judicial capital)</td>
<td></td>
</tr>
</tbody>
</table>
Rewrite this for clarity:

Students all around the world are realizing learning has to happen not just in school but throughout life including after graduation. Students worldwide understand that learning extends beyond school and continues throughout life.

Write a 300 word story involving trucks for a 3 year old to encourage them to brush their teeth.

Once upon a time in a town called Toothville, there lived a happy tow truck named Timmy.

One day Timmy noticed that his shiny bumper had a dull spot. Worried, he went to the doctor [...]
Examples of Generated Images

A picture of a woman smiling

A futuristic city scene

A cool, happy robot
Image Generation

Typically ~100 steps for diffusion model
Image generation from Text

Input (A) "green banana" → Output (B)

Image 1 → Image 2 → Image 3 → Image 4
In 2015, Bengio’s Model focuses on every phenon’s recognition as the combined weights.

\[
\alpha_i = \text{Attend}(s_{i-1}, \alpha_{i-1}, h)
\]
\[
g_i = \sum_{j=1}^{L} \alpha_{i,j} h_j
\]
\[
y_i \sim \text{Generate}(s_{i-1}, g_i),
\]

\(h\) : Input  
\(\alpha_i\) : Attention Weight  
\(y_i\) : Output

Transformer [Vaswani_2017]

In 2017, 8 Google researchers proposed Transformer Neuron Networks based on Attention, which was adopted by ChatGPT.

Noam Shazeer proposed scaled dot-product attention, multi-head attention and the parameter-free position representation.

Jakob Uszkoreit proposed replacing RNNs with self-attention and started the effort to evaluate this idea.
Transformer

- Transformer is a Deep Learning Model based on Self-Attention

- Transformer encodes and decodes data with different weights.

- Examples of transformer language models include: GPT (GPT-1, GPT-2, GPT-3, ChatGPT) and BERT models (BERT, RoBERTa, ERNIE).

Attention to Transformer

**Encoder Self Attention**
1. Multi-head Attention
2. $\text{Query} = \text{Key} = \text{Value}$

**Decoder Self Attention**
1. Masked Multi-head Attention
2. $\text{Query} = \text{Key} = \text{Value}$

**Encoder-Decoder Attention**
1. Multi-head Attention
2. Encoder Self attention = $\text{Key} = \text{Value}$
3. Decoder Self attention = $\text{Query}$

*Figure 1: The Transformer - model architecture.*
Transformer

Encoder  Transformer  Decoder

Transformer

哥大學生很棒! / Columbia University students are great!

Encoder → Decoder
Transformer
Attention

<table>
<thead>
<tr>
<th>weights</th>
<th>Columbia</th>
<th>university</th>
<th>students</th>
<th>are</th>
<th>great</th>
<th>!</th>
</tr>
</thead>
<tbody>
<tr>
<td>哥</td>
<td>1</td>
<td>0.5</td>
<td>0.2</td>
<td>0</td>
<td>0.3</td>
<td>0.2</td>
</tr>
<tr>
<td>大</td>
<td>0.5</td>
<td>1</td>
<td>0.2</td>
<td>0.1</td>
<td>0.3</td>
<td>0.1</td>
</tr>
<tr>
<td>學</td>
<td>0.2</td>
<td>0.2</td>
<td>1</td>
<td>0</td>
<td>0.5</td>
<td>0.2</td>
</tr>
<tr>
<td>生</td>
<td>0.3</td>
<td>0.3</td>
<td>0.8</td>
<td>0.5</td>
<td>0.5</td>
<td>0.6</td>
</tr>
<tr>
<td>很</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>棒</td>
<td>0.3</td>
<td>0.3</td>
<td>0.5</td>
<td>0.5</td>
<td>1</td>
<td>0.8</td>
</tr>
<tr>
<td>!</td>
<td>0.2</td>
<td>0.1</td>
<td>0.2</td>
<td>0</td>
<td>0.8</td>
<td>1</td>
</tr>
</tbody>
</table>
Transformer Translation

Transformer uses 6 layers of encoder and decoder to achieve the same quality of SOTA English-German and English-French translation.
BERT Introduction

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin  Ming-Wei Chang  Kenton Lee  Kristina Toutanova
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Abstract

We introduce a new language representation model called BERT, which stands for Bidirectional Encoder Representations from Transformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

There are two existing strategies for applying pre-trained language representations to downstream tasks: feature-based and fine-tuning. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning all pre-trained parameters. The two approaches share the same objective function during pre-training, where they use unidirectional language models to learn general language representations.

BERT Introduction

- 2018 Google’BERT has 24 layers of Transformer Encoder
- BERT’s original model is based on Wikipedia and booksorpus, using unsupervised training to create BERT.
- At Stanford’s Machine Reasoning Test SQuAD1.1 beats human performance.
- Google NLU English was replaced from seq2seq to BERT

BERT understands language’s meaning


EECS E6895 BIG DATA ANALYTICS
In 2018, BERT Comprehension test outperformed human

<table>
<thead>
<tr>
<th>Rank</th>
<th>Date</th>
<th>Model</th>
<th>EM</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Oct 05, 2018</td>
<td>BERT (ensemble)</td>
<td>87.433</td>
<td>93.160</td>
</tr>
<tr>
<td>2</td>
<td>Oct 05, 2018</td>
<td>BERT (single model)</td>
<td>85.083</td>
<td>91.835</td>
</tr>
<tr>
<td>2</td>
<td>Sep 09, 2018</td>
<td>nlnet (ensemble)</td>
<td>85.356</td>
<td>91.202</td>
</tr>
<tr>
<td>2</td>
<td>Sep 26, 2018</td>
<td>nlnet (ensemble)</td>
<td>85.954</td>
<td>91.677</td>
</tr>
</tbody>
</table>

Since the release of SQuAD 1.0, the community has made rapid progress, with the best models now rivaling human performance on the task. Here are the ExactMatch (EM) and F1 scores evaluated on the test set of v1.1.
Transformer to GPT

Transformer
Input -> Encoder -> Latent Feature + Masked Output -> Decoder -> Output

GPT
Input -> Decoder(with Casual mask) -> shift Output

Masked Language Learning

An a day keeps the doctor away
apple 95%
banana 5%

An apple a day keeps the doctor away

Autoregressive Learning
An apple
a 99%
watch 1%

EECS E6895 BIG DATA ANALYTICS
60
# ChatGPT

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software dev job</td>
<td>ChatGPT would be hired as L3 Software Developer at Google: the role pays $183,000/year.</td>
</tr>
<tr>
<td>Politics</td>
<td>ChatGPT writes several Bills (USA).</td>
</tr>
<tr>
<td>MBA</td>
<td>ChatGPT would pass an MBA degree exam at Wharton (UPenn).</td>
</tr>
<tr>
<td>Accounting</td>
<td>GPT-3.5 would pass the US CPA exam.</td>
</tr>
<tr>
<td>Legal</td>
<td>GPT-3.5 would pass the bar in the US.</td>
</tr>
<tr>
<td>Medical</td>
<td>ChatGPT would pass the United States Medical Licensing Exam (USMLE).</td>
</tr>
<tr>
<td>AWS certificate</td>
<td>ChatGPT would pass the AWS Certified Cloud Practitioner exam.</td>
</tr>
<tr>
<td>IQ (verbal only)</td>
<td>ChatGPT scores IQ=147, 99.9th %ile.</td>
</tr>
<tr>
<td>SAT exam</td>
<td>ChatGPT scores 1020/1600 on SAT exam.</td>
</tr>
</tbody>
</table>

https://lifearchitect.ai/chatgpt/
“GPT-3 is applied without any gradient updates or fine-tuning, with tasks and few-shot demonstrations specified purely via text interaction with the model.”

From Language Models are Few-Shot Learners (2020)
GPT Evolution

Not only Bigger and Bigger

Fine Tuning
Or
In context Few shot Learning

175B parameters

GPT-3

2048 token size

2020

Thinking and Answering policy optimization
Reinforcement Learning from Human Feedback
(RLHF)

Step I
Labeler
Prompts & Text
Prompts & Text^n
Training

Step II
Pre-trained model
Critic
Text
Scoring
... 5
... 4
... 3
... 2
... 1

Step III
Pre-trained model
Policy Model
Text
Reward Model
Scoring
... 5
... 4
... 3
... 2
... 1

Policy Training

Inference
ChatGPT

EECS E6895 BIG DATA ANALYTICS
63
What is Next?

What goal do you want

Connect to Internet to Find the way or answer

Like JARVIS

Write code and debug

In Iron Man

Do research and try & error

https://agentgpt.reworkd.ai/zh

Make a Hypothesis and provide it

The no longer future will come true

Summarization and Organization

AutoGPT

EECS E6895 BIG DATA ANALYTICS
A.I. for Drug Design

-- New Era is Now. Significant Scientific and Business Breakthrough
When people in 50 years later look back history, they may recognize ‘Today’ is a scientific breakthrough moment in Biology as an equivalent of Newton’s moment in Physics

-- Graphen 2023
“Tools from established companies like Google DeepMind, startups like Graphen, and AI chipsets from vendors like NVIDIA and Intel will help accelerate the speed of drug discovery, development, and testing, allowing pharmaceutical companies and healthcare authorities to combat the pandemic.” – ABI research, May 2020
Graphen Ardi Full-Brain Platform’s Graph Models enable Graphen Atom Tools that better simulate biological functions.

https://www.graphen.ai/products/ardi.html

Ardi Graph Analytics and Database of zillions of nodes and edges were deployed in one of the world’s largest institutes in 2018.

Graphen’s Differentiator: Graph Modeling + Deep Learning + Generative AI
Graphen Small Mole Drug Dev ➔ 1/27 of the Time; 1/9000 of the Cost, comparing to traditional methods

1. Generate Synthesis and low side effects Drug
   - Generate de novo new compound
     - Samples: 1.5M
     - Required Time: 7 Days

2. Filtering drug by ADMET and Solubility
   - Manufacturable and Safety Screening
     - Samples: 30K
     - Required Time: 3 Days

3. Filtering Drug by Graphen QF Energy model and Kinase model
   - Dynamic Target-Lead interaction simulation
     - Samples: 2.5K
     - Required Time: 5 Days

4. Disease-Clinical/Cell Target mapping (Spectrum Mutant Prediction)
   - Target Cell and Disease Finding
     - Samples: 0.1K
     - Required Time: 5 Days

5. Synthesis and Wet Lab Evaluation
   - Chemical Synthesis Kinase Assay/Cell Assay
     - Samples: 30
     - Required Time: 30+30 Days

AI is making a paradigm shift to reduce the risk of drug development via:
- Enrich pipeline
- Increase Probability of Success
- Cost Reduction
- More Precise, Fewer Side Effects
- Rare Disease & Personalized Drugs