



**BERKELEY LAB**

Bringing Science Solutions to the World



# Large Language Models

Xiangyang Ju

28 July 2023

[US ATLAS Machine Learning Training](#)

# Natural Language Processing

## Tasks:

- question answering (QA)
- language translation
- sentiment analysis
- completing a sentence
- reading comprehension
- picking the best ending to a story
- and many others

## Data representation: Sequence

How to express words with numbers?

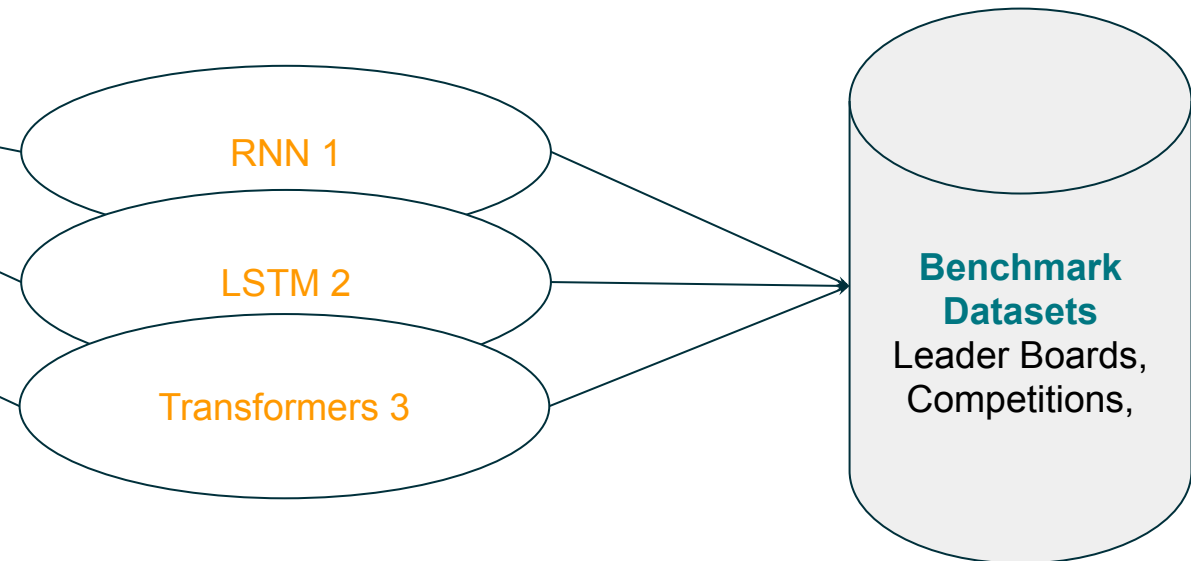
# Tokenization and Embedding

- Tokenization is to divide text into smallest units for processing
  - In English, tokens are usually words or roots (i.e -ing, -ment)
  - Some special tokens:
    - [PAD], [UNK], [CLS], [SEP]
- Static vocabulary → Only process a fixed number of token
  - Each token is indexed in the vocabulary
- Embedding or word2vec → represent words as dense vectors in a continuous space
  - It should capture semantic relationships between tokens and beyond. e.g.
    - BERT: Bidirectional Encoder *Representations* from Transformers

# Natural Language Processing Models

## Tasks:

- reading comprehension
- question answering (QA)
- language translation
- sentiment analysis
- completing a sentence
- picking the best ending to a story
- and many others



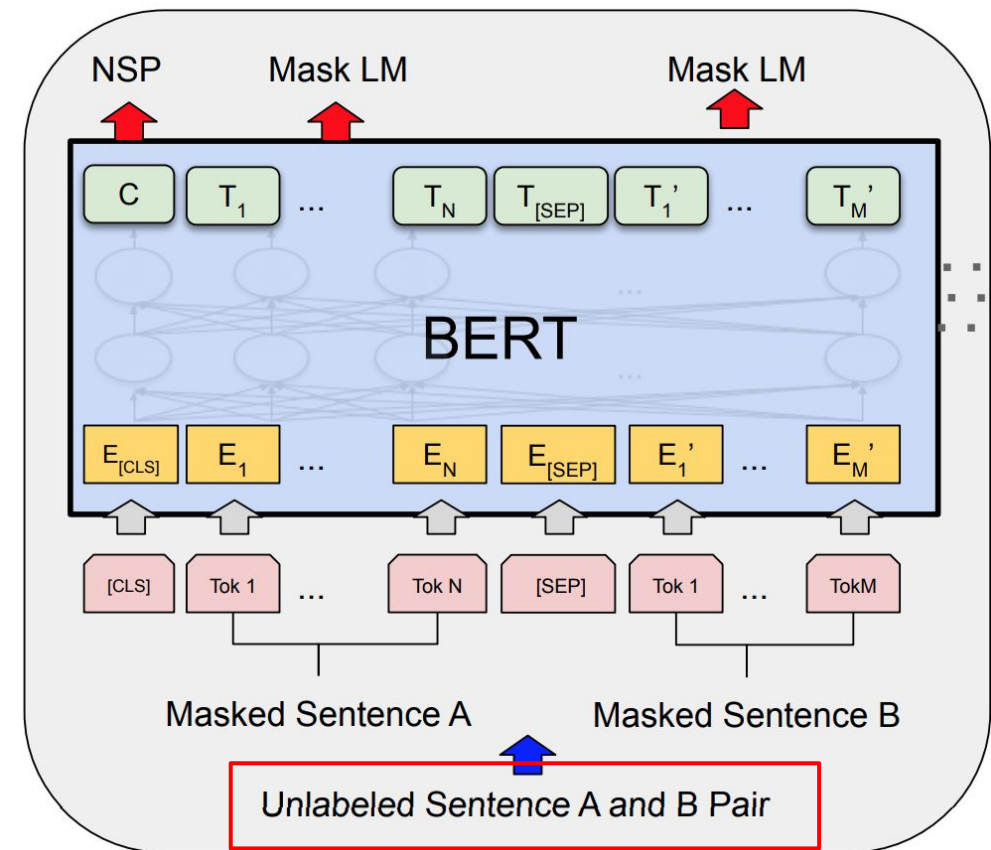
- For each task, one can beat another by creating a new model or new training strategies or other techniques.
- Creating tasks-specific, high-quality datasets is tedious and time consuming; It basically prevents the scaling

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

A new paradigm for language understanding: **Pretraining + Fine Tuning**

## Pretraining with large general dataset

- Train a Transformer-based encoder on *surrogate tasks*



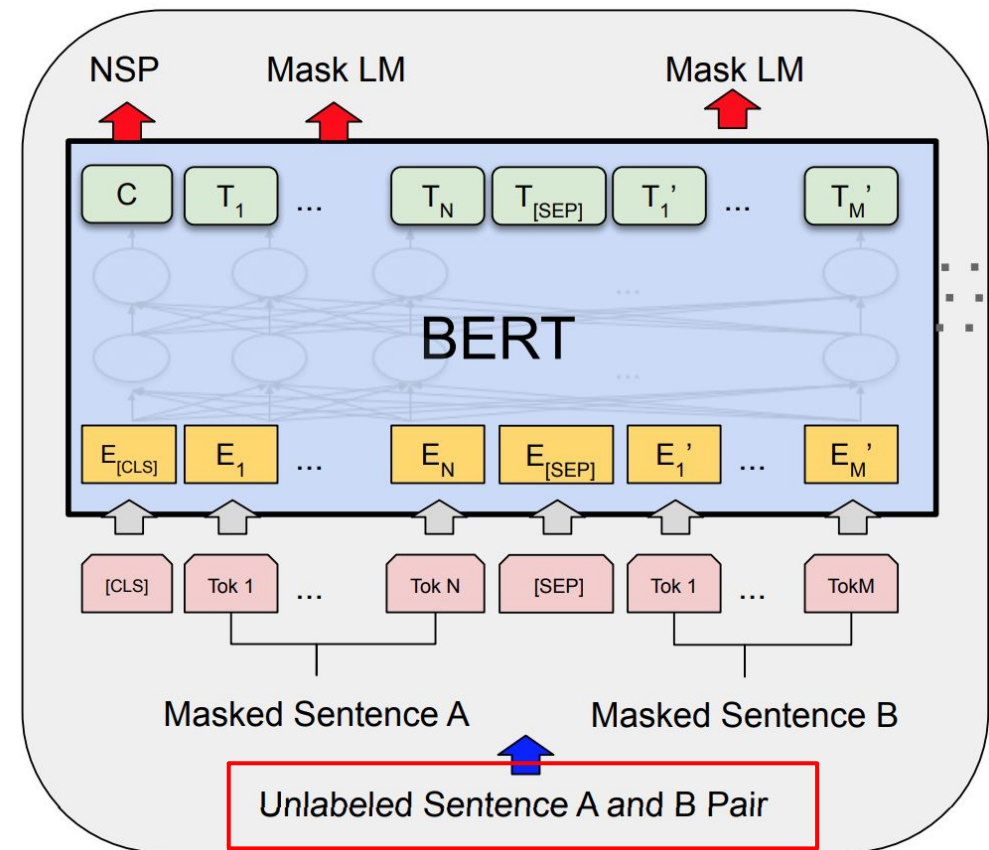
# BERT: Pre-training of Bidirectional Transformers for Language Understanding

A new paradigm for language understanding: **Pretraining + Fine Tuning**

## Pretraining with large unlabeled dataset

- Train a Transformer-based encoder on *surrogate tasks*
- Each word (token) is then encoded in a dense representation

The new representation can be used in *multiple tasks*



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

A new paradigm for language understanding: **Pretraining + Fine Tuning**

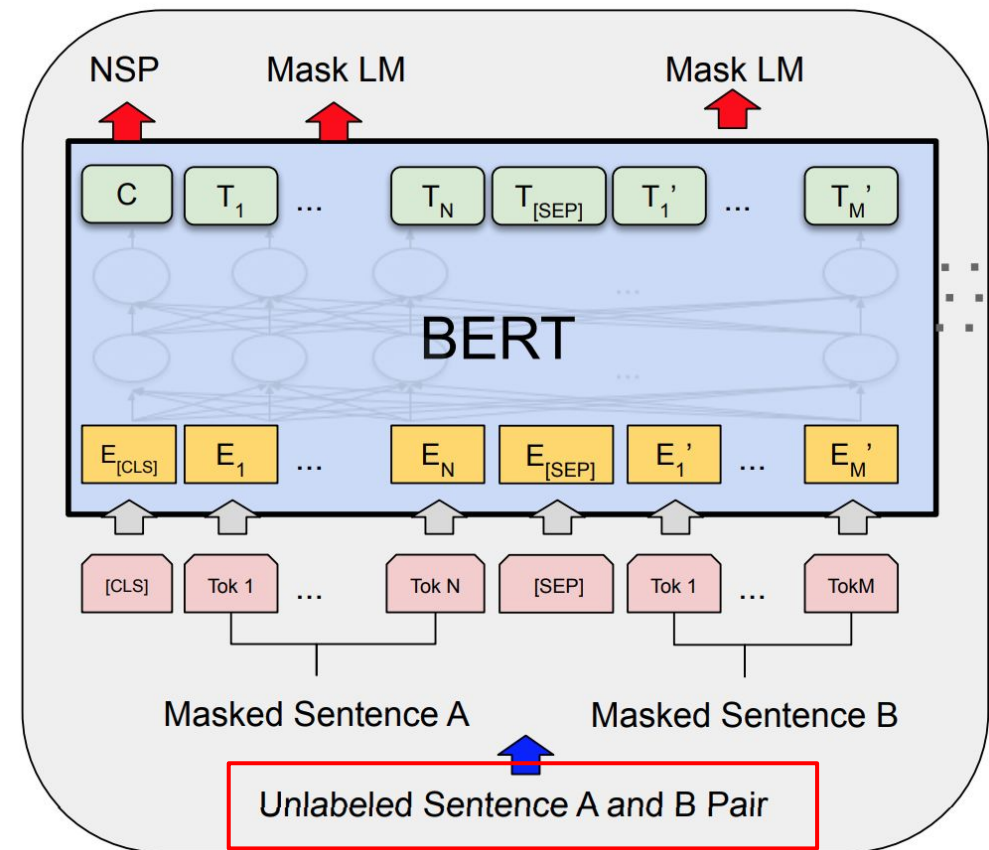
## Pretraining with large unlabeled dataset

- Train a Transformer-based encoder on *surrogate tasks*
- Each word (token) is then encoded in a dense representation

The new representation can be used in *multiple tasks*

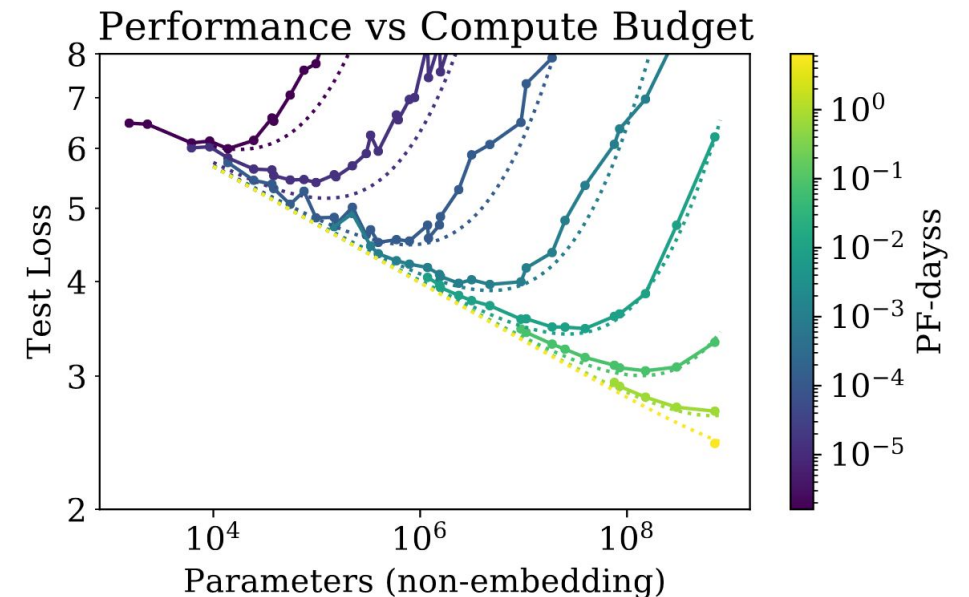
## Fine Tuning on a smaller labeled dataset

- Either train a new model for the *target* task
- Or continue train the same model with very small LR.



# The Scaling Law for Language Models

- Performance depends strongly on scale, weakly on model shape
  - Three factors: *model size*, *dataset size*, and *the amount of compute* used for training
- Universality of overfitting:
  - Performance scales as  $N^{0.74} / D$ , meaning every time we increase model size,  $N$ , by 8x, we only need to increase the data,  $D$ , by 5x to avoid a performance penalty
- Universality of training:
  - Training curves follow power-laws whose parameters are roughly independent of the model size.





# GPT3: Language Models are Few-Shot Learners

Say no to “pretraining + fine tuning”, say yes to “meta-learning”

Why not “pretraining + fine tuning”?

- Creating labeled data for fine tuning is tedious
- Fine tuning may potentially pick up spurious correlations, thus could not generalize for out-of-distribution data

# GPT3: Language Models are Few-Shot Learners

Say no to “pretraining + fine tuning”, say yes to “meta-learning”

Meta-learning is that LLM is trained with a broad set of tasks and can “adapt to” or “recognize” the desired task.

## Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

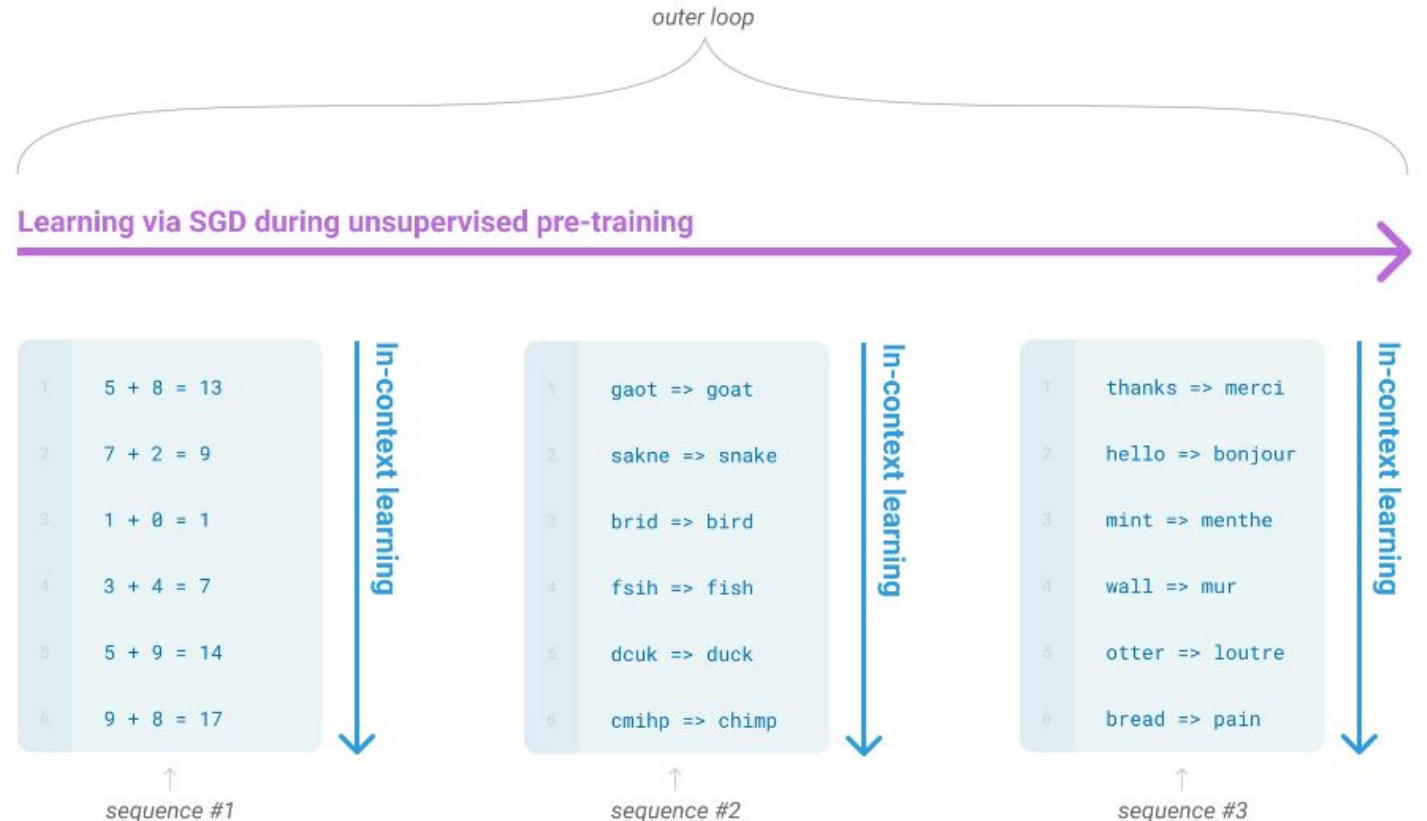
```
1 Translate English to French: ← task description
2 cheese => ..... ← prompt
```

## One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 sea otter => loutre de mer ← example
3 cheese => ..... ← prompt
```

inner loop



# GPT3: Language Models are Few-Shot Learners

Say no to “pretraining + fine tuning”, say yes to “meta-learning”

- Say no to fine tuning, but yes to “[Reinforcement Learning Human Feedback](#) (arxiv:1706.03741)” or “[follow instructions with human feedback](#) (arxiv:2203.02155)”
- The idea is to train a NN with labeled data. OpenAI hired 40 contractors to label the data.
- The trained NN served as a reward function in the reinforcement learning.
- The LLM has to adapt their outputs to get the maximum reward
- The reward can also serve as a test of the goodness of the output

# Training data and Benchmarks

Following are used in [GPT3](#)

Common Crawl datasets have trillion, unfiltered, low quality, duplicated words

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4



+



A woman is outside with a bucket and a dog. The dog is running around trying to avoid a bath. She...

- A. rinses the bucket off with soap and blow dry the dog's head.
- B. uses a hose to keep it from getting soapy.
- C. gets the dog wet, then it runs away again.**
- D. gets into a bath tub with the dog.

**HellaSwag Benchmark** uses adversarial machine-generated wrong answers  
GPT-3: 79.3% accuracy with a few-shot setting

# Some interesting models

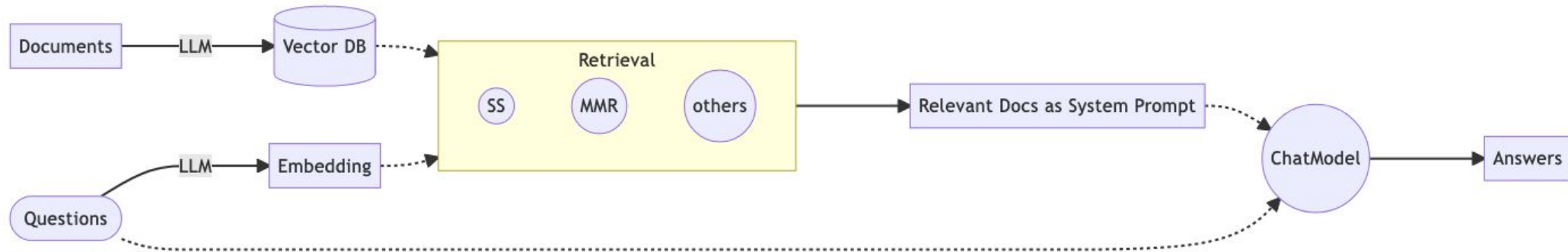
Thanks to A. Huang.

Model Name	Model Size	Functionality	Architecture	Reference
GPT-3	175B	Task-agnostic NLP (translation, cloze tasks, Q&A, etc.)	Pre-trained on 45 TB of texts, downstream tasks perform well with few-shot learning.	<a href="#">Paper</a> <a href="#">Code</a>
Bert	340M	Downstream task-specific NLP (translation, prediction, etc.)	Bidirectional model pre-trained on masked input sequences (800M + 2500M words), fine-tuned on downstream tasks.	<a href="#">Paper</a> <a href="#">Code</a>
Roberta	355M	Word prediction	Randomly mask 15% of words and predict the masked words to learn inner representation	<a href="#">Paper</a> <a href="#">Code</a>
T5	60M-11B	Text-to-text translation	Used a unified framework that converts all NP into text-to-text format	<a href="#">Paper</a> <a href="#">Code</a>

# How do we use LLM?


AI-assistant programming, bugging, summarization, Q&A on twikis

# Document Q&A



- Learned through the [deeplearning.ai](https://www.deeplearning.ai) open course for [LangChain](https://www.langchain.com).
- The idea is
  - 1) build a embedded database;
  - 2) embed the question;
  - 3) retrieve relevant information by querying the embedded question from the database
- It avoids the “Hallucination” and can even provides a reference

# Outlook : ATLAS ChatGPT

- 
- Imagine an ATLAS ChatGPT trained on atlas papers (and all CDS comments!), internal notes, all mailing lists, all indico meetings slides !?!?

Prompt : why is there an electron veto BDT for tau id ?

Answer : it was published in 20XX (Ref), first discussed in meetings (indico,indico), it is described in details in (CDS). It allows to reduce electron contamination by  $x\%$  with an efficiency loss of  $y\%$ . The main objection at the time was () Since then it has been used in many ATLAS analyses like (ref, ref) and the coming paper (glance)

- This would be done by « fine-tuning »



# ATLAS GPT, from Daniel

## Soft Introduction

Q I'm new to Athena. Where should I start?

Welcome to the world of Athena! To get started, you can follow these steps:

1. ATLAS holds [Analysis Software Tutorials](#) several times per year. These tutorials contain a lot of information about how to get started as an ATLAS software user and give a brief overview of how Athena works. This is a great place to begin, especially if you are a member of ATLAS and need to use ATLAS computing resources.
2. Read through the [git development tutorial](#) to understand in more detail how Athena developers use git. This will be essential once you have code you want to get accepted into Athena.
3. Learn about [Athena Nightly Builds](#) and how to use them. As a developer, you will normally be using a nightly build to have the latest version of Athena.
4. Occasionally, there are [ATLAS Developer Tutorials](#) available (this page may be restricted to ATLAS members).

To run a simple serial Athena job, follow these steps:

1. Log into LXPLUS or another CVMFS-equipped Linux cluster and then:

```
export ATLAS_LOCAL_ROOT_BASE=/cvmfs/atlas.cern.ch/repo/ATLASLocalRootBase
alias setupATLAS='source ${ATLAS_LOCAL_ROOT_BASE}/user/atlasLocalSetup.sh'
```

(These lines can go into your login script)

2. On LXPLUS or a Centos7 cluster:

```
setupATLAS
```

On any other kind of Linux cluster:

```
setupATLAS -c centos7
```

3. Set up the latest release and finally run the job:

```
asetup Athena, master, latest
python -m AthExHelloWorld.HelloWorldConfig
```

Uses [MarkPrompt](#)

# chATLAS, Gabriel

Uses [LangChain](#).

## Result: chATLAS

vDev



BeautifulSoup

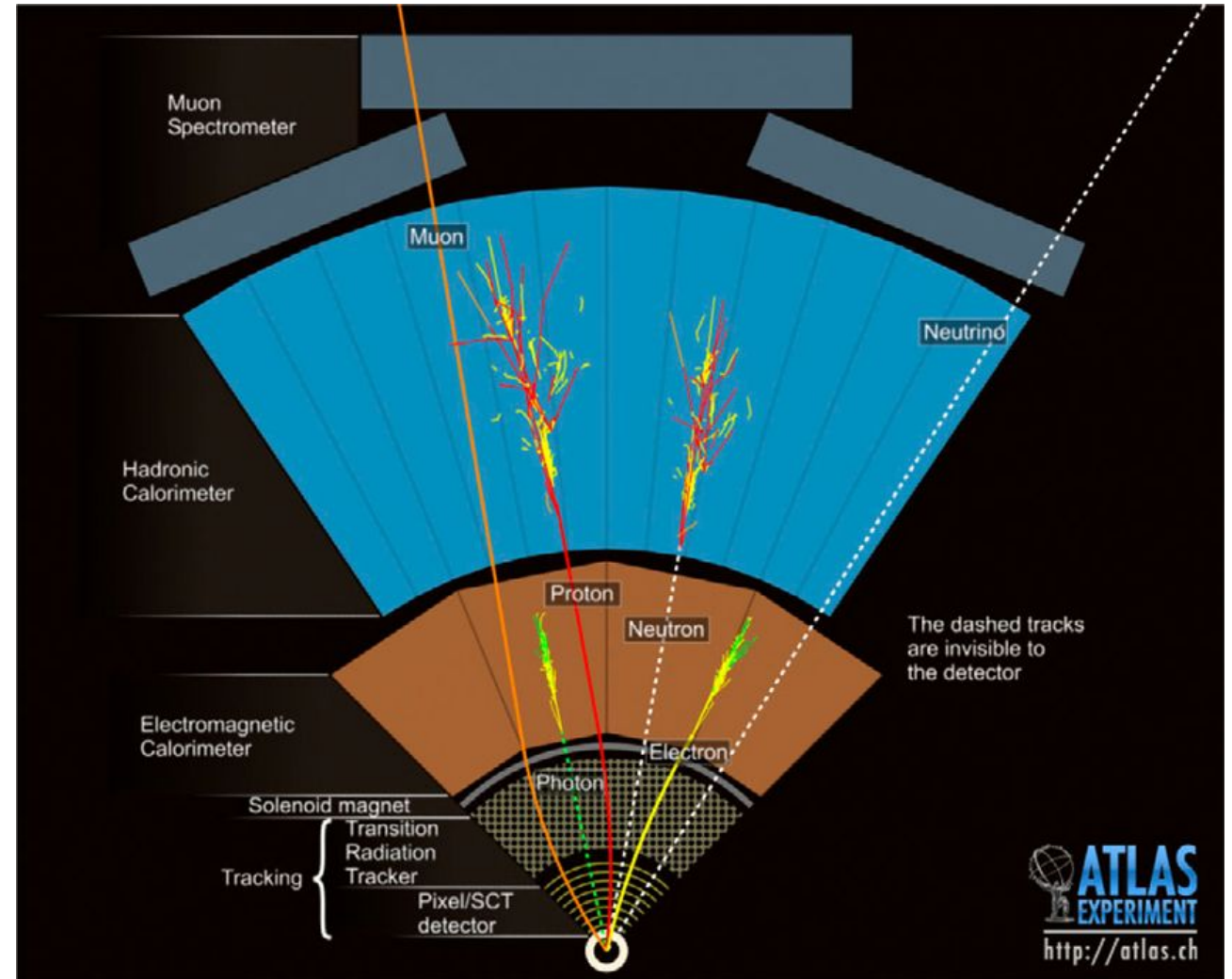


# A Large Model for ATLAS?

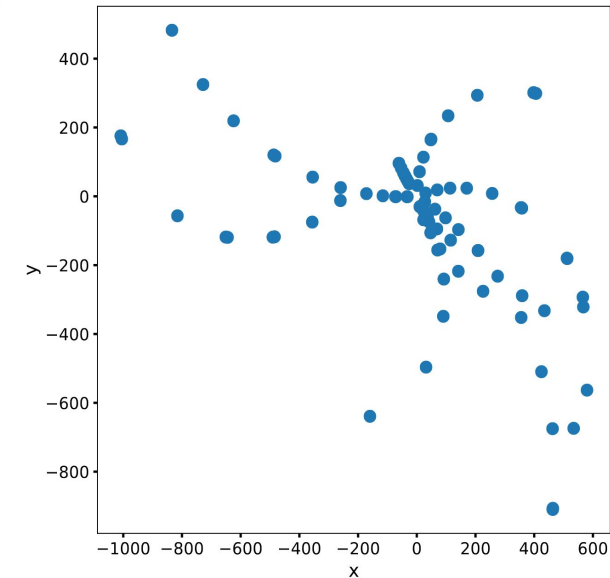
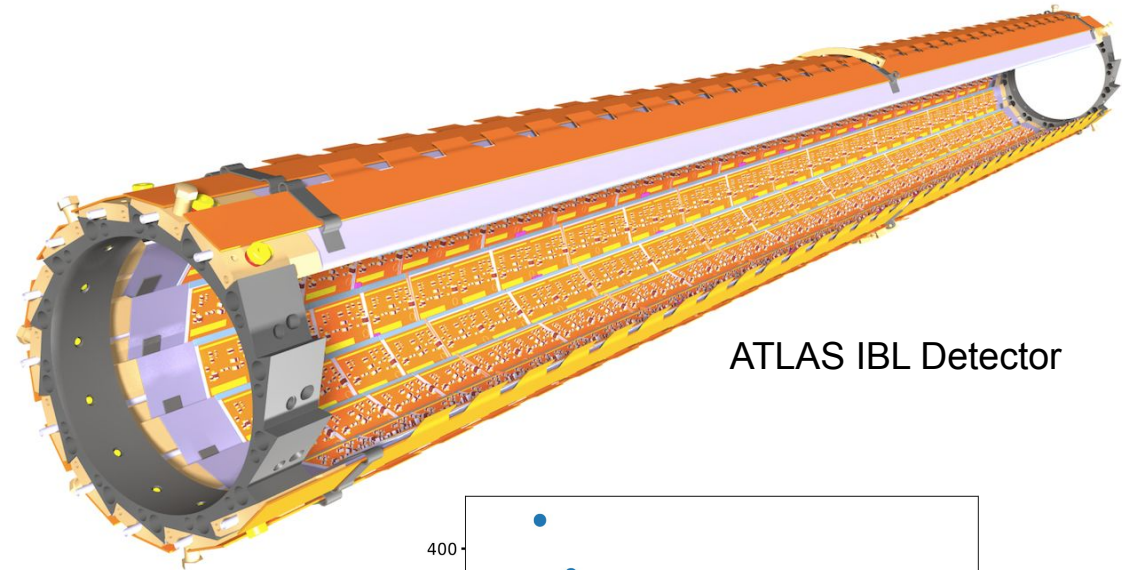
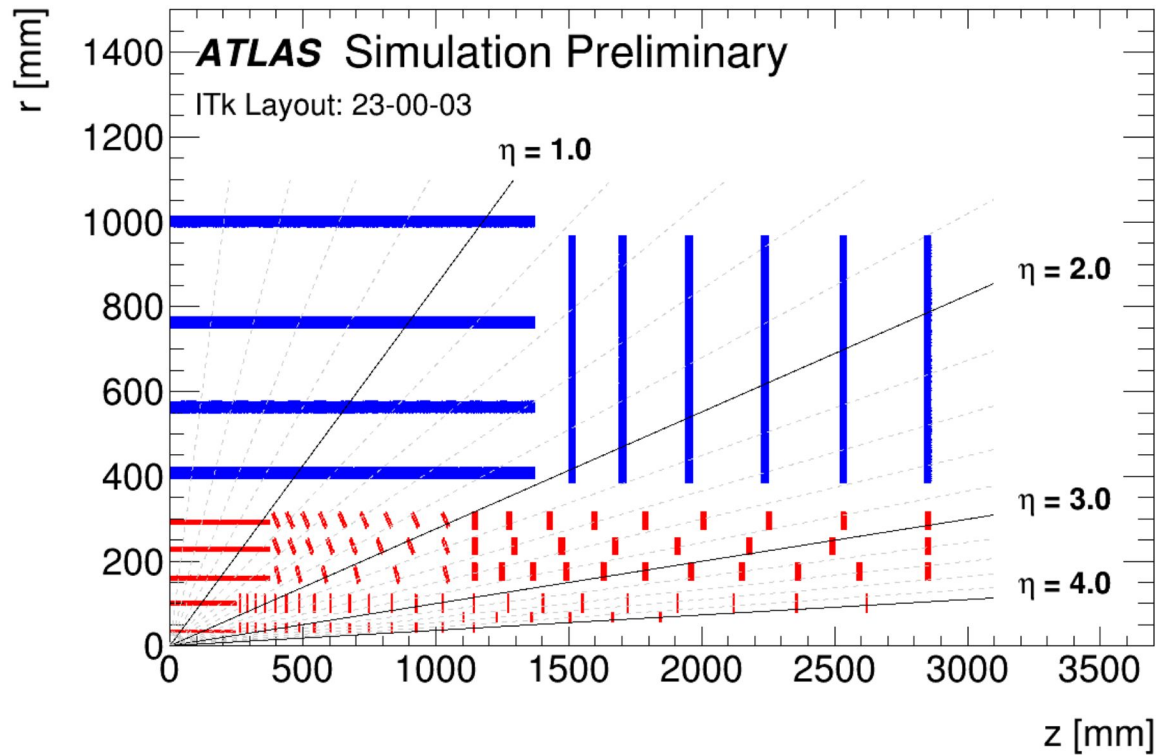
For detector simulation, particle reconstruction, or physics analysis

# NLP vs ATLAS detector

Analogy between NLP and ATLAS	
Detector elements	Words
All detector elements	Vocabulary
Particle trajectories or showers	Sentences
Collision Events	Paragraphs
Events from the same physics process	Sections

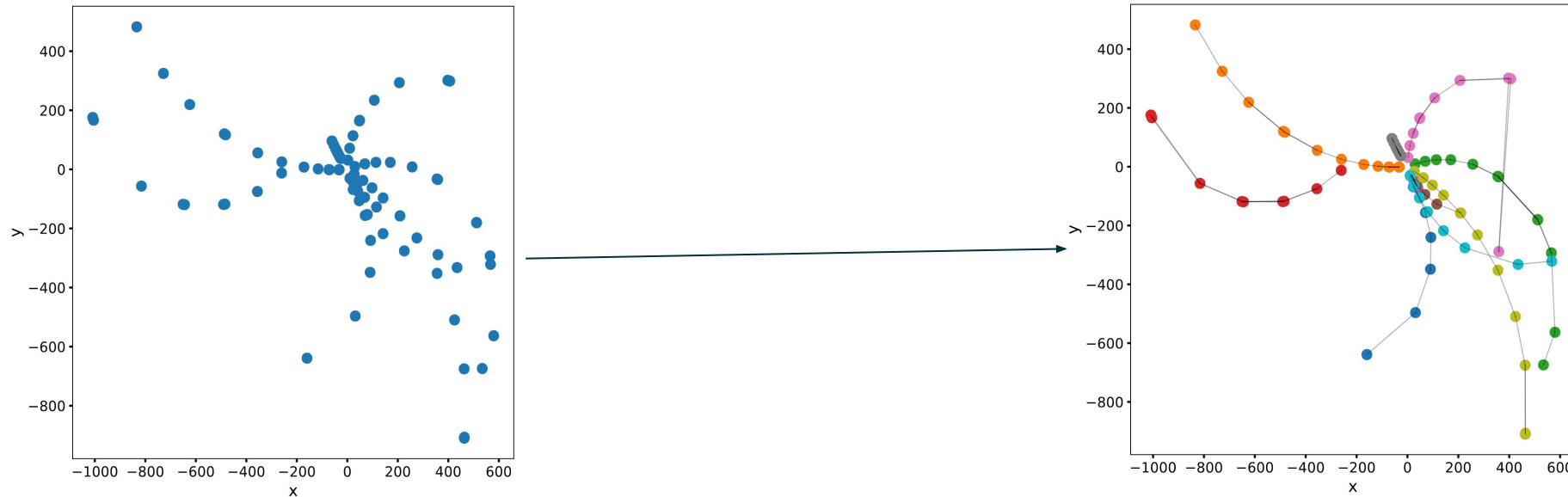


# Deep detector representation for particle tracking



- The detector provides a vocabulary and the recorded spacepoints are the words.
- The task of particle tracking reconstruction is to form sentences from those words.

# Deep detector representation for particle tracking

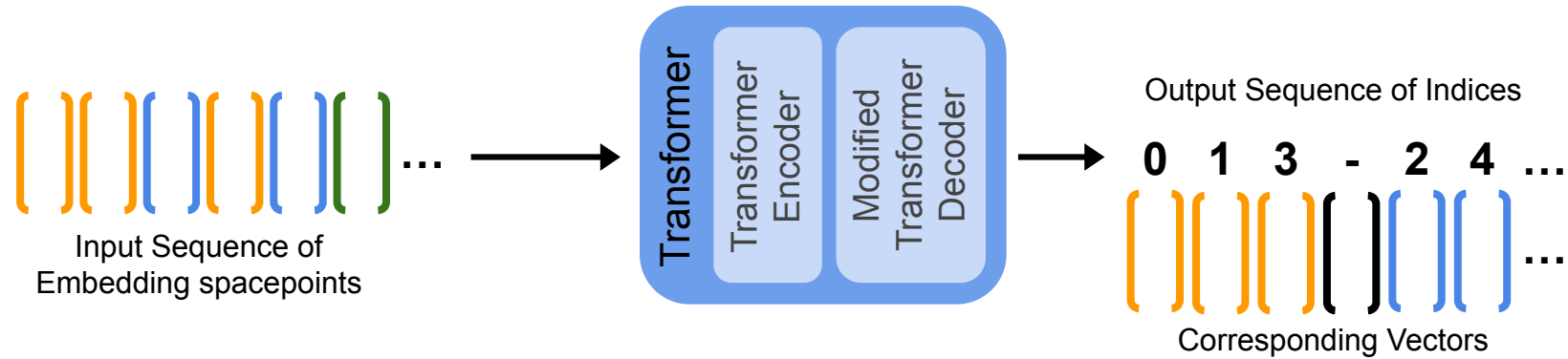


- In ITk, the Pixel system contains  $\sim 9400$  modules,  $\sim 1.4 \times 10^9$  pixels; Strip system contains  $> 20,000$  modules
- At HL-LHC, the ITk records about 300,000 hits

How would you tackle this problem in the context of Large Language Models?

# A sorting algorithm for particle tracking

Thanks to Y. Melkani

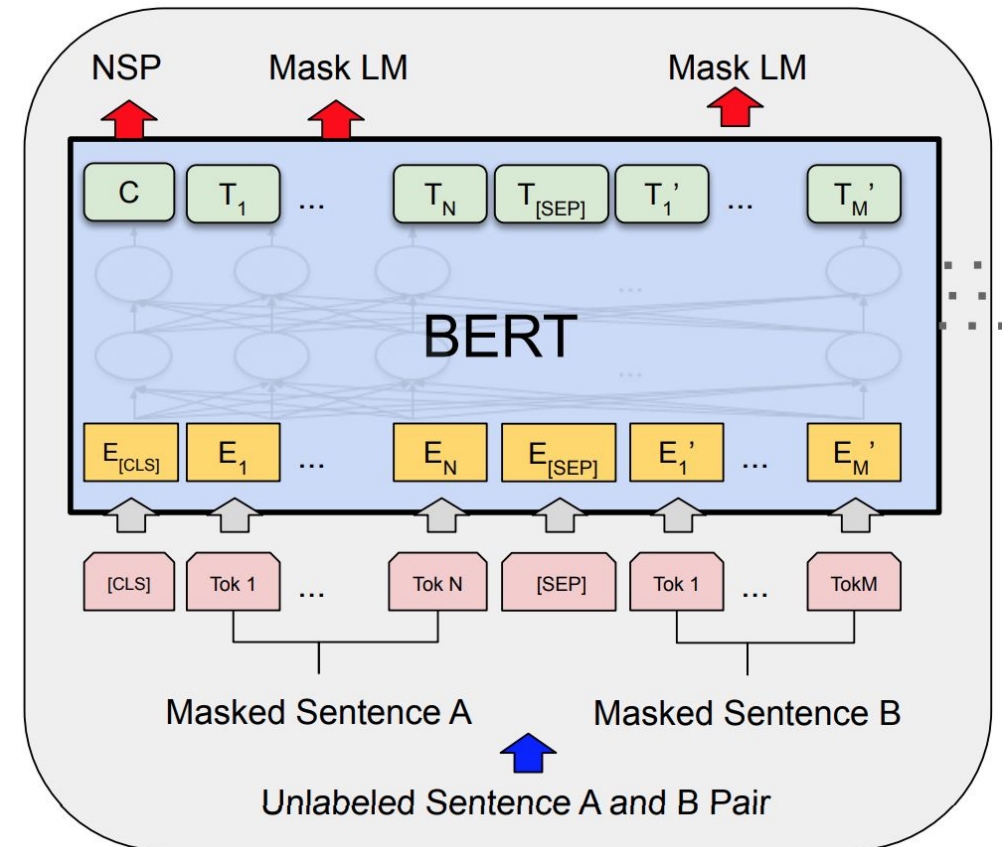
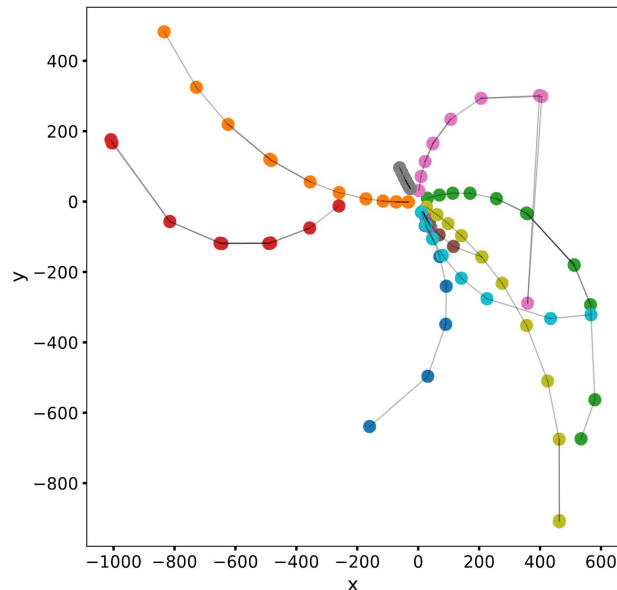


- Input is a sequence of space points ordered by their distance away from the collision point
- The Model is trained to sort the input sequence in a way space points from the same tracks are together



# Deep detector representation for particle reconstruction

- Train a Transformer to encode detector elements with a surrogate tasks: predicting missing space points in a track
- Use the detector encoder for different downstream tasks, such as b-tagging





# Conclusion

- NLP models have found a way to scale their training data and the language model and achieved impressive performance
  - We have a huge scale real data and simulated data
- Looking back the history of how NLP comes up with LLMs can be inspiring
- People in ATLAS already starts to use LLMs such as chatGPT for programming, Q&A, summarizing twiki pages, and so on
  
- Do we need a large model for ATLAS? What if you have unlimited resources to build a large model, which type of large model you would like to build to solve which problem?