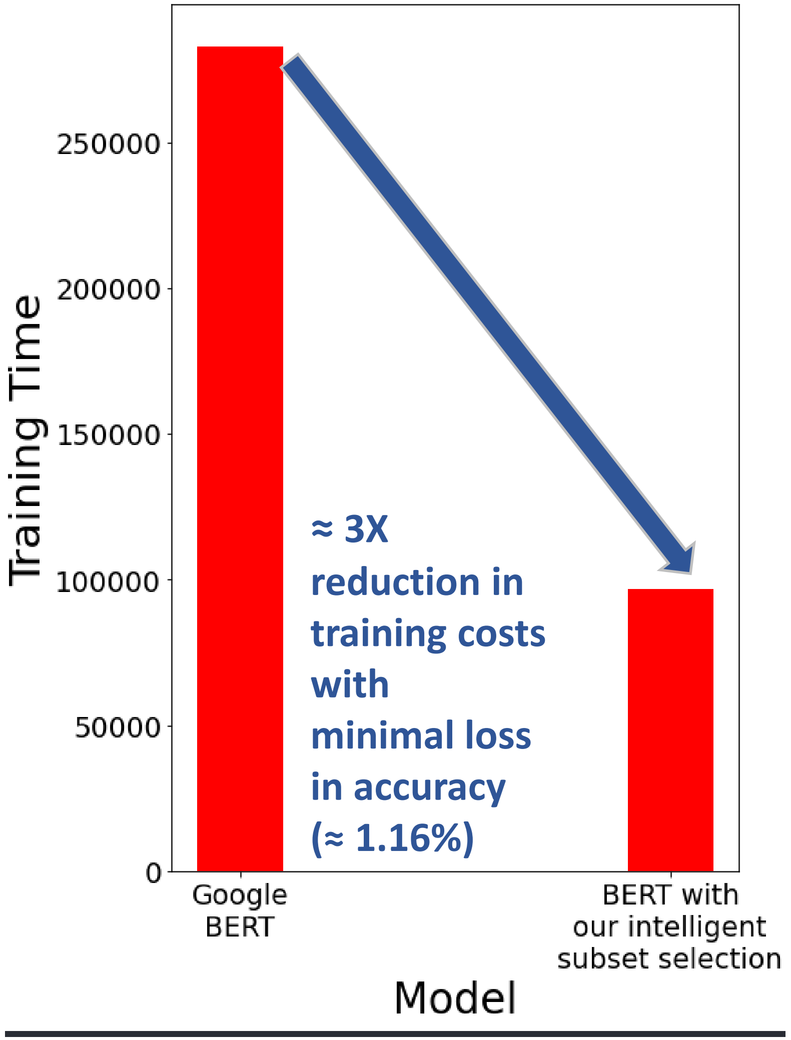
Efficient Training of Large-Language Models (LLMs) using Subset Selection

Language Model Training is Expensive

GPT-3

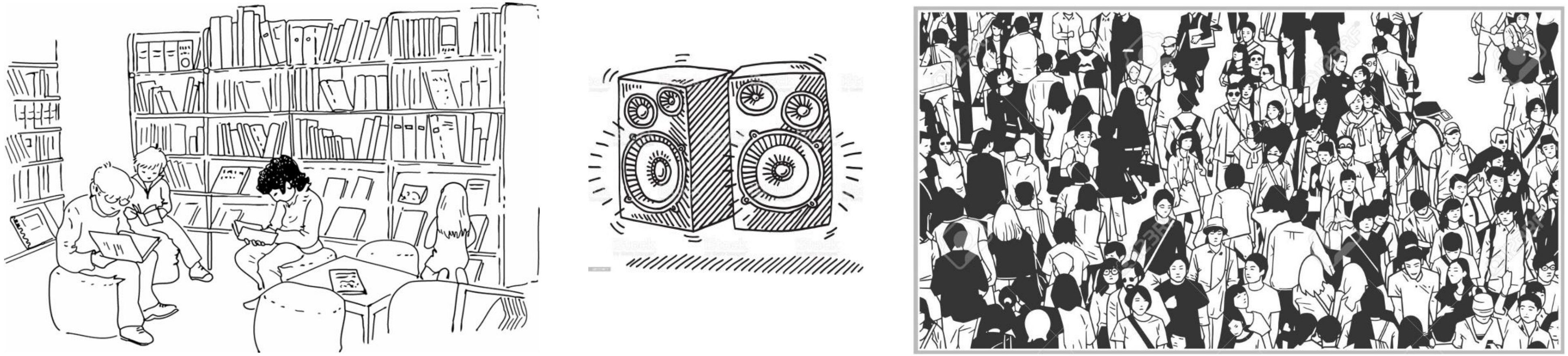
estimated cost: 12,000,000 USD
CO2 emissions: equivalent to lifetime emission of 120 cars



≈ 3X reduction in training costs with minimal loss in accuracy (≈ 1.16%)

Proposed Approach

- **Data Efficient Machine Learning**: Can we train these large state-of-the-art language models with only a sample of massive datasets(say 15% or 25%), while having negligible impact on their performance?
- **How to obtain the subset?** Model the problem as *submodular maximization*
- **The intuition behind submodular functions**:

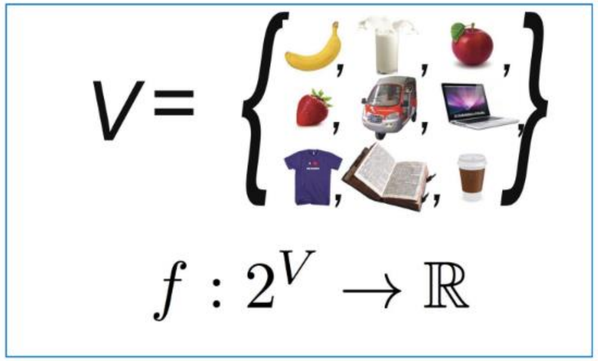


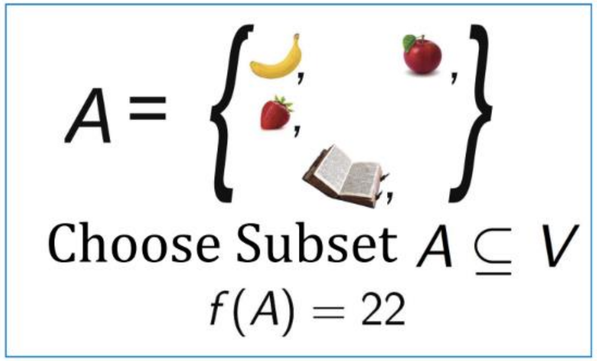
In which case does the loud-speaker make more difference?

Subset Selection using Submodular Functions

Subset Selection as a Set Function Maximization Problem:

$$\mathcal{S}^* = \underset{\mathcal{S}:\mathcal{S}\subseteq\mathcal{D},|\mathcal{S}|=k}{\arg\max} \; f(\mathcal{S})$$

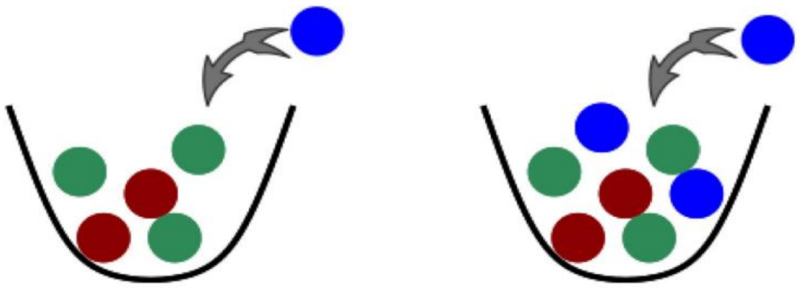




Given, a set function f, selecting a subset that maximizes it in a brute-force manner requires combinatorial number of function evaluations.

Submodularity: A set function f is submodular, if it satisfies the diminishing gains property.

$$f(A \cup v) - f(A) \geq f(B \cup v) - f(B), \text{ if } A \subseteq B$$



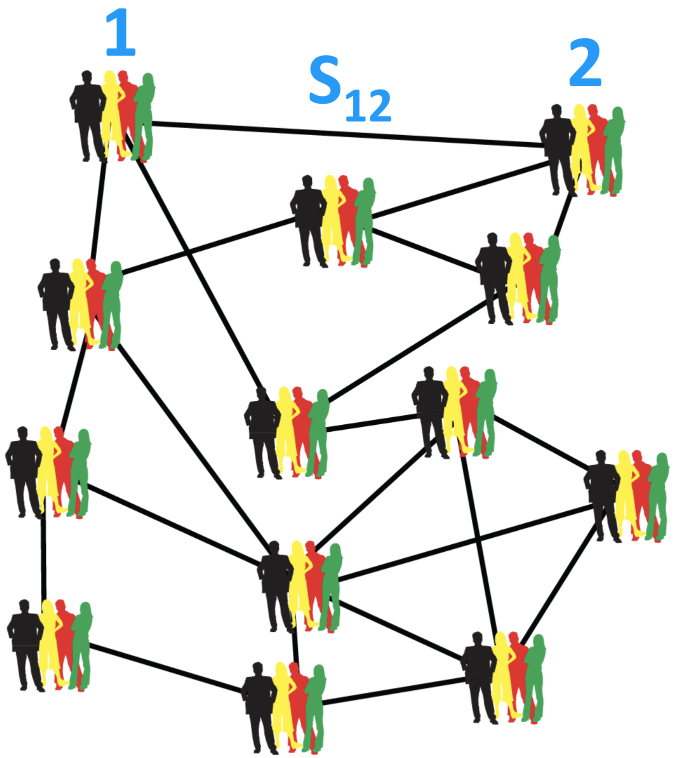
$f = \#$ of distinct colors of balls in the urn.

Why is Submodularity Important?

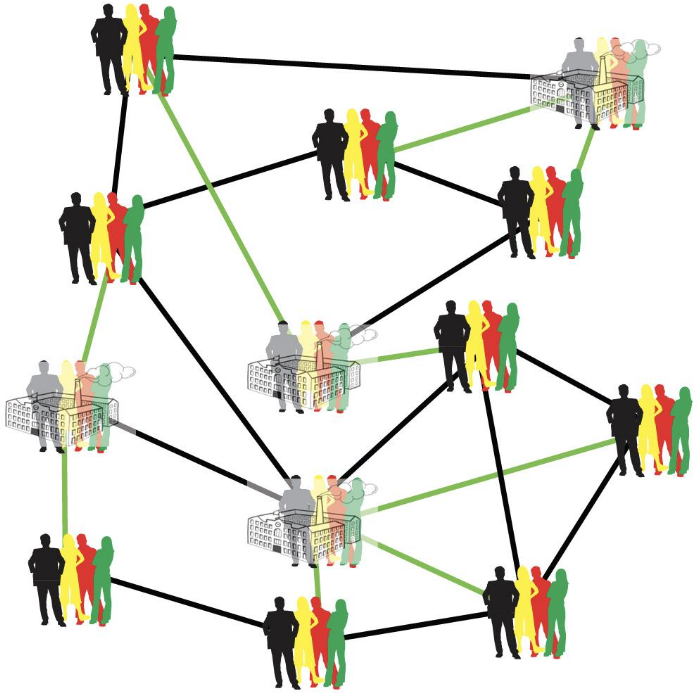
When set function is submodular, we can maximize it using standard "greedy" algorithm while achieving approximation guarantees for the selected subsets.

Facility Location Function

as efficiently as possible, place **facilities** at certain **locations** to satisfy sites having various demands

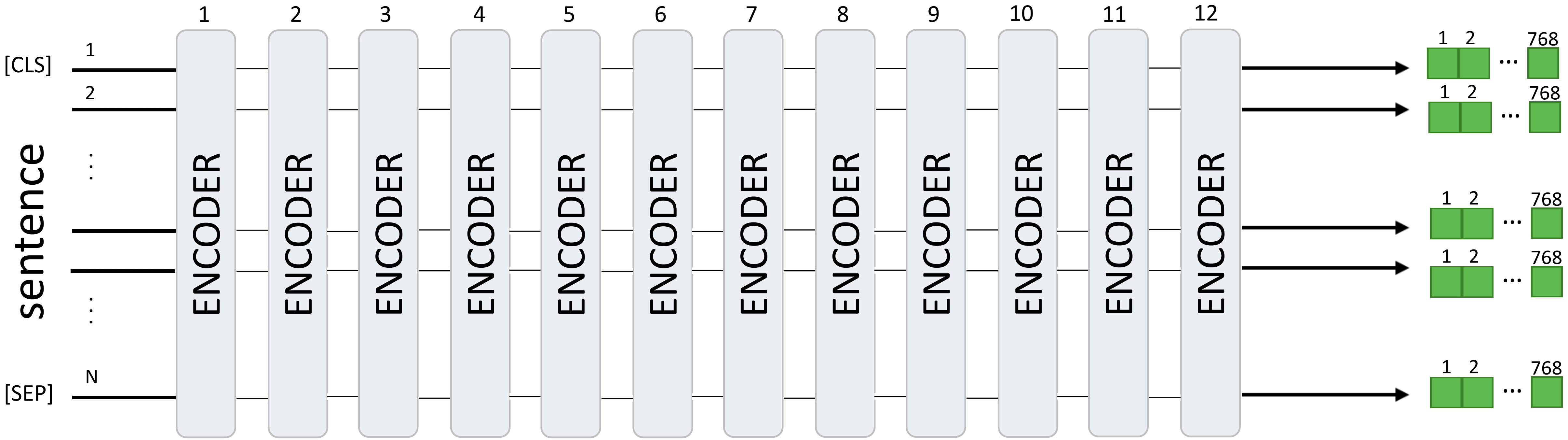


$$f_{FL}(S) = \sum_{i \in \mathcal{V}} \max_{j \in S} s_{ij}$$

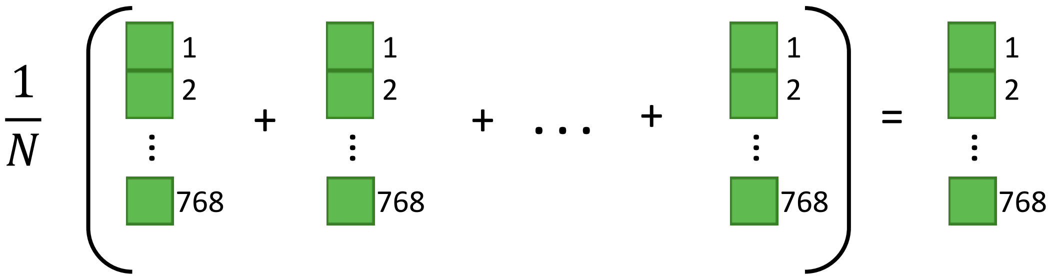


In context of data-subset selection, facility location problem can be viewed as K-Medoid Clustering.

Computing s_{ij}s

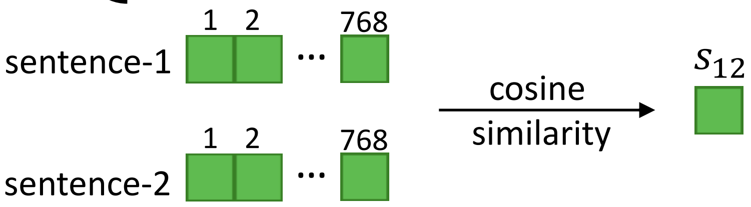


Sentence Embedding:



(Mean of i^{th} layer token embeddings)

Sentence Similarity:



Cosine similarity between sentence embeddings

Which layer embeddings to use?

A Primer in BERTology: What We Know About How BERT Works

Anna Rogers
Center for Social Data Science
University of Copenhagen

Olga Kovaleva
Dept. of Computer Science
University of Massachusetts Lowell

Anna Rumshisky
Dept. of Computer Science
University of Massachusetts Lowell
arum@cs.uml.edu

“Syntactic information is most prominent in the middle layers of BERT &
Semantics is spread across the entire model”

“Had most success reconstructing syntactic tree depth from middle layers(6-9) &
Best subject-verb agreement around layers 8-9”

“The final layers of BERT are most task-specific(MLM task in this context)
and so middle layers are more transferable”

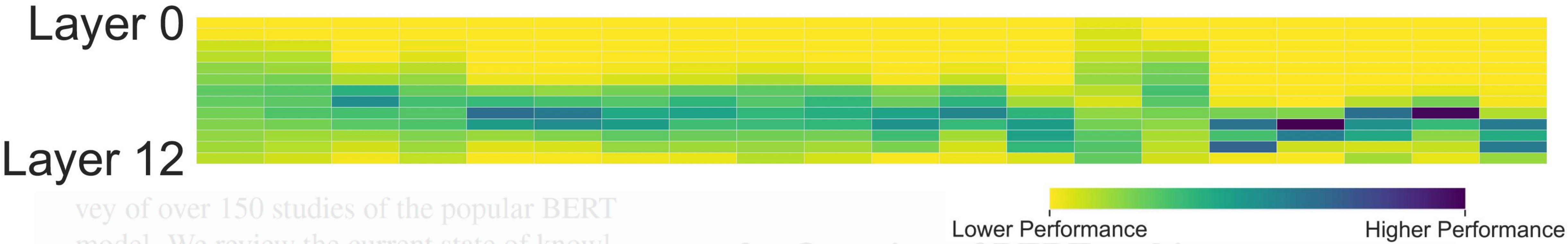
“This also explains why in fine-tuning, the final layers change the most &
restoring weights of lower layers after fine-tuning doesn’t hurt performance”

Abstract

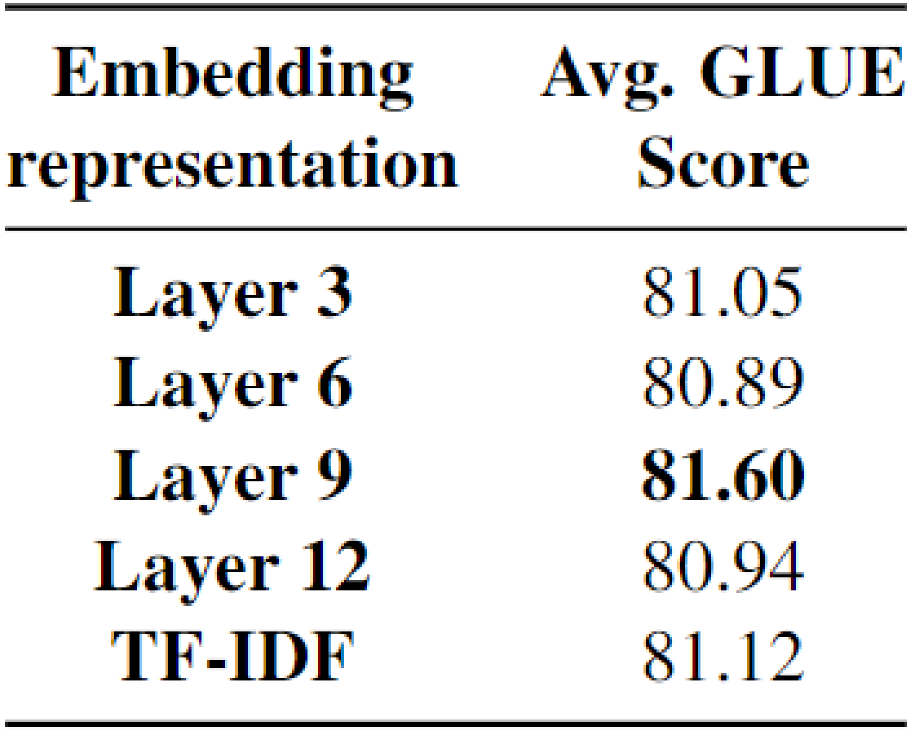
vey of over 150 studies of the popular BERT model. We review the current state of knowledge about how BERT works, what kind

improve BERT’s architecture, pre-training and fine-

2 Overview of BERT architecture



Embedding Layer Ablation



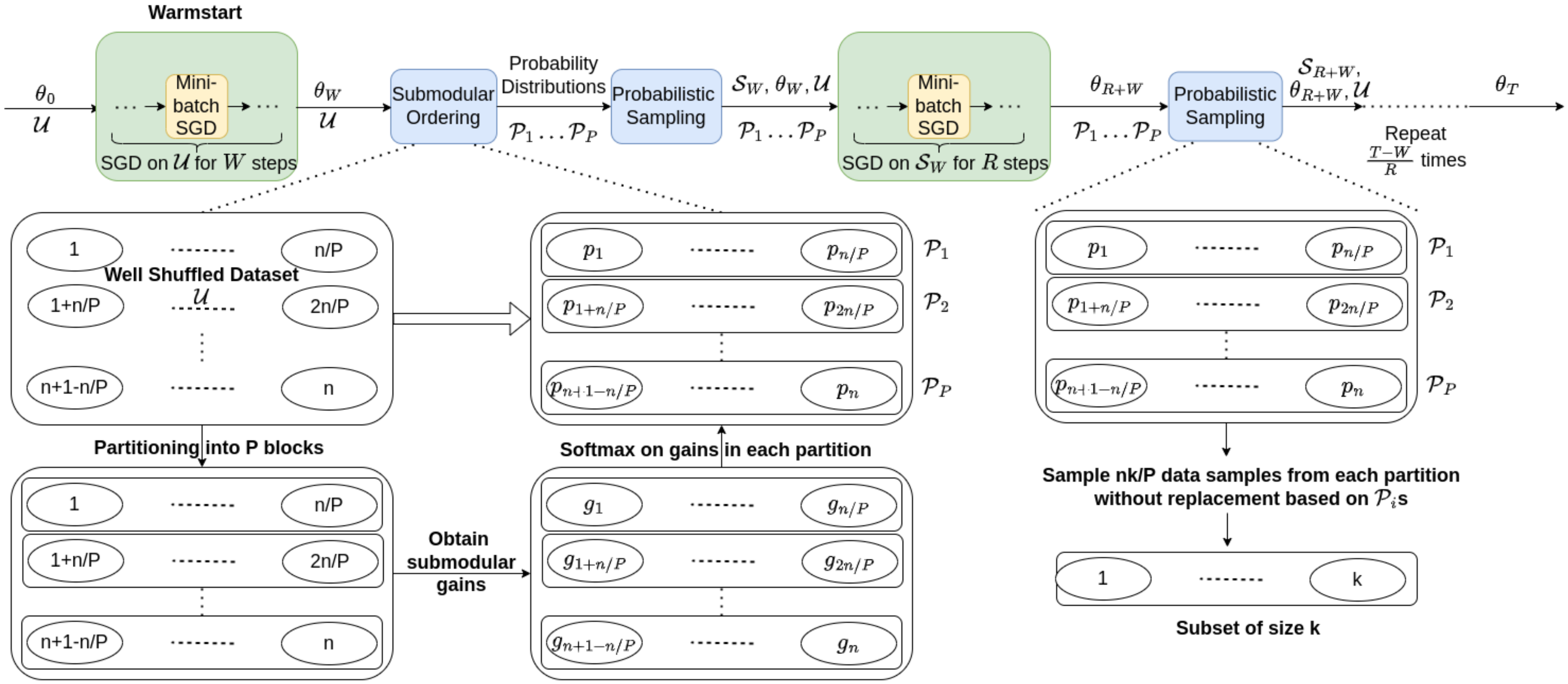
Embedding representation	Avg. GLUE Score
Layer 3	81.05
Layer 6	80.89
Layer 9	**81.60**
Layer 12	80.94
TF-IDF	81.12

Performance of BERT model trained on subsets selected by using embeddings from different layers.

Some scaling issues & mitigating them

- Initial BERT embeddings are not good(since model weights are initialized randomly)
  - We use an initial warm-start of ~80K steps & then get the embeddings for subset selection
- Impractical to store huge similarity matrices (40million x 40million)
  - Randomly partition the dataset and select subsets from each partition
- It doesn't help to use the same subset repetitively for the entire training
  - Adaptive subset selection is done by introducing some diversity in the subset
  - Training on subsets can lead to overfitting of the model

Training Pipeline


Warmstart
θ_0
$\mathcal{U}$
Mini-batch SGD
SGD on $\mathcal{U}$ for W steps
θ_W
$\mathcal{U}$
Submodular Ordering
Probability Distributions
$\mathcal{P}_1 \dots \mathcal{P}_P$
Probabilistic Sampling
$\mathcal{S}_W, \theta_W, \mathcal{U}$
$\mathcal{P}_1 \dots \mathcal{P}_P$
Mini-batch SGD
SGD on $\mathcal{S}_W$ for R steps
θ_{R+W}
$\mathcal{P}_1 \dots \mathcal{P}_P$
Probabilistic Sampling
$\mathcal{S}_{R+W}, \theta_{R+W}, \mathcal{U}$
θ_T
Repeat $\frac{T-W}{R}$ times
Well Shuffled Dataset $\mathcal{U}$
1, n/P, 1+n/P, 2n/P, n+1-n/P, n
Partitioning into P blocks
Obtain submodular gains
g_1, $g_{n/P}$, $g_{1+n/P}$, $g_{2n/P}$, $g_{n+1-n/P}$, g_n
Softmax on gains in each partition
p_1, $p_{n/P}$, $p_{1+n/P}$, $p_{2n/P}$, $p_{n+1-n/P}$, p_n
$\mathcal{P}_1$
$\mathcal{P}_2$
$\mathcal{P}_P$
Sample nk/P data samples from each partition without replacement based on $\mathcal{P}_i$s
1, k
Subset of size k

Note that submodular maximization uses a greedy algorithm. The submodular gain of an element is the gain associated with adding that element to the current solution. Since the function is submodular, elements that are added earlier have greater gains and therefore a greater probability of being selected.

Experiments

- We analyze the efficiency vs performance tradeoff of the considered LLMs pre-trained on a subset of data with different baselines.

LLMs Considered:

- BERT Base (110 M parameters) - Wikipedia + Book Corpus
- GPT2 Small (124 M parameters) - Open WebText

Baselines:

- LLMs pre-trained on randomly selected subsets of same size.
- LLMs pre-trained on full datasets.

LLMs Performance Evaluation:

- GLUE Benchmark
- LAMA Probe Benchmark (Knowledge retention analysis)

BERT Results

Method	Average GLUE Score
Vanilla BERT (1M steps)	82.76
Vanilla BERT (early stopping, 250K steps)	81.27 (-1.49)
Random Online (250K steps)	81.04 (-1.72)
Loss-based Sampling (250K steps)	81.05 (-1.71)
OURS (BERT + FL Subsets) (250K steps)	**81.6 (-1.16)**

Baselines:

Random Online: BERT model trained on randomly selected subsets of size 25%

Loss-based Sampling: BERT model trained on subsets containing top 25% samples based on loss value.

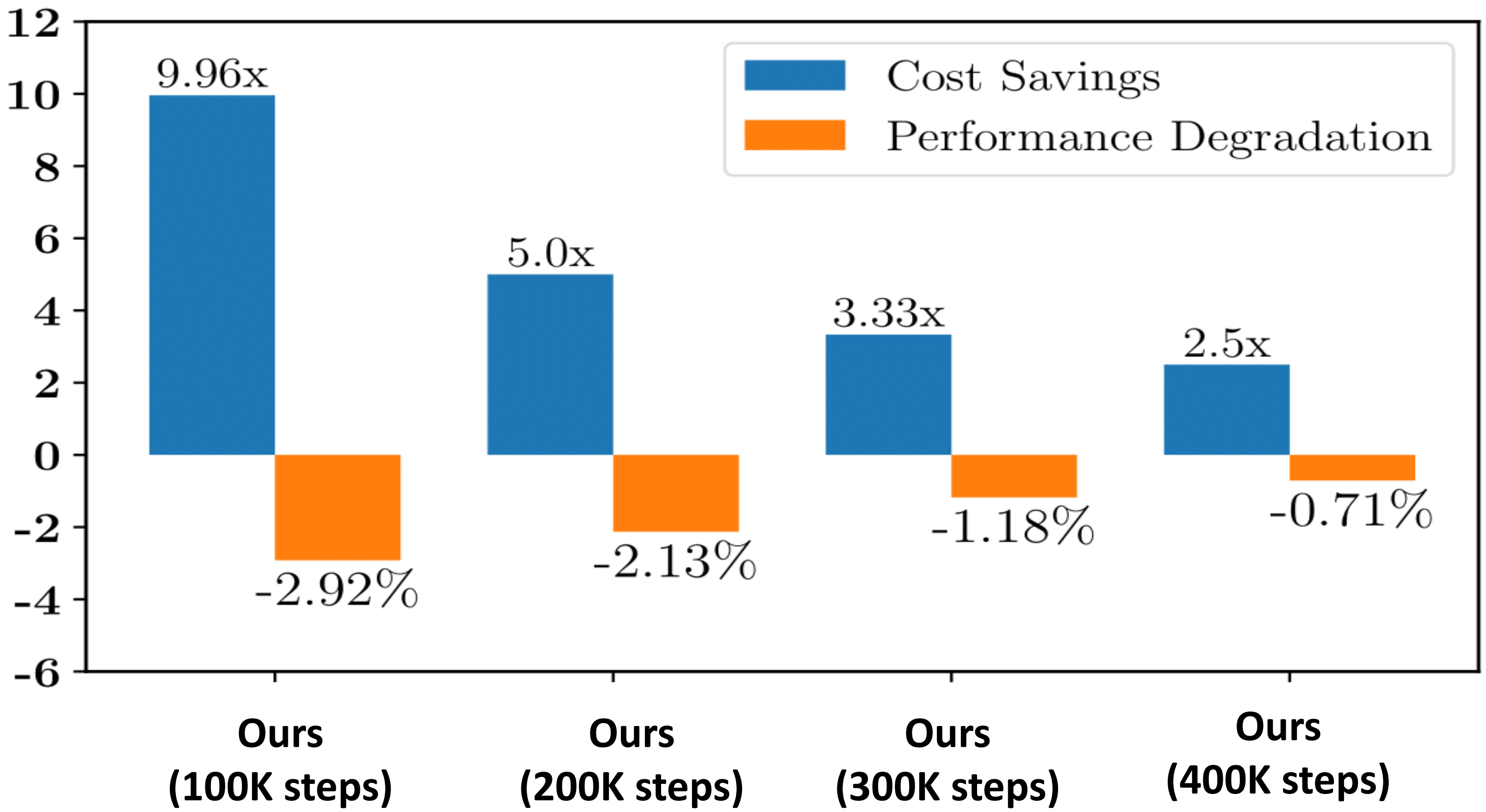
Vanilla BERT(Early Stopping): BERT model trained on the entire dataset for 250K steps

Subset size: 25%

Subset selected: Every 25000 steps

Number of partitions: 1500

BERT Training Efficiency



Performance Degradation:

Vanilla BERT Avg. Glue Score – BERT (OURS) Avg. Glue Score

Cost Savings:

Vanilla BERT Training Cost/BERT (OURS) Training Cost

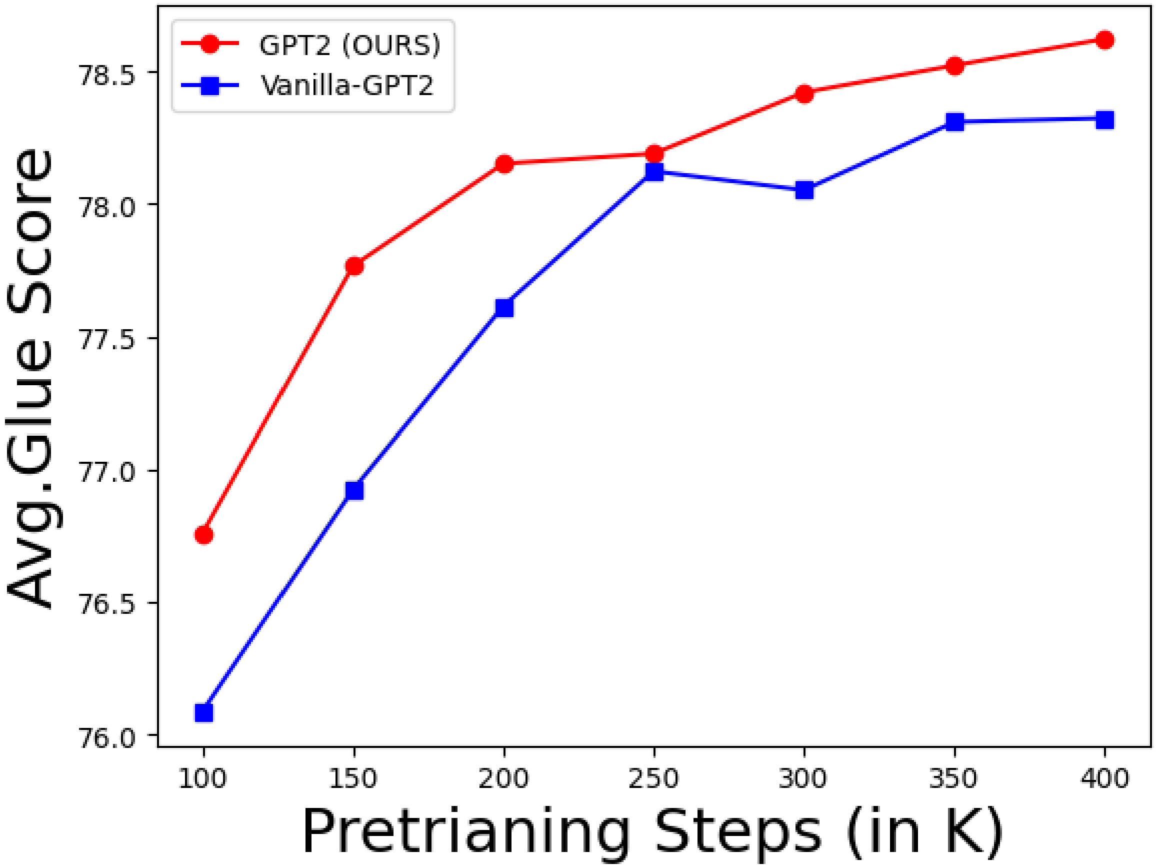
Knowledge Retention Analysis

Knowledge retention of different models as measured by LAMA probe. We report P@1 scores for all the four different subtasks in LAMA.

Method	Google-RE	T-REx	ConceptNet	SQuaD
Vanilla-BERT (1M steps)	3.99	25.76	11.48	14.77
BERT – Early Stopping (250K steps)	2.23	24.28	9.51	12.41
Random Online (250K steps)	3.1	23	7.87	11.37
Loss-Based Sampling (250K steps)	2.08	21.72	10.82	12.15
OURS (BERT + FL) (250K steps)	3.39	24.08	11.15	13.71

BERT model trained on facility location subsets retains knowledge better compared to baselines.

GPT2 Results



GPT2 model trained on facility location subsets achieved faster convergence than GPT2 model trained on the entire dataset.

Questions

Thank You!