More Neural Language Models
Logistics

• **Office hours**: Wed 12.30 – 1.30 pm (over zoom, note the channel)
• **A2**: due **Mar 10, 2023** – *errata recap*.
• **A2 tutorials planned schedule**:
  • Feb 17: A2 tutorial – 1
  • Mar 3: A2 tutorial – 2 (ft. Frank Niu)
  • Mar 10: A2 – Q/A and OH (*submission due at mid-night*)
• **A3**: release Mar 11, 2023
• **Final exam**: date to be finalized soon

• **Lecture feedback**:
  • Anonymous
  • Please share any thoughts/suggestions

• **Questions?**
More Neural Language Models

Lecture plan for today (L7 – 1/1)

• Emergent NLM architectures:
  • Encoder only (BERT, BERTology findings)
  • Encoder-Decoder: unified text-to-text format (T5)
  • Decoder only auto-regressive models (GPT):
    • covered in detail at a later lecture (L13)
  • Token-free models:
    • Importance, and the whys
    • Selective example: CANINE

• Trends in Neural Language Models
  • Scaling laws of NLMs
  • NLMs as foundation models & implications
### BERT: Bidirectional Encoder Representations from Transformers

<table>
<thead>
<tr>
<th>Rank</th>
<th>Name</th>
<th>Model</th>
<th>URL</th>
<th>Score</th>
<th>CoLA</th>
<th>SST-2</th>
<th>MRPC</th>
<th>STS-B</th>
<th>QQP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T5 Team - Google</td>
<td>T5</td>
<td></td>
<td>89.7</td>
<td>70.8</td>
<td>97.1</td>
<td>91.9/89.2</td>
<td>92.5/92.1</td>
<td>74.6/90.4</td>
</tr>
<tr>
<td>2</td>
<td>ALBERT-Team Google Language</td>
<td>ALBERT (Ensemble)</td>
<td></td>
<td>89.4</td>
<td>69.1</td>
<td>97.1</td>
<td>93.4/91.2</td>
<td>92.5/92.0</td>
<td>74.2/90.5</td>
</tr>
<tr>
<td>3</td>
<td>王玮</td>
<td>ALICE v2 large ensemble (Alibaba DAMO NLP)</td>
<td></td>
<td>89.0</td>
<td>69.2</td>
<td>97.1</td>
<td>93.6/91.5</td>
<td>92.7/92.3</td>
<td>74.4/90.7</td>
</tr>
<tr>
<td>4</td>
<td>Microsoft D365 AI &amp; UMD</td>
<td>FreeLB-RoBERTa (ensemble)</td>
<td></td>
<td>88.8</td>
<td>68.0</td>
<td>96.8</td>
<td>93.1/90.8</td>
<td>92.4/92.2</td>
<td>74.8/90.3</td>
</tr>
<tr>
<td>5</td>
<td>Facebook AI</td>
<td>RoBERTa</td>
<td></td>
<td>88.5</td>
<td>67.8</td>
<td>96.7</td>
<td>92.3/89.8</td>
<td>92.2/91.9</td>
<td>74.3/90.2</td>
</tr>
<tr>
<td>6</td>
<td>XLNet Team</td>
<td>XLNet-Large (ensemble)</td>
<td></td>
<td>88.4</td>
<td>67.8</td>
<td>96.8</td>
<td>93.0/90.7</td>
<td>91.6/91.1</td>
<td>74.2/90.3</td>
</tr>
<tr>
<td>7</td>
<td>Microsoft D365 AI &amp; MSR AI</td>
<td>MT-DNN-ensemble</td>
<td></td>
<td>87.6</td>
<td>68.4</td>
<td>96.5</td>
<td>92.7/90.3</td>
<td>91.1/90.7</td>
<td>73.7/89.9</td>
</tr>
<tr>
<td>8</td>
<td>GLUE Human Baselines</td>
<td>GLUE Human Baselines</td>
<td></td>
<td>87.1</td>
<td>66.4</td>
<td>97.8</td>
<td>86.3/80.8</td>
<td>92.7/92.6</td>
<td>59.5/80.4</td>
</tr>
<tr>
<td>9</td>
<td>Stanford Hazy Research</td>
<td>Snorkel MetaT</td>
<td></td>
<td>83.2</td>
<td>63.8</td>
<td>96.2</td>
<td>91.5/88.5</td>
<td>90.1/89.7</td>
<td>73.1/89.9</td>
</tr>
<tr>
<td>10</td>
<td>XLM Systems</td>
<td>XLM (English only)</td>
<td></td>
<td>83.1</td>
<td>62.9</td>
<td>95.6</td>
<td>90.7/87.1</td>
<td>88.8/88.2</td>
<td>73.2/89.8</td>
</tr>
</tbody>
</table>

• The age of humans is over?
Think of the **encoder** part of the transformer architecture

- Landmark, pivotal neural LM that has become an ubiquitous baseline in NLP.
- BERT is conceptually simple (multi-layer, bidirectional transformer), empirically powerful.
- Unlike predecessors (ELMo) or contemporaneous LMs (GPT), BERT is deeply bidirectional and independent of task-specific features with unified architecture across different tasks.

---


Code and models: [https://github.com/google-research/bert](https://github.com/google-research/bert) [Colab](https://colab.research.google.com/)
BERT: Bidirectional Encoder Representations from Transformers

• First, **pre-trained** on (large) unlabeled data on two unsupervised tasks/objectives:
  • Masked LM (**MLM**), and
  • Next Sentence Prediction (**NSP**)

• Then, **fine-tuned** using labeled data from downstream tasks

• Training entails feeding the final hidden vectors to an output FFN layer with softmax over the possibilities (e.g. the vocabulary as in a standard LM)


Code and models: [https://github.com/google-research/bert](https://github.com/google-research/bert) [Colab]
**Pre-training objectives**

- **Masked LM (MLM):** predict randomly masked words:
  
  
  **Input:** The man went to the [MASK]_{1}. He bought a [MASK]_{2} of milk.
  **Labels:** [MASK]_{1} = store; [MASK]_{2} = gallon

- 80% of the target words are masked with: [MASK]. 10% are replaced with another word, and 10% are kept as-is, to bias ‘towards the observation’.

- **Variants:** masking granularity can be varied (word-piece, word, span) with respective quirks. E.g., masking named entities improves structured knowledge representation.

- **Next sentence prediction (NSP):** does sentence B follow A?
  
  Sentence A = The man went to the store.
  Sentence B = He bought a gallon of milk.
  Label = IsNextSentence

  Sentence A = The man went to the store.
  Sentence B = Penguins are flightless.
  Label = NotNextSentence

- 50% of the time true, 50% of the time it’s a random sentence.
- Later research finds removing the NSP task does not hurt, or slightly improves performance. [2]

BERT: Bidirectional Encoder Representations from Transformers

Findings from ablative studies [1,2,3]

- **Heads**: Analysis of the multi-headed attention mechanism in BERT shows attention heads exhibiting attentions on various linguistic (e.g. syntax, coreference) patterns. [1]

  ![BERT Attention Heads](image)

- **Layers**: Linear word order and surface features captured most by lower layers. Syntactic information most prominent in middle layers. Semantic and task specific features are best captured in higher/final layers.

- **Research**: Research on proposed improvements and modifications to BERT, both architectural choices (e.g. # of layers, heads) and training methods is voluminous and ongoing. Due to overall trend towards larger model sizes, systematic ablations have become prohibitively expensive.

**Limitations:** BERT’s possession of impressive syntactic, semantic, and world knowledge has caveats.

**World Knowledge:**
- BERT struggles with pragmatic inference, and role-based event knowledge.
- It can ‘guess’ object affordances and properties, but cannot reason about relationships between them. Example: it ‘knows’ people can walk into houses, houses are big, but cannot infer that houses are bigger than people.

**Semantic Knowledge:**
- Struggles with representations of numbers.
- Surprisingly brittle to named entity replacements: e.g. 85% drop in performance in coreference task with names replaced.

**Syntactic Knowledge:**
- Does not ‘understand’ negations and is insensitive to malformed input.
- Findings suggest that either its syntactic knowledge is incomplete, or not dependent on it for solving its tasks.
Aside – BERT → BART → NMT

- Explosion of variants to BERT
- Pretrained BERT language model used to re-score/fine-tune downstream NLP tasks
- BART (Lewis et al., 2020) adds the decoder back to BERT, keeping the BERT objective
- Add some source language layers on top to train for NMT

T5: Text-to-Text Transfer Transformer

A refined Transformer updated with better methodologies

- T5 is an unified framework that casts all NLP problems into a ‘text-to-text’ format.
- Architecturally (almost) identical to the original Transformer (Vaswani et al., 2017).
- Draws from a systematic study comparing pre-training objectives, architectures, unlabeled data sets, transfer approaches, and other factors on dozens of language understanding tasks.
- Introduces and uses a new curated dataset: “Colossal Clean Crawled Corpus” (C4) for training.

Distinguishing features:

- Consistent, task-invariant MLE training objective.
- Self-attention “mask” with prefix.
- Unsupervised “denoising” training objectives: span corruption (conceptually same to MLM, mask ‘spans’ instead of words).

T5: Text-to-Text Transfer Transformer

Example Task: English to German (En-De) translation:

Input sentence: “That is good.”
Target: “Das ist gut.”

- **Training**: task specification is imbued by prepending task prefix to the input sequence. Model trained on next sequence prediction over the concatenated input sequence:
  
  “translate En-De: That is good. Das ist gut.”

- For prediction, the model is fed prefix:
  - “translate En-De: That is good. target:”

- For classification tasks, the model predicts a single word corresponding to the target label.

- E.g. MNLI task of entailment prediction:
  - “mnli premise: I hate pigeons. hypothesis: I am hostile to pigeons. entailment.”

- Model predicts label: {“entailment”, “neutral”, “contradiction”}.
The Open AI GPT papers

- The GPT papers:
  - GPT (2018)
  - GPT2 (2019)
  - GPT3 (2020)

- Each builds on the predecessor
- Auto-regressive, unidirectional \textit{(left to right)} architecture
- Detailed discussion in \textbf{lecture 13: LLMs}
GPT: model & architecture

- Architecture evolution: GPT3 ← GPT2 + mods ← GPT + mods
- Core architecture follows classic ‘language modeling’:
  \[
p(x) = \prod_{i=1}^{n} p(s_n|s_1, \ldots, s_{n-1})
\]
- Learning to perform a task as estimating distribution \( P(output \mid input) \)
- Original GPT\(^1\) trains a standard LM objective to maximize the likelihood:
  \[
  L(\mu) = \sum_{i} \log P(u_i|u_{i-k}, \ldots, u_{i-1}; \Theta)
  \]
    - Given an unsupervised corpus of tokens \( \mu = \{\mu_1, \ldots, \mu_n\} \), where \( k \) is context window, \( P \) is modelled using a neural network with parameters \( \theta \)
- GPT uses a multi-layer Transformer \textit{decoder} for the language model

Key architectural differences

- **GPT vs. BERT-variants:**
  - GPT uses ‘transformer’ blocks as *decoders*, and BERT as *encoders*.
  - Underlying (block level) ideology is same
  - GPT (later Transformer XL, XLNet) is an *autoregressive* model, BERT is not
    - At the cost of auto-regression, BERT has bi-directional context awareness.
  - GPT, like traditional LMs, outputs (predicts) one token at a time.

- Compare with T5, BART that uses encoder-decoder

Token free models

- Unlike the ubiquitous pre-trained LMs that operate on sequences of tokens corresponding to word or sub-word units, **token free models**:
  + Operate on raw text (bytes or characters) **directly**.
  + Removes necessity for (error-prone, complex) text preprocessing pipelines.
  - Con: raw sequences significantly **longer than token sequences**, increases computational complexity. (Reminder: ‘attention’ costs are quadratic to the length of input sequence)

- **Pitfalls** of explicit (word, sub-word) tokenization:
  - Need for large language dependent (fixed) **vocabulary** mapping **matrices**.
  - Applies **hand-engineered**, costly, language-specific string tokenization/segmentation algorithms (e.g. BPE, word-piece, sentence-piece) requiring linguistic expertise.
  - **Heuristic string-splitting**, however nuanced, cannot capture full breadth of linguistic phenomena, (e.g. morphologically distant agglutinative, non-concatenative languages). Other examples include languages without white-space (Thai, Chinese), or that uses punctuation as letters (Hawaiian, Twi). **Fine-tuning** tokenization needs to match **pretraining** tokenization methods.
  - **Brittle** to noise, corruption of input (typos, adversarial manipulations). Corrupted tokens lose vocabulary coverage.

2. Xue et al. "**ByT5**: Towards a token-free future with pre-trained byte-to-byte models." (2022). [link](#)
CANINE: Character Architecture with No tokenization In Neural Encoders.

- CANINE is a large language encoder with a deep transformer stack at its core.
- Inputs to the model are sequences of Unicode characters. 143,698 Unicode codepoints assigned to characters covers 154 scripts and over 900 languages!
- To avoid slowdown from increasing sequence length, it uses stride convolutions to down-sample input sequences to a shorter length, before the deep transformer stack to encode context.

- Three primary components:
  - Vocab free embedding technique;
  - Character-level model (CLM) with efficiency measures (up/down sampling of sequences); and
  - Perform unsupervised masked LM (MLM) pretraining on the CLM using variants:
    - Autoregressive character prediction
    - Subword prediction

Aside: Token free models - CANINE

The overall functional composition form uses [UP|DOWN]-sampling, and primary encoder:

\[ Y_{seq} \leftarrow \text{UP(ENCODE(DOWN}(e)) \quad \text{where} \quad e \in \mathbb{R}^{n \times d} \quad \text{is an input characters sequence, and} \quad Y_{seq} \in \mathbb{R}^{n \times d} \quad \text{is output of sequence predictions} \]

- **Down-sampling:**
  \[ h_{init} \leftarrow \text{LOCALTRANSFORMER}(e); \quad h_{down} \leftarrow \text{STRIDEDCONV}(h_{init}, r) \]
  \[ \text{where} \quad h_{down} \in \mathbb{R}^{m \times d} \quad \text{and} \quad m = \frac{n}{r} \quad \text{is the number of downsampling positions} \]

- **Up-sampling:** prediction require model’s output layer sequence length to match input’s length
  \[ h_{up} \leftarrow \text{CONV}(h_{init} \oplus h'_{down}, w); \quad y_{seq} \leftarrow \text{TRANSFORMER}(h_{up}) \]
  \[ \text{where} \quad \oplus \quad \text{is vector concatenation, CONV projects} \quad \mathbb{R}^{n \times 2d} \quad \text{back to} \quad \mathbb{R}^{n \times d} \quad \text{across a window of} \quad w \quad \text{characters. Applying a final transformer layer yields a final sequence representation:} \quad Y_{seq} \in \mathbb{R}^{n \times d} \]
NLM TRENDS & IMPLICATIONS
NLM: the bigger is better trend
NLM: the bigger is better trend

• Cons:
  • Deep learning == Deep pockets? Democratisation of compute power
  • Social impact e.g. (environmental): “training BERT on GPU is roughly equivalent to a trans-American flight”\(^1\)

\(^1\) S. Emma, A. Ganesh, and A. McCallum. "Energy and policy considerations for deep learning in NLP. (2019)" [arxiv]
Scaling laws for NLMs

- Kaplan et al. (2020) does a systematic review of scaling laws for NLMs [1]

Three scale factors:
- Compute: the amount of compute $C$ used for training
- Dataset size: the size of the dataset $D$
- Model parameters: the number of model parameters $N$, excluding embeddings

Language modelling performance (decreasing test loss is better), as the factors are scaled up

Scaling laws for NLMs

Key Findings: Performance of (Transformer based) NLMs:

• Has power-law relationship with the three scale factors: C, D, N (excluding embeddings).

• Depends most strongly on these scale factors; architectural hyperparameters (like depth, width) does not have much effect.

• Improves smoothly when the factors (N, D) are scaled up in tandem. Diminishing returns if either N or D bottlenecks the other. Roughly, an 8x model size increase should match 5x data size increase to avoid performance penalty.

• **Transfer learning**: out-of-distribution generalization depends almost exclusively on the in-distribution (train set) validation loss performance that improves with the scaling factors.

• **Sample efficiency**: Large models are more sample-efficient than small models, reaching the same level of performance with fewer optimization steps, data points.

LLMs as Foundation Models

• **Homogenization**: (almost) all SOTA NLP LLM models are now adapted from one of a few foundation models (like BERT, BART, T5, etc.). [1]

![Diagram showing the process of data from various modalities being adapted to a foundation model and used for various tasks](image)

- Data from various modalities
- Adoption to a wide range of downstream tasks

• **Social Impact**
  - Exacerbation of social inequalities.
  - Democratization: increased computation demands – power/capability concentrated to few corporations/start-ups.
  - Gap between industry models and community models are large.
  - Increasing proprietary moat and closed source nature.
  - Solution: *government intervention?*