CS 263: Advanced NLP — Midterm Review Sheet

1. Word Representations

- **Distributional Hypothesis** (Harris, 1954): Words in similar contexts have similar meanings.
- **Static Word embeddings** (word2vec, GloVe): Dense vectors pre-trained on co-occurrence stats. *Problem*: context-free — "bank" gets same vector regardless of meaning.
- **Contextualized representations**: Transformer models produce embeddings that change with context, handling **polysemy** (e.g., "bank" in finance vs. river).
- **Byte Pair Encoding**: Learn subword vocabulary; split rare words into components. Common words stay whole; rare words decomposed. Solves OOV (out-of-vocabulary) / UNK problem of word-level models.

2. Attention Mechanism

(Bahdanau et al., 2015)

- **Dot product:** $a(q, k) = \mathbf{q}^T\mathbf{k}$
- **Scaled dot product:** $a(q, k) = \frac{\mathbf{q}^T\mathbf{k}}{\sqrt{|\mathbf{k}|}}$ (stabilizes gradients)
- **Benefits:** Better use of input; helps address long-range dependencies and vanishing gradient (shortcut to faraway states); provides some degree of interpretability/alignment.

3. Transformers

(Vaswani et al., 2017)

- **Key insight:** No recurrence – relies (almost) entirely on attention; easier to **parallelize**. You should familiarize yourself with the self-attention computation from the Illustrated Transformer blog (`https://jalammar.github.io/illustrated-transformer/`).
- **Positional Encoding:** Since no recurrence, position info added via sinusoidal functions.

4. Transformer Components

- **Residual Connections**: LayerNorm(x+Sublayer(x)) — prevents information loss through layers.
- **Layer Normalization**: $\text{LayerNorm}(x) = \gamma \odot \frac{x-\mu}{\sqrt{\sigma^2+\epsilon}} + \beta$, helps to stabilize training
- **GPT block components**: Masked Multi-Head Self-Attention, LayerNorm + Residual, Feed Forward

5. LLM Pre-training

- Labeled data is scarce; unlabeled text is abundant (web-scale).
- **Self-supervised objectives** (masking, next-token prediction) require no manual labels.
- **Chinchilla Scaling Law** (Hoffmann et al., 2022): For compute-optimal training, *scale model size and training data proportionately.*

6. Pre-training Architectures

1. Unidirectional / Decoder-Only (GPT)

- Left-to-right; **causal/masked self-attention** (can only attend to past tokens). Suited for **generation**.

2. Bidirectional / Encoder-Only (BERT)

- Attends to **both left and right** context. Suited for **classification/understanding**.

3. Encoder-Decoder (T5)

- Encoder processes corrupted input; decoder generates missing spans. Combines benefits of both.

7. In-Context Learning & Prompting

- **Zero-shot**: Task description only; no examples. No gradient updates.
- **Few-shot**: Task description + few examples in context. No gradient updates. Performance improves more rapidly with model size.

8. LLM Post-training

- **Supervised Finetuning (SFT)** is used *before* RLHF/DPO to teach the model expected output *format* via examples. Learning format from reward signals alone is inefficient.
- **RLHF / DPO** then refines outputs for desired qualities (helpfulness, harmlessness).
- **PEFT**: Freeze most parameters; update only a small subset for each downstream task.

9. Basic Decoding Strategies

1. Greedy Decoding

- Select highest-probability token at each step: $y_j = \arg\max\ P(y_j|X, y_1, \ldots, y_{j-1})$
- **Pros:** Fast, deterministic
- **Cons:** Ignores long-tail; generic/repetitive outputs; locally optimal $\neq$ globally optimal

2. Ancestral Sampling

- Sample randomly: $y_j \sim P(y_j|X, y_1, \ldots, y_{j-1})$
- Can produce very improbable/incoherent sequences

3. Top-k Sampling

- Restrict to **top** k most probable tokens, renormalize, then sample
- Fixed k can be too narrow (peaked dist.) or too broad (flat dist.)

4. Nucleus / Top-p Sampling

- Sample from smallest token set whose cumulative probability $\geq p$
- **Adaptive:** includes fewer tokens when distribution is peaked, more when flat
- Addresses top-k's fixed-size limitation

5. Temperature Scaling

$$p_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}, \quad T > 0$$

- **Low T (e.g., 0.2):** Peaky distribution $\rightarrow$ more deterministic, conservative
- $T = 1$**:** Original distribution (no change)
- **High T (e.g., 1.5+):** Flatter distribution $\rightarrow$ more random/creative
- $T \rightarrow 0$: equivalent to greedy; $T \rightarrow \infty$: uniform distribution
- Can combine with top-k/top-p

10. Beam Search

- **Idea:** Explore multiple partial sequences (breadth-first) before committing

- **Algorithm:** (1) Set beam width k; (2) Start with top-k tokens; (3) Expand each beam by top-k continuations (k^2 candidates); (4) Prune to top-k by cumulative log-prob; (5) Repeat until EOS
- **Length normalization:** Divide log-prob by sequence length (avoids bias toward short sequences)
- **Pros:** Better coherence/overall quality than greedy
- **Cons:** Computationally expensive; outputs can be very similar; large beam width can *degrade* quality (Cohen & Beck, 2019)

Diverse Beam Search

- Split beams into **groups**; group 1 expands normally; subsequent groups **penalize similarity** to prior groups

Stochastic Beam Search (Kool et al., 2019)

- Add **Gumbel noise** to logits: $X_i = z_i + G_i, \quad G_i \sim \text{Gumbel}(0, 1)$
- Select top-k of perturbed values $\Rightarrow$ sampling without replacement
- $P(X_i = \max) = \frac{\exp(\mu_i)}{\sum_j \exp(\mu_j)}$ (softmax!)

11. Test-Time Scaling

- **Test-time scaling:** Improve performance via extra inference compute (more tokens) rather than retraining
- **Chain-of-Thought (Wei et al., 2022):** Few-shot examples with intermediate reasoning steps $\Rightarrow$ large gains
- **Zero-Shot CoT (Kojima et al., 2022):** Simply append *"Let's think step by step"* – no task-specific examples needed

12. Classification Metrics

- **Accuracy** $= \frac{TP+TN}{TP+TN+FP+FN}$
- **Precision** $= \frac{TP}{TP+FP}$ (of predicted positives, how many correct?)
- **Recall** $= \frac{TP}{TP+FN}$ (of actual positives, how many found?)
- **F1** $= 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$ (harmonic mean)

13. Open-Ended Generation Metrics

N-gram Based (Reference-Based)

- **BLEU, ROUGE, METEOR**: Measure *lexical similarity* (n-gram overlap) between generated and gold-standard text. Fast but lack semantic understanding.

Model-Based (Reference-Based)

E.g.

- **BERTScore** (Zhang et al., 2020)
- **BLEURT** (Sellam et al., 2020)

Crowdsourced references may be *uncorrelated* with faithfulness; **expert references** yield better correlation with actual quality.

LLM-as-a-Judge (Reference-Free)

- Use a strong LLM to score/compare outputs. No human reference needed. Increasingly popular.
- **Self-bias**: LLMs prefer their own outputs. Self-preference correlates linearly with self-recognition capability (Panickssery et al., 2024).

14. Human Evaluation (Gold Standard)

- Most important form of evaluation for text generation; used to validate automatic metrics.
- **Dimensions**: fluency, coherence/consistency, factuality, commonsense, style/formality, grammaticality, redundancy.
- Used in both *model training* (e.g., Anthropic RLHF with crowdworkers) and *model evaluation* (e.g., BIG-bench).

Inter-Annotator Agreement (IAA)

- **Cohen's Kappa** (2 annotators): $\kappa = \frac{Pr(a)-Pr(e)}{1-Pr(e)}$
  $Pr(a)$ = observed agreement; $Pr(e)$ = expected (chance) agreement. Computed from marginal label distributions.
- **Fleiss' Kappa**: Generalizes Cohen's κ for $n > 2$ annotators.
- **Krippendorff's Alpha**: Most robust; handles missing data.
- Interpretation (Landis & Koch): 0–0.2 slight, 0.2–0.4 fair, 0.4–0.6 moderate, 0.6–0.8 substantial, 0.8–1.0 perfect.
- Low IAA can signal annotator error, bad protocol, *or* genuine task subjectivity.

Quality Issues in Human Eval

- **Order bias**: Randomize question/example positions.
- **Inattentive annotators**: Use attention checks.
- **Spamming**: Time checks, consistency checks, free-text quality.
- **Reproducibility**: Human eval is *not* reliably reproducible; experts vs. crowdworkers give very different scores. Report minimum details (participants, preparation, task construction, annotations, IAA).

15. Benchmarking Ecosystem

- **Static vs Dynamic Benchmarking**
- **Monoculture problem**: Most papers evaluate only on English + accuracy; neglect multilinguality, fairness, efficiency, interpretability.
- **Prevention of Test Set Contamination**: Models may be *trained on test data*. Hard to verify for closed models. Some workarounds are private test sets (limit access) and dynamic test sets (e.g. DynaBench — continuously create new adversarial examples).