**Presentation Speaker Notes: BERT Fact Injection Project**

**Slide 1: Title Slide – "Teaching BERT New Facts"**

**Speaker Notes:**  
Hi everyone, I'm [Name], and this is our final project for CS 5322. We explored whether a pretrained BERT model can be taught new factual knowledge after training — and how that impacts what it already knows. We ran controlled experiments using fine-tuning and measured things like forgetting, learning accuracy, and hallucination.

**Slide 2: Project Goal**

**Speaker Notes:**  
Large Language Models like BERT are trained on massive static corpora and aren’t naturally designed to update knowledge post-training. But in real-world scenarios, facts change — new presidents, sports wins, scientific updates — and we’d like models to adapt without retraining from scratch.

Our core question: Can BERT be taught new facts — and what’s the cost?  
This is interesting because if it works, it could lead to faster updates, better misinformation handling, and more adaptive AI systems.  
But it’s hard because BERT is prone to forgetting old knowledge, hallucinating incorrect answers, and masked language modeling doesn’t easily support clean fact insertion.

**Slide 3: Our Approach**

**Speaker Notes:**  
We selected three target facts:

* The Eagles won the Super Bowl in 2025
* The Chiefs won in 2024
* Jay Hartzell became president of SMU

These varied in difficulty — some were similar to known facts, others had rare tokens. For each fact, we wrote over 50 templated factual sentences, along with test prompts. We also included unrelated trivia and known facts to measure hallucination and forgetting.

We fine-tuned BERT with different training sizes and epochs, then evaluated performance changes.

**Slide 4: Dataset & Injection Design**

**Speaker Notes:**  
Each fact group had 50+ injection sentences, using varied wording and sentence lengths. For example: “The Eagles claimed Super Bowl LIX in 2025.”

We also built evaluation prompts like: “The [MASK] won the Super Bowl in 2025.”  
We used:

* **Fact prompts** to check if the model learned
* **Known prompts** to test for forgetting
* **Random control prompts** to monitor hallucination

We paid attention to rare token issues — especially for “Hartzell,” which didn’t appear in BERT’s pretraining.

**Slide 5: Training Setup**

**Speaker Notes:**  
We used bert-base-uncased with HuggingFace’s Trainer and MLM objective.

Our training loop covered 3 sentence sizes: 5, 10, 50 and 3 epoch counts: 1, 3, 5.  
We tested with and without adding known facts.

Each config got a fresh model. We tracked changes in accuracy, confidence, forgetting, and hallucination.  
All outputs were saved in a pandas DataFrame and used for downstream analysis and plotting.

**Slide 6: Evaluation Metrics**

**Speaker Notes:**  
Our evaluation used several key metrics:

* **Top-1 accuracy**: was the correct answer ranked first?
* **Top-5 accuracy**: did it appear at all?
* **Confidence**: how sure was the model?
* **Δ Accuracy**: change from the base model
* **Forgetting**: drop in accuracy on known facts
* **Hallucination**: accuracy on random unrelated prompts

We also analyzed most frequent incorrect answers, like . or something, to understand fallback behavior.

**Slide 7: Δ Accuracy & Learning Results**

**Speaker Notes:**  
Here’s our ΔTop1 accuracy heatmap. We saw the best learning with 50 sentences over 3–5 epochs.

The *Eagles* and *Chiefs* facts were learned effectively — ΔTop1 increased by over 10% in some configs.

But Jay Hartzell failed — BERT never predicted the correct name. This reflects issues with multi-token names and token rarity. Hartzell gets split into subwords BERT likely never saw during pretraining.

**Slide 8: Forgetting & Hallucination**

**Speaker Notes:**  
This bar chart shows forgetting — i.e., drop in accuracy on known facts. Most configurations retained knowledge well, but large fine-tuning runs did lead to up to 20% degradation.

On the hallucination side, we tested 15 unrelated trivia prompts. Some models predicted random but confident answers — especially when trained on small data for many epochs. However, hallucination dropped significantly in higher-data settings.

**Slide 9: Prediction Trends**

**Speaker Notes:**  
We also looked at frequent incorrect answers. BERT often filled [MASK] with:

* .
* something
* him, he, or high-frequency names

This fallback behavior is typical when BERT lacks strong priors — it guesses based on language patterns, not fact grounding.

This aligns with results in the TruthfulQA benchmark, where LLMs confidently generate plausible but incorrect completions.

**Slide 10: Key Findings**

**Speaker Notes:**  
In summary:

* BERT **can** learn new facts through small-scale fine-tuning — especially single-token facts.
* Learning is sensitive to training size and duration.
* Forgetting was modest but visible in high-exposure configs.
* Hallucination remained controlled when enough training data was used.
* Multi-token answers like “Jay Hartzell” are currently impractical with vanilla BERT.

**Slide 11: Future Work**

**Speaker Notes:**  
To improve, we want to:

* Use **span masking** or **T5** for multi-token predictions
* Try **adapter layers** or **LoRA** to avoid overwriting core weights
* Explore **prompt tuning** or **contrastive objectives**
* Use external evaluation sets like **TruthfulQA** or adversarial quizzes
* Simulate **sequential updates** to test retention over time

These ideas build on the “From Static to Dynamic” paper and Retrieval-Augmented Generation approaches.

**🧠 Du et al., From Static to Dynamic**

**Goal:** Proposes a **continual learning framework** for updating large language models (LLMs) without forgetting old knowledge.

**Key Ideas:**

* Introduces strategies to **inject new knowledge** over time while preserving prior capabilities.
* Combines **parameter-efficient tuning** (like adapters) with **selective rehearsal** of old data.
* Frames LLMs as **dynamic learners**, not frozen snapshots.

**Use in your project:** This inspired your idea of updating BERT incrementally and trying to preserve old knowledge (i.e., prevent forgetting).

**🔍 Gao et al., Retrieval-Augmented Generation (RAG) Survey**

**Goal:** Surveys methods that combine **retrieval systems** with LLMs to ground answers in real-world data.

**Key Ideas:**

* Instead of storing all facts in the model weights, RAG fetches external documents at inference time.
* Covers models like RAG, REALM, and Fusion-in-Decoder.
* Highlights benefits for **factual accuracy**, scalability, and **reducing hallucination**.

**Use in your project:** Reinforces that fine-tuning isn't the only way to inject facts — retrieval could help avoid hallucination and forgetting.

**✅ Lin et al., TruthfulQA**

**Goal:** Introduces a benchmark to test if language models generate **factually correct** answers, especially under misleading or false prompts.

**Key Ideas:**

* Tests models on **questions designed to trigger falsehoods** (e.g., conspiracy theories).
* Measures how often models mimic common misconceptions.
* Finds that larger models can be **more confidently wrong**.

**Use in your project:** Validates your hallucination tests — shows that confident nonsense is a real issue in LLMs and needs to be measured.

**Slide 12: Q&A**

**Speaker Notes:**  
Thanks for listening — happy to take any questions!

**✅ Span Masking**

* **What it is:** Instead of masking just one word with [MASK], span masking hides **multiple tokens at once** (e.g., entire phrases).
* **Why it matters:** Helps with multi-word facts like "Jay Hartzell" that can’t be predicted in a single [MASK] slot.
* **Example:** T5 uses this to let the model fill in longer blanks like "The president of SMU is <extra\_id\_0>" → Jay Hartzell.

**✅ T5 (Text-to-Text Transfer Transformer)**

* **What it is:** A model that treats every task as **text-in, text-out** (not just masking).
* **Why it helps:** You can prompt it like a quiz — "Who won the Super Bowl in 2025?" — and it can output **multiple tokens**, like "The Eagles".
* **Compared to BERT:** T5 isn’t limited to one masked word; it’s better for answering questions or filling in phrases.

**✅ Adapters**

* **What it is:** Tiny neural layers inserted inside a model like BERT.
* **Why it's useful:** You can fine-tune **just the adapters** instead of all of BERT’s weights — this keeps the original knowledge safe.
* **Think of it like:** Plug-in memory — new facts go into the adapters, and you can swap them out later.

**✅ LoRA (Low-Rank Adaptation)**

* **What it is:** A method that fine-tunes just a few **efficient, low-rank matrices** inside the model.
* **Why it's useful:** Like adapters, it’s lightweight and doesn’t disturb the original model much.
* **Think of it like:** Teaching the model new tricks without making it forget old ones.

**✅ Prompt Tuning**

* **What it is:** Instead of changing the model, you learn a **custom prompt** (a string of tokens) that nudges the model to give the right answer.
* **Why it's useful:** You don’t touch the model weights at all — just optimize the input.
* **Example:** You might find the best way to ask BERT about 2025 is “In Super Bowl LIX, the winning team was [MASK]”.

**✅ Contrastive Learning**

* **What it is:** You train the model to tell apart **true facts from false ones**.
* **Why it's useful:** Helps the model distinguish between closely related but incorrect facts (e.g., Eagles 2025 vs Eagles 2018).
* **Think of it like:** Teaching by comparison — not just saying what’s right, but also showing what’s wrong.

**✅ TruthfulQA**

* **What it is:** A benchmark that tests if LLMs give **factually accurate** answers — not just plausible ones.
* **Why it’s helpful:** It catches models that confidently hallucinate wrong info.
* **How you’d use it:** After fine-tuning, you’d evaluate your model on TruthfulQA to check if it’s trustworthy.

Let me know if you want me to format this as a printable “answer key” or include it in your speaker notes document.