**Professional Notes on Pretraining Strategies Lecture**

**Overview**

This lecture delves into the evolution of **pretraining strategies in NLP**, focusing on the shift from static word embeddings to **contextual embeddings** using language models. The discussion bridges classical word embedding techniques (like Word2Vec and GloVe) with modern **contextualized language models** like **ELMo**, and sets the stage for the development of **Transformer-based models**.

**Motivation**

* **Quote Reference**: "We shall know a word by the company it keeps" emphasizes **distributional semantics**.
* Updated interpretation: "The complete meaning of a word is always contextual" — highlighting the **necessity of context** in semantic understanding.
* **Problem with Static Embeddings**:
  + E.g., The word *"record"* in "I will record the video" vs. "The record is broken".
  + GloVe and Word2Vec yield **same embedding** for both uses, failing to capture semantic nuance.
  + Similarly, "bat" in sports vs. animal domain.

**Contextualized Embeddings**

* Solution: Produce **context-sensitive word embeddings** that **dynamically change** depending on the sentence context.

**ELMo (Embeddings from Language Models)**

* **Introduced**: 2018, before widespread use of Transformers.
* **Not Transformer-based** — Uses **bidirectional RNNs (Bi-LSTMs)**.

**Architecture**

* Input: Tokenized sentence.
* Uses two **RNN layers**:
  1. **Bottom Layer**: Pretrained on language modeling objective (predict next token).
  2. **Top Layer**: Task-specific layer (e.g., sentiment analysis).

**Training Process**

1. **Language Model Pretraining** (Bottom Layer)
   * Uses RNNs (e.g., LSTM or GRU).
   * Trained on large corpus with a **language modeling objective**:
     + Predict next word (left-to-right) or previous word (right-to-left).
     + Weights are updated (e.g., matrix W\_h).
   * Hidden state of RNN is used as the **contextualized embedding**.
   * Embedding dictionary is updated dynamically during training.
2. **Freezing the Model**
   * After pretraining, bottom layer is **frozen**.
   * Embeddings now represent **context-aware representations** of words.
3. **Fine-Tuning (Top Layer)**
   * A second RNN is trained on **task-specific data**.
   * Task examples: Sentiment analysis, classification.
   * Output layer is updated during training, while the lower pretrained embeddings remain static.
4. **Inference**
   * Input goes through pretrained bottom RNN (produces contextual embeddings).
   * These are passed into the top task-specific RNN.
   * Final predictions are made (e.g., class labels).

**Key Characteristics of ELMo**

* **Dynamic embeddings**: Different for the same word in different contexts.
* **Deep contextualization**: Embeddings incorporate full sentence meaning.
* **Bidirectionality**: Uses both forward and backward RNNs to understand full sentence context.

**Alternatives to Bidirectional RNN**

* If only left-to-right RNN is available:
  + One can **reverse the input sentence** to simulate right-to-left behavior.

**Summary**

* **ELMo** introduced a powerful method to generate **contextual embeddings** before Transformers gained dominance.
* It solved the limitations of static word embeddings like Word2Vec and GloVe.
* This approach laid the **foundation** for Transformer-based models (e.g., BERT), which later refined and improved on ELMo's core principles.

**Next Steps**

* Further exploration will look into how the **pretraining paradigm shifted** from word-level to language model-level (contextual language understanding).
* Comparison with Transformer-based models like **BERT**, **GPT**, etc., which solve similar problems with different architectural choices.

**Contextual Word Embeddings, ELMo, and Pretraining Strategies in Transformers**

**1. Introduction to Contextual Embeddings (ELMo)**

* **ELMo (Embeddings from Language Models)** is a contextual word embedding technique.
* Unlike static embeddings like **Word2Vec** or **GloVe**, which assign a single vector to a word irrespective of its context, ELMo produces **dynamic vectors** depending on the word’s **context** within a sentence.
* ELMo uses a **bidirectional LSTM** trained on a language modeling objective.
* This allows the same word to have different embeddings in different contexts.

Example: "bank" in "river bank" vs. "money bank" will have different embeddings in ELMo but identical ones in Word2Vec.

**2. From Pretrained Word Vectors to Pretrained Models**

* Earlier approaches (Word2Vec, GloVe) pretrained only the **embedding layer**.
* The rest of the model (task-specific layers) had to be trained from scratch.
* The shift is now towards **pretraining the entire model**, including all layers (attention layers, feed-forward networks, etc.).

**3. Types of Transformer Pretraining Architectures**

**A. Decoder-Only Models**

* Only use the **decoder** component of the transformer.
* **Examples**: GPT, GPT-2, GPT-3
* These models are **auto-regressive**, predicting the next token based on previous ones.
* No **cross-attention**, only **masked self-attention**.

**B. Encoder-Only Models**

* Only use the **encoder** part of the transformer.
* **Examples**: BERT, RoBERTa
* Use **unmasked self-attention**, allowing full bidirectional context.

**C. Encoder-Decoder Models**

* Utilize both the encoder and decoder.
* **Examples**: T5, BART, mBART
* Typically used for sequence-to-sequence tasks (e.g., translation, summarization).

**4. BERT: Bidirectional Encoder Representations from Transformers**

**Architecture:**

* Only uses the **encoder** from the Transformer.
* Allows each token to **attend to all others** in the sequence.
* Enables **bidirectional context understanding**.

**Pretraining Objectives:**

BERT is trained on two **self-supervised** objectives using raw text:

**A. Masked Language Modeling (MLM)**

* Randomly mask **15%** of input tokens during training.
* Model is trained to **predict** these masked tokens.

**Masking Strategy**:

* Of the 15% masked tokens:
  + **80%** are replaced with [MASK] token.
  + **10%** are replaced with a random token.
  + **10%** are left unchanged, but still predicted.

**Why 15% and Not 50%?**

* Masking too many tokens (e.g., 50%) **destroys the context**.
* 15% is empirically found to balance learning while preserving sentence context.

**Loss Calculation**:

* Loss is computed **only at the masked token positions**.
* Non-masked tokens do not contribute to the training loss.

**B. Next Sentence Prediction (NSP)**

* Input: A pair of sentences (Sentence A, Sentence B).
* Task: Predict if Sentence B **naturally follows** Sentence A.

**Data Generation**:

* **Positive samples**: Consecutive sentences from the corpus.
* **Negative samples**: Randomly paired sentences not adjacent in the corpus.

**Why NSP?**

* Helps the model learn **sentence-level relationships**, useful for QA and natural inference tasks.

**5. Summary of Key Concepts**

| **Concept** | **Static Embedding** | **Contextual Embedding** |
| --- | --- | --- |
| Examples | Word2Vec, GloVe | ELMo, BERT |
| Word Representation | Same for all contexts | Varies with context |
| Model Trained | Only embeddings | Full model parameters |

| **Model Type** | **Encoder** | **Decoder** | **Encoder-Decoder** |
| --- | --- | --- | --- |
| Examples | BERT, RoBERTa | GPT, GPT-2 | T5, BART |

**6. Important Notes**

* **MLM and NSP** are jointly trained in BERT.
* BERT does not use autoregressive generation; it's designed for **understanding**, not **generation**.
* Pretraining enables **transfer learning**: pretrained models are **fine-tuned** on downstream tasks with smaller labeled datasets.

**Understanding BERT's Input Structure, Pretraining Tasks, and Downstream Fine-Tuning (QA)**

**1. BERT Pretraining Tasks Overview**

BERT (Bidirectional Encoder Representations from Transformers) is pretrained using two simultaneous tasks:

**a. Masked Language Modeling (MLM)**

* Certain tokens in the input sequence are replaced with a [MASK] token.
* The model is trained to predict the original value of these masked tokens.

**b. Next Sentence Prediction (NSP)**

* The model is fed two sequences:
  + **Label YES** if sentence B logically follows sentence A.
  + **Label NO** if sentence B is a random sentence.
* Helps BERT learn sentence-level relationships.

**2. Input Representation for Pretraining**

**a. Example Input:**

"I enjoyed reading the mask." and "The novel was great."

**b. Special Tokens:**

* **[CLS]**: Prepended to the input sequence for classification tasks.
* **[SEP]**: Separates sentence A and B.

**c. Token Embeddings:**

Each token is represented using the sum of:

1. Token Embeddings (WordPiece)
2. Segment Embeddings (A or B)
3. Positional Embeddings

**3. Why Use [CLS] Token?**

* The **[CLS]** token is a learned embedding that aggregates context from the entire sequence.
* Unlike regular tokens (e.g., "I", "reading"), [CLS] has **no inherent meaning** and is not biased by any particular token.
* It attends to all tokens and can be used as a global representative for classification.
* Reduces the number of learnable parameters (only one projection layer: 512 x 2 instead of using all tokens).
* Efficient and unbiased feature for classification.

**4. Loss Functions in Pretraining**

* **Loss 1 (MLM Loss)**: For predicting the masked tokens.
* **Loss 2 (NSP Loss)**: For predicting whether sentence B follows sentence A.
* Both losses are backpropagated jointly to update encoder weights.

**5. Training Details**

* **Corpus**:
  + Wikipedia: 2.5 Billion words
  + BookCorpus: 800 Million words
* **Model Variants**:
  + **BERT Base**: 12 encoder layers, 12 attention heads
  + **BERT Large**: 24 encoder layers, 16 attention heads

**6. Downstream Task: Question Answering (QA)**

**a. Input Setup**

* Concatenate **Question (Q)** and **Passage (P)** with [SEP] separator and a leading [CLS]:

[CLS] Q [SEP] P [SEP]

**b. Token Classification**

* Add **two classification layers** (heads) on top of every token:
  + One for predicting **Start** of the answer span
  + Another for **End** of the answer span

**c. Training Objective**

* For each token:
  + **Start Label (CB)** and **End Label (CE)** are one-hot encoded.
  + All other tokens: CB = 0, CE = 0
  + Correct start token: CB = 1
  + Correct end token: CE = 1
* Use cross-entropy loss for both classifiers and backpropagate.

**d. Inference Strategy**

* For every token, calculate:
  + **P(start)** = probability it's the start of the answer
  + **P(end)** = probability it's the end of the answer
* Find the span *(i, j)* such that:
  + j > i (valid span)
  + **P(i) \* P(j)** is maximized

**7. Adaptability and Fine-tuning**

* During fine-tuning, only the task-specific heads (and optionally the encoder layers) are updated.
* Pretrained BERT embeddings (contextualized) serve as a strong foundation.

**8. Conclusion**

BERT’s design with [CLS], [SEP], and MLM/NSP objectives allows it to capture deep bidirectional relationships. When adapted for tasks like QA, it shows superior performance by leveraging span prediction using learned contextual embeddings and classification heads.