**Pre-Training Strategies:**

Contexualized word embedding:

Helps tackel same word having diff meaning in diff context

ELMo- Embeddind from language model (non – transformer method)

* Language objective based rnn
* Relies on rnn(lstm or gru)
* Hidden state in rnn act as embedding for the corresponding word – but how possible only one rnn but many words
* Feed the embedding of input I, hidden state update, hidden state corresponding to input I is also updated. When training is done take the hidden states and replace prev embedding of I with this hidden state.
* Now another traning sentence comes use same rnn, we have word matrix from prev iteration and again update, when hidden stae update dictionary/matrix also updated.
* Keep doing this with all instance, we can also train batch wise, or update rnn per instance
* This first objective based rnn is freezed after the task, then after getting another task specific data to fine tune model, on top of this we apply another rnn(say sentimental analysiz), task specific input token embedding coming from frozen rnn , in second layer we have another weight specific matrix (corresponding to that layer), we hava the task produce some output then backprop and update the weights, task specific weights are updated.
* We have two layer first – language model layer, second – task specific layer
* In inference time, bottom layer Wh already stored.
* Take the embedding of the word from dict and mul with weight Wh, this and prev hidden state will be used to get next hidden state and so on, i.e we roll out
* Current hidden state of bottom layer is very diff from hidden state store earlier,

As these are the function weight matrix , This is contextual.

* Botom layer produce hidden state these moved to task specific layer. Tsl have corresponding weight learnt earlier and used to produce the task.
* Uni directional rnn not a great approach
* Along with Left to right rnn we also hve right to left rnn.
* In fact used a stacked rnn (language model rnn – bottom rnn)

Theta s = softmax parameter, only Wh changes

For every token k we have representation from forward layer and backward layer we can either concatenate them or instead of 4 outputs we take only from top layer above to layer we have U matrix and softmax , so when we do backprop top layer affected more than bottom layer,

We can also add a layer specific parameter – gamma, s- s- specific layer, gamma – specific task

On top of language model layer i.e apply task specific layer (language model layer is frozen) , instead of feasing completely add parameter dependent on the task. In inference time when feed specific instance/seq we fetch s and gamma. Now hidden state dependent on the parameter of the task, if need we can discard these parameter and use only hidden state or use this.

ELMo’s token representation:

Elmo use token level representation, token is character level token not word level,

2048 character n-gram convolutional filters with two highway layers, followed by a linear projection to 512 dimention

Character level encoding level overcome UNK token problem.

ELMo first large scale model.

ELMO can be used to initiate transformer

ElMO is another word embedding technique – i.e contextual , dynamic

Pre-Trained Word Vectors:

In pretrain word vector embedding layer is frozen and task specific layer is our intrest.

Model is not pre trained only input vectors are pretrained

Pre-Trained Model:

All parameters are pretrained (like one correspondint to attention,feed forward) and for specific task fine tuned

Types of pretraining:

3 kinds depending on which part of transformer(encoder part, decoder part)

* **Decoder only model: GPT mode;**

What pre traing considers only decoder part of transformer

All gpt are decoder based model – wont consider encoder part –

so cross attention not there

* **Encoder only model: BERT Series**

Only encoder part is focused

* **Encoder-decoder type model: TS / BART model**

BERT:

* In encoder model No mask self attention,
* Sort of bidirectional in nature. Scan from both direction.
* Objective – masked language model – it is self supervised approach ( no labling needed) ,
* Feed a sentace and mask some tokens(say words) and let model predict those tokens

Question – which token to mask? how much token to mask?

* When masking mask 15 percent of tokens: but we do not replace by {MASK] 100% of time

But 3 diff ways of masking:

* + - 80% of time Replace with [MASK]
    - 10% of time Replace with random word
    - 10% of time Keep same word (why ? – cuz when we train model to predict only the masked token sometimes model forgets to predict the actual token)
* Why 15 percent? Since higher masking(like 50%) leads to context destruction

When predictin the error is only computed in the {MASK} token.

**Next Word Prediction:**

**Masked Language Models (MLMs)** like **BERT**, which use **two training tasks**:

1. **Masked Language Prediction**
2. **Next Sentence Prediction (NSP)**

Let me break it down step by step.

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

* The goal is to train the model to **predict if one sentence follows another** in a coherent text.
* The model is given **two sentences** (W1 and W2) and must determine if W2 **logically follows** W1.
* **How the training data is prepared:**
  + **Positive Samples**: Sentences that actually appear **side by side** in real text.
  + **Negative Samples**: Sentences that are **randomly shuffled** so they do **not** follow naturally.

For example:

* **Positive Pair**:
  + Sentence 1: *I enjoy reading the book.*
  + Sentence 2: *I finished the novel.*
  + **Label** = YES
* **Negative Pair (Random Shuffle Example)**:
  + Sentence 1: *I enjoy reading the book.*
  + Sentence 2: *She went to the park.*
  + **Label** = NO

This helps the model learn **contextual relationships** between sentences.

**2. How is Input Given to the Model?**

* The input consists of **two sequences** (Sentence 1 and Sentence 2).
* They are separated by a **special token** called **[SEP]**.
* Another special token **[CLS]** (Classification Token) is **prepended** at the beginning.
* Example input:

[CLS] I enjoyed reading the [MASK] [SEP] The novel was great [SEP]

* + **[MASK]** is a placeholder for a masked word.
  + **[SEP]** separates the two sentences.
  + **[CLS]** is used for classification tasks.

**3. Why Do We Need the [CLS] Token?**

The main question asked in the explanation is **why we need [CLS] instead of just using other word embeddings for classification.**

**Why Not Just Concatenate All Token Embeddings?**

* If we **concatenate embeddings** of all tokens (except [MASK]), the vector would be **too large**.
* The classifier would need **more parameters** to process this.

**Advantages of Using [CLS]:**

1. **It is Not Biased Toward Any Specific Word**
   * Words like "I" or "reading" have their **own meaning** and might be biased when used for classification.
   * **[CLS] does not belong to the sentence**, so it represents a more **neutral** context.
2. **It Attends to All Tokens**
   * The **self-attention mechanism** ensures that **[CLS] learns from all words** in both sentences.
   * This makes it a **good summary representation** of the entire input.
3. **Reduces the Number of Parameters**
   * If we used all token embeddings for classification, we would need **many more parameters**.
   * Using **just one vector ([CLS])**, we only need a **512 × 2 weight matrix** (for a 512-dimension embedding).
   * If we used all token embeddings, the parameter count would be **much higher**.

**4. Backpropagation and Learning Process**

* **Two losses are used for training**:
  1. **Masked Language Loss** → Predicts the missing word (**[MASK]**)
  2. **Next Sentence Prediction (NSP) Loss** → Predicts if the two sentences are connected.
* Both losses are **backpropagated** through all encoder layers.
* The final trained encoder can be used for:
  1. **Contextual word embeddings** (like ELMo)
  2. **Sentence-level tasks** (like classification)

**Questions That Were Asked**

1. **Why do we need the [CLS] token?**
2. **Can't we just take the concatenation of all token embeddings and classify?**
3. **Why not ignore the [MASK] token and use the other embeddings?**
4. **Why is [CLS] considered unbiased?**
5. **What are the benefits of using [CLS] instead of another token?**
6. **How does using [CLS] reduce the number of parameters?**

**Final Takeaway**

* **Next Sentence Prediction (NSP)** helps models **learn sentence relationships**.
* **Masked Language Modeling (MLM)** helps models **understand word context**.
* **[CLS] is a special token** that provides a **neutral, summarized representation** of the input.
* The model **backpropagates two losses** (Masked Word and NSP) to update its parameters.
* The trained **encoder can be used for multiple NLP tasks** like sentence embeddings and classification.

Eg working:

* Input fed to model
* Model produce embedding
* Embedding fed to classification layer
* Classification layer produce probability – 2 prob – one current token as beginning Cb , current token act as end Ce: Cb = 1 at the start of the req sol sentence , Ce = 1 at the end of sol sentace
* If it dosnt happen theres loss and it is backpropagated

Pb – prob token act as beginning, Pe – token act as endding

Try to maximize the product Pb x Pe , whenr i >= j where i = start pos, j= end pos, but there are many occu of solution sentace. In that case we take the first to occur from beginning

This method was used in **BERT**, but later models like **RoBERTa** removed **NSP** because it was found to be unnecessary in some cases.