AI Club: Project Vivēka

Application for project members.

Project Leads:
Harshith M R, me23b049@smail.iitm.ac.in
Jayden, ep23b035@smail.iitm.ac.in

1 Instructions

Please use the resources provided in the resources doc.

The app is quite long, but we've focused on breaking down each question into smaller pieces that you can spend time on, think and piece together to get the final picture. Ideally there is also a learning outcome associated with each question that we hope will get through to you. The philosophy is to point towards a certain direction, help you discover something interesting about the model, and you come up with an explanation, and in the process of which, you end up learning the concept.

For everything you answer, cite references you used, this is absolutely critical and not doing it will not be taken lightly.

Using ChatGPT/ Gemini/ Claude/ any other inanimate entity is permitted, but cite your conversation/ interaction in your app.

Note: We will prefer people who do DFS on the questions and complete them in-depth than doing BFS and attending all the questions.

2 Submission

Submit the application in format **"cfi-ai-club-project-viveka-name-roll-number.pdf"** and add your grade card as **"roll-number-grade-card.pdf"** in the submission link

3 The General Section

Please be as honest and thoughtful as possible when answering these questions. The goal is to prompt reflection and help both you and us determine if this project is a good fit. Some questions don't have a "right" answer—we're looking for your unique perspective.

3.1 Understanding the Field

1. What do you understand by Mechanistic Interpretability? Explain in your own words, without necessarily relying on formal definitions.
2. Name 3 of your favourite articles, blog posts or videos on Mechanistic Interpretability. What did you like the most about each?
3. With the articles, blog posts or videos that you have seen, find a appropriate name for the project or if you like the current one why do you so?

3.2 Project Goals & Expectations

1. Describe the goals of the project in as much detail as possible. What do you expect working on it to look like? (For any clarification, you may ask in the aspiring Project Members' group.)
2. Imagine yourself a year later, looking back at the work you did. What achievements or progress would make you feel that the experience was a tremendous success?
3. Considering your definition of tremendous success, how much effort do you think it would take to reach that level?

3.3 Motivation & Drive

1. What is your primary motivation for joining this project? What excites you the most?
2. What could dampen your enthusiasm or make you disengage?
3. This project would require you to put in a lot of work. Committing to this could also mean that you might have lesser breathing room to do other things. As always there is a tradeoff between exploring multiple domains in a shallow manner vs doing one in great detail. Considering that you are in early UG, do you think this would a be a good opportunity for you? Don't you think taking such a big time commitment would come at a cost? Are you sure the positives outweigh the challenges?

4 The Technical Section

Problem 1: Directionality is Important! (15 points)

In the first section, we will look at writing out and training a neural network.
Alongside this, we will take a first step towards getting a feel for an important idea: concepts are often represented by directions in a vector space.
A archetype for this example is Word2Vec - a paper that introduced the use of embeddings (dense vectors) for each word. Your goal in this section will be to try and recreate that in a very small scale.
For useful resources, please look at the Word2Vec section in the resources doc.

1. Describe the continuous bag-of-words method introduced in the paper briefly.
2. Write out the equations to train the continuous bag-of-words neural network. Use index notation and the Einstein summation convention.
3. Which part of the neural network gives you the final word embeddings? Does this make sense? Explain in your own words.
4. What changes do you anticipate if the equations had/ did not have a bias term? (To clarify, suppose you have the relation below, the intercept term is called the bias term.)

$$y_i = W_{ij}x_j + b_i$$

5. What do you think the size of the embeddings would have an effect on?
6. What if you associated a different basis with every word? Say that you index every word in the dictionary and for the i^{th} word, you assign a vector such that the $i-1^{th}$ element is 1 and everything else is 0.

   Note: this is commonly called a one-hot vector.

   What do you think are the disadvantages with this approach?

   Are the vectors associated with the words related? (Think in terms of dot-product between vectors.)
7. What is the cosine-similarity metric?
8. Go to TensorFlow Projector and play around with the embeddings there. Find interesting clusters of words having similar meaning and attach screenshots of them in your app. What is the cosine-similarity of vectors within the same cluster, in comparison to vectors in different clusters?
9. Recreate the CBOW approach. Choose an appropriate size for the embeddings. Based on the questions you tried answering above, explain what goes into choosing the size of the embeddings. What happens if its too large or too small?
10. From your own trained CBOW, load embeddings onto the TensorFlow Projector and look for clusters.

11. Check if you are able to replicate the following relation:

$$king - man + woman \approx queen$$

Note: You may use the following dataset.
You are free to use a tokenizer of your choice.

An interesting thing you might observe is how neural networks might allocate neurons/ features for concepts that they might not exactly be trained for. We'll look into this in the next exercise.

Problem 2: Sentiment Detection in Text Completion Models (21 points)

Before we move forward, let us understand the kind of architecture we will be working with.

4.1 Understanding RNNs and LSTMs

1. Describe what RNNs and LSTMs are. Write out equations for their forward pass in index notation. (Describe the vanilla RNN and a single layer LSTM)
2. What is the role of the hidden state in an RNN/ LSTM?

Now let us see how you'll actually go about finding the representation of sentiment in the RNN.

4.2 Interpreting LSTMs

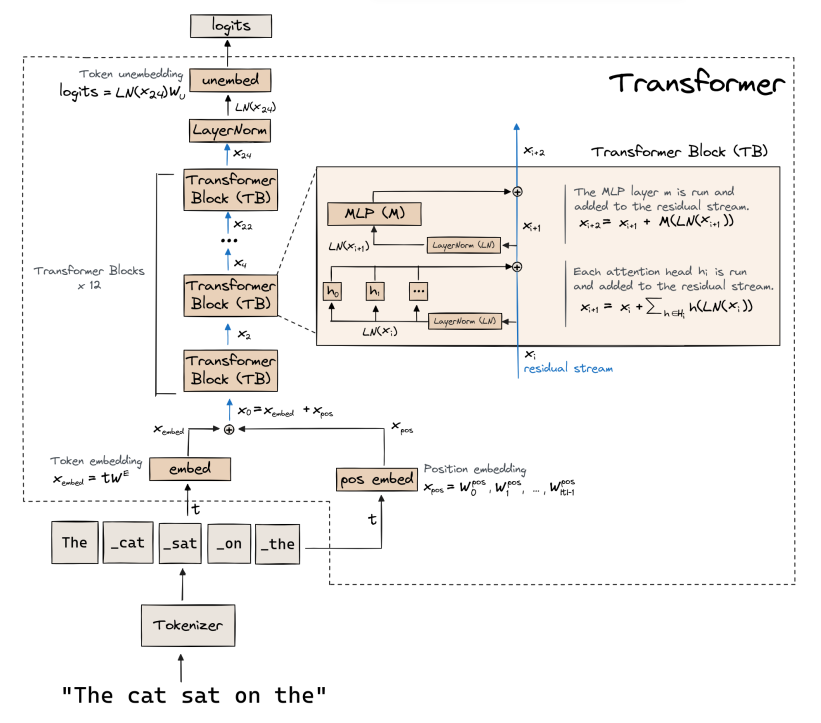
Note: This problem comes with a starter notebook. You'll find code for the training loop. Use that to train the initial LSTM. Your interpretability work will start after that training. Do it in Kaggle so that the runtime lasts long enough for the training, doing it on the free tier in colab will lead to the session terminating before training is complete. Link to the starter notebook.

1. How will you go about isolating the sentiment vector and distinguish it from every other vector that could exist in that vector space? Maybe look for ideas from similar tasks in transformers. Look at the lesswrong article in the resources doc for inspiration. For this question, think in terms of the data you will input to the model.
2. The hidden state changes with every token, and on top of that, each token in this case is just a letter, so at what point will you probe the hidden state to find the sentiment vector? Give a list of the possibilities you would like to try.
3. This LSTM has 2 layers. What does that mean? Can you write the forward pass equations for it? Which layer would you look to for the sentiment analysis vector? Can you come up with a test to figure out which layer might be more relevant?
4. Now list out methods you could use to isolate this feature vector. Again, take inspiration from the lesswrong post shared in the resources doc.
5. Pass inputs from the Amazon Reviews Dataset. Apply the methods you suggested so far and report your classification score. DO NOT train on this dataset. Provide code for whatever you tried.
6. Are the methods that you suggested above proof that the vector you found is really the sentiment detection vector? Are you sure that it is not detecting something that is correlated to sentiment detection?
7. Design a test to show that the feature you have with you is indeed a sentiment detector.

All LLMs use an architecture called the transformer. In this question, we'll try to find out what they're all about.

Problem 3: Entering the Transformer World (16 points)

4.3 Introducing the architecture



Taking a closer look at the transformer block.

1. What forms the transformer block? What are its parts? Go through the resources in the doc to help answer this question.

4.4 Using a transformer for a classification task

Check out Disaster tweets classification
We would like you to try the following approach:

1. Read up about what BERT is. Refer to the resources in the provided doc. What are the [CLS] and the [SEP] tokens?

2. Load ModernBERT. This is pretty much a drop in substitute for the original BERT but is faster and has a lot of the improvements that have been implemented in LLMs over the years. You need not go to the details of what all they are for this app.

3. Train a simple classifier on the top of the [CLS] token embeddings from the last layer of ModernBERT. Do not train the entire transformer, train only the classifier on top. Make a submission and report your accuracy along with your code. Also train a classifer on the [CLS] token just after the initial embedding and report the accuracy.

4. Instead of the [CLS] token, suppose you took the mean of embeddings of all the tokens from the last layer. Would it make sense to take a weighted mean of the embeddings, i.e., scale each embedding by its norm and add them together? Based on your understanding from the attention mechanism, do you suspect that the norm of different embedding vectors will be different? Print out the norm to check for yourself? Are you surprised? Explain what you're seeing.

4.5 A Closer Look at Attention

Time to look at something at the very heart of the transformer architecture itself: Attention.

1. By this point, you would have become quite familiar with the idea of tokens being converted to vectors that have represent some concept. Based on this, does the attention mechanism make sense for how it helps figure out 'important' parts from a sentence? Explain in your own words by giving an example.

2. In the previous section, you used the embedding associated with the [CLS] token from the first and the last layer of ModernBERT to train your classifer. Did you notice a difference in accuracy? Come up with an explanation for why this happens.

4.6 Looking at the Residual Stream

1. What do you understand by the residual stream in transformer models?

2. The residual stream has to represent a lot of different concepts at once. How many unique concepts can it represent at a particular layer? Does this seem like a large/ small number? Justify your thoughts.