Full name: ______________________________

Initials: ______________________________

Instructions

- This is a *closed book* exam. No course material or other additional material is allowed.
- Fill in your full name as well as your initials in block letters at the top of this page.
- Write your initials on each page.
- This exam consists of **14 pages**. Make sure you have all pages.
- If you have questions, raise your hand to clarify any uncertainties.
- Use the designated space for your answers. You may use the back of the page as additional space. If you do, indicate that your answer continues on the back.
- Write clearly and legibly. Only readable answers give points.
- You can answer in German or English (and you can mix).
- Sign the declaration of academic integrity below.
- Good luck with the exam!

Question:	1	2	3	4	5	6	7	8	9	10	Total
Points:	5	10	9	10	4	18	10	14	4	6	90
Score:											

Declaration of Academic Integrity

By signing below, I pledge that the answers of this exam are my own work without the assistance of others or the usage of unauthorized material or information.

Signature: ....

1. **Introduction to NLP**

   (a) [2 points] Consider a tokenizer with BPE tokenization (e.g. the BertTokenizer). What happens to the tokens it outputs when the input text has spelling mistakes?

   ....
   ....
   ....
   ....
   ....
   ....
   ....
   ....
   ....

   (b) [3 points] We now run the input text with spelling mistakes through BERT. How will the spelling mistakes affect the quality of our model's outputs? What does it depend on?

   ....
   ....
   ....
   ....
   ....
   ....
   ....
   ....
   ....

2. **Embeddings**

(a) [4 points] Explain the TF and the IDF part of TF-IDF. For each, say what it is and why it's a useful metric of token importance.

. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .

(b) [4 points] Name one advantage each of low-dimensional embeddings (e.g. word2vec) and high-dimensional embeddings (e.g. bag-of-words).

. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .

(c) [2 points] Given the following vectors for man, woman and king, estimate the vector for queen.

$$\text{man} = \begin{pmatrix} 4 \\ 1 \\ 0 \end{pmatrix}, \quad \text{woman} = \begin{pmatrix} 5 \\ -2 \\ 9 \end{pmatrix}, \quad \text{king} = \begin{pmatrix} 1 \\ 17 \\ -3 \end{pmatrix}$$

. . . .
. . . .
. . . .
. . . .
. . . .
. . . .

3. **Recurrent Neural Networks**

Here is the formulation of an LSTM:

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f) \tag{1}$$
$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i) \tag{2}$$
$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o) \tag{3}$$
$$\tilde{c}_t = \tanh(W_c h_{t-1} + U_c x_t + b_c) \tag{4}$$
$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t \tag{5}$$
$$h_t = o_t * \tanh(c_t) \tag{6}$$

(a) [3 points] Explain the purpose of each of the 3 gates: $f_t, i_t,$ and o_t.

....
....
....
....
....
....
....
....
....

(b) You need to get rid of one gate.

i. [3 points] What do you do? Explain why.

....
....
....
....
....
....
....
....
....

ii. [1 point] Which equation(s) above (1-6) do you remove?

....
....
....

iii. [2 points] Which equation(s) do you add?

. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .

4. **Advanced RNNs**

   (a) [10 points] The code below shows the Python implementation of the `__init__` method of a 2-layer bidirectional sequence-to-sequence LSTM. Unfortunately, parts of the code have been lost. Fill in the gaps to reconstruct it.

```
class Seq2seqBiLSTM(nn.Module):
    """ Bidirectional 2-layer sequence-to-sequence LSTM. """

    def __init__(self, embedding_dim, hidden_dim):
        super().__init__()
        self.num_encoder_layers = 2
        self.num_decoder_layers = 2
        self.bidirectional = True
        self.num_directions = ........................
        self.encoder_layer_1 = nn.LSTM(
            input_size=..............................,
            hidden_size=.............................,
            num_layers=1,
            bidirectional=...........................,
        )
        self.encoder_layer_2 = nn.LSTM(
            input_size=..............................,
            hidden_size=.............................,
            num_layers=1,
            bidirectional=...........................,
        )
        self.decoder = nn.LSTM(
            input_size=..............................,
            hidden_size=.............................,
            num_layers=self.num_decoder_layers,
            bidirectional=...........................,
        )
```

5. Attention

(a) [4 points] Explain the difference between self-attention and the type of attention used in sequence-to-sequence RNNs.

6. Transformer

```python
def forward(self, x):
    output = x + self.pos_emb(x)  # add absolute position embeddings
    for layer in self.layers:  # iterate over layers
        # project to queries, keys and values
        q = self.project_q(output)
        k = self.project_k(output)
        v = self.project_v(output)
        # split to attention heads
        qs, ks, vs = self.split(q), self.split(k), self.split(v)
        # dot-product attention
        head_outputs = []
        for q_h, k_h, v_h in zip(qs, ks, vs):
            attention_scores = q_h @ k_h.tranpose(-1, -2)
            attention_scores /= math.sqrt(self.head_dim)
            attention_probs = F.softmax(attention_scores, dim=-1)
            head_outputs.append(attention_probs @ v_h)
        attn_out = torch.cat(head_outputs, dim=-1)  # concatenate
        attn_out = self.output_projection(attn_out)
        attn_out += layer_input  # add residual
        attn_out = self.layer_norm_1(attn_out)  # layer norm
        # FFN
        ffn_out = self.up_projection(attn_out)
        ffn_out = F.relu(ffn_out)
        ffn_out = self.down_projection(ffn_out)
        ffn_out += attn_out  # add residual
        output = self.layer_norm_2(ffn_out)  # layer norm
    return output  # final layer hidden states
```

(a) [12 points] Above you see an implementation of the forward pass of a Transformer encoder. What is required to change it for a Transformer decoder? For each change, list the line number at which you change (or after which you insert) something, then write the replaced/inserted line. You can also copy lines by giving the line numbers ("copy 42"). Correct Python syntax will not be graded.

(b) [4 points] In the Transformer architecture, why can't we add relative position embeddings just once to the input embeddings like we can for absolute position embeddings?

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

(c) [2 points] Where do we have to add it instead?

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

7. **Pretraining**

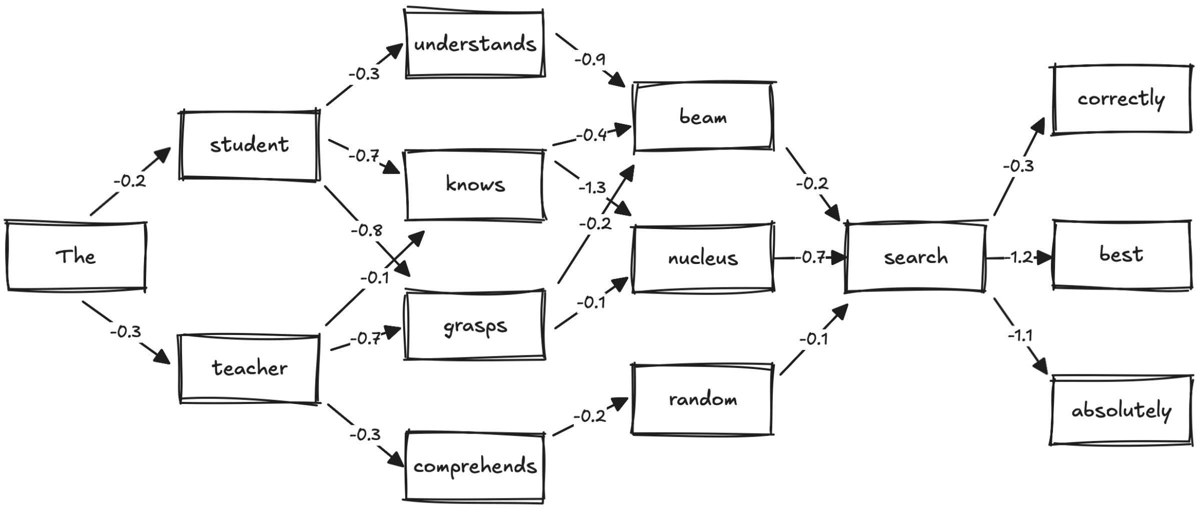
(a) [6 points] Create BERT pretraining data from the sentence: *The quick brown fox jumps over the lazy dog.* Create a single sentence. Apply BERT's pretraining objective and use each possible distortion of the input at least once. Label the distortions you applied.

. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .

(b) [4 points] Fact check the following statement s: *Joe Biden won the 2024 US presidential election.* You are only given a pure pretrained language model (no instruction tuning). Use PET-style verbalizers to fact check statement s.

. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .

8. **Text Generation**



Beam search with transitions showing log probabilities.

(a) [10 points] Run beam search with length normalization on the example shown above. The starting token *The* is already given. Use the following settings: `num_beams` $= 2$, `min_new_tokens` $= 5$, `length_penalty` $= 1$. The length-normalized score with length penalty α is computed as:

$$\text{score}(h) = \frac{\log(p(h))}{\text{length}(h)^{\alpha}}$$

Write down the final hypotheses together with their logprobs.

(b) [4 points] Now do it again, this time with `min_new_tokens` = 3. Write down the final hypotheses together with their logprobs.

. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .

9. **Text Classification**

   (a) [4 points] You train four models for text classification. For each of the following models, check the boxes of all the operations that **absolutely should not** be performed.

   word2vec-GoogleNews-300

   - ☐ Lemmatize the input.
   - ☐ Lowercase the input.
   - ☐ Stopword removal.
   - ☐ Train your own classifier.

   A randomly initialized Transformer.

   - ☐ Lemmatize the input.
   - ☐ Lowercase the input.
   - ☐ Stopword removal.
   - ☐ Train your own classifier.

   bert-base-cased

   - ☐ Lemmatize the input.
   - ☐ Lowercase the input.
   - ☐ Stopword removal.
   - ☐ Train your own classifier.

   distilbert-base-uncased-qa-boolq

   - ☐ Lemmatize the input.
   - ☐ Lowercase the input.
   - ☐ Stopword removal.
   - ☐ Train your own classifier.

10. **Guest Lectures**

(a) [2 points] In the NLP in Pharma guest lecture, we have seen that medical coding is a tough challenge, with roughly 80k labels (lowest level terms) in the Medical Dictionary for Regulatory Activities (MedDRA) to select from. Assume that ChatGPT has seen the (proprietary) dictionary in its pretraining. Write a prompt to make ChatGPT output the correct label for the symptom *stomach ache*.

....
....
....
....
....
....
....
....
....

(b) [4 points] ChatGPT answers with: *The MedDRA label is "abdominal pain upper".* How do you check this result with the help of an NLI model and a medical article describing stomach ache (which contains the correct MedDRA label)?

....
....
....
....
....
....
....
....
....
....
....
....