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 **12 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.
- Sign the declaration of academic integrity below.
- Good luck with the exam!

Question:	1	2	3	4	5	6	7	8	9	Total
Points:	6	8	12	12	10	16	8	12	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] What is the idea of the *NLP pipeline*?

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

   (b) [2 points] Name as many stages as you know.

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

   (c) [2 points] Which stages of the NLP pipeline are taken care of by pretrained neural networks (such as BERT), and which stages do still have to be done by our (or a library's) code?

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

2. **Embeddings**

(a) [4 points] Why is cosine similarity a good word similarity metric for word2vec embeddings but not for one-hot encoding?

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

(b) [4 points] In topic modeling, we perform topic analysis by manually assigning topic names, for example by inspecting the words with the highest weight for a given topic. In class, we saw an example where we assigned the topic name *arts* to a topic with the words *new*, *film*, *show*, *music*, *movie*, *play*, *musical*, *best*, *actor*, and *opera*. A cheap automatic solution is to pick the word with the highest weight, but this may not be representative of the topic; in the previous example we would have picked *new*. Suggest a better fully automatic solution.

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

3. **Recurrent Neural Networks**

(a) You train a tri-gram language model on the following data: "the quick brown fox jumps over the lazy dog. the fox jumps across the field." N-grams do not cross sentence boundaries. Give all probabilities as fractions.

i. [2 points] What is the probability of the next word being "over" given the prefix "the quick red fox jumps"?

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

ii. [2 points] What is the probability of the next word being "down" given the prefix "the fox jumps"?

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

iii. [4 points] What is the probability of the next word being "down" given the prefix "the fox jumps" when you use a tri-gram LM with add-one smoothing?

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

(b) Exploding gradients in RNNs.

i. [2 points] Explain how gradients can explode in RNNs.

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

ii. [2 points] Name two techniques (not neural network architectures) to fix exploding gradients.

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

4. **Attention**

(a) [12 points] The code below shows the forward pass of a decoder LSTM with attention. Unfortunately, a part of the loop over decoder time steps has been lost. Reconstruct it with the help of some of the functions below. Write Python code (exact syntax will not be graded). The number of dotted lines is no indication of how many code lines you need. If you use a different version of attention in RNNs than the one we have used in class, describe it.

- F.layer_norm
- F.relu
- F.softmax
- self.compute_context_vector
- self.project_to_higher_dim
- self.project_to_lower_dim
- self.project_to_same_dim
- torch.add
- torch.cat
- torch.matmul
- torch.mul
- torch.ones
- torch.stack
- torch.zeros

```
def forward(self, y, encoder_hidden_states):
    h = torch.zeros(self.hidden_dim)
    c = torch.zeros(self.hidden_dim)
    hidden_states = []

    # loop over the target sequence
    for y_i in y:
        ..................................................
        ..................................................
        ..................................................
        ..................................................
        ..................................................
        ..................................................
        ..................................................
        ..................................................
        ..................................................
        h, c = self.cell(y_i, (h, c))  # LSTM cell forward pass
        ..................................................
        ..................................................
        ..................................................
        ..................................................
        hidden_states.append(h)

    return torch.stack(hidden_states), (h, c)
```

5. **Transformer**

   (a) [4 points] Name 2 fundamental reasons why the Transformer outperforms the RNN.

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

   (b) [4 points] Why do we need position encoding in the Transformer, but not in the RNN?

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

   (c) [2 points] What is the big weakness of the attention mechanism in the Transformer, and the main reason researchers have been looking at recurrent architectures again in the past year?

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

6. **Pretraining**

Here is an example from Winogrande: *Sarah was a much better surgeon than Maria so _ always got the easier cases.* The options are *Sarah* and *Maria*.

(a) [3 points] Explain why this example (and the entire dataset) is hard to solve for word embeddings, as opposed to RNNs and Transformers.

(b) [3 points] Explain why this example (and the entire dataset) is hard to solve for a model finetuned from a randomly initialized neural network, as opposed to a pretrained one.

(c) [5 points] Create a one-shot prompt for which a good model should output the correct solution to the introductory example in this section (by actually solving the task, not just repeating/outputting the solution).

(d) [5 points] Create a one-shot chain-of-thought prompt for the above example.

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

7. **Text Generation**

(a) Nucleus sampling.

i. [1 point] What value do you have to set for p in Nucleus sampling to get the same results as greedy decoding? Round to 2 decimal points.

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

ii. [4 points] Apply Nucleus sampling with $p = 0.75$ to the model's following next word predictions and associated probabilities. Write down the updated distribution (word and updated probability, in fractions) we sample our next word from in Nucleus sampling.

- where – 0.4
- that – 0.2
- which – 0.1
- so – 0.1
- such – 0.05
- in – 0.04
- a – 0.02
- . – 0.01

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

(b) [3 points] Will the following candidate translation achieve high BLEU score? Why? (Hint: 1 mile = 1.60934 kilometers)
Candidate: I ran a mile.
Reference: My running went for 1.61 kilometers.

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

8. **Text Classification**

(a) [12 points] Your colleague AM has been working on a text classification task and created a pull request (see next page). You were assigned to peer review his code. Check the code for correctness and efficiency, and suggest improvements. State the line number, what needs to be changed and why. You do not have to provide the implementation for the changes. You can assume that all imports are present and the comments are not misleading.

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

```python
def evaluate():
    # read data
    data = load_dataset('winogrande', 'winogrande_l', split='validation')

    # preprocess
    processed = []
    stemmer = nltk.stem.PorterStemmer()
    for example in data:
        options = []
        options.append(stemmer.stem(example['option1'].lower()))
        options.append(stemmer.stem(example['option1'].lower()))
        options = sorted(options)
        sentence = f"Sentence: {example['sentence']}, Option 1: {options[0]}, Option 2: {options[1]}"
        processed.append(sentence)

    # load model
    tokenizer = AutoTokenizer.from_pretrained(
        'google-bert/bert-base-cased'
    )
    model = AutoModelForSequenceClassification.from_pretrained(
        'google-bert/bert-base-cased'
    )

    # run evaluation
    inputs = tokenizer(processed, padding=True)
    outputs = model(**inputs)

    # outputs.logits shape: [batch_size, seq_len, hidden_dim]
    logits = outputs.logits[:, -1]
    if self.num_classes == 1:
        predictions = torch.sigmoid(logits)
    else:
        predictions = torch.nn.Softmax(logits)

    # normalize and regularize
    predictions = F.layer_norm(predictions)
    predictions = F.dropout(predictions)

    return predictions
```

Evaluation function for text classification.

9. **Research Topics and Guest Lectures**

(a) [3 points] In the argument mining guest lecture, we have seen that the labels that a large language model gives for the same question can flip for various reasons. Devise a strategy which can reduce the uncertainty in the output label.

(b) [3 points] In the NLP in health guest lecture, we have seen that not all models perform well on medical text data. Since the models are usually large (e.g. EPFL's Meditron model is built from Llama 3 70B), we need a lot of resources to evaluate them. In contrast, the tokenizer is cheap to download and run. Given a list of medical terms you care about and only the tokenizers of your candidate models, how can you select the most promising models for an in-depth evaluation?