NLP and Information Retrieval
Sample Exam 1 with Answers

Generated by ChatGPT

Instructions

- Answer all questions.
- Show all calculations where required.
- Justify your answers clearly in transfer questions.

Task 1: Tokenization & Subword Methods

1a) (Knowledge)
Explain the difference between **word-level tokenization** and **subword tokenization** (e.g., Byte-Pair Encoding). Why is subword tokenization widely used in modern NLP?

Answer:
Word-level tokenization treats each complete word as a token (usually split on whitespace and punctuation), whereas subword tokenization breaks words into smaller units (subwords). This method, as used in Byte-Pair Encoding (BPE), reduces vocabulary size and handles out-of-vocabulary (OOV) words by composing them from common subword units.

1b) (Understanding)
Given the word `"unbelievable"`, provide a possible BPE tokenization output (you can invent merges). Briefly explain why this split might help handle out-of-vocabulary words.

Answer:
A possible tokenization could be: `"un", "believ", "able"`. Splitting into these subwords enables the model to recognize common prefixes and suffixes even if the entire word was not seen during training, thereby mitigating OOV issues.

Task 2: Text Classification Foundations

2a) (Knowledge)
Name three possible **text classification** tasks and briefly describe the feature extraction steps you might perform in a **traditional** (non-neural) pipeline (e.g., TF-IDF vectors).

Answer:
Examples of text classification tasks include sentiment analysis, spam detection, and topic categorization. A traditional pipeline involves:

1. Preprocessing (tokenization, stopword removal, stemming/lemmatization)
2. Feature extraction using methods such as bag-of-words or TF-IDF
3. Training a classifier (e.g., logistic regression or Naïve Bayes)

2b) (Transfer)
A social media company wants to **detect hate speech** vs. **harmless content**. Given a small labeled dataset of tweets, would you choose a simple Naïve Bayes classifier or a large Transformer-based classifier (like BERT)? Justify your choice by weighing **time/resources** vs. **potential accuracy**.

Answer:
Given the small dataset and resource constraints, a Naïve Bayes classifier is likely preferable due to its simplicity and efficiency. Although a Transformer-based model like BERT might achieve higher accuracy, it requires significantly more data and computational power for fine-tuning.

Task 3: IR Basics & BM25

3a) (Knowledge)
Briefly define the **BM25** scoring function. What are the roles of term frequency (TF), inverse document frequency (IDF), and document length in BM25?

Answer:
BM25 is a ranking function used to assess the relevance of a document to a query. It uses:

- **Term Frequency (TF):** Higher term occurrences increase the score.
- **Inverse Document Frequency (IDF):** Rarer terms are weighted more heavily.
- **Document Length:** Longer documents are penalized to normalize the TF.

3b) (Do Stuff / Short Calculation)
Suppose we have 3 documents, each containing the word "computer" as follows:

- Document A: 3 occurrences
- Document B: 0 occurrences
- Document C: 5 occurrences

Given a query "computer," explain how BM25 assigns higher scores to documents with more term occurrences. You do **not** need to calculate an exact numeric score—just outline how increased term frequency and shorter document length would boost BM25.

Answer:
BM25 increases the score for documents with higher term frequency; hence, Document C (5 occurrences) would receive a higher score than Document A (3 occurrences), while Document B would score lowest. Additionally, if a document is shorter, the impact of term frequency is amplified due to document length normalization.

Task 4: True or False (Conceptual)

Mark each statement **True (T) or False (F)** and provide **one sentence of explanation**:

(1) Transformers rely on recurrent connections to handle long-range dependencies.

(2) Dense retrieval uses approximate nearest neighbor search to scale to large document collections.

(3) Instruction tuning only works for small, domain-specific language models.

(4) N-gram language models struggle with zero probabilities for unseen word sequences.

(5) In a two-stage IR pipeline, the first stage must always be BM25.

Answers:

(1) **False.** Transformers use self-attention mechanisms instead of recurrent connections.

(2) **True.** Dense retrieval commonly employs ANN search to efficiently handle large-scale vector comparisons.

(3) **False.** Instruction tuning can be applied to both small domain-specific models and large general-purpose models.

(4) **True.** Without smoothing, n-gram models assign a zero probability to any unseen word sequence.

(5) **False.** The first stage can use various retrieval methods, not solely BM25.

Task 5: Neural Re-Ranking & BERT Fine-Tuning

5a) (Understanding)
Explain how a **neural re-ranking** approach (e.g., BERT re-ranker) works in combination with a **first-stage** retriever. Why might this two-stage approach outperform using only BM25 or only a single dense retrieval model?

Answer:
A two-stage pipeline first uses a fast retriever (like BM25 or dense retrieval) to shortlist candidate documents, then applies a BERT re-ranker to evaluate each query-document pair using contextualized representations. This leverages the speed of the first stage and the deep semantic understanding of BERT in the second stage, resulting in improved overall relevance.

5b) (Transfer)
You have a **legal document retrieval** system where correctness is critical. Describe a scenario where **re-ranking** with a carefully fine-tuned BERT model is crucial, and highlight potential **drawbacks** (e.g., inference speed) of adding this re-ranker.

Answer:
In legal document retrieval, subtle language differences can be vital; a BERT re-ranker can accurately interpret these nuances to prioritize highly relevant documents. However, the computational cost and slower inference times of running BERT on many candidates can impede real-time responsiveness.

Task 6: Dense Retrieval (Calculation / “Do Stuff”)

6a) (Knowledge)
What are the **three major phases** in building a dense retrieval system? Name **one** advantage dense retrieval has over BM25.

Answer:
The three phases are:

1. **Training**: Learning vector representations for queries and documents.
2. **Indexing**: Creating an index of document embeddings using ANN structures.
3. **Inference**: Retrieving documents by comparing query embeddings with document embeddings.

Dense retrieval can capture semantic similarity beyond exact term matching, which BM25 relies on.

6b) (Short Calculation)
You have a query vector $\mathbf{q} = [1.0, 0.5]$ and two document vectors:

- $\mathbf{d}_1 = [0.9, 0.4]$
- $\mathbf{d}_2 = [0.1, 0.2]$

Using **dot product** as the similarity function, compute:

- $\mathbf{q} \cdot \mathbf{d}_1 = 1.0 \times 0.9 + 0.5 \times 0.4 = 0.9 + 0.2 = 1.1$
- $\mathbf{q} \cdot \mathbf{d}_2 = 1.0 \times 0.1 + 0.5 \times 0.2 = 0.1 + 0.1 = 0.2$

Answer:
Since $\mathbf{q} \cdot \mathbf{d}_1 = 1.1$ is greater than $\mathbf{q} \cdot \mathbf{d}_2 = 0.2$, the document corresponding to $\mathbf{d}_1$ is more relevant.

Task 7: Ranking Evaluation Metrics

7a) (Knowledge)
Define **Precision@k** and **Recall** in the context of Information Retrieval. Provide a quick example of each.

Answer:
Precision@k is the fraction of relevant documents among the top k retrieved documents. For example, if 2 out of 5 retrieved documents are relevant, Precision@5 is 0.4. **Recall** is the fraction of all relevant documents that have been retrieved; for instance, if there are 3 relevant documents and 2 are retrieved, Recall is approximately 0.67.

7b) (Calculation)
A search system returns 5 results for a query. Among these results, the relevant ones appear at ranks [1, 4]. There are 3 relevant documents in total for this query.

- **Precision@5** $= \frac{2}{5} = 0.4$
- **Recall** $= \frac{2}{3} \approx 0.67$

Answer:
Recall is higher in this scenario, indicating that while a large fraction of all relevant documents were retrieved, the presence of non-relevant documents in the top 5 lowers precision.

Task 8: Adapters & LoRA (Parameter-Efficient Fine-Tuning)

8a) (Understanding)
Explain the core idea behind **adapter layers** and **LoRA**. How do they reduce the number of trainable parameters compared to fully fine-tuning a large model?

Answer:
Adapter layers and LoRA add small trainable modules to a pre-trained model while keeping the original model weights frozen. This method allows the model to adapt to new tasks by only training the additional parameters, significantly reducing memory and computational requirements.

8b) (Transfer)
You need to fine-tune a **GPT-like** model for a specialized task but have limited GPU memory. Would you choose **adapter layers** or **LoRA**, and why?

Answer:
Either method can be effective, but LoRA might be preferred because it decomposes weight updates into low-rank matrices, further reducing memory usage while still capturing task-specific nuances.

Task 9: Instruction Tuning & Prompting

9a) (Knowledge)
What is **instruction tuning**, and how does it differ from using **few-shot prompting**?

Answer:
Instruction tuning fine-tunes a model on a dataset of instruction-response pairs, effectively altering the model's weights to better follow instructions. In contrast, few-shot prompting provides examples within the prompt without changing the underlying model parameters.

9b) (Transfer)
You need an LLM to follow instructions better. Outline a **dataset** you would create for instruction tuning and explain its benefits.

Answer:
The dataset should consist of diverse instruction-response pairs across different domains (e.g., summarization, step-by-step guides, question answering). This helps the model learn to generalize instructions and produce more consistent and accurate responses, reducing reliance on prompt engineering.

Task 10: RLHF & Model Alignment

10a) (Understanding)
How does **Reinforcement Learning from Human Feedback (RLHF)** help align an LLM?

Answer:
RLHF uses human feedback to train a reward model that evaluates the quality of model outputs. The LLM is then fine-tuned using reinforcement learning to maximize the reward, thereby aligning its outputs with human preferences and reducing undesirable behavior.

Task 11: Long Context Handling

11a) (Knowledge)
Explain one approach to handling **long input sequences** in Transformers.

Answer:
One effective approach is to use sparse attention mechanisms (as seen in models like Longformer or BigBird), which limit the number of positions each token attends to, reducing the quadratic complexity of full self-attention and enabling the processing of longer sequences efficiently.