First Midterm Exam

601.467/667 Introduction to Human Language Technology

Fall 2026
Johns Hopkins University
Co-ordinator: Philipp Koehn

8 October 2026

Complete all questions.

Use additional paper if needed.

Time: 75 minutes.

Name of student: _____

Q1. Syntax *20 points*

Formal Language Theory

1. **Reading a regular grammar (4 points)**

   Consider the grammar below, with start symbol S and alphabet $\Sigma = \{a, b\}$:

   $$S \to aS \mid bA \qquad A \to bA \mid \epsilon$$

   a. Describe the language generated by this grammar in words, and give a regular expression for it. *(2 points)*

   b. Can the grammar generate `b`? Can it generate `aba`? Why or why not? *(1 point)*

   c. Give a sequence of rule applications that generates `aabb`. *(1 point)*

(a) Zero or more as followed by one or more bs. A regular expression is a^*b^+. **(b)** It generates `b` via $S \Rightarrow bA \Rightarrow b$. It cannot generate `aba`: once the grammar produces its first b, it cannot produce another a. **(c)**

$$S \Rightarrow aS \Rightarrow aaS \Rightarrow aabA \Rightarrow aabbA \Rightarrow aabb$$

2. **Applying the Chomsky hierarchy (4 points)**

   For each production below, identify the **most restrictive Chomsky-hierarchy rule format** (regular, context-free, context-sensitive, or unrestricted) that permits it, and explain your choice. Standard conventions apply: capital letters denote nonterminals, and lowercase letters denote terminals.

   a. $A \to aB$ *(1 point)*

b. $A \to BC$ *(1 point)*

c. $bAc \to bDc$ *(1 point)*

d. $bA \to c$ *(1 point)*

(a) Regular (Type 3): the right side is a terminal followed by a nonterminal. **(b)** Context-free (Type 2): there is one nonterminal on the left, but the right side is not a regular production. **(c)** Context-sensitive (Type 1): it rewrites A while preserving its surrounding context b and c. **(d)** Unrestricted (Type 0, corresponding to recursively enumerable languages): it rewrites a multi-symbol left side without preserving its context, and it shortens the string.

Constituency and Parsing

3. **From a treebank tree to a grammar (5 points)**

   A treebank contains the following constituent tree:

```
(S
  (NP (DT the) (NN student))
  (VP (VBD saw)
      (NP (DT a) (NN bird))))
```

   a. Extract all distinct CFG productions represented by this tree, including productions that generate words. List a repeated production only once. *(3 points)*

   b. Use the substitution test to provide evidence that `a bird` is a constituent. *(1 point)*

   c. Explain why a CFG extracted from a large treebank can generate sentences that native speakers judge ungrammatical. Hint: use the rules you extracted above (feel free to add some). *(1 point)*

(a) There are eight distinct productions, compactly written as:

$$\begin{aligned} \text{S} &\rightarrow \text{NP VP} \\ \text{NP} &\rightarrow \text{DT NN} \\ \text{VP} &\rightarrow \text{VBD NP} \\ \text{DT} &\rightarrow \text{the} \mid \text{a} \\ \text{NN} &\rightarrow \text{student} \mid \text{bird} \\ \text{VBD} &\rightarrow \text{saw} \end{aligned}$$

Award one point for the three structural productions and two points for the five lexical productions. **(b)** "The student saw *a bird*" can be replaced by "The student saw *it*." **(c)** The grammar can recombine rules without retaining all the restrictions that governed the original sentences, such as agreement constraints, so it can overgenerate.

4. **Binarization and a small CKY parse (7 points)**

Consider this grammar, with start symbol S:

$$\begin{aligned} \text{S} &\rightarrow \text{NP V NP} \\ \text{NP} &\rightarrow \text{D N} \\ \text{D} &\rightarrow \text{a} \mid \text{the} \\ \text{N} &\rightarrow \text{robot} \mid \text{bird} \\ \text{V} &\rightarrow \text{sees} \end{aligned}$$

a. Binarize the first production by grouping V NP under a new nonterminal X. Write the two resulting productions. *(2 points)*

$\text{S} \rightarrow \text{NP X}$ $\quad$ $\text{X} \rightarrow \text{V NP}$

b. Using the binarized grammar, apply CKY to:

$$\text{a}_1 \quad \text{robot}_2 \quad \text{sees}_3 \quad \text{the}_4 \quad \text{bird}_5$$

List **every nonempty chart entry spanning at least two words**. Use inclusive word positions: for example, $\text{NP}[1, 2]$ means an NP covering words 1 through 2. *(4 points)*

The entries are $\text{NP}[1, 2]$, $\text{NP}[4, 5]$, $\text{X}[3, 5]$, and $\text{S}[1, 5]$. Award one point for each entry.

c. Does the grammar recognize the sentence? Identify the chart entry that establishes your answer. *(1 point)*

Yes: $\text{S}[1, 5]$ covers the entire sentence.

Q2. Semantics *15 points*

Consider the following document-term count matrix:

Doc #	cat	tail	grep	dog	python	leash
1	12	8	0	0	3	7
2	0	0	0	10	0	11
3	17	0	15	0	1	1
4	5	13	7	0	9	0

1. Treating the matrix as our initial knowledge of similarity between words and between documents, which pairs of words and pairs of documents have zero similarity? Explain why. *(5 points)*

   ”cat”, ”tail”, ”grep”, and ”python” all have zero similarity with ”dog”, because they never co-occur in the same document. Document 2 has zero similarity with document 4, because they have zero overlap in vocabulary.

2. Under a semantic model trained using the distributional information in the table above, which of the initially-zero-similarity words would ”dog” be closest to? Explain why. *(5 points)*

   ”cat” and ”dog” would move particularly closer together, linked by second-order co-occurrence with respect to ”leash”.

3. Alongside the ”animal”-semantics, the table shows a ”programming”-semantics, where some words have particular meanings in each. Such words are *polysemous*, like ”tail” and ”cat”. What difference between *how* these two words are polysemous does WordNet *fail* to capture? *(5 points)*

   ”cat” is an example of *homonymy*: while spelled and pronounced the same, the programming-sense is a shortening of ”concatenate” and unrelated to the animal-sense. In contrast, the two senses of ”tail” are both referring to the trailing part of an entity.

Q3. Deep Learning *10 points*

1. Why do we use a softmax layer? *(2 points)*

   To make things a probability distribution

2. What is supervised learning? *(2 points)*

   You have labeled examples

3. Please give 2 examples of Supervised Learning Tasks in HLT. *(2 points)*

   MT, ASR, LID, etc

4. Why have deep learning models gotten better in the last 15 years? Name 2 reasons for this. *(4 points)*

   Data Availability, GPUs, Other possible solutions

Q4. Information Retrieval *10 points*

1. What is the difference between queries and topics? *(3 points)*

   A topic conveys the exact information that a user is looking for; a query is a realization of a search topic, which may be short, ambiguous, or even not present to the search engine (e.g., in a recommendation system where a user profile is an implicit query).

2. What is the benefit of learned sparse retrieval models compared to dense retrieval models? *(3 points)*

   Learned sparse retrieval models turn queries and documents into contextualized sparse vectors, which can leverage an inverted index for fast query serving, i.e., low query latency, compared to most dense retrieval methods that use approximate nearest-neighbor search engines, while achieving similar effectiveness to dense models.

3. What is the benefit of retrieval-augmented generation compared to only using an LLM to respond to users? *(4 points)*

   1. Provide grounded information from the document collection.
   2. Supplement the LLM when it does not have the knowledge to answer the user's question.

Q5. Machine Translation *10 points*

1. Define the three main application scenarios for machine translation: *(3 points)*

   - Dissemination:
   - Assimilation:
   - Communication:

   Dissemination: Translate out to many languages, e.g. localization; Assimilation: Translate into your own language, e.g. cross-lingual search; Communication: Real-time two-way conversation, e.g. tourist travel smartphone app

2. Explain how each application scenario imposes different requirements on the translation accuracy and speed/memory for the MT engine. *(2 points)*

   There are many valid answers for this question, as long as one demonstrates an understanding of the different application scenarios. For example: Communication applications require real-time interactivity, so short latency is required; it may also need to fit on device. Dissemination, especially in the cases such as company translation of official webpages or manuals, require fluent translation and accurate representation of information in the target language. Assimilation requirements may be less in comparison: MT accuracy should be good enough for the user to digest the information in the translation.

3. Please demonstrate your understanding of the attention mechanism in a Transformer encoder block. Suppose I have an input of 3 tokens with embeddings e_1, e_2, e_3, where $e_j \in \mathcal{R}^d$. I would like to compute the representations for the next layer l_1, l_2, l_3, where $l_i \in \mathcal{R}^d$. Using the idea of scaled dot-product (self)-attention in Transformers, write down the equation for one of these vectors l_1 as a function of the other terms. Also, indicate which term is query, key, and value. You do not need to use multi-head attention; just assume a single head for simplicity. *(5 points)*

   $l_1 = \sum_{j=1}^{3} \alpha_j e_j$ where $\alpha_j = (\exp e_1^T \cdot e_j) / \sum_{k=1}^{3} \exp(e_1^T \cdot e_k)$. Here e_1 acts like the query; e_1, e_2, e_3 are both keys and values. An answer that includes the normalization term is also valid: $\alpha_j = (\exp e_1^T \cdot e_j/\sqrt{d}) / \sum_{k=1}^{3} \exp(e_1^T \cdot e_k/\sqrt{d})$; a matrix form answer is valid if understanding is demonstrated: $l_1 = softmax(QK^T/\sqrt{d})V$ where Q is e_1, K and V are both a stacking of e_1, e_2, e_3.

Q6. Large Language Models *10 points*

1. What training strategy is used to make models less harmful and why? *(3 points)*

   Preference training (reinforcement learning from human feedback), since we can much more easily compute an overall harmfulness score for a response than including a harmfulness signal into next-token prediction.

2. The design of today's transformer large language models is called *decoder-only*. Explain the origin of this term. *(3 points)*

   Transformers were original machine translation models with an encoder that embedded the tokens of the input sentence and a decoder that predicted output tokens, one-by-one. Large language model remove the encoder, leaving the next-token prediction decoder.

3. Why is Low-Rank Adaptation more efficient? Compute the savings in number of weights that are trained for a 1000-dimensional matrix when using a LoRA adapter of rank 20. *(4 points)*

   Withoughy 1000x1000=1 million weights, LoRA 2x1000x20=40,000, so only 4% — or savings of 96%

Q7. Neural and Symbolic Language Processing *10 points*

1. **Prompt sensitivity.** An LLM gives the wrong answer to a question. You rephrase the question without adding any new information, and it now gives the correct answer. What does this tell us about how LLMs process knowledge? *(3 points)*

   The rephrased question matches more closely the relevant knowledge was expressed in the training data. While the LLMs abstracts knowledge to some degree, it is still sensitive to surface-level realizations.

2. **Combining neural and symbolic methods.** Consider a task in which an LLM must produce an answer that obeys a strict set of logical constraints. What advantage could a symbolic solver provide if it is combined with the LLM? *(3 points)*

   The problem could be solved either by the LLM or the symbolic solver. The LLM could format the question to fit the solver. The symbolic solver could check the formal validity of the answer. The solver may have an easier time proving the answer than coming up with the answer. Any mistake flagged by the solver could be fed back to the LLM.

3. **Agentic system design.** A task can be implemented either as a LangFlow-style fixed workflow or as a ReAct-style agent that decides which action to take next based on intermediate results. Give one type of task for which each approach would be preferable, and briefly explain why. *(4 points)*

   Langflow is better for problems that have a clear fixed process path, such as for translation (identify problems, information gathering, draft, review, refine), while reactive systems are better for open ended problems such as coding.

Extra Space