CAP 6640 – Natural Language Processing **Spring 2025**
Homework #6 Due date: April 4, 2025

Problem 1:

The softmax temperature is a parameter used to adjust how a language model selects words during text generation. It changes the probability distribution of possible next words, allowing us to tune the balance between randomness and predictability in the output. Given a vector of logits z_i for each possible token, the temperature scaled softmax is defined as:

$$P(w_i) = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}}$$

where T is the temperature, z_i is the pre-softmax score for token i, and $P(w_i)$ is the resulting probability for token i.

When the temperature is close to 0.7 or below, the model becomes more confident and focused. It strongly favors high-probability words, resulting in precise, factual, and consistent outputs. This setting is well-suited for tasks where accuracy and coherence are important, such as summarization, and QA.

In contrast, a higher temperature which can be selected as 1.2 or more makes the output distribution flatter, giving more weight to less likely words. This leads to more creative, varied, and sometimes unexpected text, which is ideal for poetry, storytelling, or brainstorming tasks. However, very high temperatures can also introduce nonsense or off-topic responses.

Problem 2:

Sampling-based decoding is a decoding strategy used in language generation where the model randomly selects the next word based on its probability, rather than always choosing the most likely one. This makes it especially suitable for creative tasks like storytelling, where unpredictability and variation are essential for producing engaging content.

Sampling-based decoding offers several benefits for open-ended text generation tasks like storytelling. By introducing randomness, sampling helps the model avoid repetitive or generic responses, creating outputs that feel more human-like and less mechanical. Furthermore, it allows for exploration of diverse narrative paths by occasionally selecting less frequent but reasonable words. Lastly, tecniques like top-n sampling where the model samples only from the top n probable words, or top-p sampling, where the model samples from the smallest set of words whose cumulative probability exceeds p, allow users to control the level of creativity and coherence of the model in the generated text.

However, sampling-based decoding also has limitations. Random selection increases the chance of the model producing disconnected or nonsensical sentences. Moreover, the model can produce very different outputs for the same set of inputs, making it harder to reproduce specific responses or maintain a consistent storytelling voice. Finally, sampling can sometimes deviate from the intended direction or tone of the story.

Problem 3:

BLEU, ROUGE, and BERTScore are three commonly used metrics for evaluating NLG, particularly in tasks like machine translation, summarization, and open-ended text generation. Each has its strengths and limitations, especially when applied to creative outputs like dialogue or storytelling.

BLEU measures the degree to which the generated text matches the reference through n-gram precision. It is particularly useful for tasks with a small set of valid outputs, such as machine translation, where phrase-level correctness is critical. However, BLEU does not account for meaning. If the model uses synonyms or rephrases the sentence, BLEU may penalize it, even if the alternative is equally valid.

ROUGE is often used in summarization tasks, and focuses on more recall than BLEU. It evaluates how much of the reference content is captured in the generated summary by comparing overlapping unigrams, bigrams, or longest common subsequences. While effective for extractive summarization,

ROUGE is less suited for abstractive summaries that creatively rephrase or reorganize content. Like BLEU, it fails to consider semantic similarity or context.

BERTScore offers a more modern and semantically aware evaluation method. It compares contextualized word embeddings between the generated and reference texts, allowing it to assess meaning alignment rather than simple word overlap. This makes it particularly effective for evaluating tasks with more expressive variability, such as dialogue generation, storytelling, or creative writing, where many phrasings may be equally valid.

In open-ended text generation like storytelling, there is hardly ever a single true, correct, and acceptable answer. Word-overlap metrics such as BLEU and ROUGE penalize valid outputs simply because they differ in wording from the reference. These tasks require understanding the intention, tone, and context of a response, which BLEU and ROUGE are not designed to measure. For example, a story may use a different reorder events creatively or analogy of wording while still conveying the same narrative. In such cases, BERTScore performs better because it can recognize that the generated output shares semantic alignment with the reference even if the wording differs.

Problem 4:

Hybrid QA architectures are designed to combine two core approaches in QA systems, which are retrieval-based and generative models. Retrieval part of the hybrid QA locates relevant content from a source while generative part is responsible for generating natural language responses. This integration allows QA systems to be both factual-based and linguistically fluent.

The process of QA begins with a retriever that identifies relevant documents from a large corpus to ensure that the model works with content that is presented. Then, a generative language model such as GPT processes the retrieved information along with dialogue history to produce coherent and conversational answer. Instead of copying a sentence from the source, the generator may involve paraphrasing, summarizing, or reasoning across multiple pieces of information.

An example of a hybrid QA system is Stanford's CQA framework. In this system, users ask follow-up questions in a multi-turn dialogue in which each question is context-dependent. The system needs to understand the flow of conversation from previous questions, retrieve the relevant passage, and generate a response that reflects both the context and current question even if the answer is not explicitly stated in the retrieved part.

Problem 5:

Coreference resolution is the process of determining when two or more expressions in a text refer to the same underlying entity. These expressions can include names, pronouns, or noun phrases. For instance, in the sentence "Mike went to the office of Jessica because he had a meeting with her.", the system must recognize that "he" refers to "Mike" and "her" refers to "Jessica". Successfully linking these references is what coreference resolution accomplishes.

Coreference resolution involves three important steps. The first step is mention detection, where the system identifies all possible mentions in the text. The second step is feature extraction, where for each mention pair, the system extracts lexical, syntactic, and semantic features. The third step is clustering, where the system predicts which mentions refer to the same entity and groups them into clusters. Following step includes using coreference resolution models such as mention pair model, entity-level model, or end-to-end neural model.

Coreference resolution is essential for accurately understanding and processing text across multiple sentences. It links information that appears in different parts of the text, helping the model form a clear and consistent understanding of the overall context. This is particularly important for full-text understanding tasks like question QA and summarization.

When answering questions in multi-sentence contexts, a system needs to trace entities across a long text. Assume that we have the following QA setup:

> **Context:** Josh's actor friends notified the police after they hadn't heard from him for 3 days. Later that evening, police went to his home and found him dead due to heart attack.

Q: Which actor has passed away recently due to heart attack?
A: Josh (requires resolution of "his"/"him")

Like in the above mentioned example, if a question refers to "he" or "she," the model must know who that refers to. Without coreference resolution, the model might miss the connection and return an incorrect or incomplete answer.

In summarization, coreference resolution is also important. Effective summarization depends on understanding which entities are being discussed across the document. By resolving pronouns and other referring expressions, a summarizing tool can produce clearer, more connected summaries. Otherwise, it might introduce ambiguity or misrepresent who did what.

Problem 6:

Coreference resolution has transitioned from rule-based systems to data-driven neural architectures. While traditional systems rely on explicit rules and hand-engineered features, end-to-end neural models aim to learn everything from the data itself.

Traditional coreference resolution systems often involve two main steps, which are identifing mentions in the text and then evaluating the relationships between them using rules or feature-based models. These models often use syntactic clues like gender and number agreement or surface features such as word overlap. Mention-pair models, classify every potential pair independently. While these systems are easy to interpret and implement, they treat each decision in isolation, which can result in contradictions across the text. Moreover, their reliance on hand-designed features means they may not generalize well beyond training data. To address consistency, entity-level models were developed. Instead of only comparing mention pairs, they cluster all mentions of the same entity. These models evaluate global coherence but still depend on feature engineering and iterative rule-based clustering strategies.

End-to-end neural models on the other hand represent a more modern approach. Rather than breaking the task into separate stages, they jointly learn mention detection and coreference resolution within a single architecture. These models use contextualized embeddings to represent spans in their surrounding context. Span scoring predicts the valid mentions. Coreference scoring then determines which spans belong to the same entity. Lastly, the final coreference clusters are formed using neural decision models. Because everything is learned from data, there is no need for manually created features or linguistic rules. This allows end-to-end models to capture complex relationships and dependencies between mentions across the entire document, leading to more accurate and coherent coreference resolution. However, they are computationally expensive and require large amounts of annotated data for training.

Problem 7:

The DecaNLP framework introduces a way to handle multiple NLP tasks using a single model without designing separate components or parameters for each task. Instead of building a separate module for every objective, DecaNLP reframes all tasks into a QA format, enabling a single architecture to process them in a unified way.

Unlike traditional multi-task learning systems that rely on custom heads or task IDs to switch between different objectives, DecaNLP uses a uniform input-output structure. Every task is rephrased as a form of question answering. Instead of giving the model a label or ID for a task, it receives a context and a question, and the model generates an answer. There are no hardcoded parameters specific to each task, and no manually added indicators to tell the model what it should do. By expressing every task in QA style, the model figures out the task based on the phrasing of the question.

Another strength of this framework is its ability to generalize to unseen tasks by rephrasing them as questions. Since the model learns how to follow natural language prompts, it can be extended to new tasks at test time by simply providing a new kind of question without the necessity of fine-tuning and retraining. This approach leads to zero-shot learning across NLP tasks.

Problem 8: Describe anti-curriculum learning. How does starting with harder tasks improve generalization in multitask training?

Problem 9: What is the MQAN architecture? List two mechanisms it uses to handle multiple tasks as QA problems.

Problem 10: What is the difference between a general language model and a conditional language model? Provide examples of tasks suitable for each.