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

Problem 1: Describe how bidirectional RNNs work, including a diagram and mathematical formulation, and explain the problems they address.

Problem 2:

1. **Convolution** is the mathematical process of applying a kernel to an input to extract semantic or syntactic features from the given text. It involves element-wise multiplication followed by summation. In the context of NLP, convolutions are applied to word embeddings to detect n-gram patterns. For example, consider the sentence: "The concert was awesome.". First of all, we embed words into vectors:

$$\text{"The"} \rightarrow [0.1, 0.2, 0.3]$$
$$\text{"concert"} \rightarrow [0.2, 0.3, 0.4]$$
$$\text{"was"} \rightarrow [0.3, 0.4, 0.5]$$
$$\text{"awesome"} \rightarrow [0.4, 0.5, 0.6]$$

Let's assume that we have a kernel of size 2, which is applied to the sentence. The kernel is represented as:

$$\begin{bmatrix} 0.1 & 0.2 & 0.3 \\ 0.3 & 0.4 & 0.5 \end{bmatrix}$$

The convolution operation is performed as follows:

$$[0.1, 0.2, 0.3] \cdot [0.1, 0.2, 0.3] + [0.2, 0.3, 0.4] \cdot [0.3, 0.4, 0.5] = 0.1 \times 0.1 + 0.2 \times 0.2 + 0.3 \times 0.3 + 0.2 \times 0.3 + 0.3 \times 0.4 + 0.4 \times 0.5 = 0.01 + 0.04 + 0.09 + 0.06 + 0.12 + 0.2 = 0.52$$

2. **Padding** involves adding extra zeros or other relevant values around the input to control the output size and preserve sequence length in text. For example, without padding the sentence "The concert was awesome.", applying a kernel size of 3 reduces output length from 4 words to 2 words. With zero padding, we maintain the original length of the sentence as follows:

$$\text{"}\phi\text{"} \rightarrow [0.0, 0.0, 0.0]$$
$$\text{"The"} \rightarrow [0.1, 0.2, 0.3]$$
$$\text{"concert"} \rightarrow [0.2, 0.3, 0.4]$$
$$\text{"was"} \rightarrow [0.3, 0.4, 0.5]$$
$$\text{"awesome"} \rightarrow [0.4, 0.5, 0.6]$$
$$\text{"}\phi\text{"} \rightarrow [0.0, 0.0, 0.0]$$

3. **Channels** are the number of feature maps generated by applying multiple kernels to the input. Each channel represents a different feature extracted from the text. This allows to combine different linguistic features. For example, if we apply 2 kernels to the sentence "The concert was awesome.", we get 2 channels, each representing a different feature. Assume that the other kernel is as follows:

$$\begin{bmatrix} 0.2 & 0.1 & 0.3 \\ 0.4 & 0.3 & 0.5 \end{bmatrix}$$

For the bigram "The concert", after applying the two kernels, we get two channels for representation as follows:

$$\text{"The", "concert"} \rightarrow [0.52, 0.50]$$

4. **Max Pooling** is a downsampling operation that selects the maximum value from a set of values in a given window while preserving key information. This operation helps to capture the most important features. For example, assume that we have the following feature map:

$$
\begin{gathered}
\text{"}\phi\text{", "The", "concert"} \rightarrow [0.52, 0.50] \\
\text{"The", "concert", "was"} \rightarrow [0.48, 0.56] \\
\text{"concert", "was", "awesome"} \rightarrow [0.64, 0.42] \\
\text{"was", "awesome", "}\phi\text{"} \rightarrow [0.40, 0.38]
\end{gathered}
$$

After applying max pooling for each channel seperately, we get the following representation:

$$\text{max p} \rightarrow [0.64, 0.56]$$

5. **Average Over Time** is a pooling operation that calculates the average of all feature activations across the sequence to capture the overall context. This operation helps to reduce the dimensionality of the feature map. Furthermore, averaging them might provide a sentence-level embedding. For example, assume that we have the following feature map:

$$
\begin{gathered}
\text{"}\phi\text{", "The", "concert"} \rightarrow [0.52, 0.50] \\
\text{"The", "concert", "was"} \rightarrow [0.48, 0.56] \\
\text{"concert", "was", "awesome"} \rightarrow [0.64, 0.42] \\
\text{"was", "awesome", "}\phi\text{"} \rightarrow [0.40, 0.38]
\end{gathered}
$$

After applying average over time for each channel seperately, we get the following representation:

$$\text{avg} \rightarrow [0.51, 0.465]$$

6. **Striding** is the process of moving the kernel by a certain number of steps across the input duing convolution to reduce the output size. This helps to control computational efficiency and feature map size. When we select stride as 1, the kernel moves one step at a time which outputs a dense feature map. For example, assume that we get the following feature map when we apply kernel size of 2 and stride of 1 to the sentence "The concert was awesome.":

$$
\begin{gathered}
\text{"}\phi\text{", "The", "concert"} \rightarrow [0.52, 0.50] \\
\text{"The", "concert", "was"} \rightarrow [0.48, 0.56] \\
\text{"concert", "was", "awesome"} \rightarrow [0.64, 0.42] \\
\text{"was", "awesome", "}\phi\text{"} \rightarrow [0.40, 0.38]
\end{gathered}
$$

For stride of 2 with the same kernel size, the feature map would be as follows:

$$
\begin{gathered}
\text{"}\phi\text{", "The", "concert"} \rightarrow [0.52, 0.50] \\
\text{"concert", "was", "awesome"} \rightarrow [0.64, 0.42]
\end{gathered}
$$

7. **K-Max Pooling** selects the top k largest activations values from the feature map instead of a fixed window. This retains significant features while preserving the order of the sequence. For example, assume that we have the following feature map:

$$
\begin{gathered}
\text{"}\phi\text{", "The", "concert"} \rightarrow [0.52, 0.50] \\
\text{"The", "concert", "was"} \rightarrow [0.48, 0.56] \\
\text{"concert", "was", "awesome"} \rightarrow [0.64, 0.42] \\
\text{"was", "awesome", "}\phi\text{"} \rightarrow [0.40, 0.38]
\end{gathered}
$$

After applying k-max pooling for each channel seperately with k=2, we get the following representation:

$$\text{k-max p} \rightarrow \begin{bmatrix} 0.52 & 0.50 \\ 0.64 & 0.56 \end{bmatrix}$$

Problem 3: Discuss the operation of multi-layer RNNs, highlighting the advantages of using multiple layers and identifying four key challenges associated with them.

Problem 4: Explain the encoder-decoder architecture, detailing its working mechanism and providing three real-world applications.

Problem 5: Explain how CNNs are applied to text understanding tasks, such as sentiment analysis.

Problem 6: Describe how neural machine translation operates and compare it to alternative approaches.

Problem 7: Identify three limitations of RNNs that CNNs overcome.