Tutorial 7: Subword Models

1. Modify the model of Question 1 of Tutorial 4 to use character embeddings of the method Character-based neural language models[1]
    a. Convert word to character indices
        i. Assume we have a pre-defined 'vocabulary' of characters (e.g. all lowercase letters, uppercase letters, numbers and punctuations).
        ii. By looking up the index of each character, we can represent the length-*l* word *x* as a vector of integers: $\boldsymbol{x} = [c_1, c_2, \cdots, c_l] \in \mathbb{Z}^l$
            1. where each c_i is an integer index into the character vocabulary
    b. Padding and embedding lookup
        i. Using a special <PAD> 'character', we pad (or truncate) every word so that it has length m_{word} (pre-defined hyper parameter, representing maximum word length):
            1. $\boldsymbol{x}_{padded} = [c_1, c_2, \cdots, c_{m_{word}}] \in \mathbb{Z}^{m_{word}}$
        ii. For each of these characters c_i, we look up a dense character embedding (which has shape e_{char}; e_{char} =50). This yields a tensor $\boldsymbol{x}_{emb}$:
            1. $\boldsymbol{x}_{emb} = \text{CharEmbedding}(\boldsymbol{x}_{padded}) \in \mathbb{R}^{m_{word} \times e_{char}}$
        iii. We will reshape $\boldsymbol{x}_{emb}$ to obtain $\boldsymbol{x}_{reshaped} \in \mathbb{R}^{e_{char} \times m_{word}}$ before feeding into the convolutional network of the Convolutional network below.
            1. Necessary because PyTorch Conv1D performs only on the last dimension of the input
    c. Convolutional network
        i. To combine these character embeddings, we'll use 1-dimensional convolutions.
        ii. The convolutional layer has two hyper-parameters:
            1. the kernel size *k* (also called window size; *k*=5), which dictates the size of the window used to compute features, and
            2. the number of filters *f* (also called number of output features or number of output channels)
            3. Assume no padding is applied and the stride is 1
        iii. The convolutional layer has a weight matrix $\boldsymbol{W} \in \mathbb{R}^{f \times e_{char} \times k}$ and a bias vector $\boldsymbol{b} \in \mathbb{R}^f$.
        iv. To compute the i^{th} output feature (where $i \in \{1, \ldots, f\}$) for the t^{th} window of the input, the convolution operation is performed between the input window $(\boldsymbol{x}_{reshaped})_{[:,t:t+k-1]} \in \mathbb{R}^{e_{char} \times k}$ and the weights $\boldsymbol{W}_{[i,:,:]} \in \mathbb{R}^{e_{char} \times k}$, and the bias term $b_i \in R$ is added:
            1. $(\boldsymbol{x}_{conv})_{i,t} = \text{sum}\left(\boldsymbol{W}_{[i,:,:]} \odot (\boldsymbol{x}_{reshaped})_{[:,t:t+k-1]}\right) + b_i \in \mathbb{R}$
            2. where $\odot$ is element-wise multiplication of two matrices with the same shape and "sum" is the sum of all the elements in the matrix. This operation is performed for every feature *i* and every window *t*, where $t \in \{1, \cdots, m_{word} - k + 1\}$. Overall this produces output $\boldsymbol{x}_{conv}$:
            3. $\boldsymbol{x}_{conv} = \text{Conv1D}(\boldsymbol{x}_{reshaped}) \in \mathbb{R}^{f \times (m_{word} - k + 1)}$

[1] Originally from Assignment of https://web.stanford.edu/class/cs224n/

v. For our application, we'll set *f* to be equal to e_{word}, the size of the final word embedding for word *x* (the rightmost vector in Figure 1). Therefore,
   1. $\boldsymbol{x}_{conv} \in \mathbb{R}^{e_{word} \times (m_{word} - k + 1)}$
vi. Finally, we apply the ReLU function to $\boldsymbol{x}_{conv}$, then use max-pooling to reduce this to a single vector $\boldsymbol{x}_{conv-out} \in \mathbb{R}^{e_{word}}$, which is the final output of the Convolutional Network:
   1. $\boldsymbol{x}_{conv-out} = \text{MaxPool}\big(\text{ReLU}(\boldsymbol{x}_{conv})\big) \in \mathbb{R}^{e_{word}}$
vii. Here, MaxPool simply takes the maximum across the second dimension.
viii. Given a matrix $M \in \mathbb{R}^{a \times b}$, then $\text{MaxPool}(M) \in \mathbb{R}^{a}$ with $\text{MaxPool}(M)_i = \max_{1 \le j \le b} M_{ij}$ for $i \in \{1, \cdots, a\}$
ix. This output is fed into the Convolutional Network of Question 1.

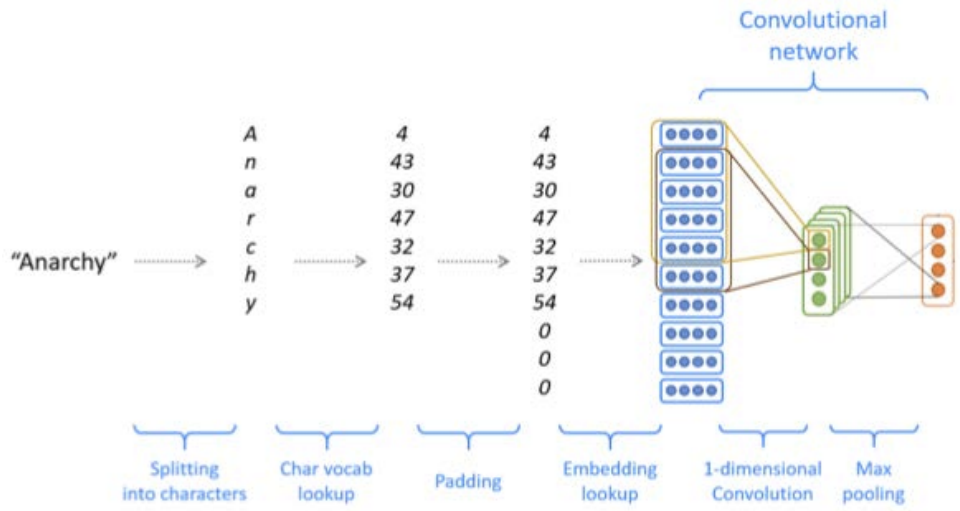

Figure 1. Character-based convolutional encoder, which ultimately produces a word embedding of length eword