A Quick Summary: Character-Aware Neural Language Models

Original Paper: https://arxiv.org/pdf/1508.06615.pdf

21st Feb 2019

1 Ideas:

(a) Words are often represented as word-embeddings, through mechanisms such as Word2vec (skipgram, CBOW). In this paper, the authors embed each word differently, through using a character level mechanism (Word2vec is a word-level mechanism).

(b) After encoding words with their character-level representations, the authors used a convolution layer with a maxpool layer to extract out a vector representation $y \in \mathbb{R}^k$ for the word.

(c) The vector y is then passed through a highway network before being passed into the LSTM.

2 Explanations:

(a) Instead of having a $\mathbb{R}^n$ vector that encodes each word in a corpus, this model encodes each of the characters (A-Z, 0-9, etc.) into a vector $\mathbb{R}^n$, so that now, each word is essentially represented as a matrix of dimension $\mathbb{R}^{n \times m}$, where m is the number of letters in the word.

(b) The convolution can be interpreted as utilizing some form of local structure to make sense of each word. The way that the convolution and maxpool ensures that the dimension of the resultant vector y is independent of the length of each word. I.e. The dimensions of the vector representation is the same for all words (refer to image on the next page).

(c) The highway network allows some of the vector y to be directly carried to the next layer.

3 Model:

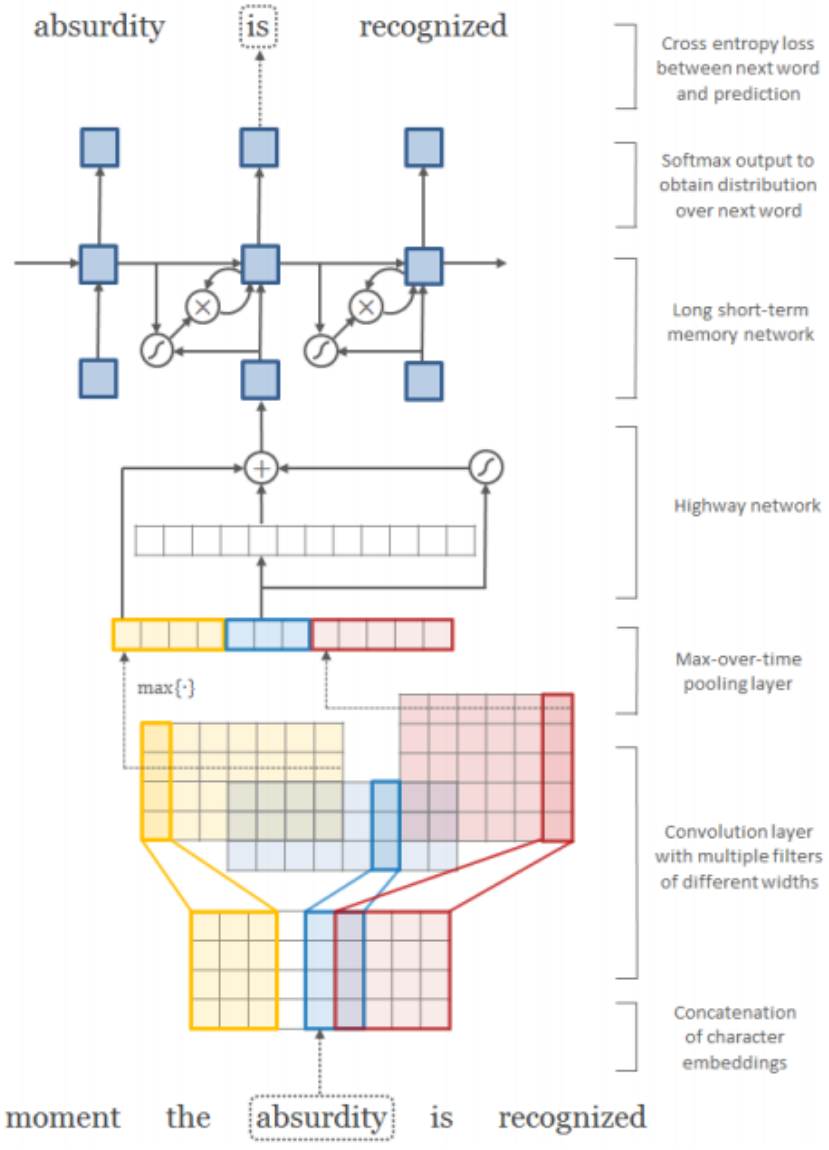

Figure 1: Illustration of the model

4 Results:

Character-level model outperforms word-level models, despite being smaller and less memory intensive.

5 Notes:

Maybe word embeddings are unnecessary.