1. **Encoder-Decoder Models and Capacity**

   (a) The architecture will not be performant on long sequences. One reason is error signal needs to travel long distances to the encoder such that it learns the underlying structure of the language model. This introduces vanishing gradient problem during backpropagation. Another reason would be the fact that the decoder only sees the final hidden encoding h_T^{enc} between encoder and decoder, each character of the input have dimishing contribution to the h_T^{enc} for longer and longer sentences.

   (b) Outputs

$$
\begin{aligned}
seat &\rightarrow eatsay && \text{(short, starts with consonant)}\\
fact &\rightarrow aftmay &&\\
how &\rightarrow owhay &&\\
able &\rightarrow ableway && \text{(short, starts with vowel)}\\
eight &\rightarrow eitttway &&\\
aches &\rightarrow amfesway &&\\
shops &\rightarrow omsshay && \text{(short, starts with consonant pair)}\\
shew &\rightarrow emthay &&\\
shades &\rightarrow afessway &&\\
misconstruction &\rightarrow ismosairorotesmay && \text{(long, starts with consonant)}\\
prepossessing &\rightarrow epessiiensdpray &&\\
reasonableness &\rightarrow efseseneeoeedway &&\\
instantaneously &\rightarrow inlaaaaaa - eayylsay && \text{(long, starts with vowel)}\\
insurmountable &\rightarrow insaa - icaa - eesway &&\\
acknowledgments &\rightarrow amkmesinoineendway &&
\end{aligned}
$$

   The result is usually not exactly correct. We observe that the model is able to learn to append $-ay$ to the end of words, preserve vowel characters at start of word and remove consonant characters at start of word for both short and long words. However, the model cannot remember long words as well as short words.

2. **Teacher-Forcing**

   (a) During training, we utilize the actual output from the training set at current time step as inputs to the decoder in the next time step. However during testing, we do not know beforehand the groundtruth token from the previous time step to feed into the current step. There is a discrepancy between training and testing which may yield error that accumulate quickly along the generated sequence.

(b) We can try to mitigate such discrepancy between training and testing by exposing the model, at training time, to the scenario it will experience at testing time, specifically when we feed the output of model at previous time step as input to the model at current time step. The paper proposed to, at each time step during training time, randomly pick either the groundtruth token or the token generated by the model as input to the decoder, on the account that the such sampling probability decays as training progresses to facilitate faster convergence.

3. **Gated Recurrent Unit (GRU)**

4. **Implementing Attention**

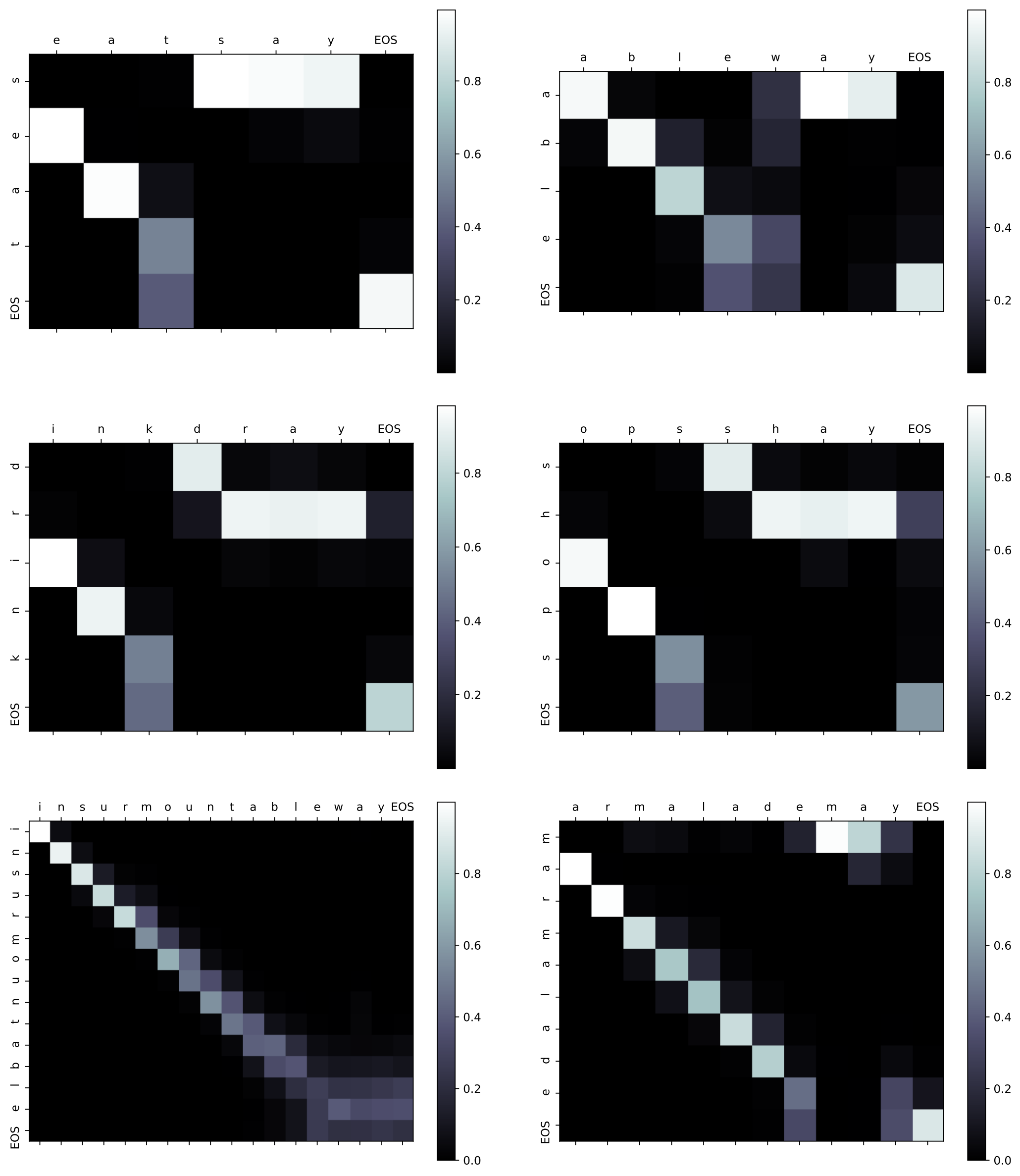
5. **Attention Visualization**

(a) Outputs

$seat$	$\rightarrow$	$eatsay$	(short, starts with 1 consonant)
$fact$	$\rightarrow$	$actfay$	
how	$\rightarrow$	$owhay$	
$drink$	$\rightarrow$	$inkdray$	(short, starts with 2 consonant)
$treat$	$\rightarrow$	$eattray$	
$knees$	$\rightarrow$	$eesknay$	
$shops$	$\rightarrow$	$opsshay$	
$shew$	$\rightarrow$	$ewshay$	
$shades$	$\rightarrow$	$adesshay$	
$able$	$\rightarrow$	$ableway$	(short + starts with vowel)
$eight$	$\rightarrow$	$eightway$	
$aches$	$\rightarrow$	$achesway$	
$misconstruction$	$\rightarrow$	$isconssrccteteway$	(long + starts with consonant)
$prepossessing$	$\rightarrow$	$epossesssnwpay$	
$reasonableness$	$\rightarrow$	$easonableneway$	
$instantaneously$	$\rightarrow$	$instantaaneualway$	(long + starts with vowel)
$insurmountable$	$\rightarrow$	$insurmountableway$	
$acknowledgments$	$\rightarrow$	$acknowledgeenway$	
$aardvark$	$\rightarrow$	$aardvarkwa$	(unusual/rare letter combination)
$marmalade$	$\rightarrow$	$armalademay$	
$apothecary$	$\rightarrow$	$apothecaryway$	
$turnpike-road$	$\rightarrow$	$urnpiksstaa-adabay$	(compound word with a dash)
$bowling-green$	$\rightarrow$	$owblnybayy-eeay$	
$toothpick-cases$	$\rightarrow$	$oothktcktay$	

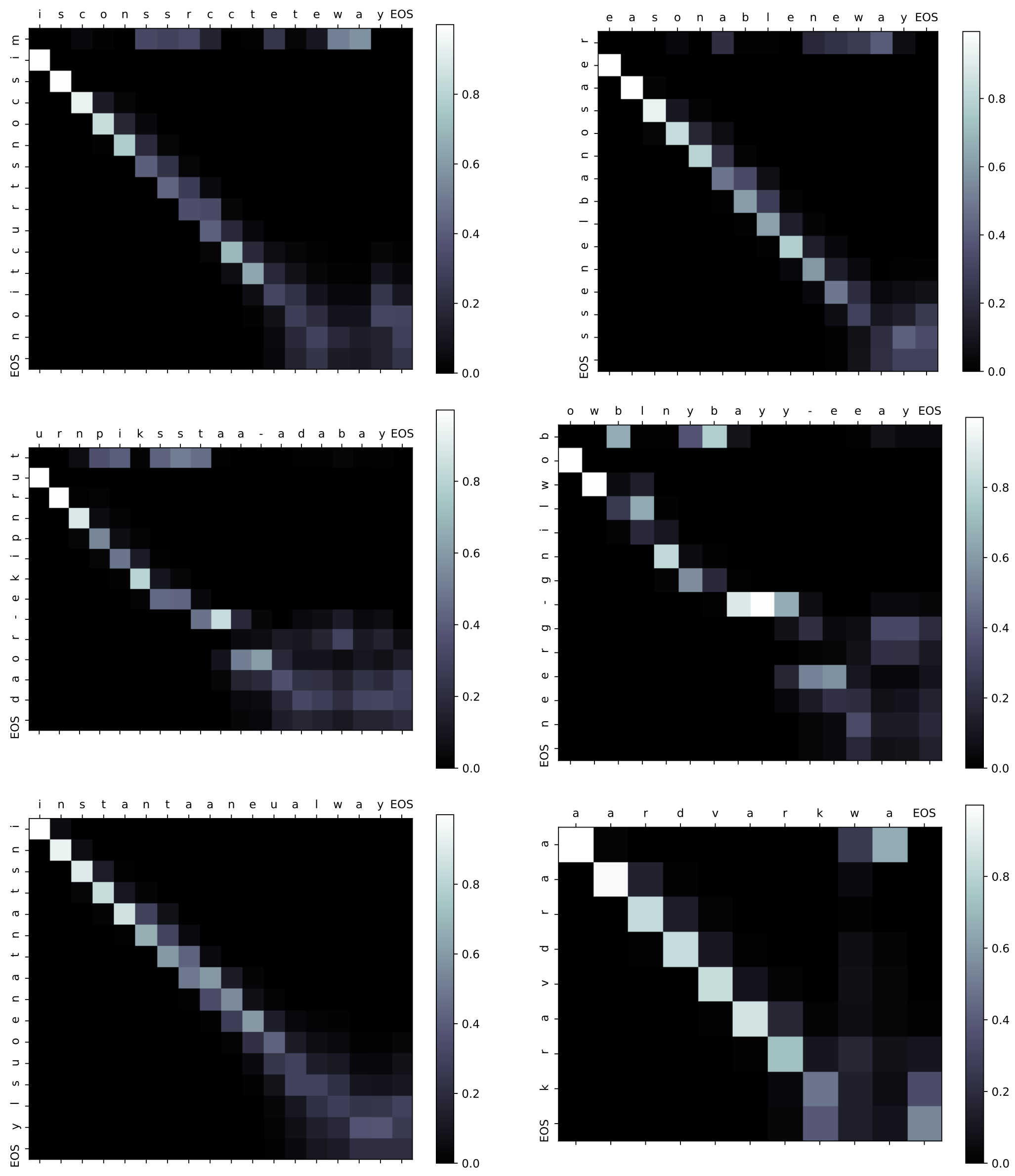
Overall, the attention-based model does a much better job at remembering words, a drawback which we noted in the non-attention-based model. The model works really well on short words (either starts with 1 consonant, consonant pairs, or vowels). The model suffers from inaccuracy for longer words and compound words with a dash, although the it is able to retain most characters from the original word, an improvement from non-attention-based model.

(b) Success Cases



From the attention map, we note that the model generated the correct output as it is able to pick up the patterns of the initial character(s) of the word (vowel, 1 or 2 consonants) and learns the correct identity mapping for the subsequent characters until the very end. For the last few characters, the model is able to appends the appropriate suffices for the different cases by referencing the first few characters of input word as shown in the attention map. However, we notice that `insurmountable`, one of the only few correctly generated long word, does not refer back to the inital characters when generating the last few characters.

(c) Failure Cases



i. One failure case is when the model prematurely references the initial characters of the input words. We observe that the model often outputs the incorrect character when the attention map for the initial characters and the characters on the diagonals are equally activated, in case of `misconstruction`, `reasonableness`, and `turnpike-road`. This might be because of the tendency of model to generate shorter words, as a result of the idiosyncracy of the training dataset.

ii. Another failure case arises when the input word is long enough such that the model is ambivalent about where in the input word to reference. Take `instantaneously` as an example, for latter characters output from the decoder, the attention is focused on several nearby characters, centered along the diagonals. At one point, the letter `a` is output twice because of such ambivalence. The failure is probably a result of the limitation of the model architecture, as it allows errors to accumulate quickly as the decoder generates the output characters.