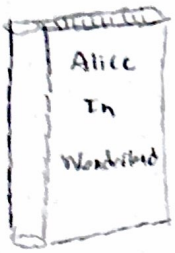
Basic Script Steps:

1) Text Preprocessing and Tokenization



- = "Alice was beggining to get tired" => words = $[w_1, \ldots, w_N]$
  - Here pre-processing takes place:
    - All characters converted to lowercase
    - Removes all punctuation
    - Split into a cleaned list of words

- Create a vocabulary in which we take the 5,000 most frequent $w_i \in$ words and map them to unique integers, the rest are converted to <UNK> and ignored by the script.

  we now have a two way mapping with:

  $\{V = \{w_1, w_2, \ldots w_v\}$ where $v = 5{,}000\}$ "vocabulary set"

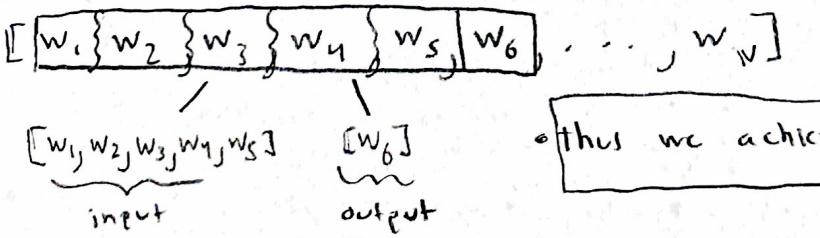
  and $\{$ word_to_ix("alice") = 42, ix_to_word : $\{0, 1, \ldots, V-1\} \rightarrow V\}$ "mapping functions"

  - Now we can encode the full text as a set of integers

2) DataSet Construction

> Note W is transformed, only elements $w_i \in V$ are retained in W and others are transformed to <UNK>.

- Now we want to use our words set W to create input output pairs
  - create sliding window of size S; seq_len = S, since we want to use S words to predict the sixth
  - the window moves across the list one word at a time



- thus we achieve (N − L) input output pairs

3) Defining The Model

- First we turn input sentences $[x_1, x_2, x_3, x_4, x_5] \in \mathbb{Z}^5$, generated from turning words to integers via our vocabulary, into embeddings:

Embedding:

- E is an embedding layer:

$$e_1 = E[x_1],\ e_2 = E[x_2],\ e_3 = E[x_3],\ e_4 = E[x_4],\ e_5 = E[x_5]$$

where each embedding layer has a size d so each word-integer is represented by a d-dimensional vector. ← "embedding dimension"

We then specify a "batch size" of inputs to throw into our model:

Component:	Description:	Shape:
Input	64 sequences of 5 word ID's	(5, 64)
Embedded Input	64 sequences of 5 word vectors	(5, 64, 32)
Target	64 next-word ID's	(64)

← Given to GRU model

- embedding is done so that the model can actually understand the words thrown into it better than simply knowing V_i as the ith most frequent word up to 5000.
- input is sent in batches, after enough batches have been sent for all data to pass through the model, then 1 epoch, or run through the data is completed.

- **GRU Model** "Model using hidden states or propagated memory"
  "Gated Recurrent Unit"

INPUT/EMBEDDING LAYER

$[e_1, e_2, \dots, e_5] \in \mathbb{R}^{5\times 32}$

GRU LAYER/HIDDEN LAYER:

Start with time: $h_0 = \vec{0}$

and for $t = 1$ to 5 do:

(a) Update Gate: $z_t = \sigma(W_z e_t + U_z h_{t-1} + b_z)$
- Controls how much of the previous hidden state to carry forward and how much of the new-candidate state to use.
  - if $z_t \approx 1$, keep new memory
  - if $z_t \approx 0$, stick with old memory

(b) Reset Gate: $r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$
- Controls how much of the previous memory to ignore when calculating the new candidate state
  - if $r_t \approx 0$, forget old hidden state
  - if $r_t \approx 1$, use full hidden state

(c) Candidate Memory: $\tilde{h}_t = \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h)$
- new hidden state

(d) Final step: combine everything $h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$

OUTPUT LAYER:

$\hat{y} = W_{out} \cdot h_5 + b_{out} \in \mathbb{R}^{5000}$

↓

This output is a logit of dimension 5000 and assigns probabilities to each word in our vocabulary based off of the input.

3) Defining the Model continued

- Output Layer gives us $\hat{y} \in \mathbb{R}^{5000}$

and we then apply a softmax: $p_i = \frac{e^{z_i}}{\sum_{j=1}^{V} e^{z_j}}$, $\hat{y} = [z_1, z_2, \ldots, z_V]$ to get the probabilities of each word being next.

- then pick the position in the vocabulary with the highest probability and return the corresponding word as output.