Sequence Models

Week 2

Transfer learning and word embeddings

1. Learn word embeddings from large text corpus. (1-100B words)

   (Or download pre-trained embedding online.)

2. Transfer embedding to new task with smaller training set. (say, 100k words) → 10,000 → 300

3. Optional: Continue to finetune the word embeddings with new data.

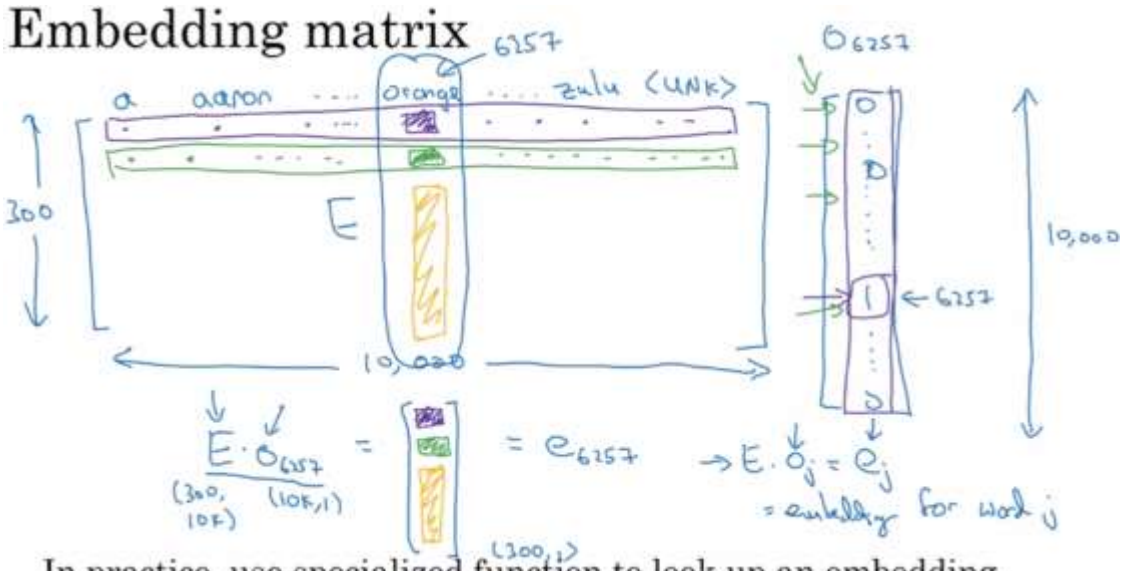
[Mikolov et. al., 2013, Linguistic regularities in continuous space word representations]

Cosine similarity

→ $sim(e_w, e_{king} - e_{man} + e_{woman})$

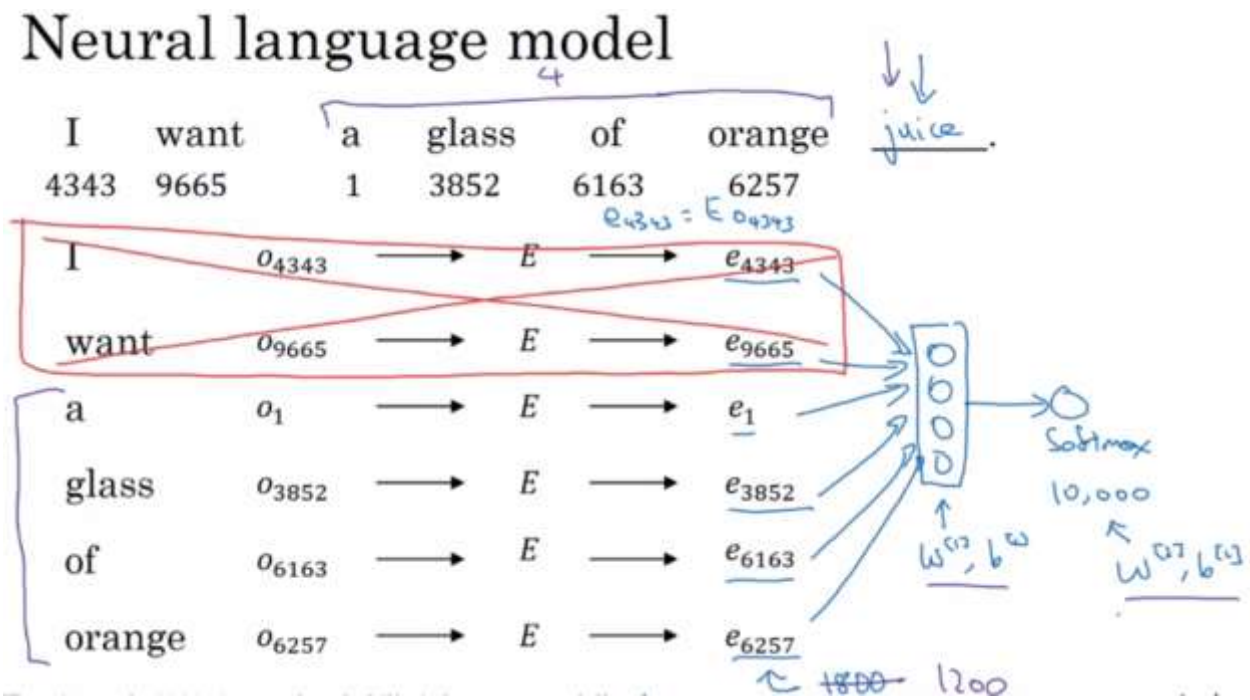
$$Sim(u, v) = \frac{u^T v}{\|u\|_2 \|v\|_2}$$

It is between -1 and 1.



In practice, use specialized function to look up an embedding.

Learning word embedding



Context: Last 4 words.

4 words on left & right

Last 1 word

Nearby 1 word

skip gram

Word2Vec

Model

Vocab size = 10,000k

$x \longrightarrow y$

Context c ("orange") 6257 $\longrightarrow$ Target t ("juice") 4834

$O_c \rightarrow E \rightarrow e_c \rightarrow O \rightarrow \hat{y}$ (Softmax)

$e_c = E O_c$

Softmax: $p(t|c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10,000} e^{\theta_j^T e_c}}$ θ_t = parameter associated with output t

$\mathcal{L}(\hat{y}, y) = -\sum_{i=1}^{10,000} y_i \log \hat{y}_i$

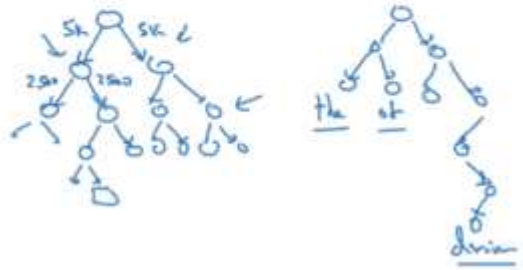
$y = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \leftarrow 4834$

Problems with softmax classification

$$p(t|c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10,000} e^{\theta_j^T e_c}}$$

Hierarchical softmax.

$\log |V|$



About hierarchical softmax:

These distributions are then modelled using the logistic sigmoid σ:

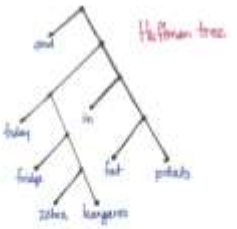
$$P_n(\text{left}|C) = \sigma({\gamma_n}^T \alpha_C),$$

where for each internal node n of the tree, γ_n is a coefficient vector – these are new model parameters that replace the β_w of the softmax. The wonderful thing about this new parameterisation is that the probability of a single outcome $P(w|C)$ only depends upon the γ_n of the internal nodes n that lie on the path from the root to the leaf labelling w. Thus, in the case of a balanced tree, the number of parameters is only logarithmic in the size $|W|$ of the vocabulary!

Mikolov et al. (2013)

The approaches above all use trees that are semantically informed. Mikolov et al, in their 2013 word2vec papers, choose to use a Huffman tree. This minimises the expected path length from root to leaf, thereby minimising the expected number of parameter updates per training task. Here is an example of the Huffman tree constructed from a word frequency distribution:

word	count
fat	3
fridge	2
zebra	1
potato	3
and	14
in	7
today	4
kangaroo	2

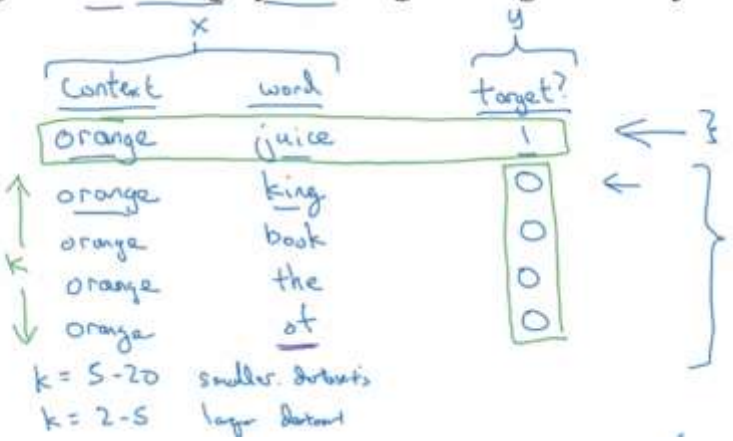


What is interesting about this approach is that the tree is random from a semantic point of view.

[Mikolov et. al., 2013. Distributed representation of words and phrases and their compositionality]

Defining a new learning problem

I want a glass of orange juice to go along with my cereal.

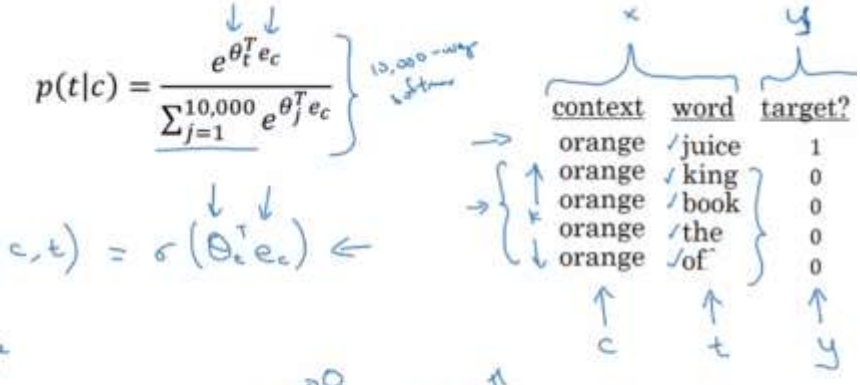


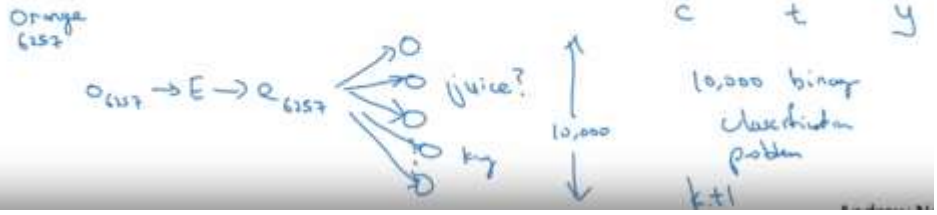
Model

Softmax: $p(t|c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10,000} e^{\theta_j^T e_c}}$

10,000-way softmax

$P(y=1 \mid c,t) = \sigma(\theta_t^T e_c)$





Selecting negative examples

context	word	target?
orange	juice	1
orange	king	0
orange	book	0
orange	the	0
orange	of	0

the, of, and, ...

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=1}^{10{,}000} f(w_j)^{3/4}} \qquad \frac{1}{|V|}$$

GloVe (global vectors for word representation)

I want a glass of orange juice to go along with my cereal.

c, t

X_{ij} = # times i appears in context of j

$X_{ij} = X_{ji}$ ←

Model

Minimize $\sum_{i=1}^{10{,}000} \sum_{j=1}^{10{,}000} f(X_{ij})(\theta_i^T e_j + b_i + b_j' - \log X_{ij})^2$ ←

$\theta_t^T e_c$

weighting term

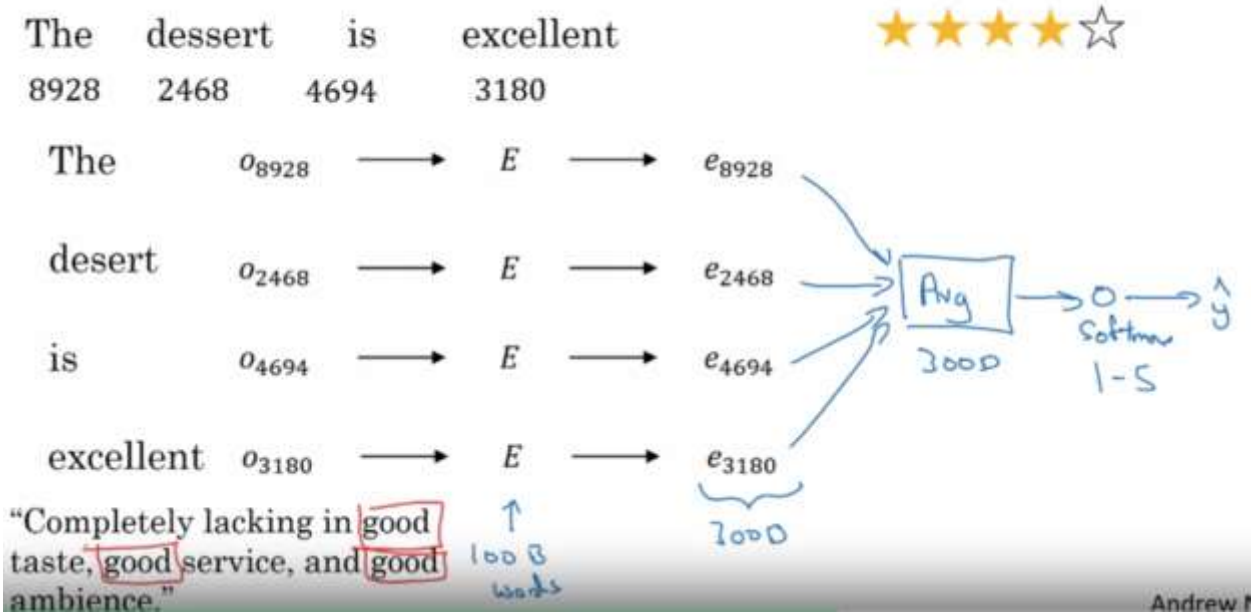
$f(X_{ij}) = 0$ if $X_{ij} = 0$ "$0 \log 0$" = 0

this, it, of, a, ...

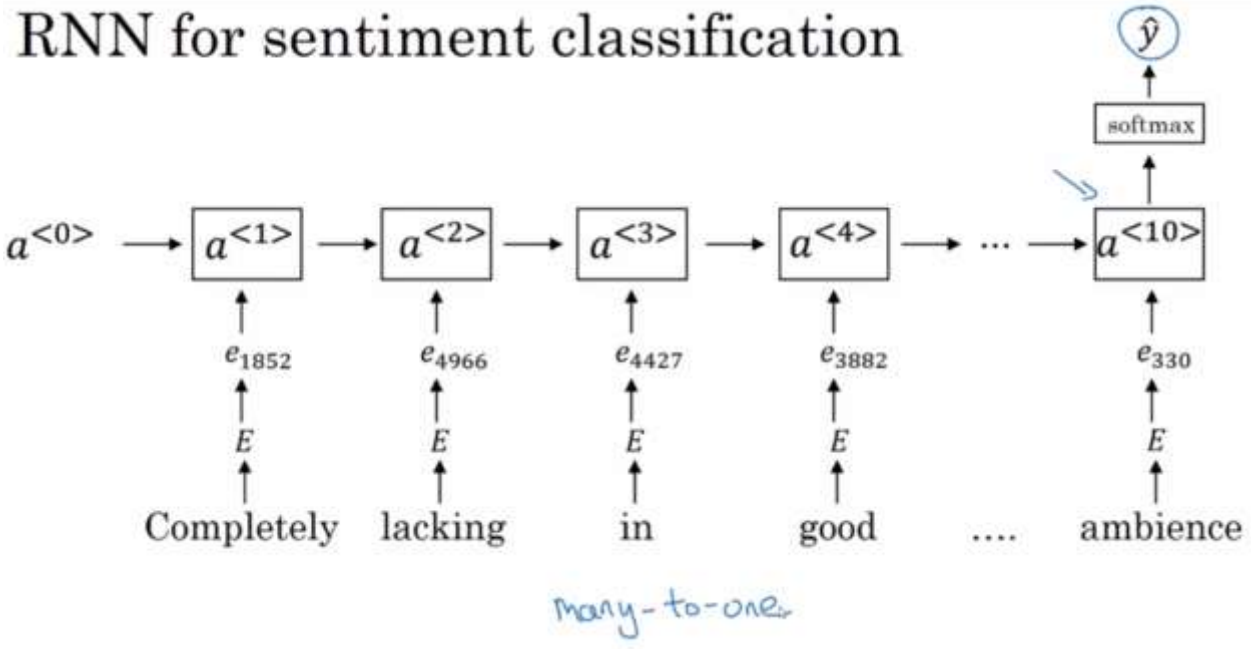
θ_i, e_i are symmetric

$e_w^{(final)} = \frac{e_w + \theta_w}{2}$

Simple sentiment classification model

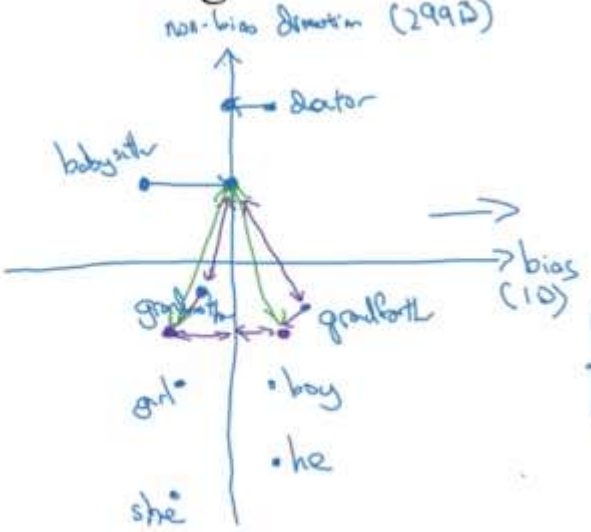


RNN for sentiment classification



Word embeddings can reflect gender, ethnicity, age, sexual orientation, and other biases of the text used to train the model.

Addressing bias in word embeddings



1. Identify bias direction.

$e_{he} - e_{she}$

$e_{male} - e_{female}$

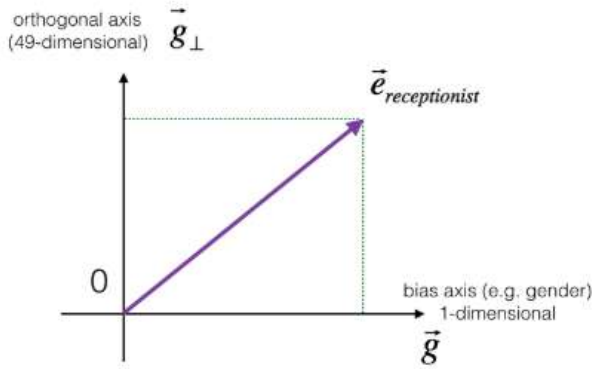
average

2. Neutralize: For every word that is not definitional, project to get rid of bias.

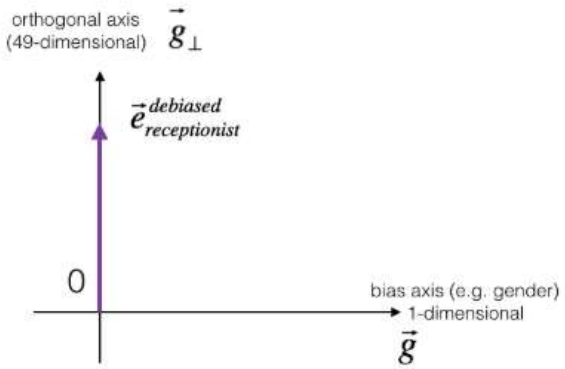
3. Equalize pairs.

grandmother − grandfather

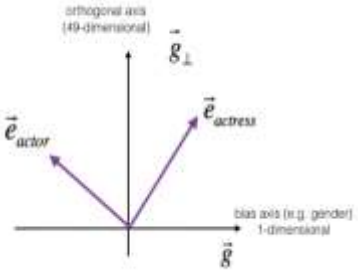
girl boy



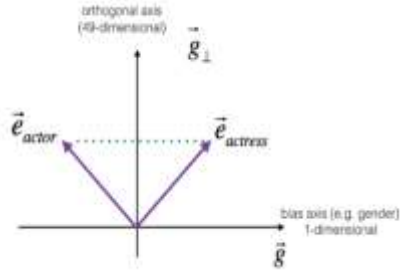
before neutralizing,
eceptionist" is positively correlated with the bias axis



after neutralizing,
debased version, with the component in the direction of the bias axis (g) zeroed out



before equalizing,
"actress" and "actor" differ in many ways beyond the direction of $\vec{g}$



after equalizing,
"actress" and "actor" differ only in the direction of $\vec{g}$, and further are equal in distance from $\vec{g}_\perp$

- Because *adore* has a similar embedding as *love*, the algorithm has generalized correctly even to a word it has never seen before.
- Words such as *heart*, *dear*, *beloved* or *adore* have embedding vectors similar to *love*.

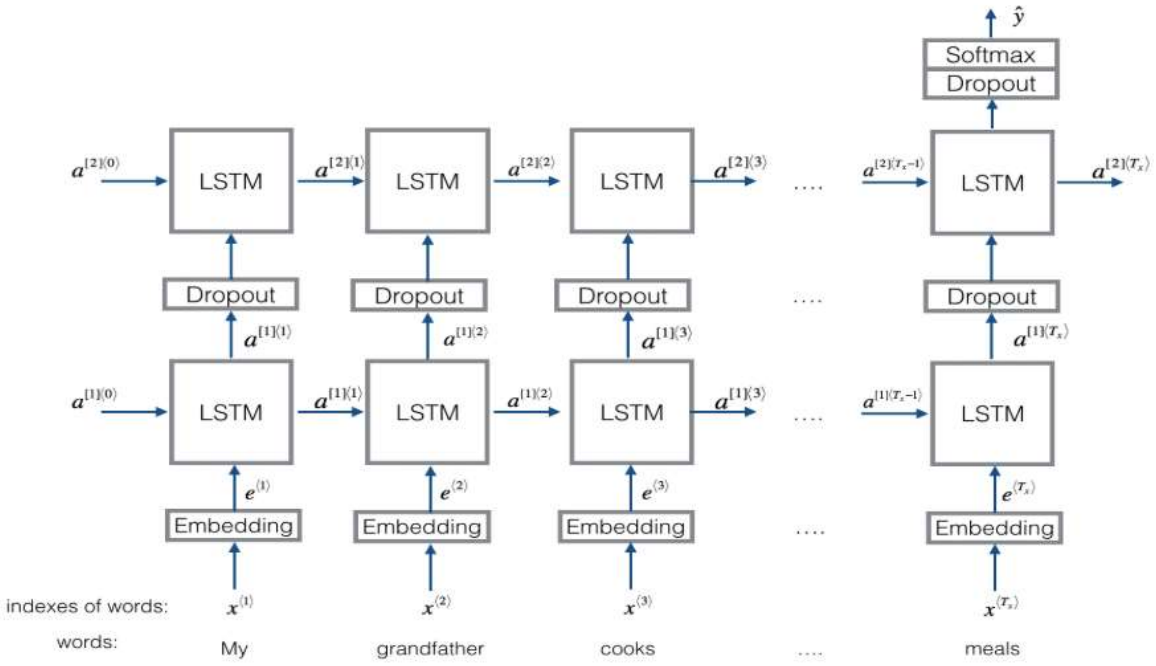


Figure 3: Emojifier-V2. A 2-layer LSTM sequence classifier.

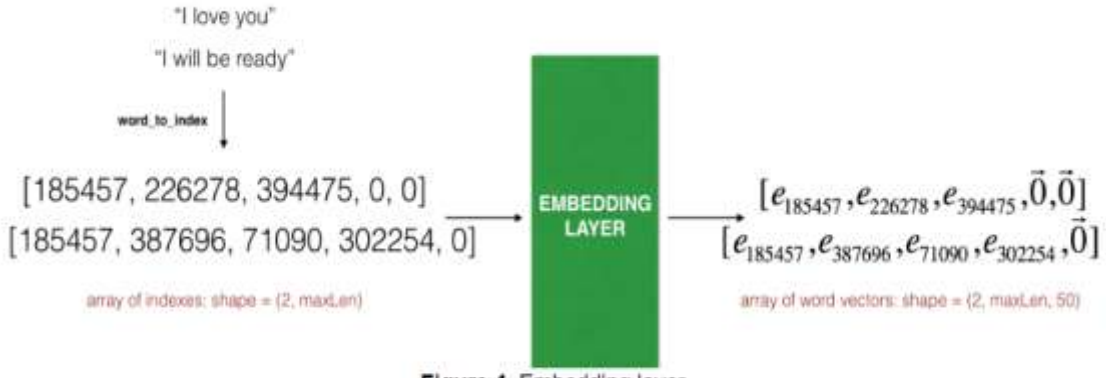


Figure 4: Embedding layer

- Input()
  - Set the `shape` and `dtype` parameters.
  - The inputs are integers, so you can specify the data type as a string, 'int32'.
- LSTM()
  - Set the `units` and `return_sequences` parameters.
- Dropout()
  - Set the `rate` parameter.
- Dense()
  - Set the `units`,
  - Note that `Dense()` has an `activation` parameter. For the purposes of passing the autograder, please do not set the activation within `Dense()`. Use the separate `Activation` layer to do so.
- Activation().
  - You can pass in the activation of your choice as a lowercase string.
- Model Set `inputs` and `outputs`.

What you should remember

- If you have an NLP task where the training set is small, using word embeddings can help your algorithm significantly.
- Word embeddings allow your model to work on words in the test set that may not even appear in the training set.
- Training sequence models in Keras (and in most other deep learning frameworks) requires a few important details:
  - To use mini-batches, the sequences need to be **padded** so that all the examples in a mini-batch have the **same length**.
  - An `Embedding()` layer can be initialized with pretrained values.
    - These values can be either fixed or trained further on your dataset.
    - If however your labeled dataset is small, it's usually not worth trying to train a large pre-trained set of embeddings.
  - `LSTM()` has a flag called `return_sequences` to decide if you would like to return every hidden states or only the last one.
  - You can use `Dropout()` right after `LSTM()` to regularize your network.