

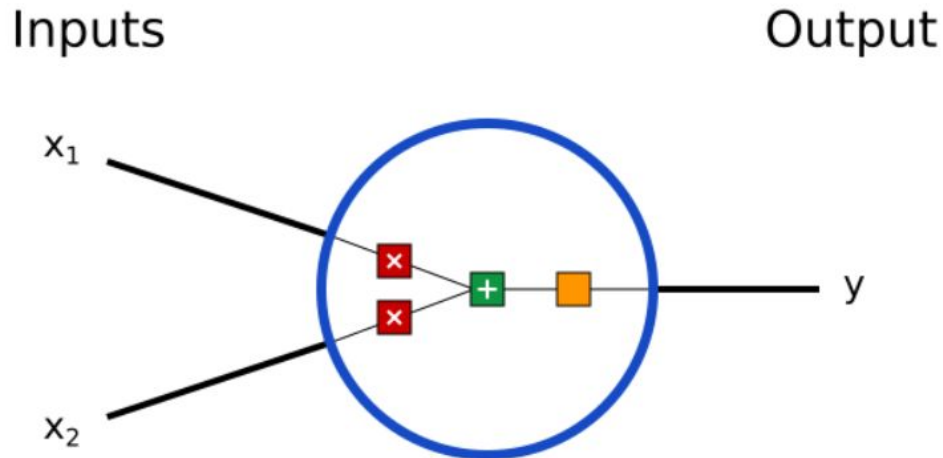
NLP with Deep Learning

Introduction to Neural Networks

Building Blocks: **Neurons**

Neurons - basic unit of a neural network

- A neuron takes inputs, does some math with them and **produces one output**



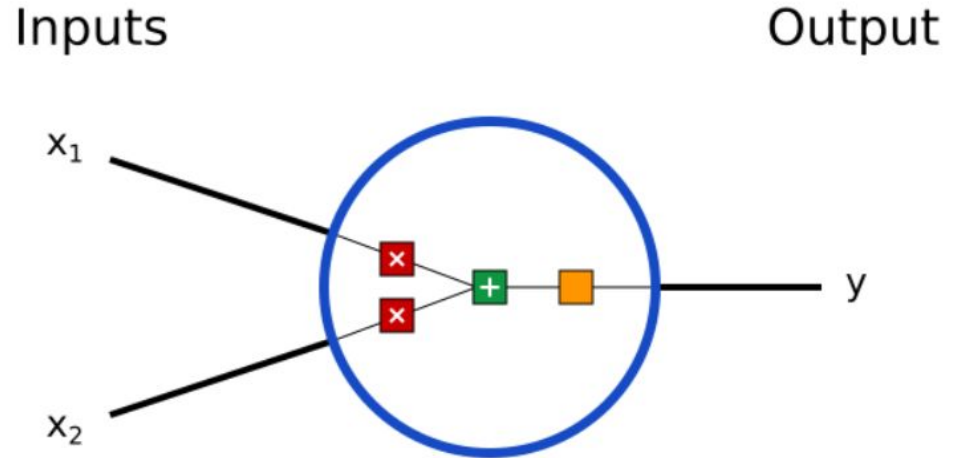
Introduction to Neural Networks

Three things happening here:

1. First, each input is multiplied by a weight:

$$x_1 \rightarrow x_1 * w_1$$

$$x_2 \rightarrow x_2 * w_2$$

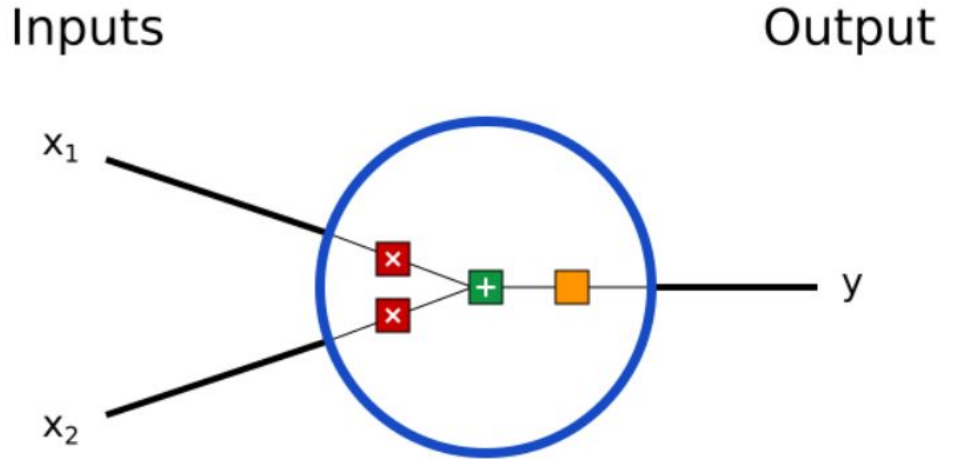


Introduction to Neural Networks

Three things happening here:

2. Next, all the weighted inputs are added together with a bias b :

$$(x_1 * w_1) + (x_2 * w_2) + b$$



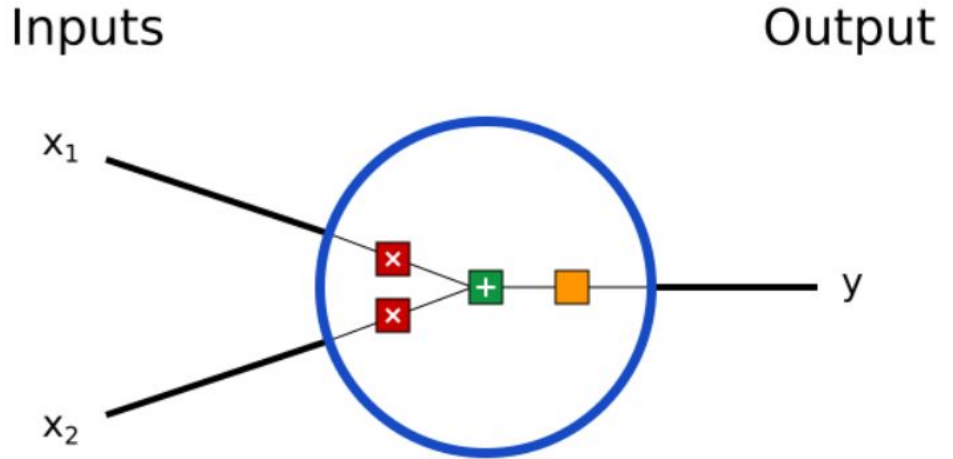
Introduction to Neural Networks

Three things happening here:

3. Finally, the sum is passed

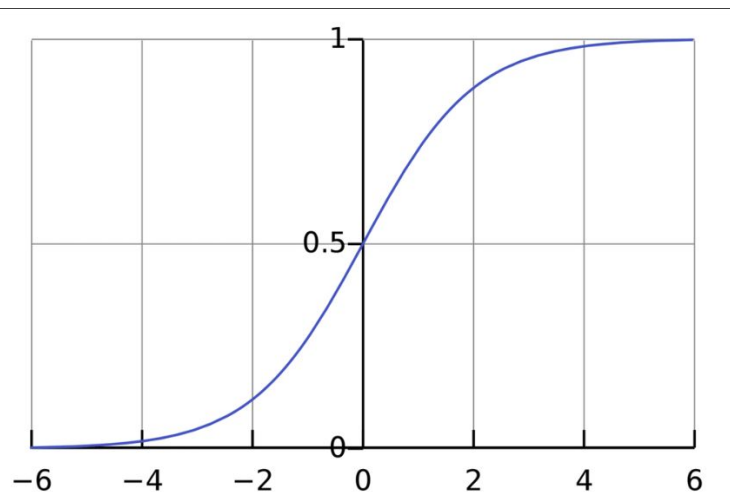
through an activation function:

$$y = f((x_1 * w_1) + (x_2 * w_2) + b)$$



Introduction to Neural Networks

- The activation function is used to turn an unbounded input into an output that has a nice, predictable form.
- A commonly used activation function is the sigmoid function:



Check this out: Neural Network from Scratch!

[Neural Network from Scratch by Victor Zhou](#)

Introduction to Neural Networks

- Feedforward neural network
 - The information moves only in one direction - forward - from the input nodes, through the hidden nodes (if there are any) and to the output nodes
- Back Propagation
 - Practice of fine-tuning weights of a neural net based on the error rate (i.e. loss) obtained in the previous epoch (i.e. iteration)
 - Proper tuning of weights ensures lower error rates, making the model reliable by increasing its generalization

Important Reminder

- A key step to ensuring that your model obtains reasonable performance on the task at hand is to perform preprocessing on your data

The unbalanced classes create a problem due to two main reasons:

1. We don't get optimized results for the class which is unbalanced in real time as the model/algorithm never gets sufficient look at the underlying class
2. It creates a problem of making a validation or test sample as it's difficult to have representation across classes in case number of observation for few classes is extremely less

Handling Unbalanced Datasets

- Undersampling
- Oversampling
- SMOTE

Loss Function

- Method of evaluating how well your algorithm models your dataset (determine the error (aka “the loss”) between the output of our algorithms and the given target value)
- In the context of an optimization algorithm, the function used to evaluate a candidate solution (i.e. a set of weights) is referred to as the **objective function** (we want to know the minimum/maximum value of this objective function)
- Typically, with neural networks, we seek to ***minimize the error***. As such, the objective function is often referred to as a cost function or a loss function and the value calculated by the loss function is referred to as simply “loss.”

Language Modeling

- You can also think of a Language Model as a system that assigns a probability to a piece of text
- For example, if we have some text , then the probability of this text (according to the Language Model) is:

$$\begin{aligned} P(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(T)}) &= P(\mathbf{x}^{(1)}) \times P(\mathbf{x}^{(2)} | \mathbf{x}^{(1)}) \times \dots \times P(\mathbf{x}^{(T)} | \mathbf{x}^{(T-1)}, \dots, \mathbf{x}^{(1)}) \\ &= \prod_{t=1}^T P(\mathbf{x}^{(t)} | \mathbf{x}^{(t-1)}, \dots, \mathbf{x}^{(1)}) \end{aligned}$$



This is what our LM provides

Language Modeling

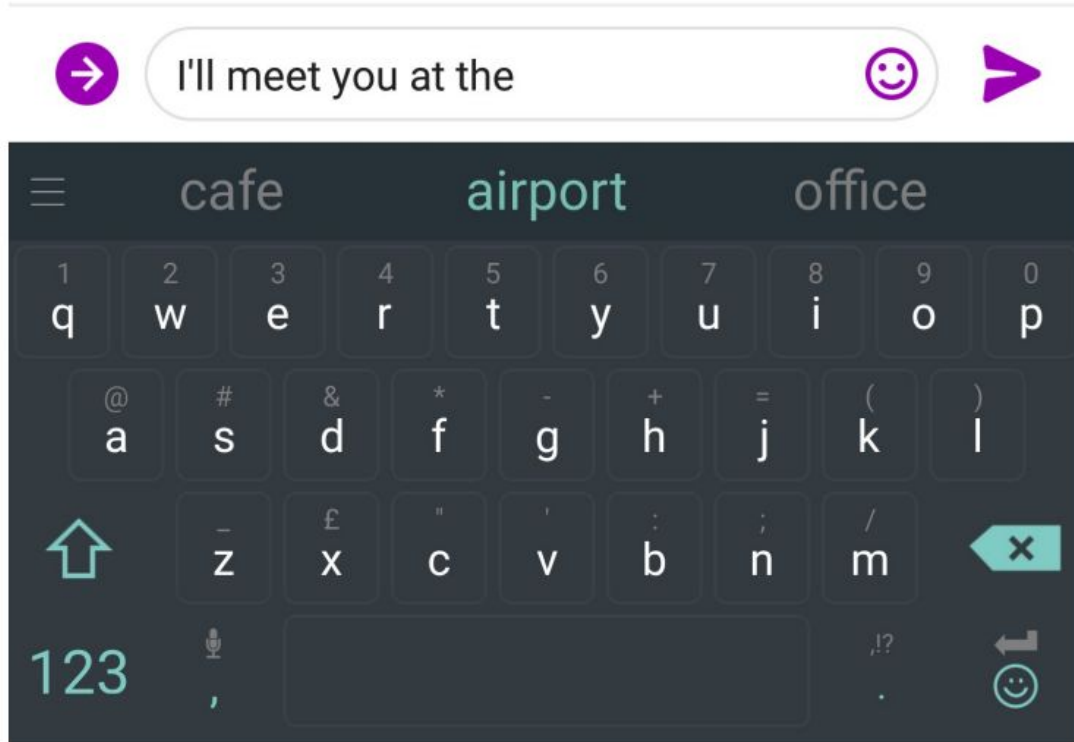
- Useful for **speech and translation systems** when determining whether a word sequence is an accurate translation of an input sentence.
- e.g. {I have, I had, I has, me have, me had}
 - Then scores them to identify the most likely translation sequence
 - In machine translation, the model chooses the best word ordering for an input phrase by assigning a goodness score to each output word sequence alternative

Language Modeling

- To do so, the model may choose between different word ordering or word choice alternatives
- It would achieve this objective by running all word sequence candidates through a probability function that assigns each a score
- The sequence with the highest score is the output of the translation

- For example,
 - "the cat is small" has higher score vs "small the is cat"
 - higher score to "walking home after school" compared to "walking house after school"

We Use Language Models Everyday!



n-gram Language Models

the students opened their _____

- **Question:** How to learn a Language Model?
- **Answer** (pre- Deep Learning): learn an n-gram Language Model!
- Definition: An n-gram is a chunk of n consecutive words.

unigrams: “the”, “students”, “opened”, “their”

bigrams: “the students”, “students opened”, “opened their”

trigrams: “the students opened”, “students opened their”

four-grams: “the students opened their”

- **Idea:** Collect statistics about how frequent different n-grams are and use these to predict next word.

$$p(w_2|w_1) = \frac{\text{count}(w_1, w_2)}{\text{count}(w_1)}$$

$$p(w_3|w_1, w_2) = \frac{\text{count}(w_1, w_2, w_3)}{\text{count}(w_1, w_2)}$$

n-gram Language Models

- Consider the sentence:
 - "As the proctor started the clock, the students opened their _____".
 - If the window only conditions on the previous three words "the students

opened their", the probabilities calculated based on the corpus may suggest that the next word be **"books"**

- If n had been large enough to include the "proctor" context, the probability might have suggested **"exam"**.

Evaluating Language Models

- The standard evaluation metric for Language Models is perplexity. (*Lower perplexity is better*)

$$\text{perplexity} = \prod_{t=1}^T \left(\frac{1}{P_{\text{LM}}(\mathbf{x}^{(t+1)} | \mathbf{x}^{(t)}, \dots, \mathbf{x}^{(1)})} \right)^{1/T}$$

Normalized by number of words

Inverse probability of corpus, according to Language Model

Disadvantages of n-gram Language Models

1. Sparsity problems with n-gram Language Models
 - Smoothing and backoff (need to do these if words do not appear together in the corpus)

Smoothing and Backoff

Sparsity Problem 1

Problem: What if “students opened their w_j ” never occurred in data? Then w_j has probability 0!

(Partial) Solution: Add small δ to count for every $w_j \in V$. This is called *smoothing*.

$$P(w_j | \text{students opened their}) = \frac{\text{count}(\text{students opened their } w_j)}{\text{count}(\text{students opened their})}$$

Sparsity Problem 2

Problem: What if “students opened their” never occurred in data? Then we can’t calculate probability for *any* w_j !

(Partial) Solution: Just condition on “opened their” instead. This is called *backoff*.

Disadvantages of n-gram Language Models

2. Storage problems

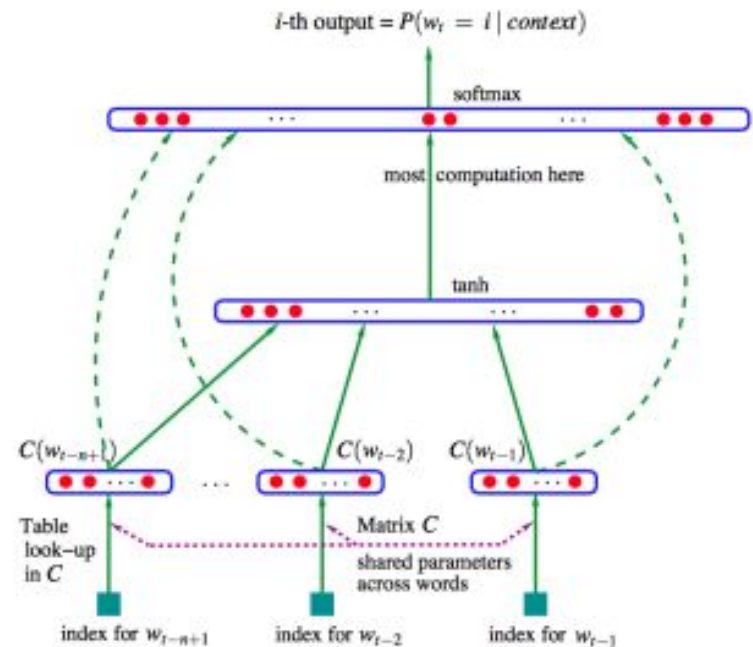
As the corpus size increases, the model size increases as well

Storage: Need to store count for all possible n -grams. So model size is $O(\exp(n))$.

$$P(\mathbf{w}_j | \text{students opened their}) = \frac{\text{count}(\text{students opened their } \mathbf{w}_j)}{\text{count}(\text{students opened their})}$$

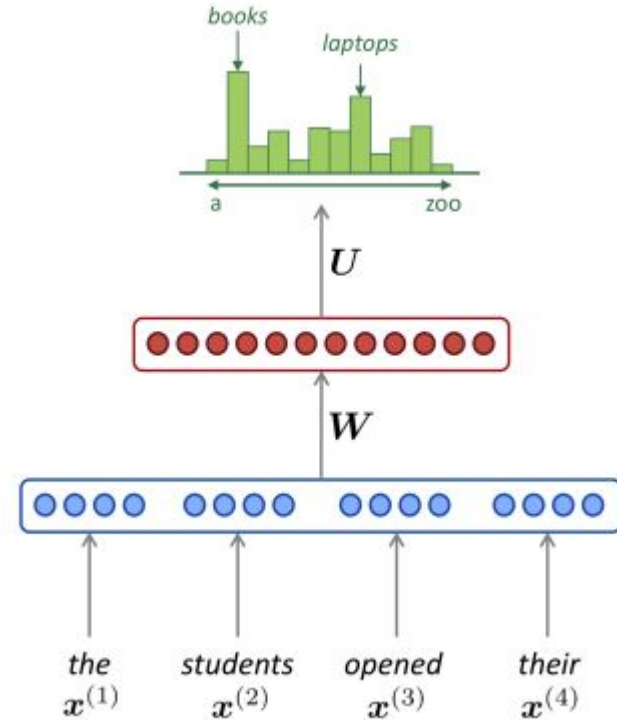
Window-Based Neural Model

This model learns a distributed representation of words, along with the probability function for word sequences expressed in terms of these representations



Window-Based Neural Model

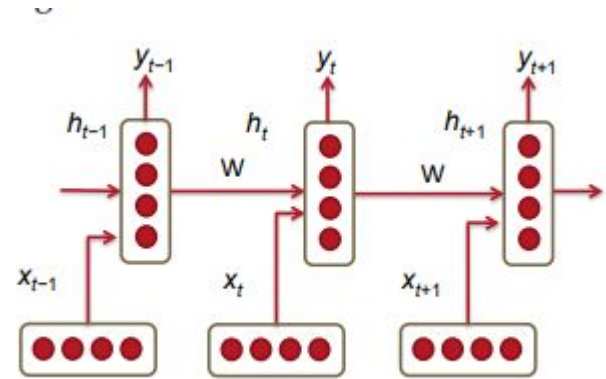
This model learns a distributed representation of words, along with the probability function for word sequences expressed in terms of these representations



Recurrent Neural Network (RNN)

RNN Advantages:

- Can process any length input
- Computation for step t can (in theory) use information from many steps back
- Model size doesn't increase for longer input context
- Same weights applied on every timestep, so there is symmetry in how inputs are processed.



Vanishing Gradient & Gradient Explosion Problems

Recall: the goal of a RNN implementation is to enable propagating context information through faraway time-steps.

Sentence 1:

"Jane walked into the room. John walked in too. Jane said hi to ____"

Sentence 2:

"Jane walked into the room. John walked in too. It was late in the day, and everyone was walking home after a long day at work. Jane said hi to ____"

Vanishing Gradient & Gradient Explosion Problems

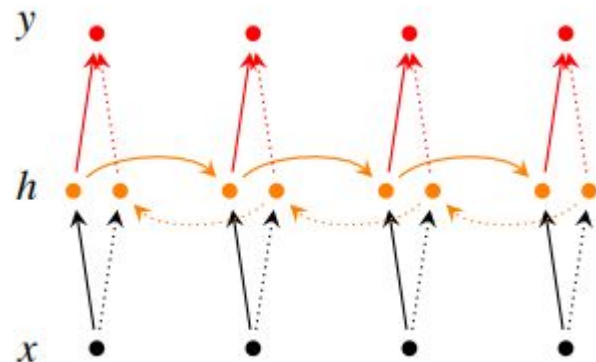
- It turns out RNNs are more likely to correctly predict the blank spot in Sentence 1 than in Sentence 2.
- This is because during the back-propagation phase, the contribution of gradient values gradually vanishes as they propagate to earlier timesteps
- Thus, for long sentences, the probability that "John" would be recognized as the next word reduces with the size of the context.

Solving Vanishing Gradient Problems

1. Initialize a random weight, start off from an identity matrix initialization.
2. Use the Rectified Linear Units (ReLU) instead of the sigmoid function. The derivative for the ReLU is either 0 or 1 . This way, gradients would flow through the neurons whose derivative is 1 without getting attenuated while propagating back through time-steps.

Deep Bidirectional RNNs

- This network maintains two hidden layers, one for the left-to-right propagation and another for the right-to-left propagation.
- To maintain two hidden layers at any time, this network consumes twice as much memory space for its weight and bias parameters.



Gated Recurrent Units (GRU)

- RNNs have been found to perform better with the use of more complex units for activation
- The use of a gated activation function modifies the RNN architecture. Although RNNs can theoretically capture long-term dependencies, they are very hard to actually train to do this
- Gated recurrent units are designed in a manner to have more persistent memory thereby making it easier for RNNs to capture long-term dependencies

To know more: [Gated Recurrent Units](#)

Long Short-Term Memory RNNs (LSTMs)

- Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems.
- This is a behavior required in complex problem domains like machine translation, speech recognition, and more.
- LSTM networks combat the RNN's vanishing gradients or long-term dependence issue.
- Gradient vanishing refers to the loss of information in a neural network as connections recur over a longer period. In simple words, LSTM tackles gradient vanishing by ignoring useless data/information in the network.

Building a Recurrent Neural Network Using Keras

The code for a simple LSTM is below with an explanation following:

```
from keras.models import Sequential

from keras.layers import LSTM, Dense, Dropout, Masking, Embedding


model = Sequential()


# Embedding layer
model.add(
    Embedding(input_dim=num_words, input_length = training_length,
output_dim=100, weights=[embedding_matrix], trainable =False, mask_zero =True))
```

```
# Masking layer for pre-trained embeddings
model.add(Masking(mask_value=0.0))
```

```
# Recurrent layer
model.add(LSTM(64, return_sequences=False,
              dropout=0.1,
              recurrent_dropout=0.1))
```

```
# Fully connected layer
model.add(Dense(64, activation='relu'))
```

```
# Dropout for regularization
model.add(Dropout(0.5))
```

```
# Output layer
model.add(Dense(num_words,
                activation='softmax'))
```

```
# Compile the model
model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy'])
```


Keras

We are using the Keras `Sequential` API which means we build the network up one layer at a time. The layers are as follows:

An `Embedding` which maps each input word to a 100-dimensional vector. The embedding can use pre-trained weights (more in a second) which we supply in the `weights` parameter. `trainable` can be set `False` if we don't want to update the embeddings.

- A `Masking` layer to mask any words that do not have a pre-trained embedding which will be represented as all zeros. This layer should not be used when training the embeddings.
- The heart of the network: a layer of `LSTM` cells with [dropout to prevent overfitting](#). Since we are only using one LSTM layer, it *does not* return the sequences, for using two or more layers, make sure to return sequences.
- A fully-connected `Dense` layer with `relu` activation. This adds additional representational capacity to the network.
- A `Dropout` layer to prevent overfitting to the training data.
- A `Dense` fully-connected output layer. This produces a probability for every word in the vocab using `softmax` activation.

Keras

The model is compiled with the `Adam` optimizer (a variant on Stochastic Gradient Descent) and trained using the `categorical_crossentropy` loss.

During training, the network will try to minimize the log loss by adjusting the trainable parameters (weights). As always, the gradients of the parameters are calculated using [back-propagation](#) and updated with the optimizer. Since we are using Keras, we [don't have to worry about how this happens](#) behind the scenes, only about setting up the network correctly.

Layer (type)	Output Shape	Param #
embedding_3 (Embedding)	(None, 50, 100)	1175500
masking_3 (Masking)	(None, 50, 100)	0
lstm_3 (LSTM)	(None, 64)	42240
dense_5 (Dense)	(None, 64)	4160
dropout_3 (Dropout)	(None, 64)	0
dense_6 (Dense)	(None, 11755)	764075
Total params: 1,985,975		
Trainable params: 810,475		
Non-trainable params: 1,175,500		

LSTM network layout.

Generating text with an RNN Language Model

- You can train an RNN-LM on any kind of text, then generate text in that style.
- RNN-LM trained on Obama speeches:



The United States will step up to the cost of a new challenges of the American people that will share the fact that we created the problem. They were attacked and so that they have to say that all the task of the final days of war that I will not be able to get this done.

Final Project Topics

Machine Translation / Grammar Error Correction

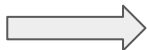
Keywords: seq2seq models, cross-corpora analysis

INPUT

OUTPUT

Translation

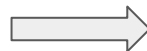
I have never been to Taiwan



我從來沒去過台灣

GEC

I never been to Taiwan



I have never been to Taiwan

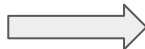
Summarization / Paraphrasing

Keywords: seq2seq models

INPUT

The tower is 324 metres (1,063 ft) tall, about the same height as an 81-storey building, and the tallest structure in Paris. Its base is square, measuring 125 metres (410 ft) on each side. It was the first structure to reach a height of 300 metres.

Excluding transmitters, the Eiffel Tower is the second tallest free-standing structure in France after the Millau Viaduct.



OUTPUT

The tower is 324 metres (1,063 ft) tall, about the same height as an 81-storey building. It was the first structure to reach a height of 300 metres.

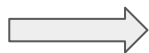
Sequence Labelling

Keywords: LSTM, CRF

INPUT

Adding punctuations

今天風很大把我的車吹走了
我現在很難過

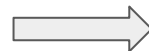


OUTPUT

CCCCC, CCCCCCC, CCCCC
C。

POS tagging

I have never been to Taiwan



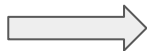
Pron Aux Adv V Prep Prop_N

Word Sense Disambiguation


Mainstream models: Masked-LM w/ nearest neighbor, contextual gloss classification

INPUT

The underdog team "beat"
the reigning champion.



OUTPUT

1. To defeat (someone) in a game or other competitive situation.
2.  To defeat (someone) in a game or other competitive situation.
3. A main accent or rhythmic unit in music or poetry.

Sentence Classification Models

Topics:

- Topic sentence identification (given a paragraph)
- Move structure Classification: predict sentence label given a paragraph
- Language competency level classification (e.g. Levels 1~5)

A popular topic in classification is natural language inference (NLI). You may start with this keyword and see if other people's methods are useful for your task.

Note: NO sentiment analysis

Useful Tools

[Tensorflow](#) (often used with [Keras](#))

[Pytorch](#)

[Huggingface](#) 🤗

[Scikit-learn](#)

Important Dates

- Submit your group and chosen topic [here](#): 11/10, before 23:59
- Topic Proposal: 11/17, in class
- Proposal slides due: 11/20, before 23:59
- Progress check: 12/1 or 12/8 (same order as your final presentation), in class
- Final presentation: 12/29 or 1/5, in class
- Project due: 1/12, before 23:59