Convolutional Neural Networks For NLP

Rudra Murthy

Center for Indian Language Technology,
Indian Institute of Technology Bombay

rudra@cse.iitb.ac.in

https://www.cse.iitb.ac.in/~rudra

Deep Learning Tutorial.
ICON 2017, Kolkata
21th December 2017

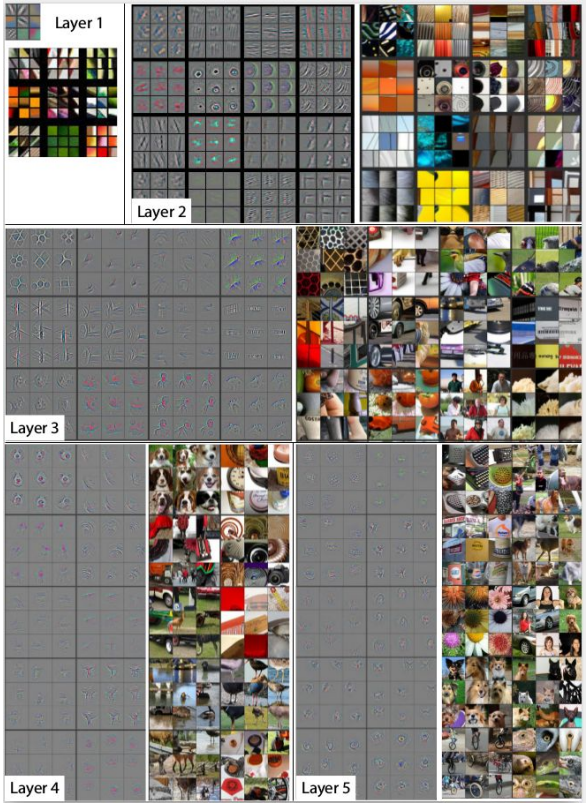
Outline

- Motivation
- What are CNNs?
- CNNs for various NLP Tasks
- Summary

Motivation

Consider the task of Image Recognition

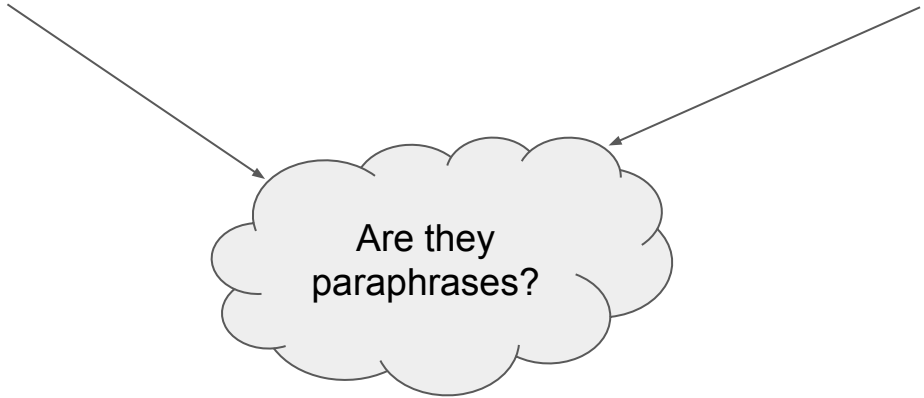
- CNNs are shown to learn hierarchical features from the data [Zeiler et.al 2014]
- Layer 1 recognizes lines and curves, layer 2 composes them to recognize simple shapes like squares, circles, *etc.*
- Layer 3 composes the output from layer 2 to recognize more complex shapes like humans, cars, *etc.*
- Two charactersitics are prominent here:
    - Recognizing position-independent features
    - Composing the features to obtain more complex features



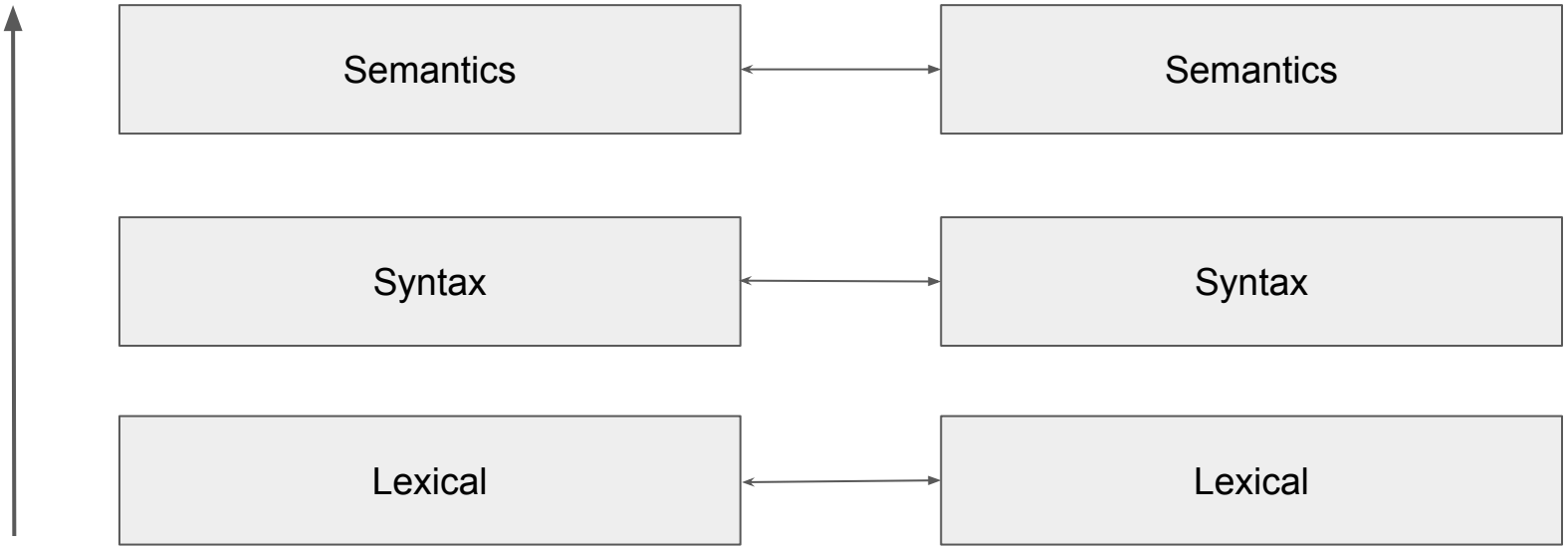
Consider the task of Paraphrase Detection

There is a Deep Learning Tutorial at ICON

DL Tutorial is scheduled @ ICON



Consider the task of Paraphrase Detection



There is a Deep Learning Tutorial at ICON

DL Tutorial is scheduled @ ICON

Consider the task of Paraphrase Detection

- Paraphrase detection can be handled at various layers of nlp layers
- At lexical layer, we compare the word overlap between the two sentences
- At Semantic layer we compare the meaning of the two sentences
- Each higher layer requires information coming from the lower layer
- We need a hierarchy of trainable feature extractors
- The lower layer feature extractor should be position independent

What are CNNs?

What is CNN?

Convolutional neural network (CNN, or ConvNet) is a type of feed-forward artificial neural network where the individual neurons are tiled in such a way that they respond to overlapping regions in the input field. (wikipedia)

CNNs are good at learning features from the data

Specifically, we will discuss the Time-Delay Neural Network used by Collobert et.al (2011)

What is CNN?

CNN is a type of feed-forward neural network with

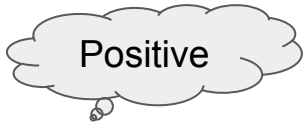
- Local connectivity
- Share weights/parameters across spatial positions

CNNs for various NLP tasks

Sentiment Analysis

Consider the task of sentiment analysis,

The movie is very good

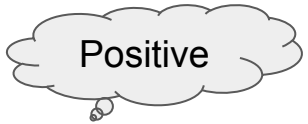


The movie is very bad

Train a supervised machine learning system to predict the sentiment of the text

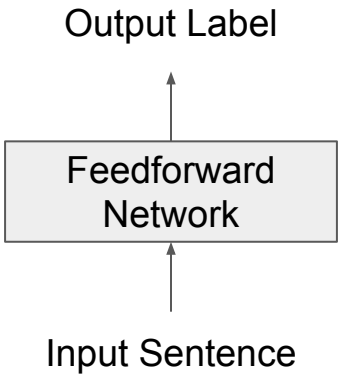
Consider the task of sentiment analysis,

The movie is very good



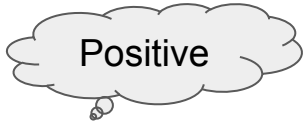
The movie is very bad

Train a simple Feedforward neural network to predict the sentiment of the text



Consider the task of sentiment analysis,

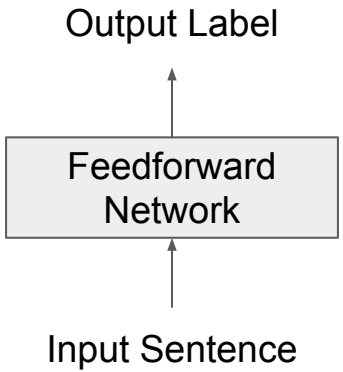
The movie is very good



The movie is very bad

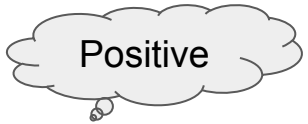
Train a simple Feedforward neural network to predict the sentiment of the text

- How to represent the input sentence?
  - Bag-of-words representation
    - Disregards the word order
  - Concatenation of Word Embeddings
    - How to handle variable-length sentences?



Consider the task of sentiment analysis,

The movie is very good

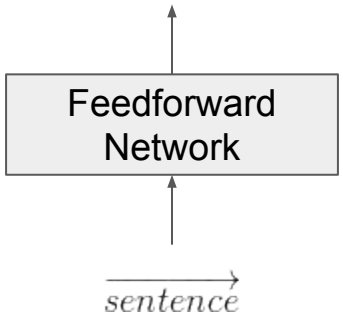


The movie is very bad

Train a simple Feedforward neural network to predict the sentiment of the text

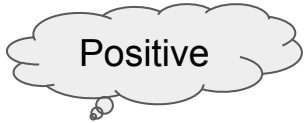
Bag of Words Representation (Average of Word Embeddings)

$$\overrightarrow{sentence} = \frac{\overrightarrow{the} + \overrightarrow{movie} + \overrightarrow{is} + \overrightarrow{very} + \overrightarrow{good}}{5}$$



Consider the task of sentiment analysis,

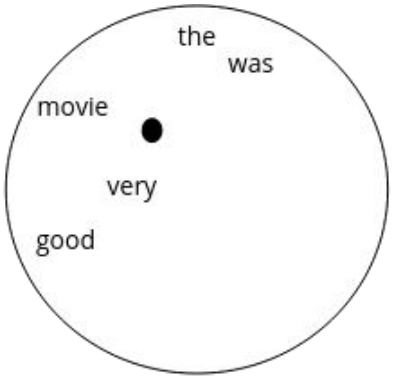
The movie is very good



The movie is very bad

Train a simple Feedforward neural network to predict the sentiment of the text

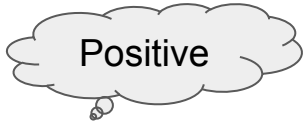
Bag of Words Representation (Average of Word Embeddings)



- Influence of unimportant words in the sentence changes the average embedding
- Use Tf-IDF to give importance to relevant words

Consider the task of sentiment analysis,

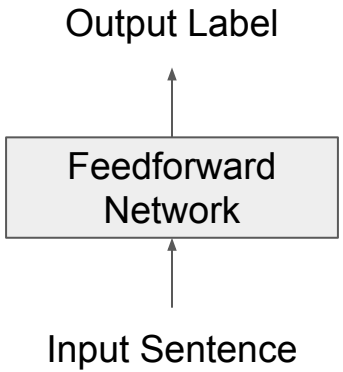
The movie is very good



The movie is very bad

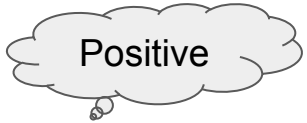
Train a simple Feedforward neural network to predict the sentiment of the text

- How to represent the input sentence?
  - Concatenate the word embeddings of all words in the sentence
    - How to handle variable-length sentences?
    - Place a restriction on the sentence length



Consider the task of sentiment analysis,

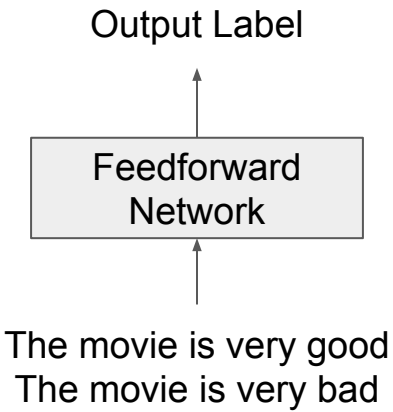
The movie is very good



The movie is very bad

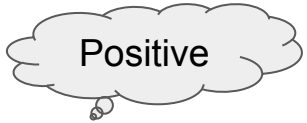
Train a simple Feedforward neural network to predict the sentiment of the text

Concatenate the word embeddings



Consider the task of sentiment analysis,

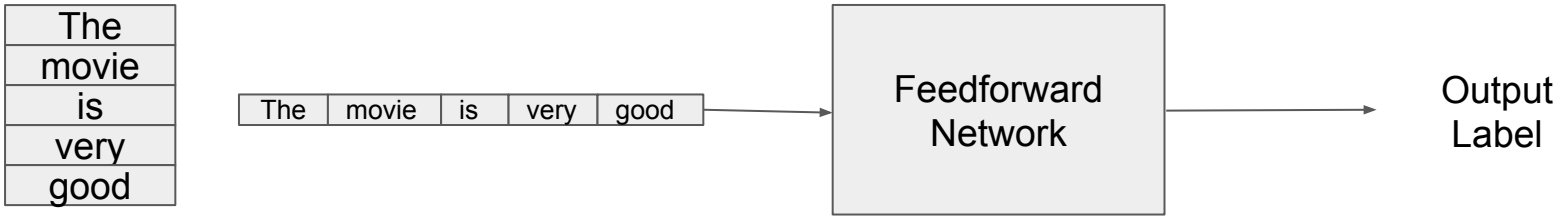
The movie is very good



The movie is very bad

Train a simple Feedforward neural network to predict the sentiment of the text

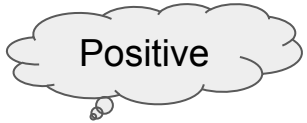
Concatenate the word embeddings



Word Embedding

Consider the task of sentiment analysis,

The movie is very good



The movie is very bad

Train a simple Feedforward neural network to predict the sentiment of the text

Concatenate the word embeddings

$$y = f(\overrightarrow{the}; \overrightarrow{movie}; \overrightarrow{is}; \overrightarrow{very}; \overrightarrow{good})$$
$$y = g(W \times [\overrightarrow{the}; \overrightarrow{movie}; \overrightarrow{is}; \overrightarrow{very}; \overrightarrow{good}])$$
$$y = g([W_1; W_2; W_3; W_4; W_5] \times [\overrightarrow{the}; \overrightarrow{movie}; \overrightarrow{is}; \overrightarrow{very}; \overrightarrow{good}])$$
$$y = g([W_1 \times \overrightarrow{the}; W_2 \times \overrightarrow{movie}; W_3 \times \overrightarrow{is}; W_4 \times \overrightarrow{very}; W_5 \times \overrightarrow{good}])$$

- *f* denotes the feedforward network here
- Let *W* denote the weights in the first layer
- *g* denotes the layers above

Concatenation of word embeddings for sentiment analysis

Let, the training data consist of instances of the form,

The ------ is very ------

The first slot is filled by words like movie, camera, …. and the second Slot is filled by words like good, bad, horrible, ….

$$y = f(\overrightarrow{the}; \overrightarrow{movie}; \overrightarrow{is}; \overrightarrow{very}; \overrightarrow{good})$$
$$y = g(W \times [\overrightarrow{the}; \overrightarrow{movie}; \overrightarrow{is}; \overrightarrow{very}; \overrightarrow{good}])$$
$$y = g([W_1; W_2; W_3; W_4; W_5] \times [\overrightarrow{the}; \overrightarrow{movie}; \overrightarrow{is}; \overrightarrow{very}; \overrightarrow{good}])$$
$$y = g([W_1 \times \overrightarrow{the}; W_2 \times \overrightarrow{movie}; W_3 \times \overrightarrow{is}; W_4 \times \overrightarrow{very}; W_5 \times \overrightarrow{good}])$$

- How will the model behave for the input sentence

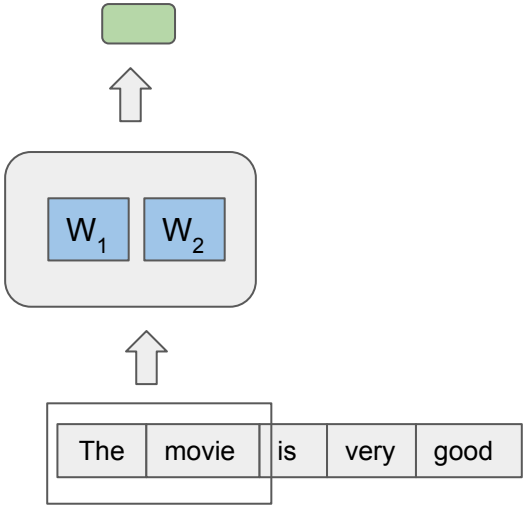
  “Very good was the movie”

CNNs for Sentiment Analysis

- Simplest approach for sentiment analysis is to check if there are any sentiment bearing words
- Assign the label based on the sentiment score of the word
- This is essentially what Tf-IDF scheme tries to simulate
- Can we do this using Deep Learning?

CNNs for Sentiment Analysis

Use Feedforward neural network on consecutive *ngram* words

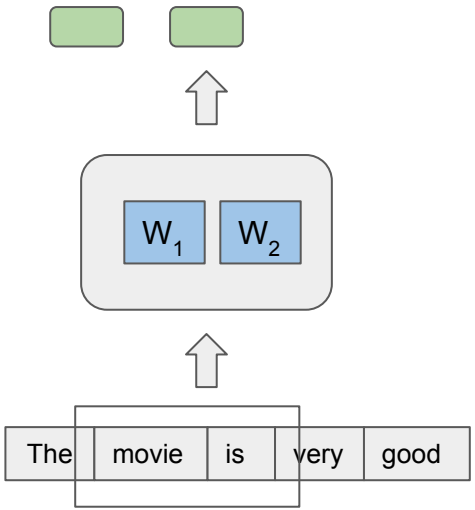


Word Embedding

Simple Linear layer looking at two words at a time

CNNs for Sentiment Analysis

Use Feedforward neural network on consecutive *ngram* words

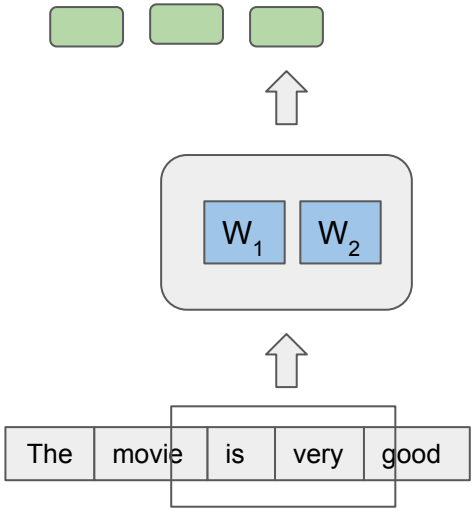


Word Embedding

Simple Linear layer looking at two words at a time

CNNs for Sentiment Analysis

Use Feedforward neural network on consecutive *ngram* words

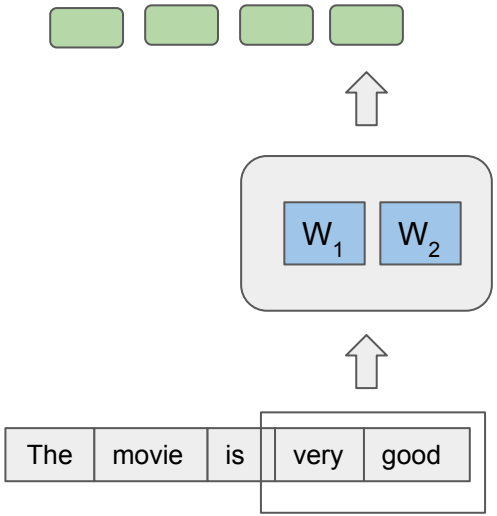


Simple Linear layer looking at two words at a time

Word Embedding

CNNs for Sentiment Analysis

Use Feedforward neural network on consecutive *ngram* words

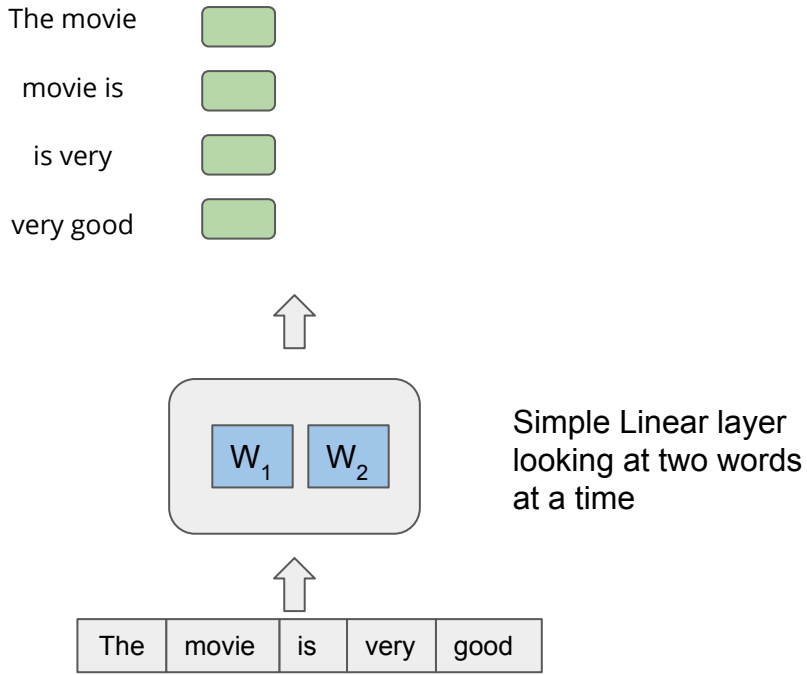


Simple Linear layer looking at two words at a time

Word Embedding

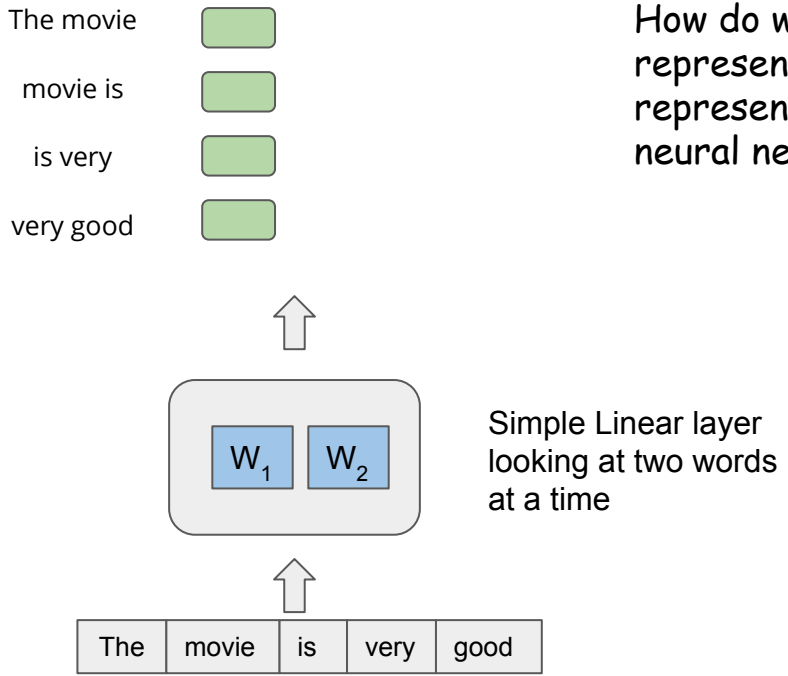
CNNs for Sentiment Analysis

Use Feedforward neural network on consecutive *ngram* words



CNNs for Sentiment Analysis

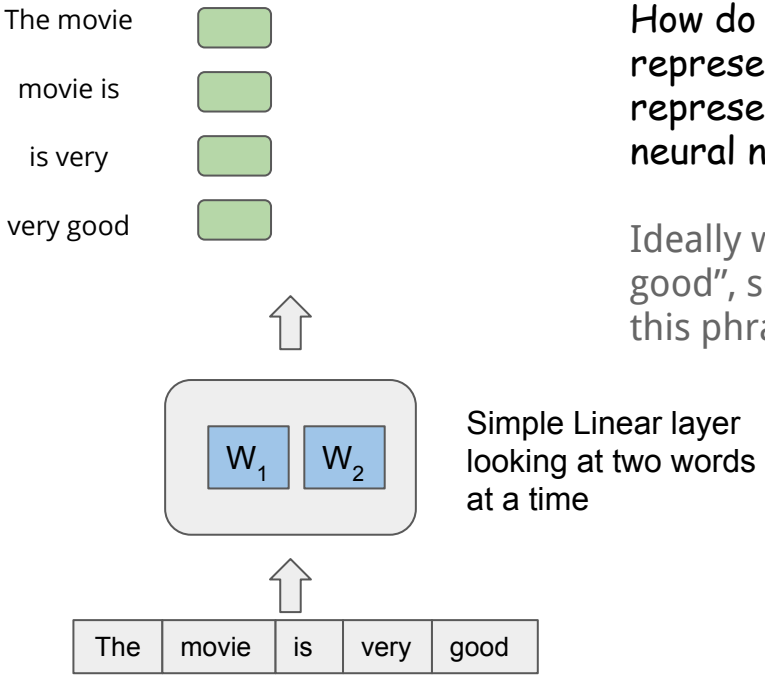
Use Feedforward neural network on consecutive *ngram* words



How do we go from variable length representation to a fixed length representation so that the feed-forward neural network can handle?

CNNs for Sentiment Analysis

Use Feedforward neural network on consecutive *ngram* words

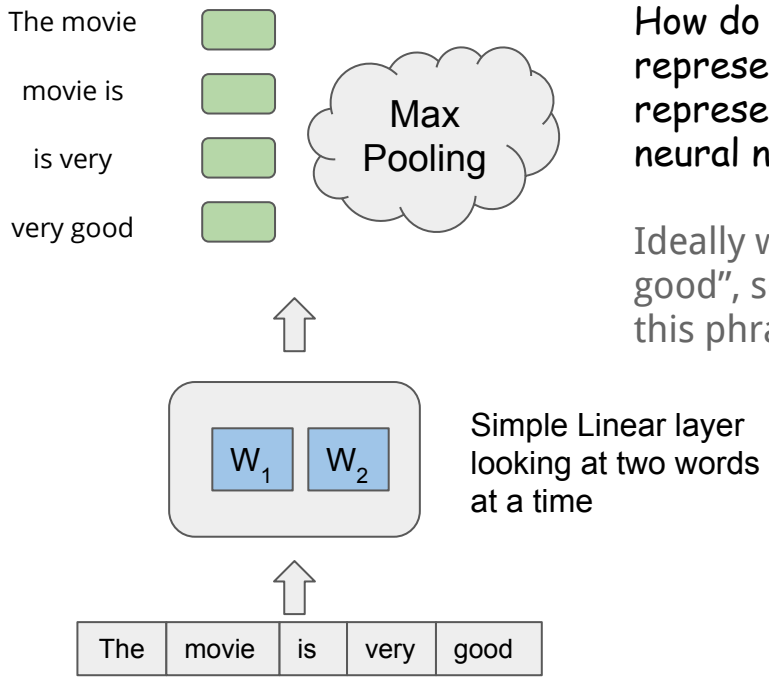


How do we go from variable length representation to a fixed length representation so that the feed-forward neural network can handle?

Ideally we have to choose the phrase "very good", so weightage have to be given to this phrase

CNNs for Sentiment Analysis

Use Feedforward neural network on consecutive *ngram* words



How do we go from variable length representation to a fixed length representation so that the feed-forward neural network can handle?

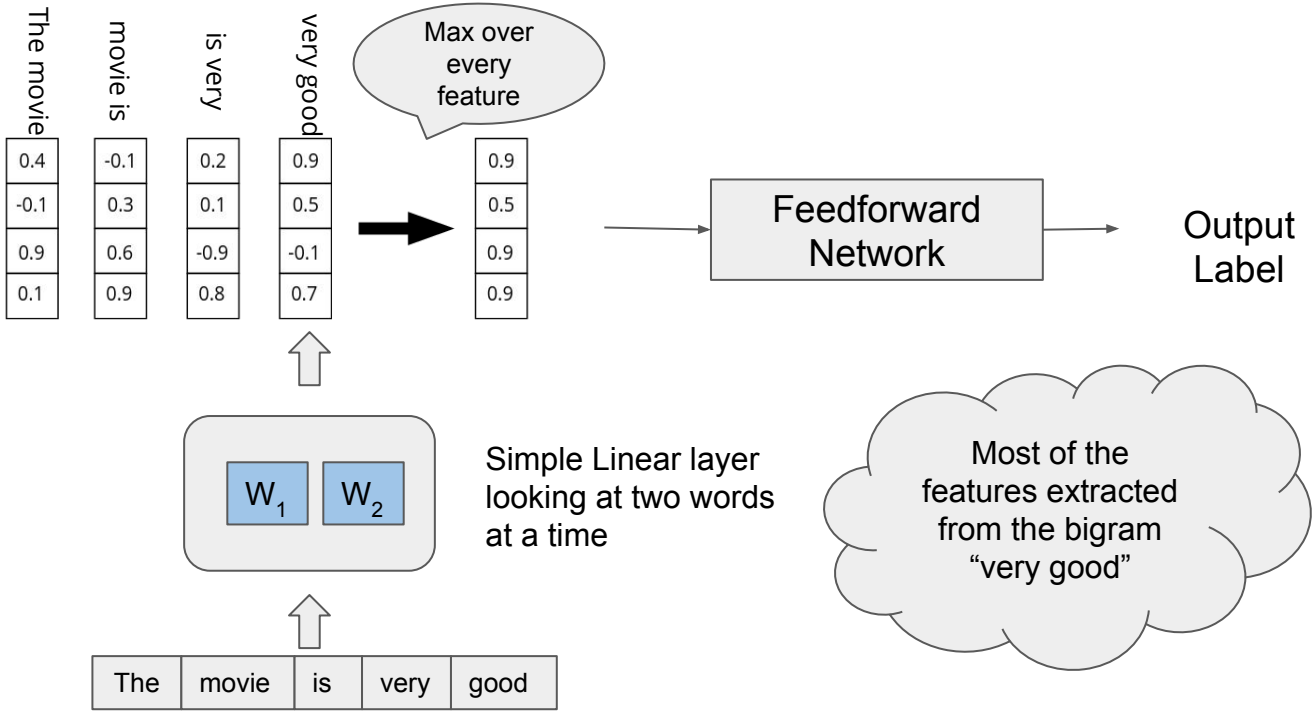
Ideally we have to choose the phrase "very good", so weightage have to be given to this phrase

Simple Linear layer looking at two words at a time

Word Embedding

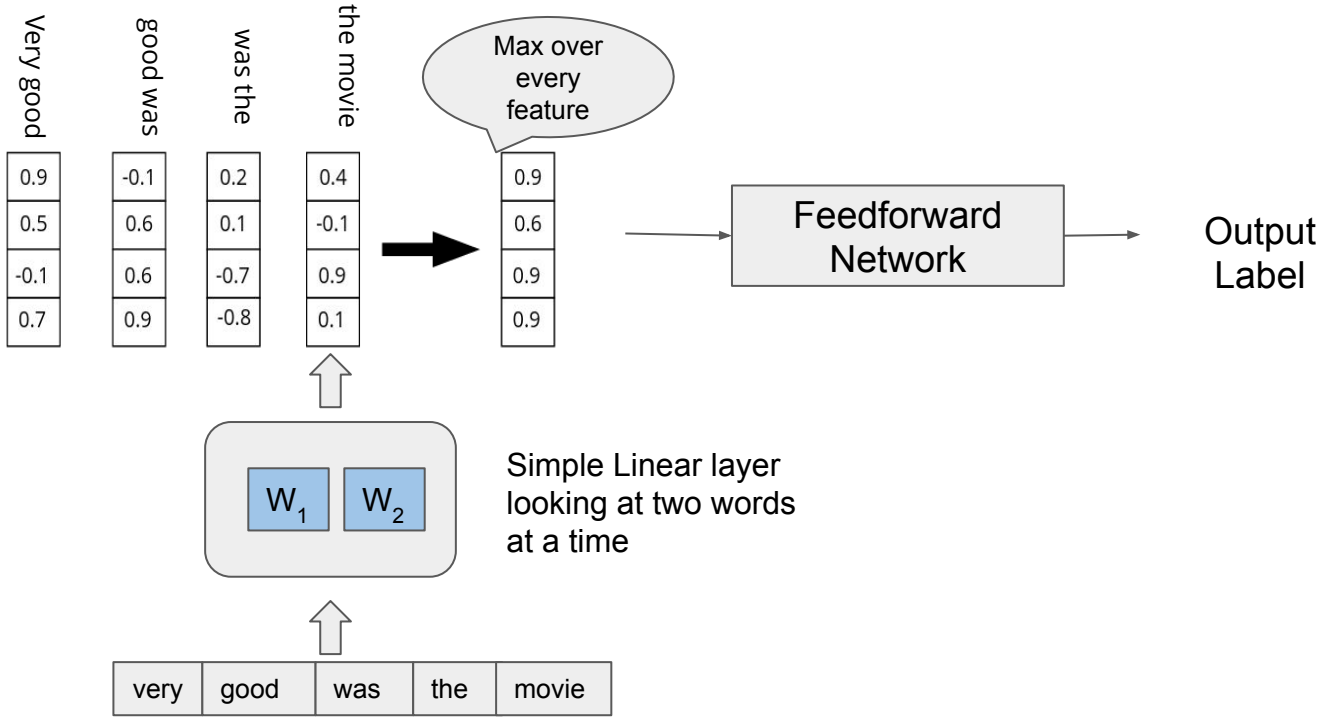
CNNs for Sentiment Analysis

Use Feedforward neural network on consecutive *ngram* words



CNNs for Sentiment Analysis

Use Feedforward neural network on consecutive *ngram* words



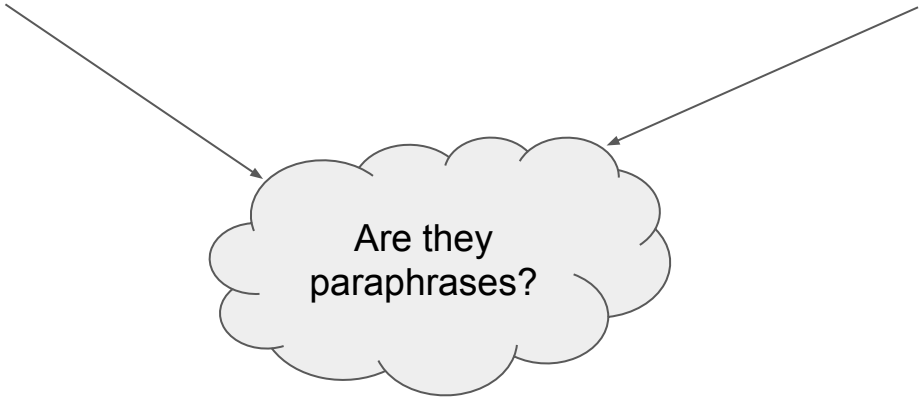
CNNs for various NLP tasks

Paraphrase Detection

Paraphrase Detection

There is a Deep Learning Tutorial at ICON

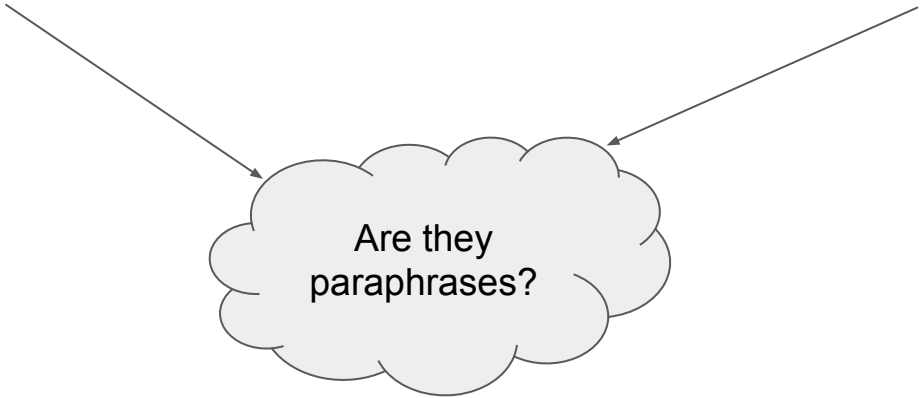
Deep Learning Tutorial is scheduled at ICON



Paraphrase Detection

There is a Deep Learning Tutorial at ICON

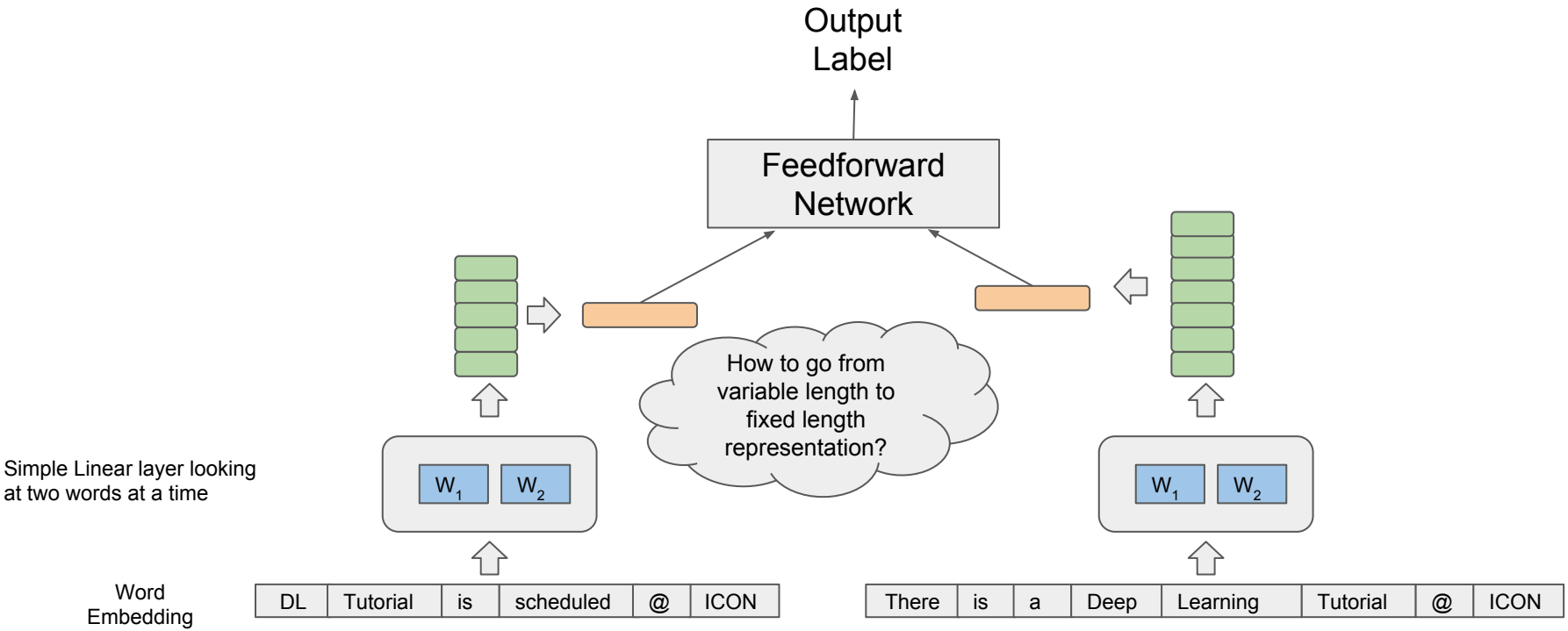
Deep Learning Tutorial is scheduled at ICON



Simplest approach for paraphrase detection using CNNs

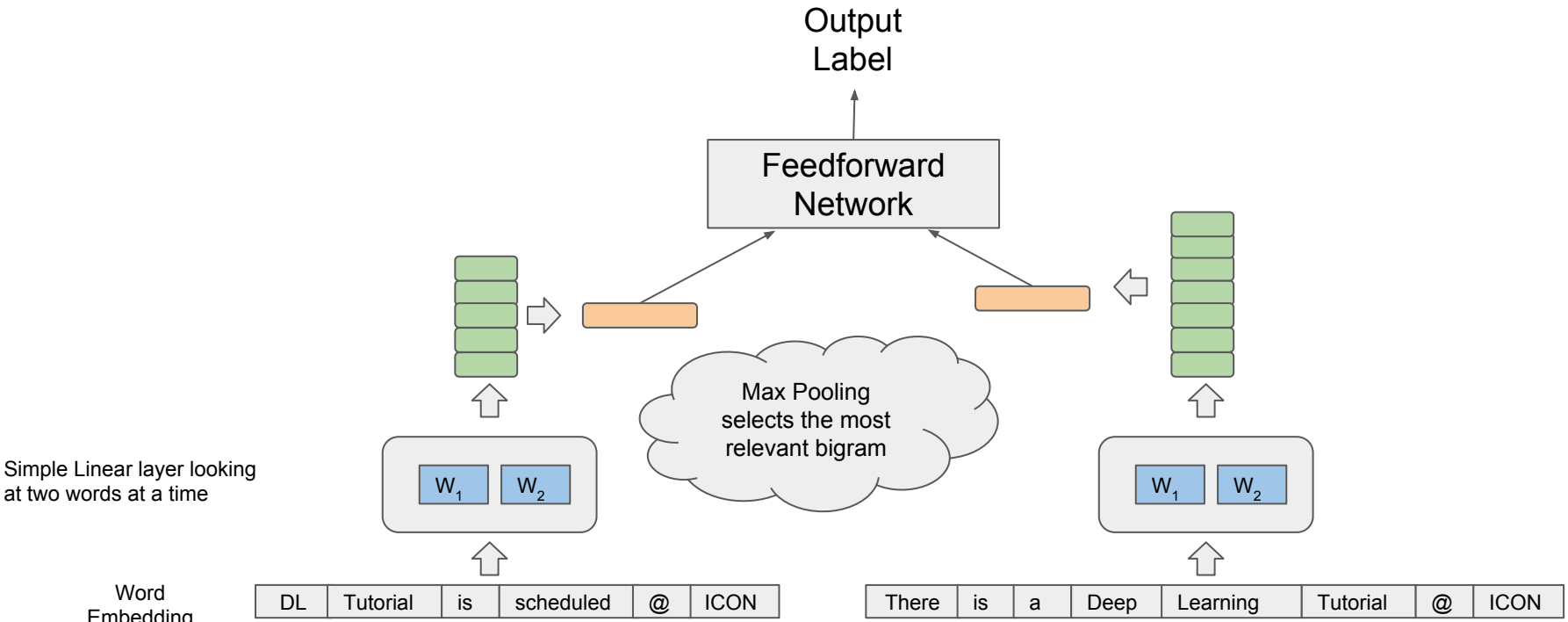
CNNs for Paraphrase Detection

Use Feedforward neural network on consecutive *ngram* words



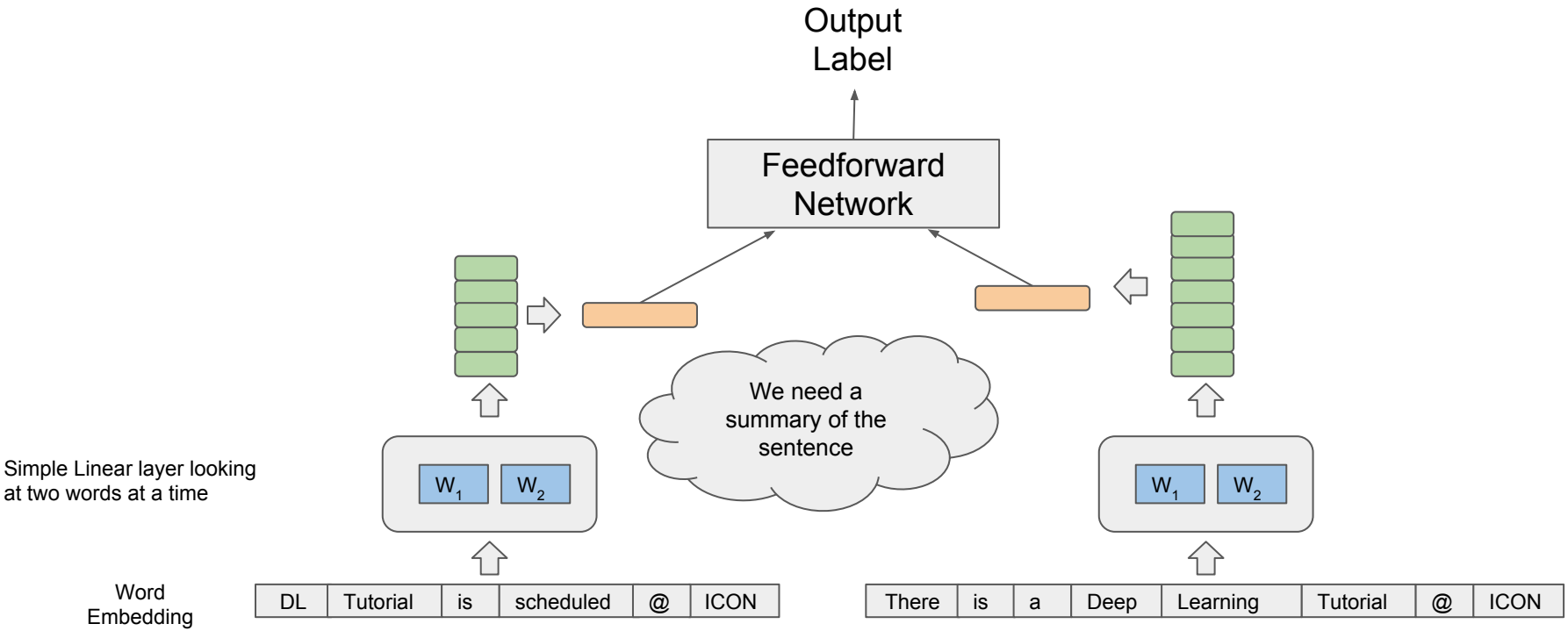
CNNs for Paraphrase Detection

Use Feedforward neural network on consecutive *ngram* words



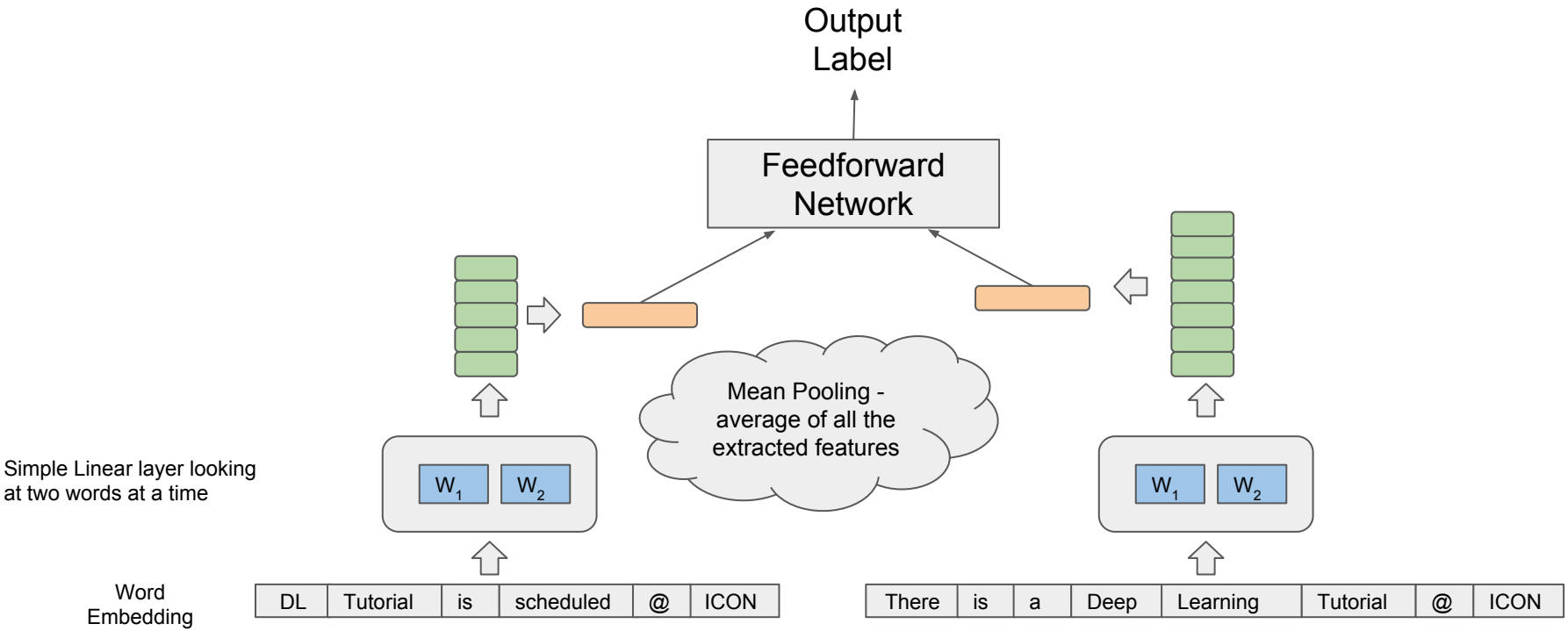
CNNs for Paraphrase Detection

Use Feedforward neural network on consecutive *ngram* words



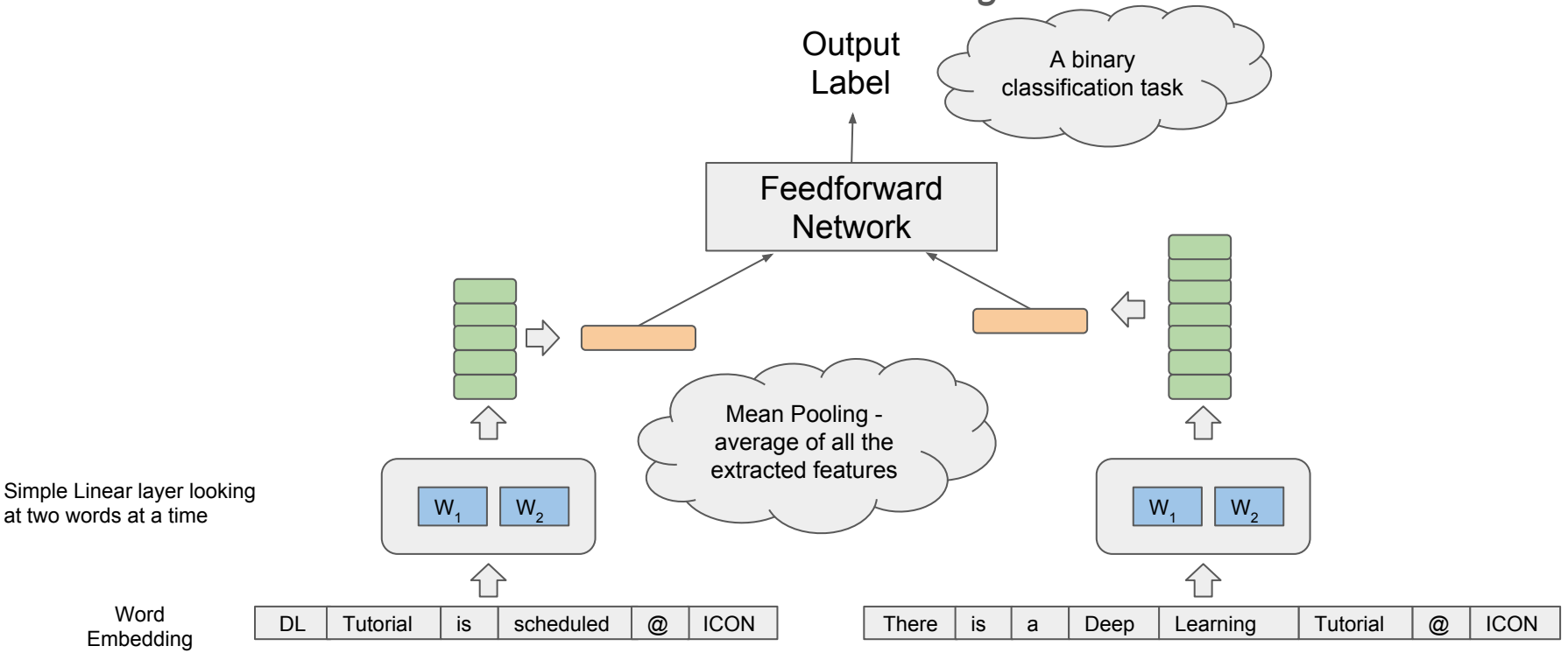
CNNs for Paraphrase Detection

Use Feedforward neural network on consecutive *ngram* words



CNNs for Paraphrase Detection

Use Feedforward neural network on consecutive *ngram* words



CNNs for Paraphrase Detection

- The architecture presented is too simple
- We can have parallel CNNs each looking at ngrams of specific length
- CNNs can be thought of performing composition operation
- We can have hierarchy of CNNs, which composes words to form simple phrases, simple phrases to form complex phrases, complex phrases to sentences, sentences to paragraphs, ...

CNNs in Torch

- First let us create word embedding matrix
  - embed = nn.LookupTableMaskZero(sourceDictionarySize, embeddingDimension)
- Given any word we send it through LookupTable to get the corresponding word embeddings
  - outputWordEmbed = embed:forward(inputWords)
- Now let us Create CNN module
  - cnn = nn.Sequential()
  - cnn:add(embed)
  - cnn:add(nn.TemporalConvolution(embeddingDimension , filterSize, nGrams))
  - cnn:add(nn.Tanh()) -- *Optional non-linearity*
  - cnn:add(nn.Max(1))

Thank You

Questions?

References

- Zeiler, Matthew D. and Fergus, Rob (2014). Visualizing and Understanding Convolutional Networks. Computer Vision -- ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I. Springer International Publishing
- Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. J. Mach. Learn. Res.,
- dos Santos, C., Guimaraes, V., Niterói, R., and de Janeiro, R. (2015). Boosting named entity recognition with neural character embeddings. Proceedings of NEWS 2015 The Fifth Named Entities Workshop, page 9.
- Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional LSTM-CRF models for sequence tagging. CoRR, abs/1508.01991

References

- Lample, G., Ballesteros, M., Kawakami, K., Subramanian, S., and Dyer, C. (2016). Neural architectures for named entity recognition. In In proceedings of NAACL-HLT (NAACL 2016)., San Diego, US