

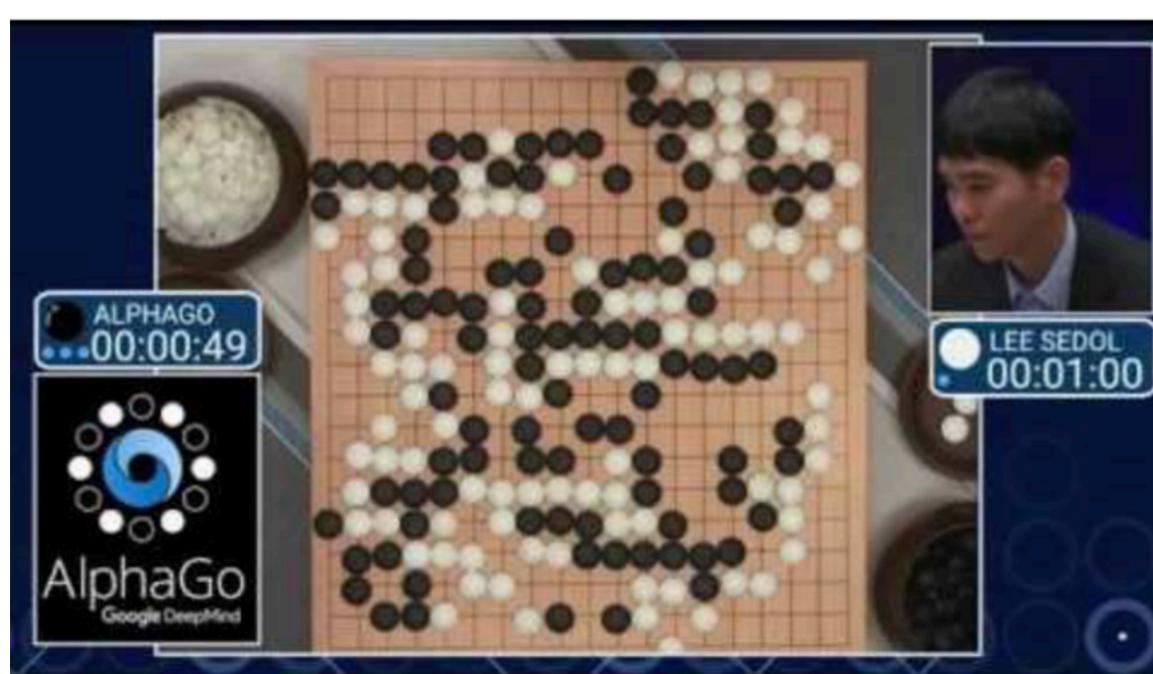
Deep Learning, NLP, and AI

Mar, 2021

Bio

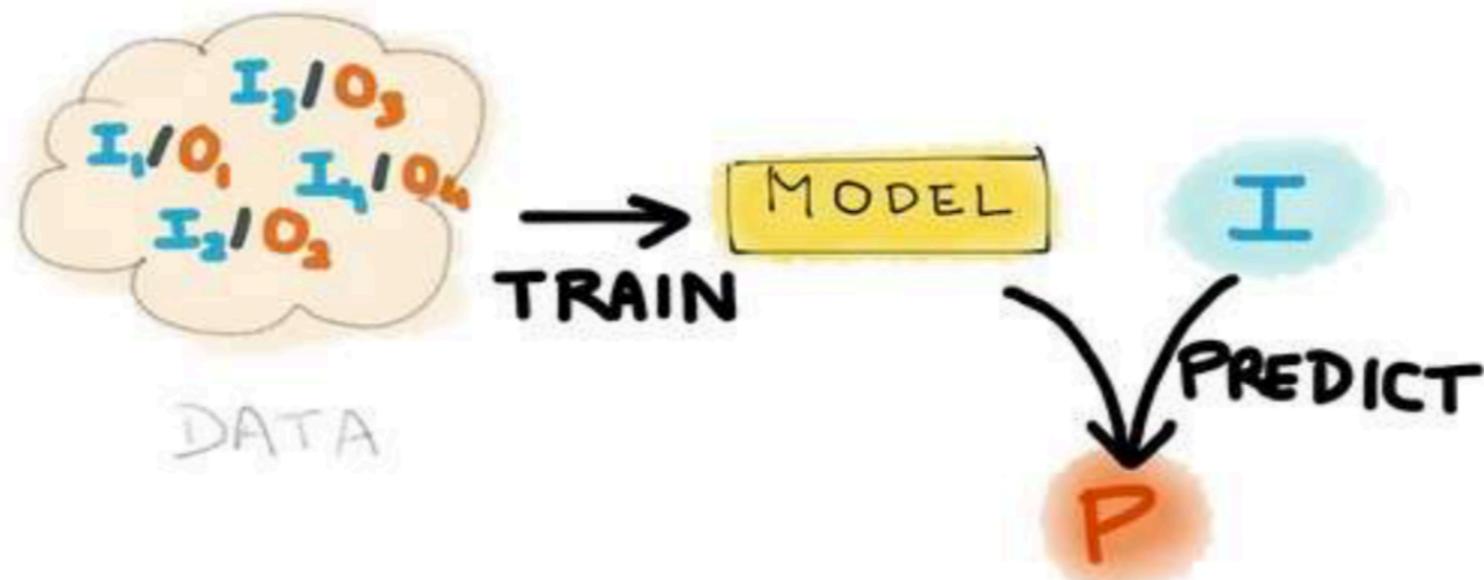
- **Ravi Ilango**
- Director - DataScience, Stealth Startup, SantaClara
- 10+ Years at Apple, Sr Data Scientist at FogHorn Systems, Sr DataScientist at StatesTitle
- Education:
 - BE Mech (Madras University, India)
 - Masters Program in Aero and Production (IIT Madras, India)
 - MBA (Santa Clara University)
 - Graduate Certificate in Data Mining and Machine Learning (Stanford)
- email: ravi.ilango@gmail.com
- LinkedIn: <https://www.linkedin.com/in/raviilango/>

Popular AI



What is machine learning

- Subfield of computer science that explores how machines can learn to perform certain task without explicit programming

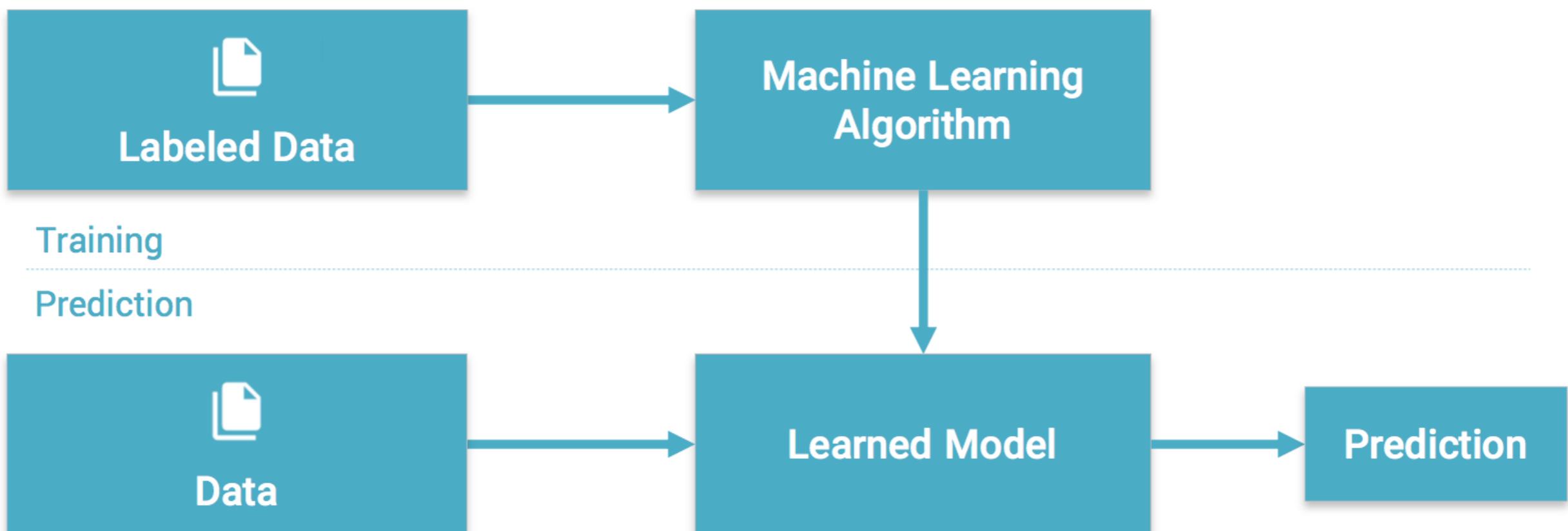


Machine Learning - Basics

Introduction

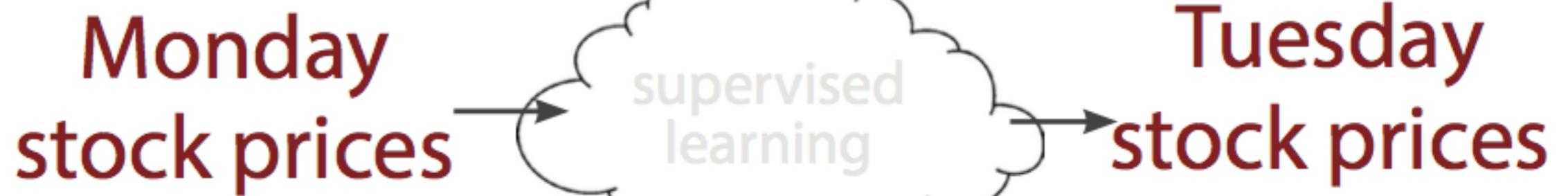


Machine Learning is a type of Artificial Intelligence that provides computers with the ability to **learn without being explicitly programmed**.

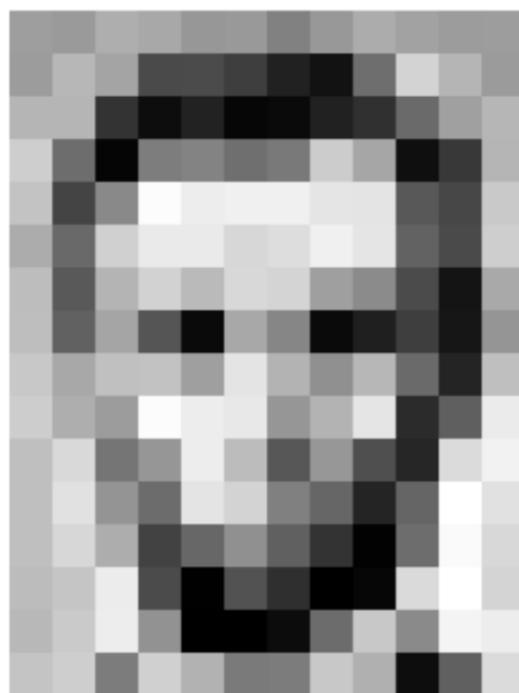


Provides **various techniques** that can learn from and make predictions on data

Supervised



Tasks in Computer Vision



Input Image

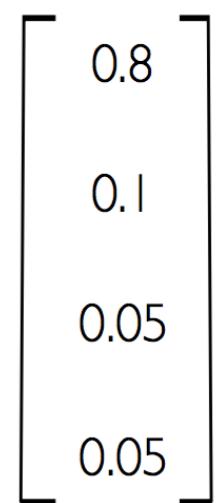


| | | | | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 157 | 153 | 174 | 168 | 150 | 152 | 129 | 151 | 172 | 161 | 155 | 156 |
| 155 | 182 | 163 | 74 | 75 | 62 | 33 | 17 | 110 | 210 | 180 | 154 |
| 180 | 180 | 50 | 14 | 34 | 6 | 10 | 33 | 48 | 106 | 159 | 181 |
| 206 | 109 | 5 | 124 | 131 | 111 | 120 | 204 | 166 | 15 | 56 | 180 |
| 194 | 68 | 137 | 251 | 237 | 239 | 239 | 228 | 227 | 87 | 71 | 201 |
| 172 | 105 | 207 | 233 | 233 | 214 | 220 | 239 | 228 | 98 | 74 | 206 |
| 188 | 88 | 179 | 209 | 185 | 215 | 211 | 198 | 139 | 75 | 20 | 169 |
| 189 | 97 | 165 | 84 | 10 | 168 | 134 | 11 | 31 | 62 | 22 | 148 |
| 199 | 168 | 191 | 193 | 158 | 227 | 178 | 143 | 182 | 106 | 36 | 190 |
| 205 | 174 | 155 | 252 | 236 | 231 | 149 | 178 | 228 | 43 | 95 | 234 |
| 190 | 216 | 116 | 149 | 236 | 187 | 86 | 150 | 79 | 38 | 218 | 241 |
| 190 | 224 | 147 | 108 | 227 | 210 | 127 | 102 | 36 | 101 | 255 | 224 |
| 190 | 214 | 173 | 66 | 103 | 143 | 96 | 50 | 2 | 109 | 249 | 215 |
| 187 | 196 | 235 | 75 | 1 | 81 | 47 | 0 | 6 | 217 | 255 | 211 |
| 183 | 202 | 237 | 145 | 0 | 0 | 12 | 108 | 200 | 138 | 243 | 236 |
| 196 | 206 | 123 | 207 | 177 | 121 | 123 | 200 | 175 | 13 | 96 | 218 |

Pixel Representation

classification

Lincoln
Washington
Jefferson
Obama

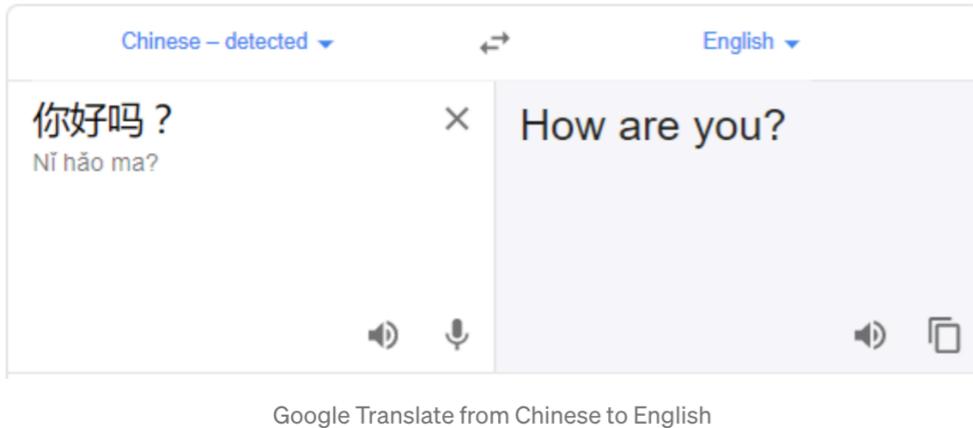


- **Regression:** output variable takes continuous value
- **Classification:** output variable takes class label. Can produce probability of belonging to a particular class

NLP Everyday

Google Translate

used by 500 million people every day



Google Translate from Chinese to English

Chatbot



Our Financial Consultants use the askPRU chatbot to provide a better service experience to customers.

Email assistant

Predictive typing suggests the next word in the sentence.



Google



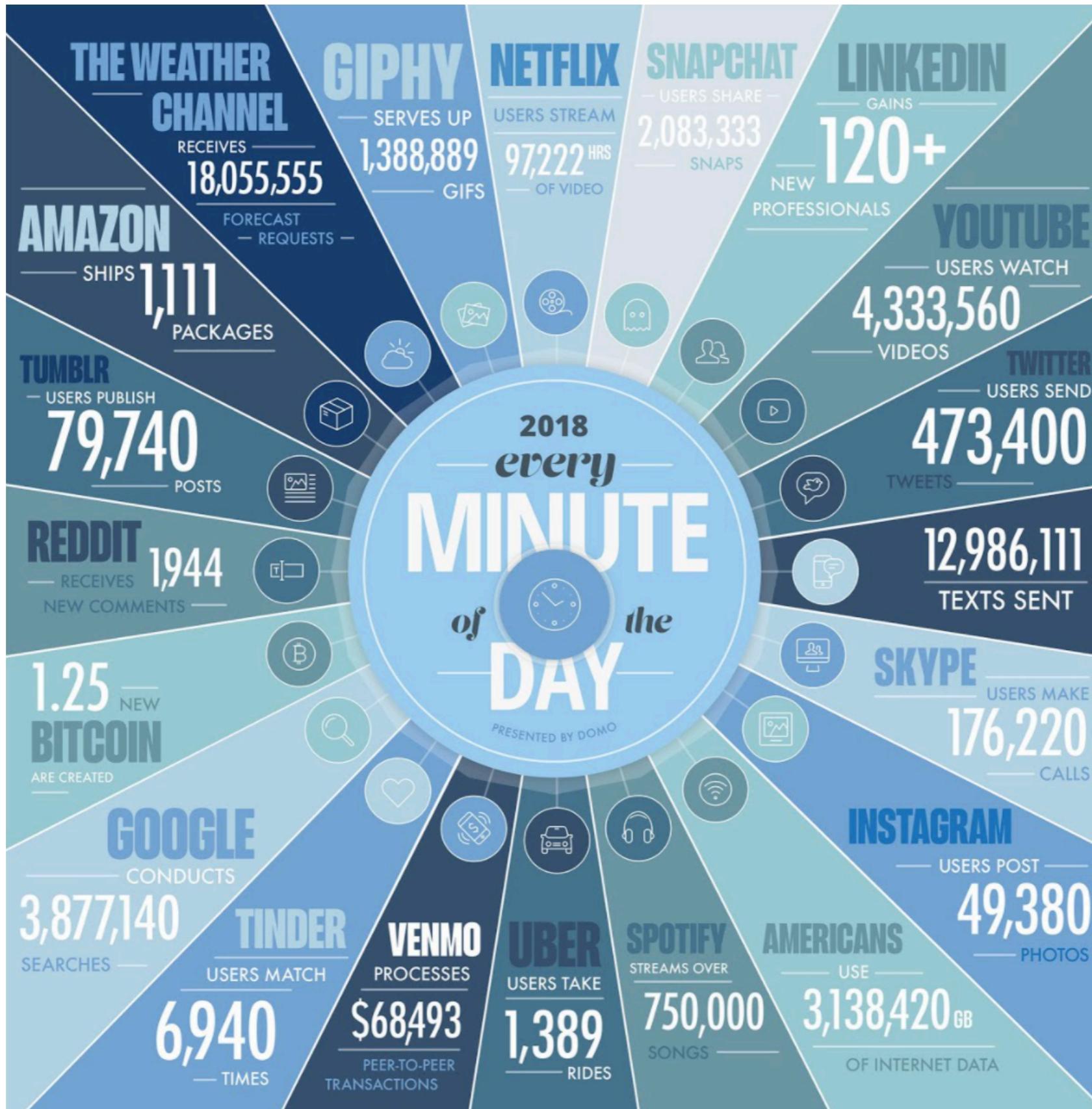
3.5B searches/daily

Extract and Summarization

NLP in business operation

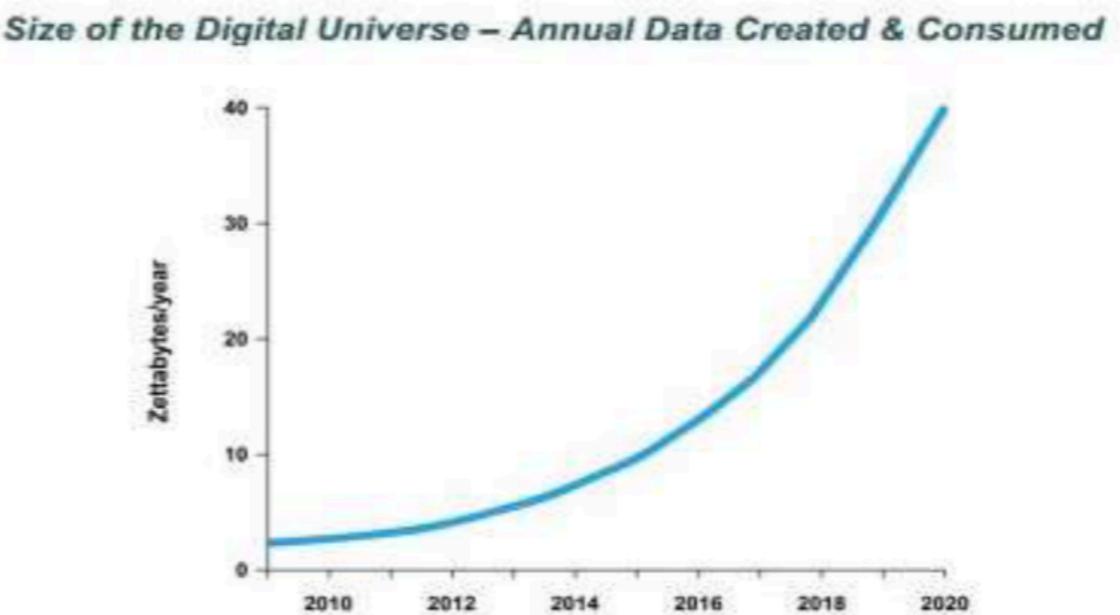
- Achieve efficiencies using machine processing of unstructured text data

Data production rate



Text data

- About 80% of data in organizations are in text format
- Harder to analyse than structured data
- Huge amount of textual documents
 - Only in biomedicine 2200 scientific papers are published every day
- Growing exponentially



NLP Use Cases in Enterprises

- **Text Classification-** It is the process of categorizing or tagging text based on its content
 - organizing customer support tickets, customer reviews, emails
- **Text Extraction-** It is the process of extracting specific information from documents
 - extract key info from business documents (invoices, agreements), emails

Spam Detection

Spam Detection

- data from 4601 emails sent to an individual (named George, at HP labs, before 2000). Each is labeled as *spam* or *email*.
- goal: build a customized spam filter.
- input features: relative frequencies of 57 of the most commonly occurring words and punctuation marks in these email messages.

| | george | you | hp | free | ! | edu | remove |
|-------|--------|------|------|------|------|------|--------|
| spam | 0.00 | 2.26 | 0.02 | 0.52 | 0.51 | 0.01 | 0.28 |
| email | 1.27 | 1.27 | 0.90 | 0.07 | 0.11 | 0.29 | 0.01 |

*Average percentage of words or characters in an email message equal to the indicated word or character. We have chosen the words and characters showing the largest difference between **spam** and **email**.*

Why is language interpretation hard?

Consider the following sentences

“I enjoy working in a bank.”

“I enjoy working near a river bank.”

Understanding language is tough

- Syntax - grammar
- Semantics - meaning
- Pragmatics - what the text is trying to achieve

Ambiguity

Resolving ambiguity is hard

Ambiguity

Find at least 6 meanings of this sentence:

I made her duck

Ambiguity

Find at least 6 meanings of this sentence:

I made her duck

I cooked waterfowl for her benefit (to eat)

I cooked waterfowl belonging to her

I created the (plaster?) waterfowl she owns

I caused her to quickly lower her head or body

I recognized the true identity of her spy waterfowl

I waved my magic wand and turned her into undifferentiated waterfowl

Ambiguity is Pervasive

I caused her to quickly lower her head or body

Part of speech: “duck” can be a Noun or Verb

I cooked waterfowl belonging to her.

Part of speech:

“her” is possessive pronoun (“of her”)

“her” is dative pronoun (“for her”)

I made the (plaster) duck statue she owns

Word Meaning : “make” can mean “create” or “cook”

Ambiguity is Pervasive: Phonetics!!!!

Aye mate, her duck

I mate or duck

I'm eight or duck

Eye maid; her duck

I maid her duck

I'm aid her duck

I mate her duck

I'm ate her duck

I'm ate or duck

I mate or duck



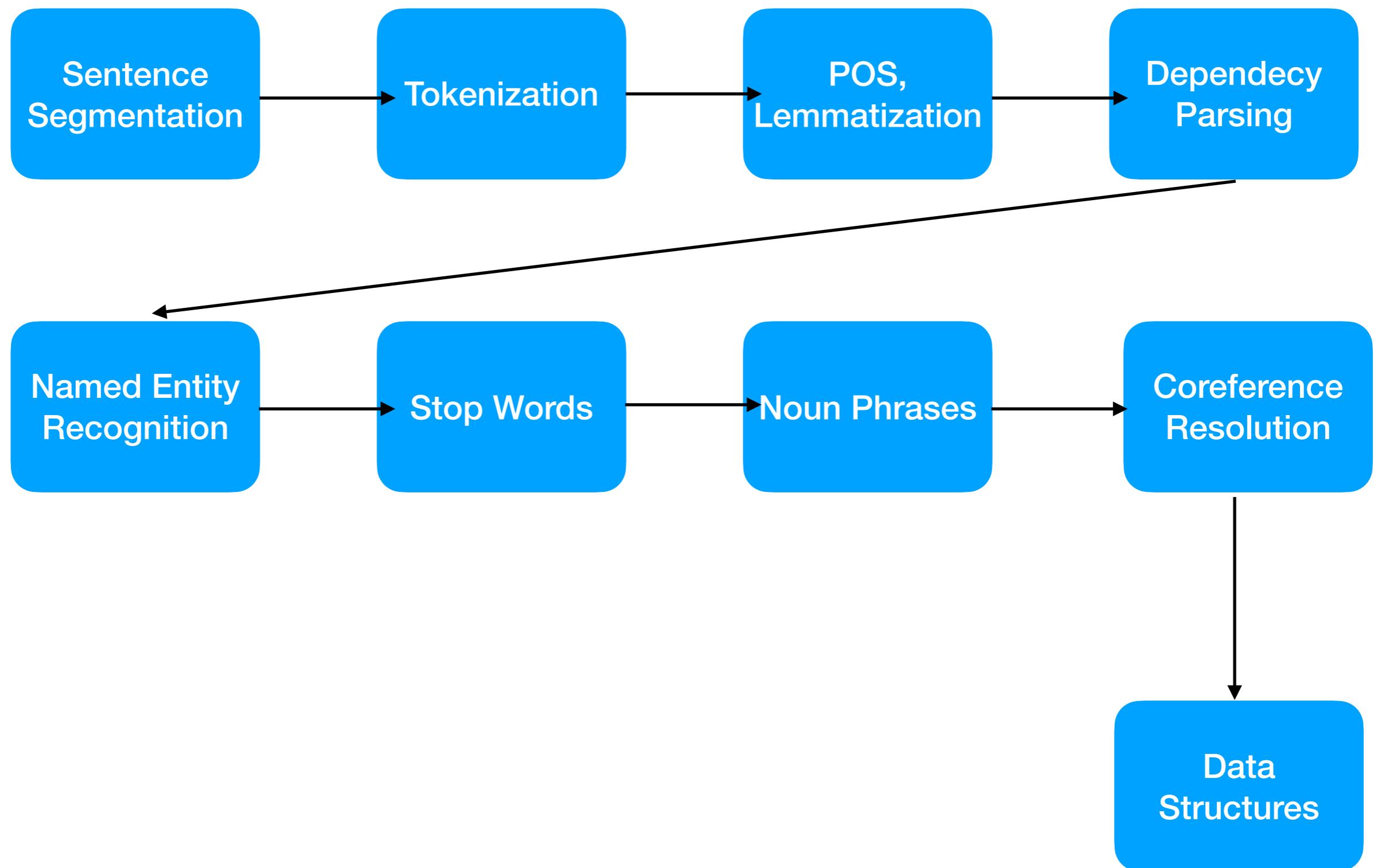
More difficulties:
Non-standard language,
emojis, hashtags,
names



chowdownwithchan #crab and #pork #xiaolongbao at
@dintaifungusa... where else? 😂🤷‍♀️ Note the cute little
crab indicator in the 2nd pic 🦀💕



NLP Pipeline to produce representative data structures



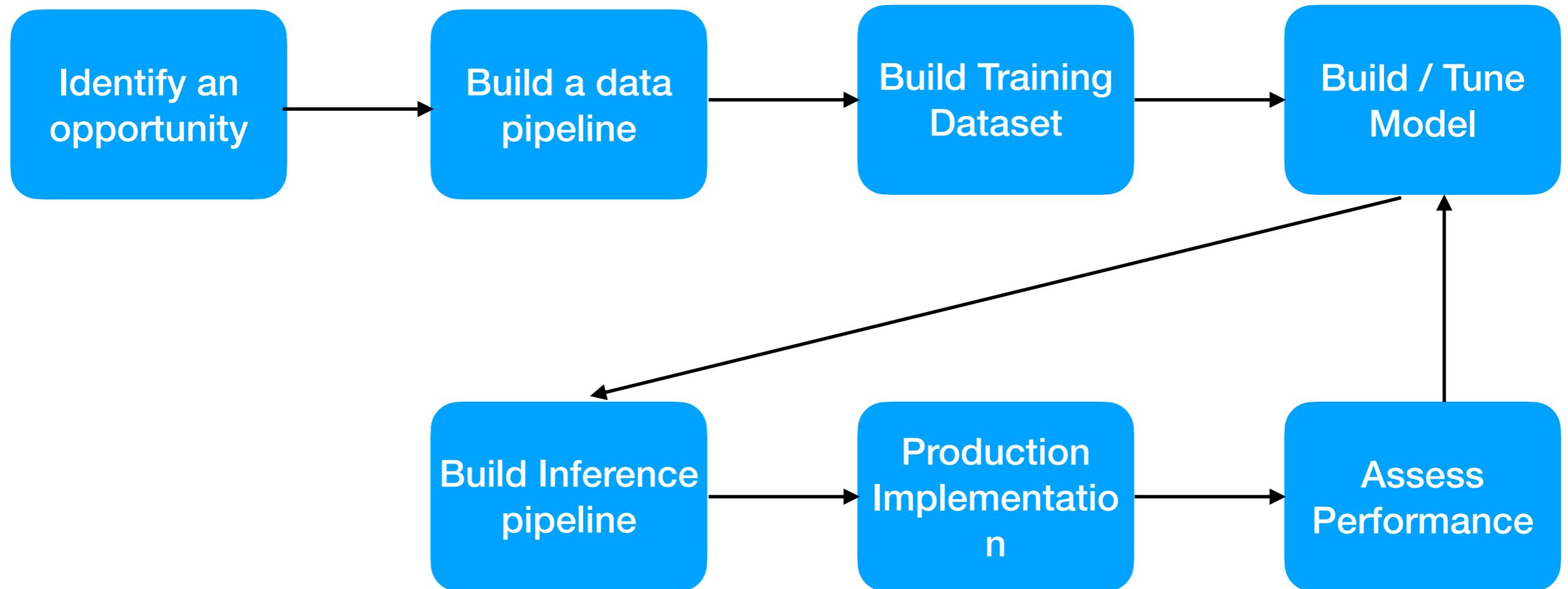
NLP Data Scientist - (some) tools



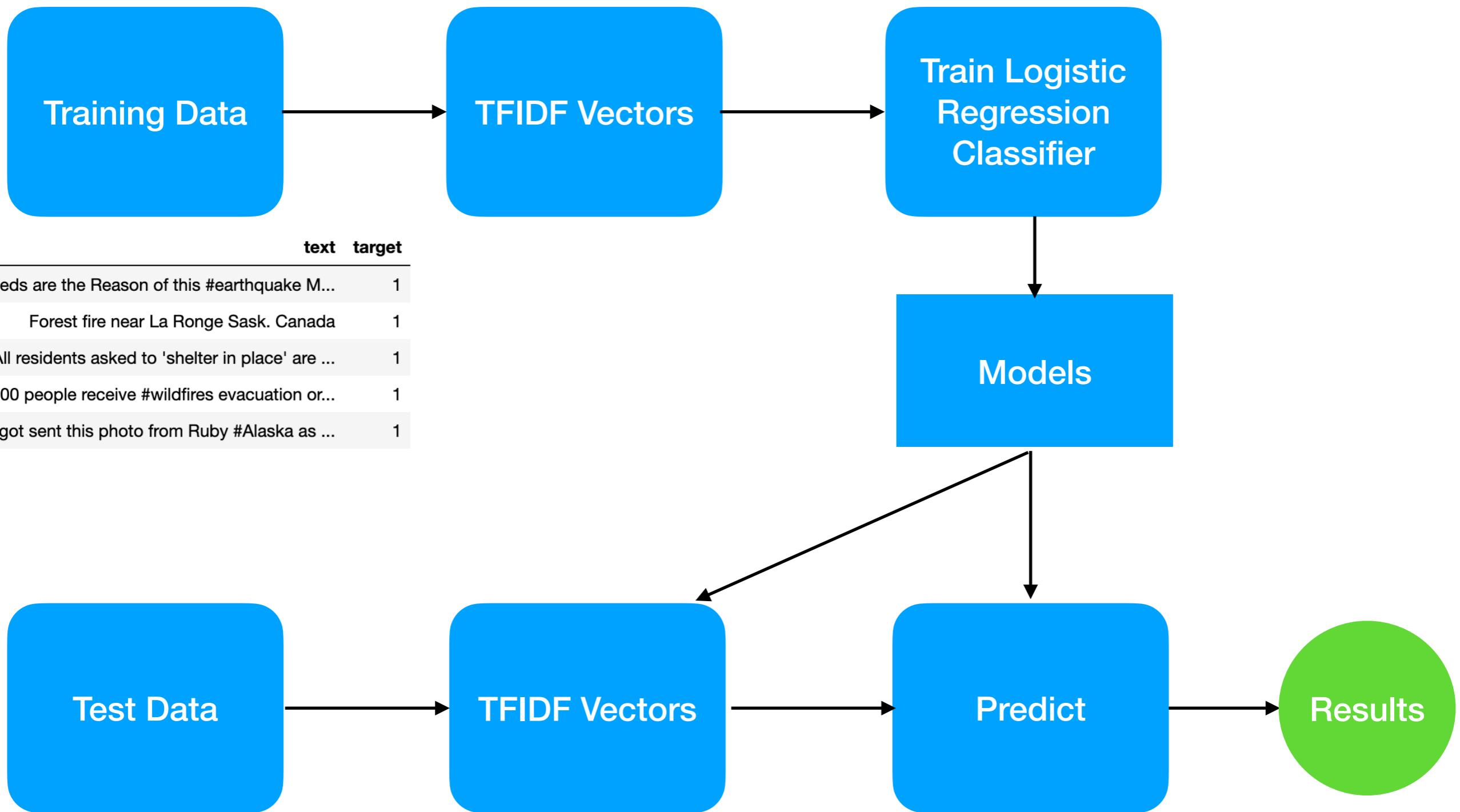
NLP Data Scientist - (some) concepts & techniques

- Word vectors, tokenization, document vectorization
- POS (parts of speech), NER
- Document parsing techniques, Regex
- Supervised ML, Classifiers, Deep Learning and how to use word embeddings
- Using Pretrained models

Typical DS model implementation



Lab: Disaster Detection



Lab: Disaster Detection using TFIDF and Logistic Regression

Click this link to begin

[https://colab.research.google.com/github/ravi-ilango/aicamp-mar-2021/blob/main/lab2/
disaster_detection_tfidf.ipynb](https://colab.research.google.com/github/ravi-ilango/aicamp-mar-2021/blob/main/lab2/disaster_detection_tfidf.ipynb)

Word Embedding

“You shall know a word by the company it keeps”
(J. R. Firth 1957)



One of the most successful ideas of modern statistical NLP!

government debt problems turning into banking crises as has happened in saying that Europe needs unified banking regulation to replace the hodgepodge

these words represent banking

Vector Space Model

- Neural word embeddings
- Combine vector space semantics with the prediction of probabilistic models
- Words are represented as a **dense** vector:

$$\text{Candy} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

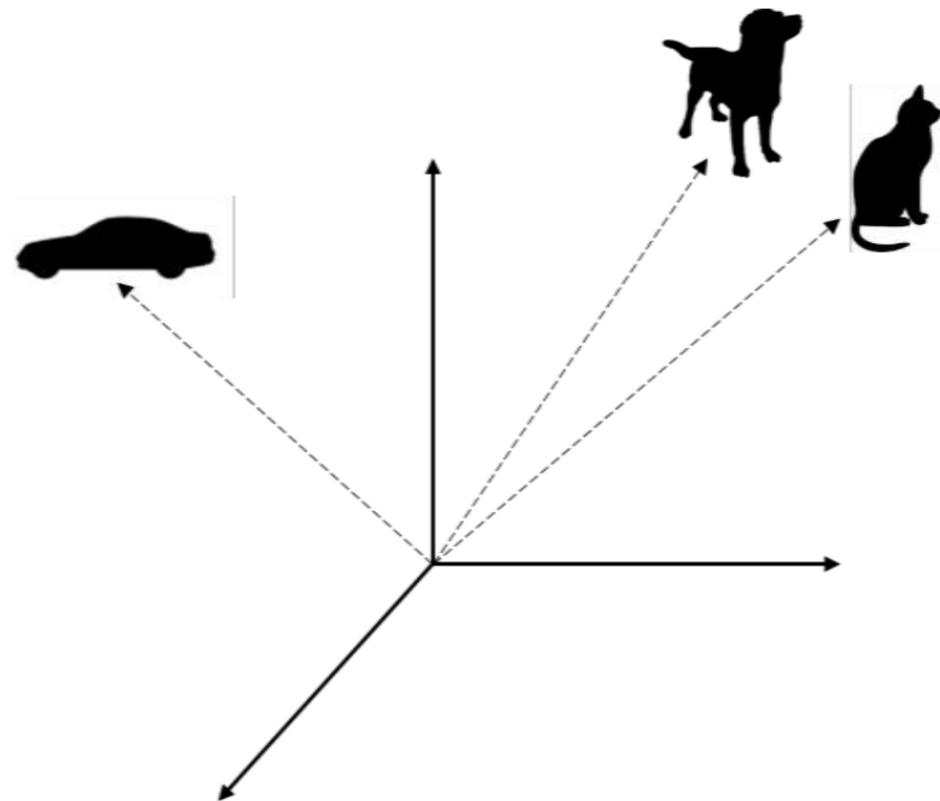
Vector Space Model

Embeddings

The vectors we have been discussing so far are very high-dimensional (thousands, or even millions) and sparse.

But there are techniques to learn lower-dimensional dense vectors for words using the same intuitions.

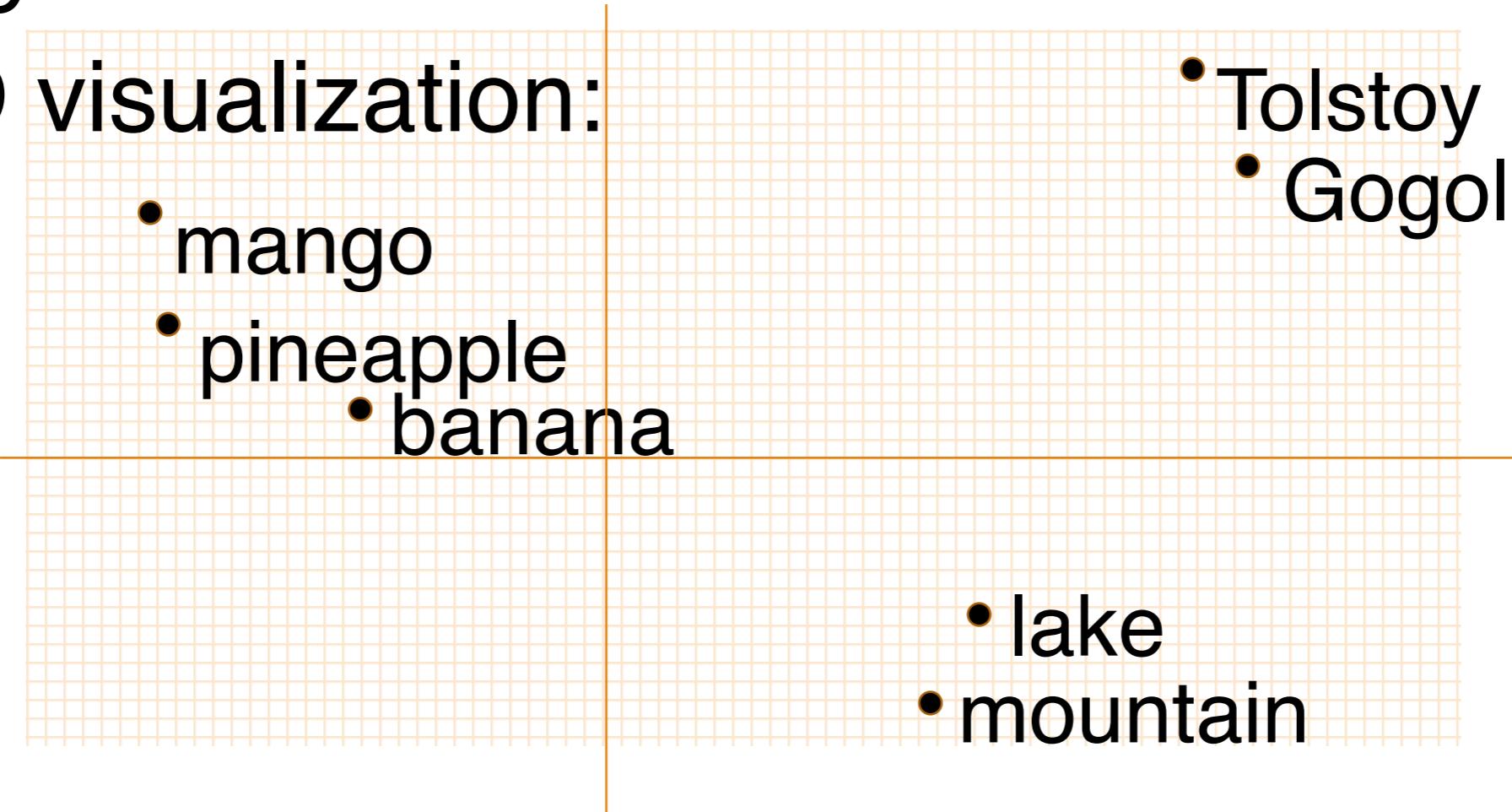
These dense vectors are called embeddings.



Neural word embeddings

A word's meaning is a point in 300-dimensional space

A 2-D visualization:



Embeddings are the core of NLP

Core technology for any NLP task (question answering, machine translation, information retrieval, etc)

- Finding synonyms for words
- Deciding if two sentences have similar meaning

How to learn these "embeddings"?

**Push words together in space they occur
together in text**

Read millions of words.

When you see:

Banana, mango, or pineapple are all delicious...

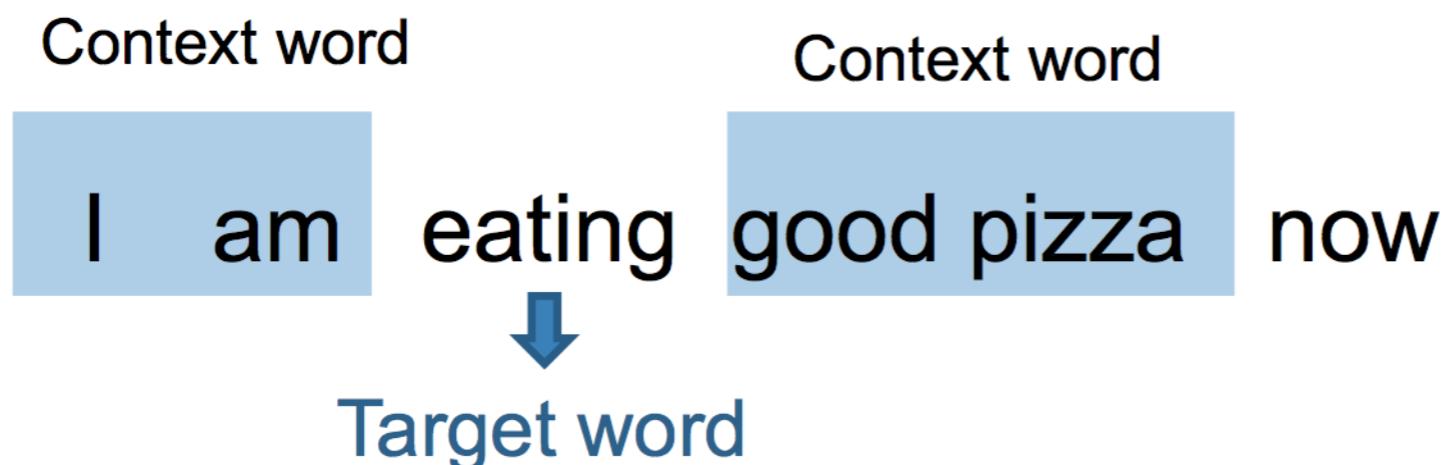
Move banana closer to mango

Move banana further from Tolstoy

Continuous Bag-of-Words Model

► Continuous Bag-of-Words Model

- **Predict target word by the context words**
- Eg: Given a sentence and window size 2



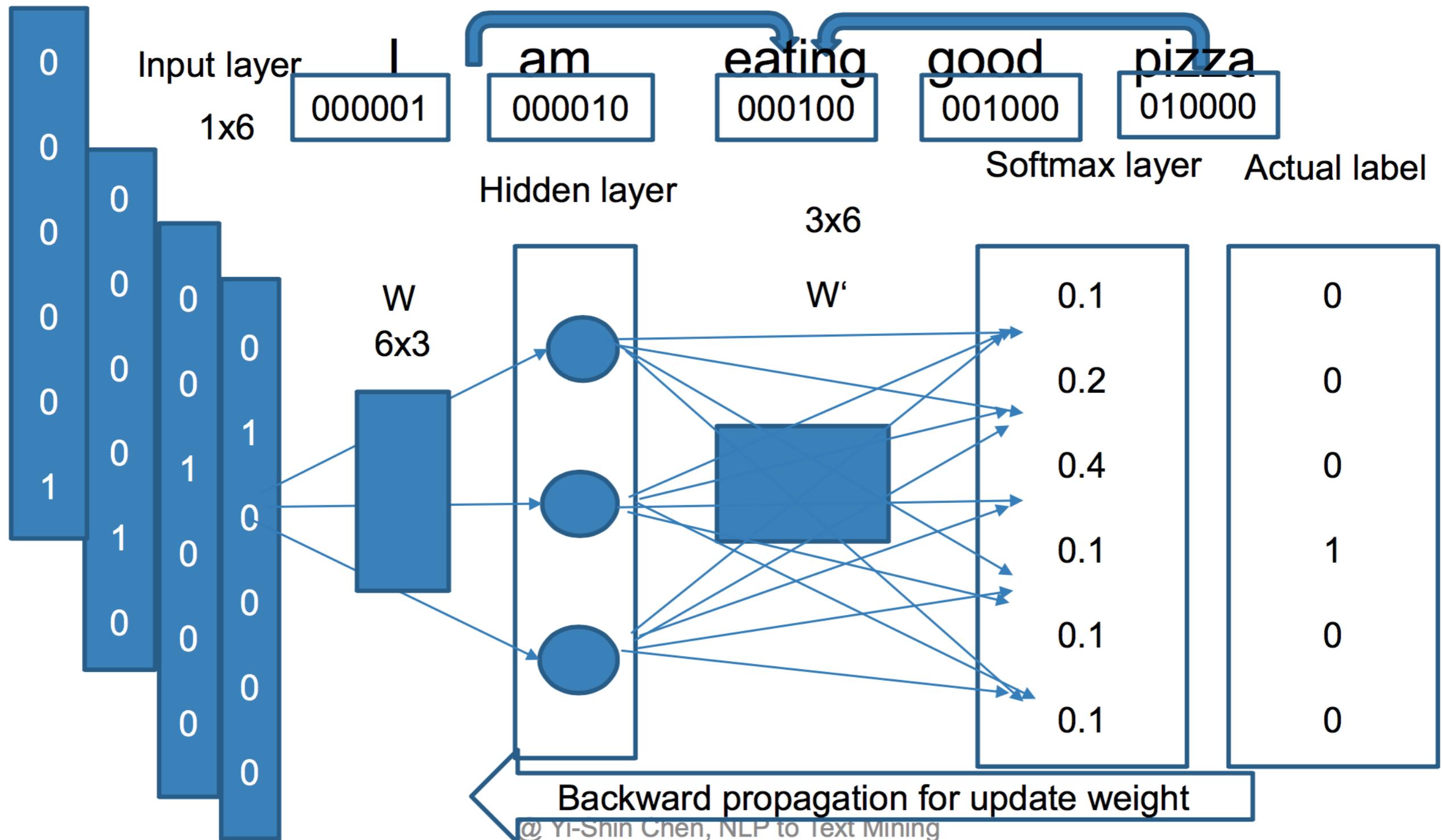
Ex: ([features], label)

([I, am, good, pizza], eating), ([am, eating, pizza, now], good) and so on and so forth.

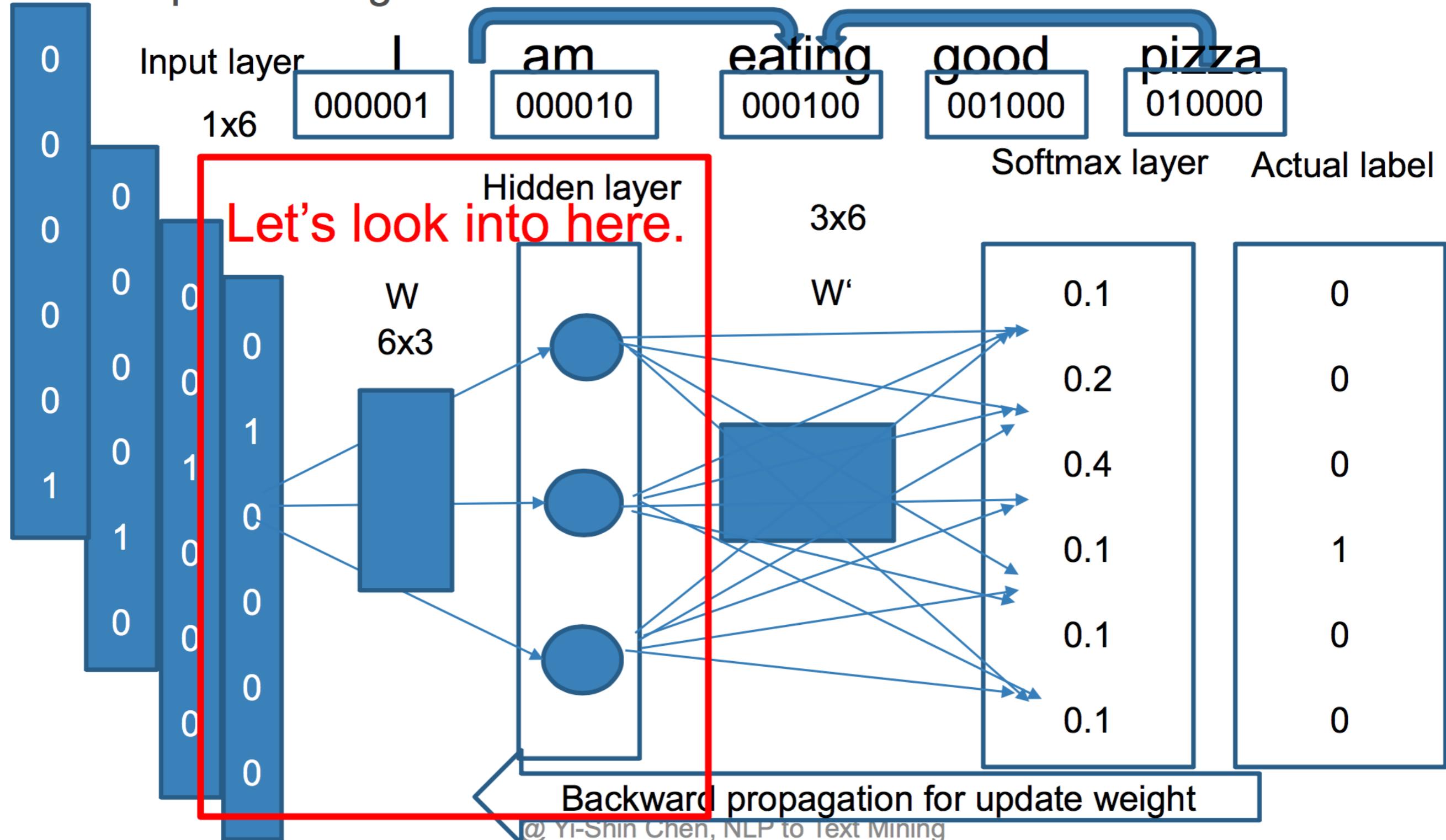
Word Embedding in a Deep Learning network

Continuous Bag-of-Words Model (Contd.)

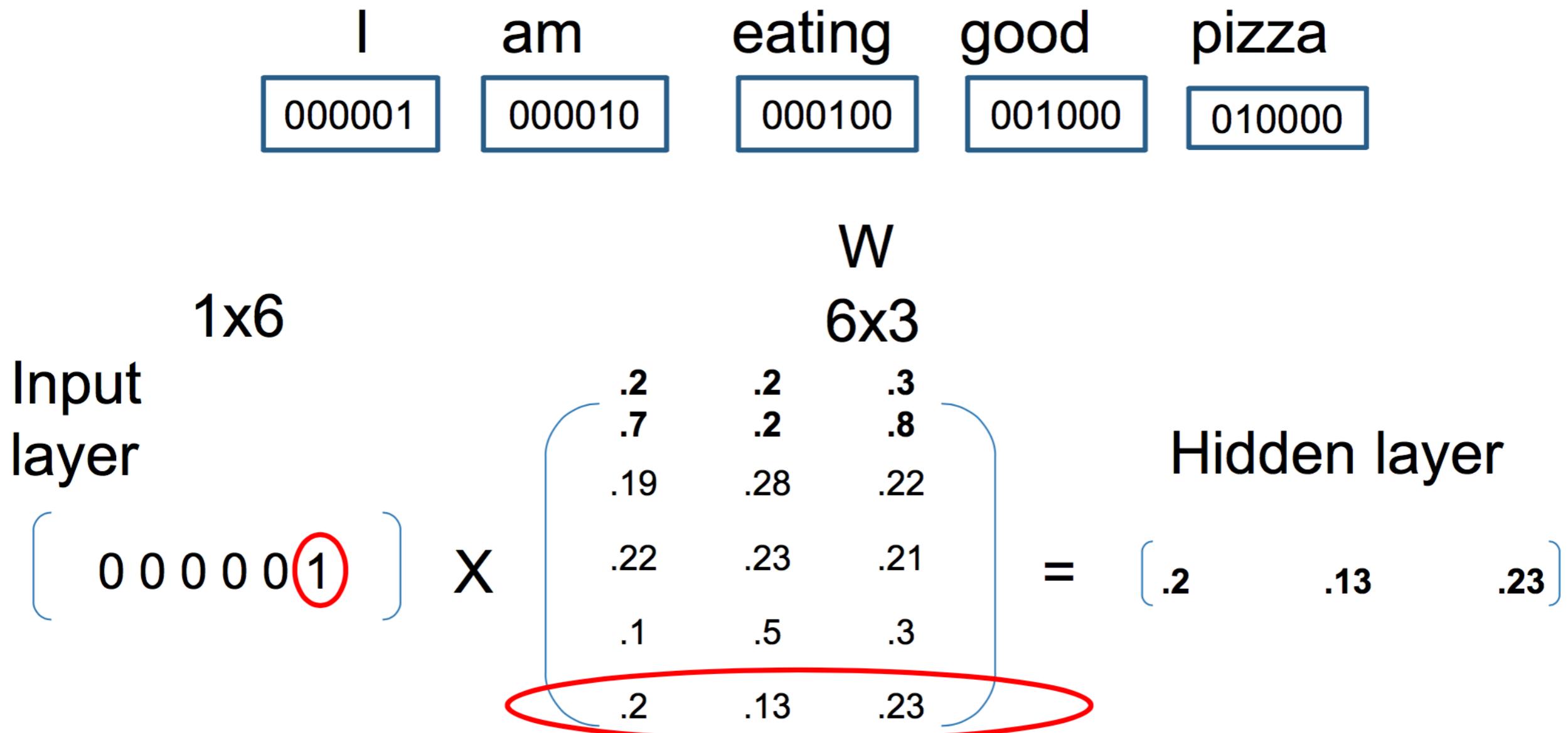
- We train one depth neural network, set hidden layer's width 3.



- ▶ The goal is to extract word representation vector instead of prob of predict target word.

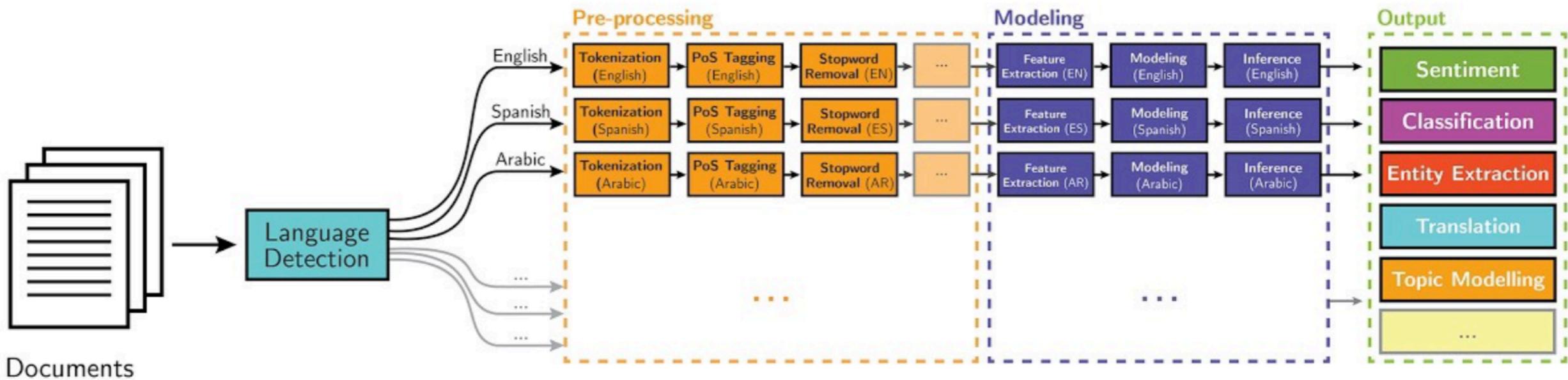


Continuous Bag-of-Words Model (Contd.)

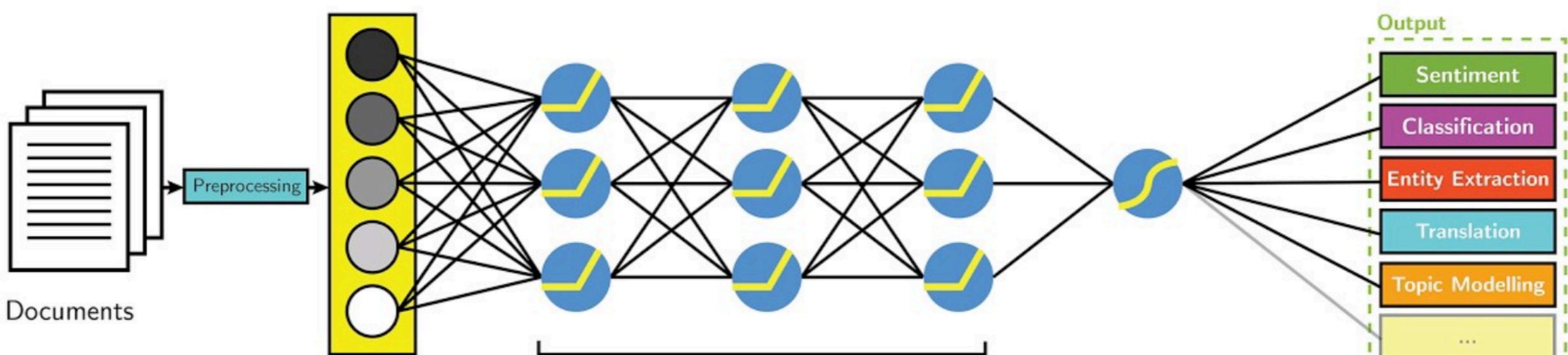


- The output of the hidden layer is just the “word vector” for the input word

NLP



Deep Learning-based NLP



Why Recurrent Neural Network?

Two main limitations of other deep nets are:

- **Data points with dependencies:** ANNs can't deal with sequential or "temporal" data.

Time Series

Gene
Sequences

Sentences

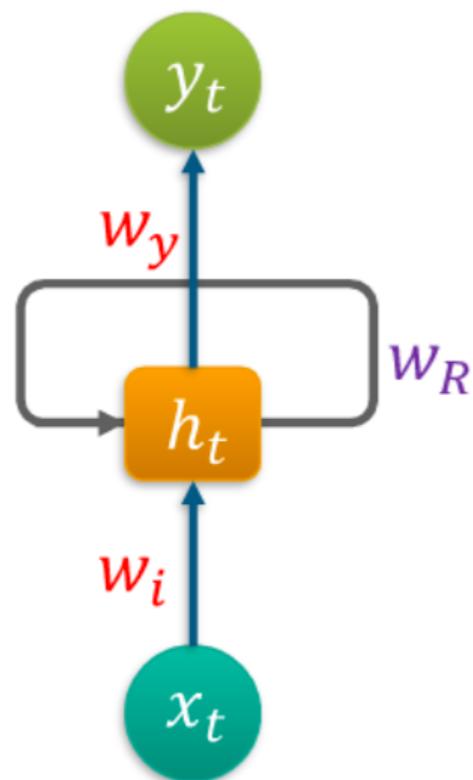
Stock prices

- **Independence:** Traditional neural network assumes that each data point is independent of other data points (inputs are analysed in isolation)

For sequential data the length of the input sequences can vary from example to example. Also In sequences there will be short and long temporal dependencies among the words, thus normal deep networks can't be used for sequential data

Structure of RNN

- The basic elements of a recurrent neuron is given below:



Here, activity $h^{(t)}$ depends on both input $x^{(t)}$ and a recurrent connection with weight w_R

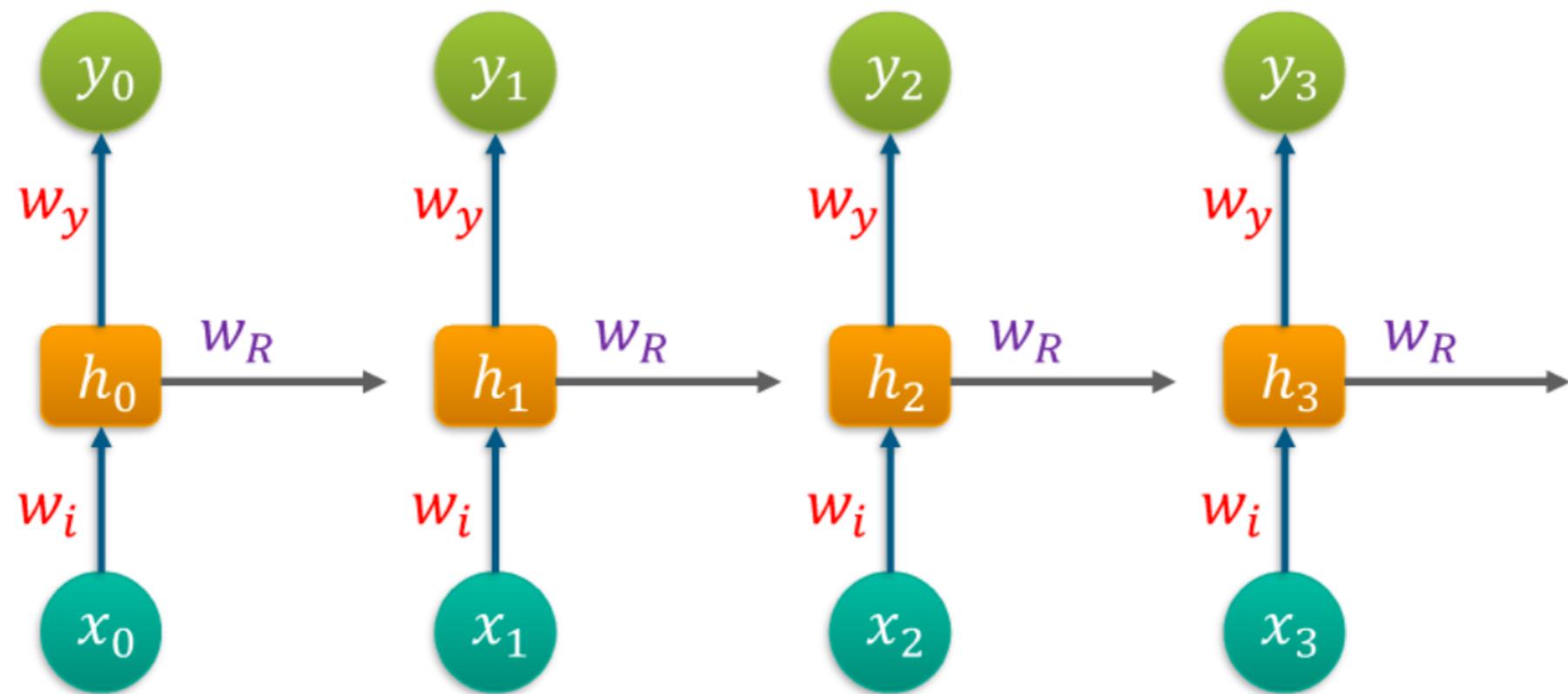
$$h^{(t)} = g_h (w_i x^{(t)} + w_R h^{(t-1)} + b_h)$$
$$y^{(t)} = g_y (w_y h^{(t)} + b_y)$$

Activity at previous
time stamp

Labelling of parameters used

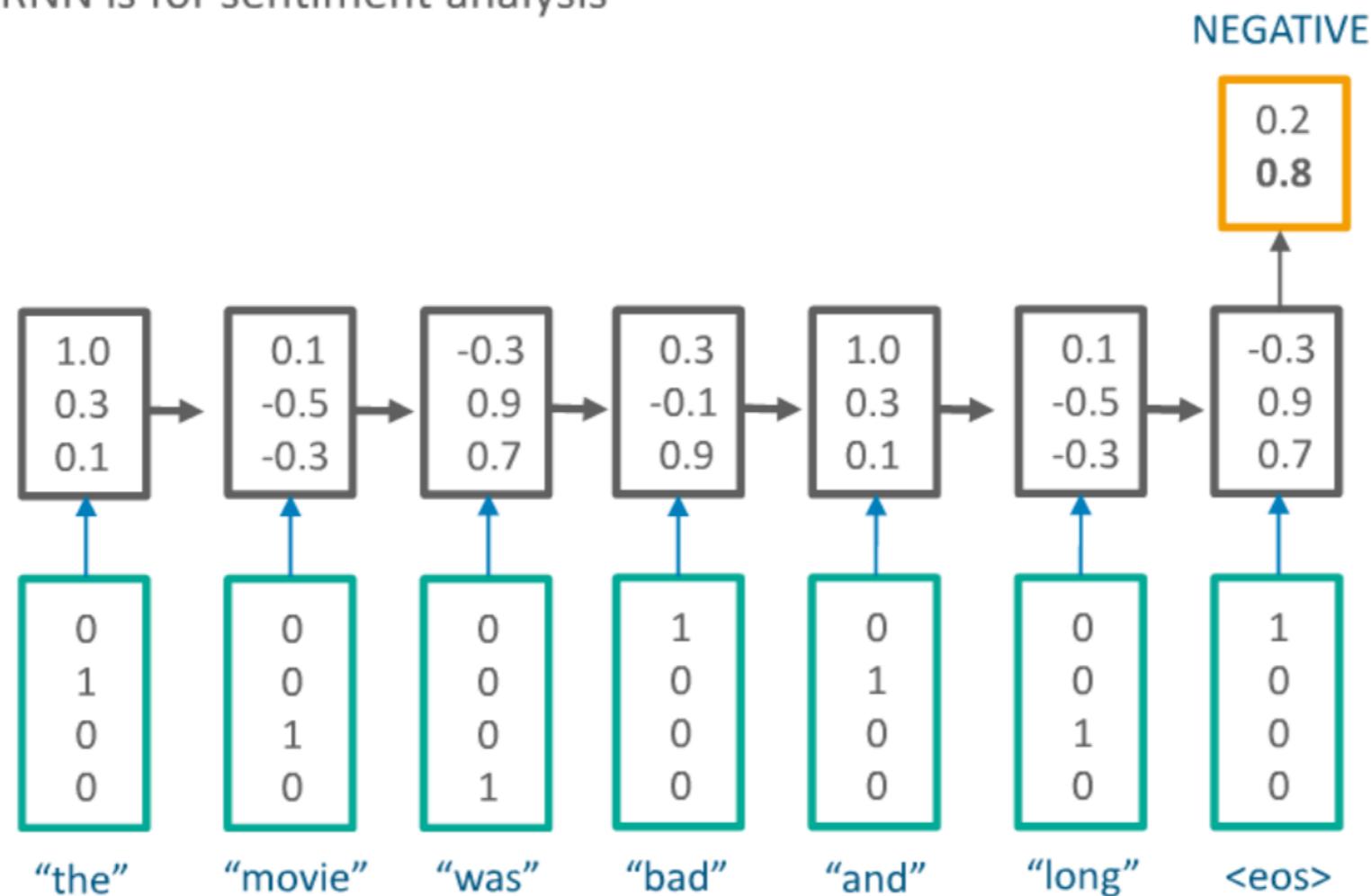
Structure of RNN

- By unrolling an RNN in time, we will get a feed forward network as:



Example

- Another use of RNN is for sentiment analysis



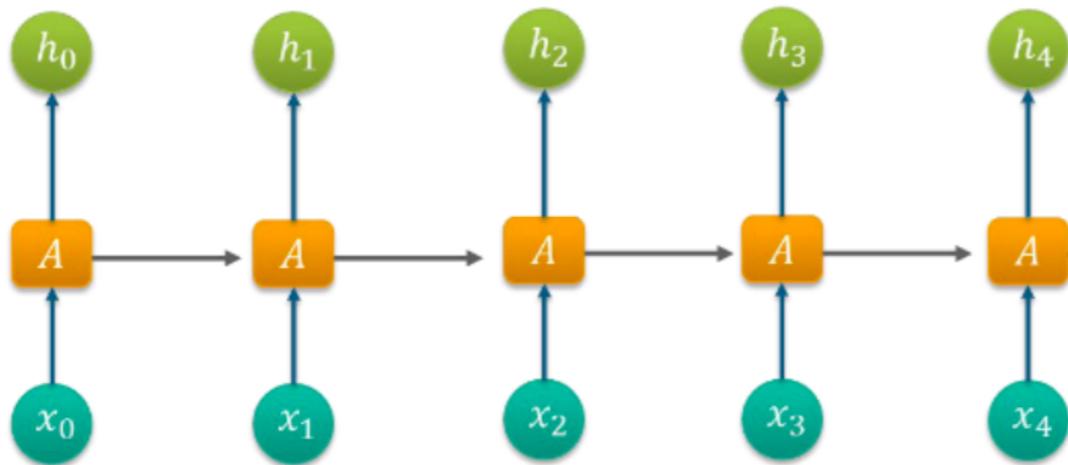
What are LSTMs?

- Long Short Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of **learning long-term dependencies**.
- LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behaviour.

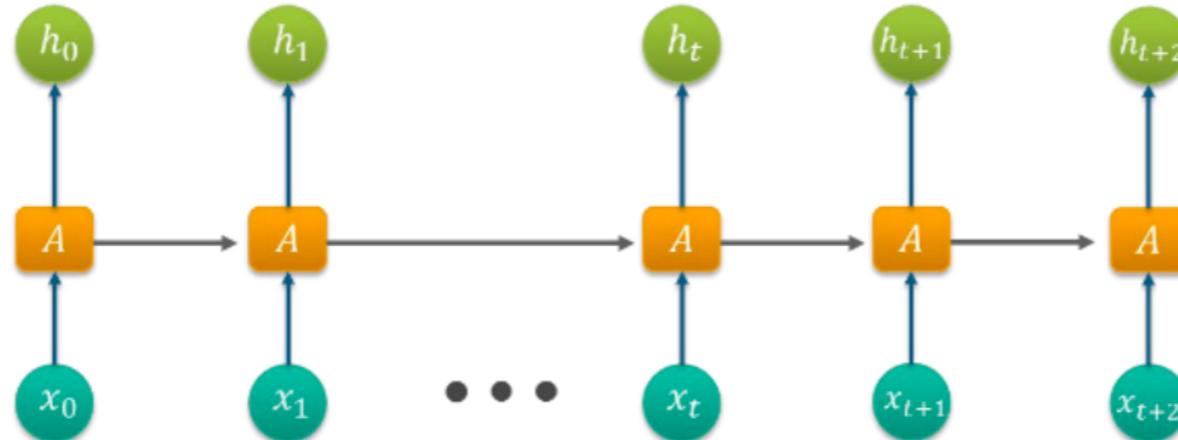
What are LSTMs?

Consider an example where we want to predict the last word of a given text.

- **Short Term Memory**

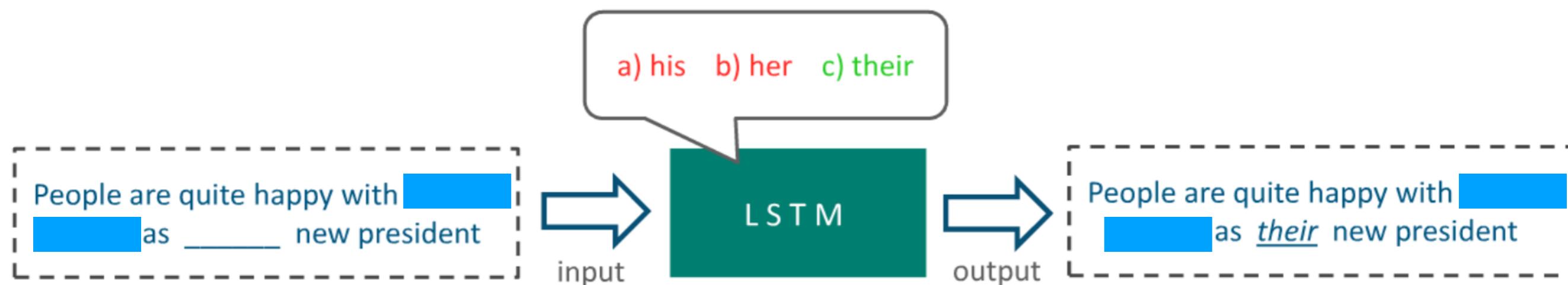


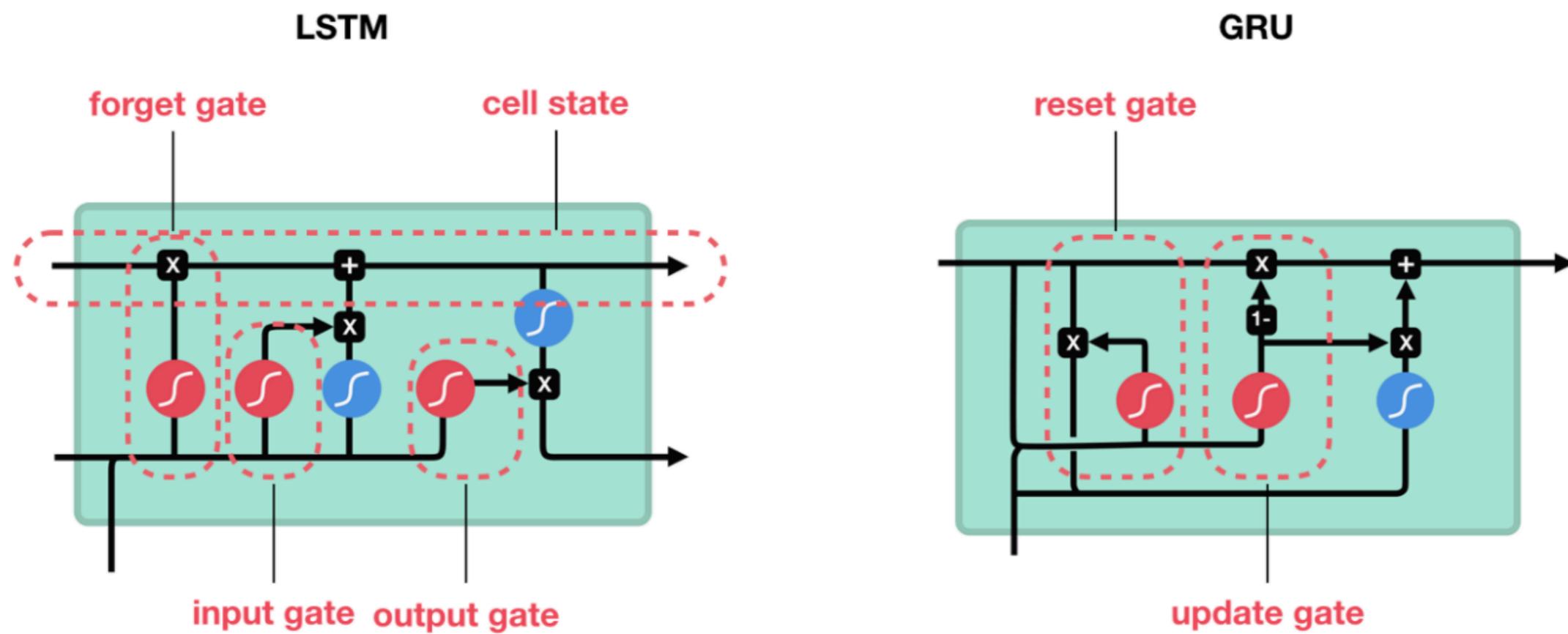
- **Long Term Memory**



LSTM Working Steps

- For the language model example, since it just saw a subject, it might want to output information relevant to a verb, in case that's what is coming next.
- For example, it might output whether the subject is singular or plural, so that we know what form a verb should be conjugated into if that's what follows next.





sigmoid



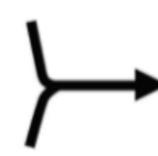
tanh



pointwise
multiplication

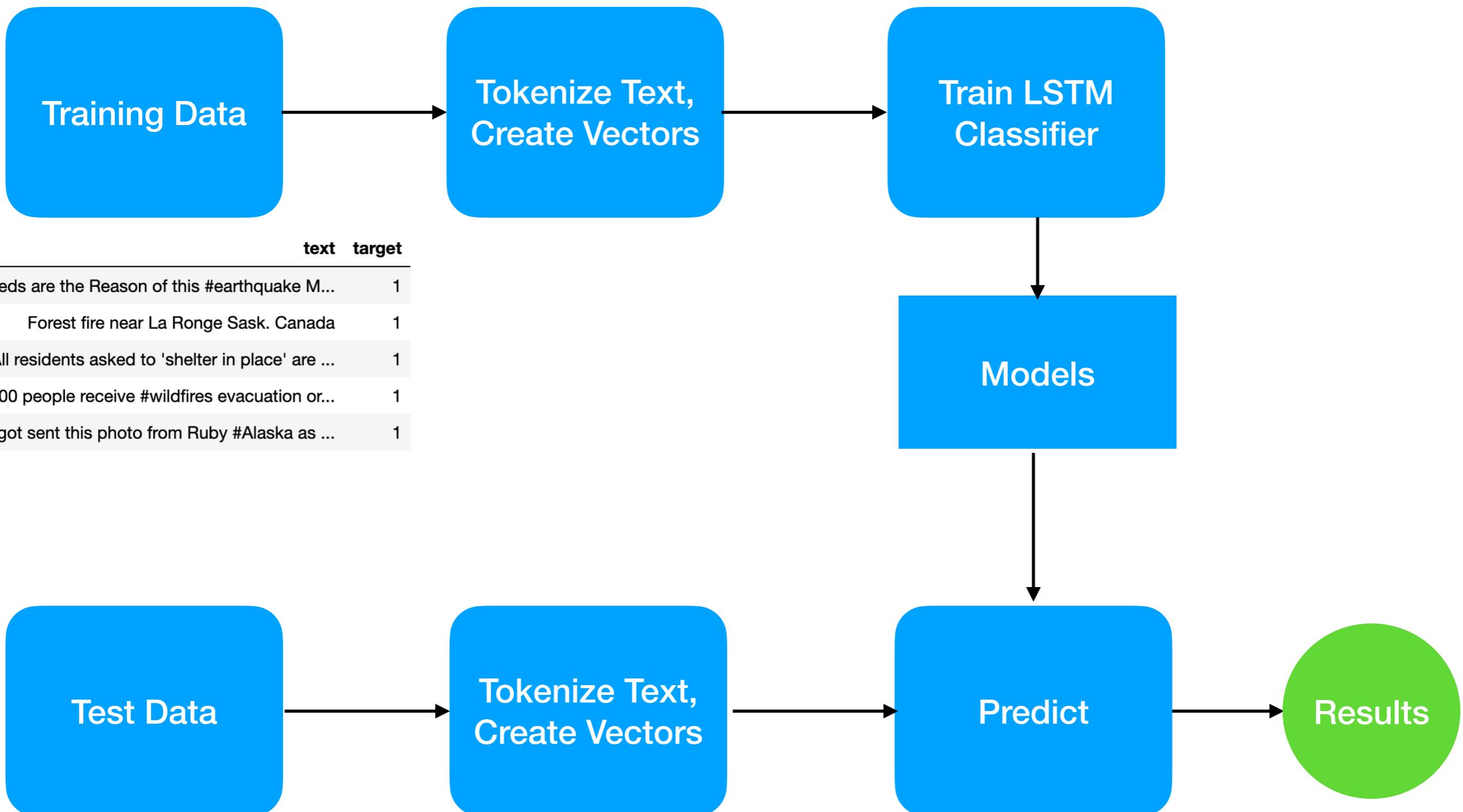


pointwise
addition



vector
concatenation

Lab: Quora Comment Classifier



Lab: Quora Insincere Questions Classification using LSTM

Click this link to begin

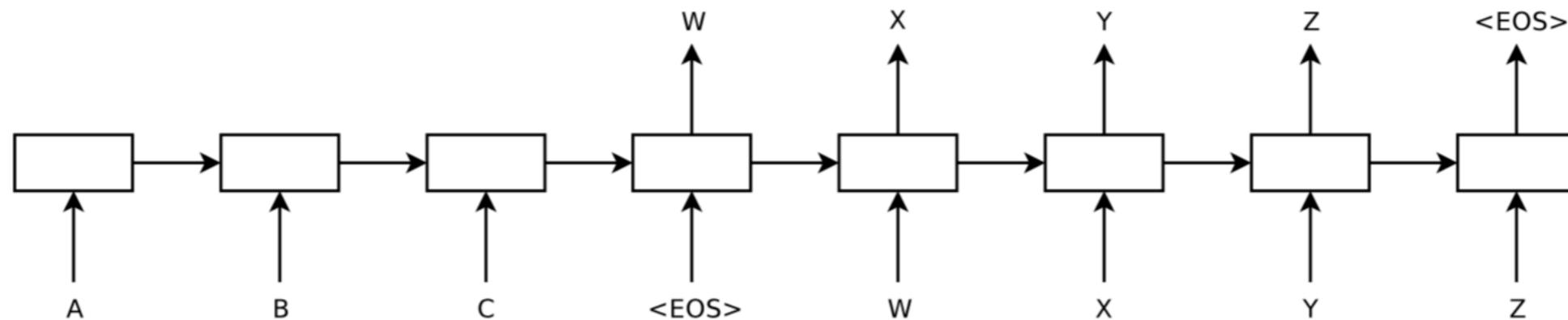
https://colab.research.google.com/github/ravi-ilango/aicamp-mar-2021/blob/main/lab2_1/quora_classifier_lstm.ipynb

Sequence to Sequence Model using RNN

- Sequence-to-sequence learning (Seq2Seq) is about training models to convert sequences from one domain (e.g. sentences in English) to sequences in another domain (e.g. the same sentences translated to French)

Sequence-to-sequence learning

Sequence-to-sequence learning (Seq2Seq) is about training models to convert sequences from one domain (e.g. sentences in English) to sequences in another domain (e.g. the same sentences translated to French)



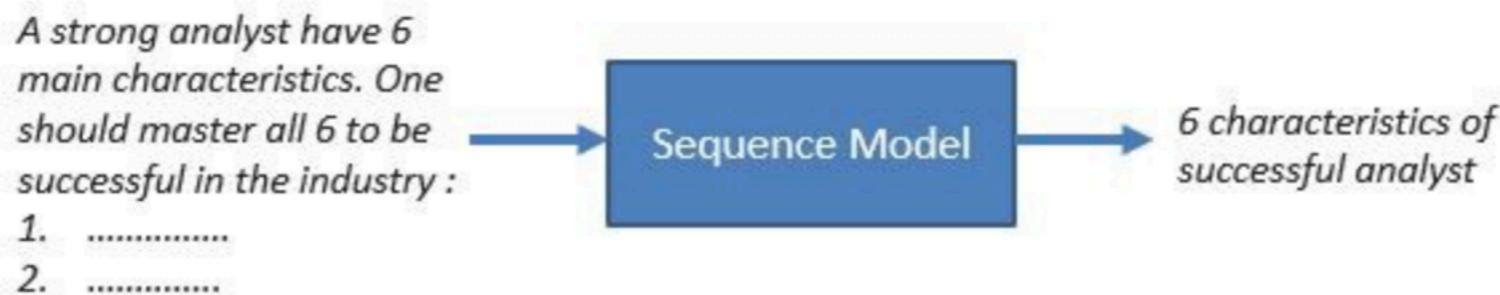
model reads an input sentence “ABC” and produces “WXYZ” as the output sentence

Sequence-to-sequence learning - use cases

Machine Language Translation



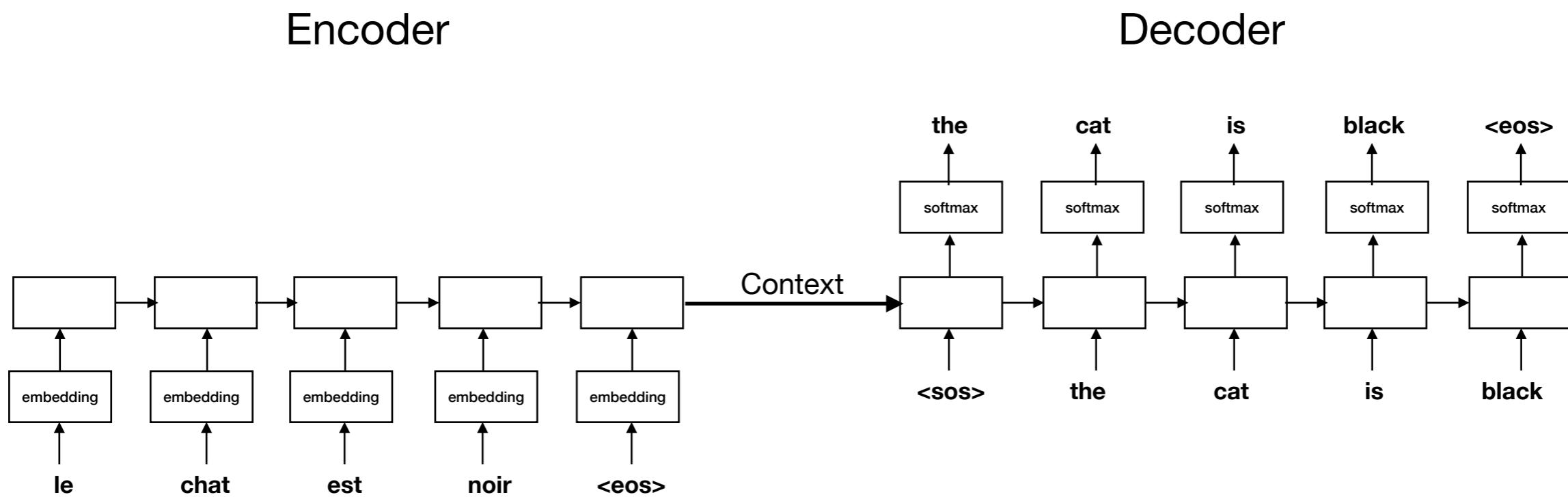
Text Summarization



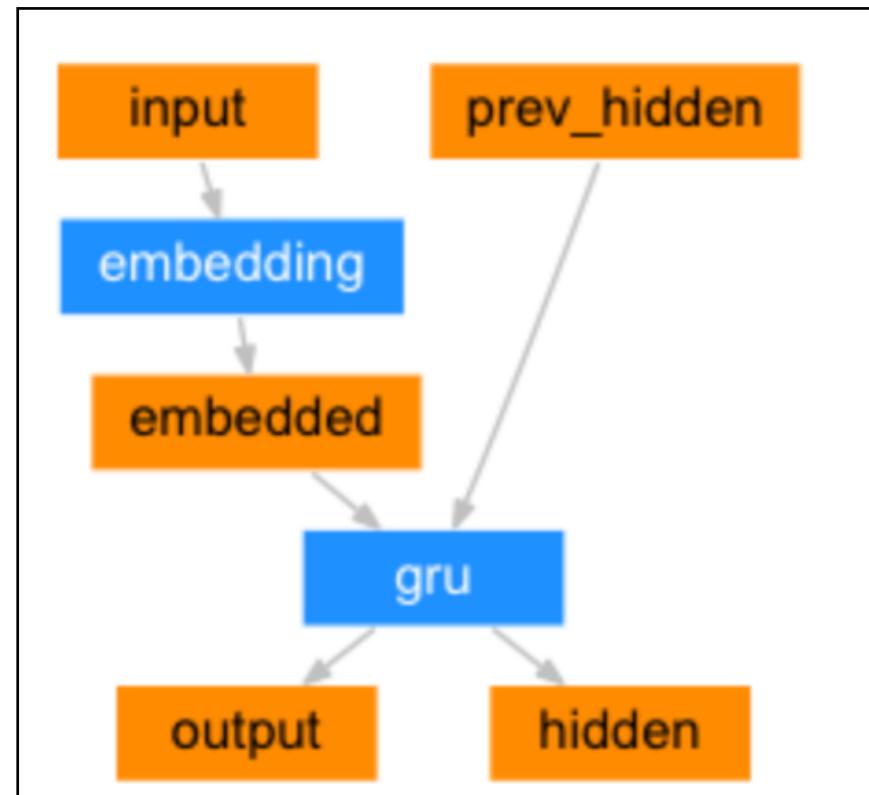
Chatbot



French to English Translation



Encoder



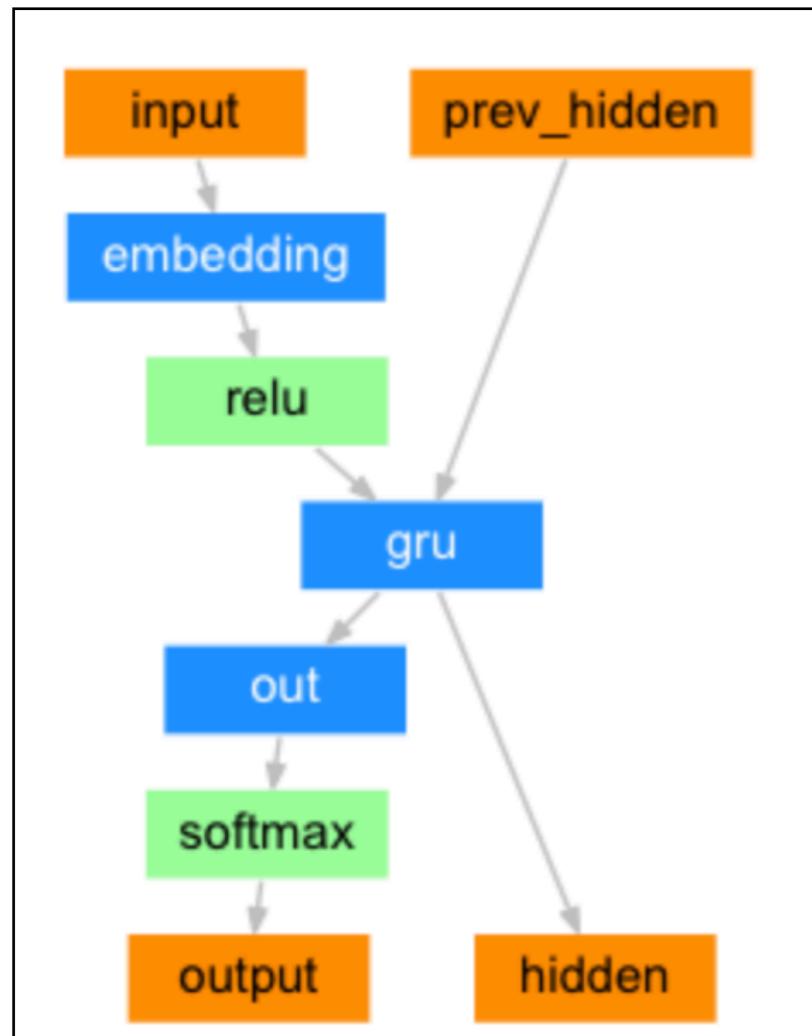
```
class EncoderRNN(nn.Module):
    def __init__(self, input_size, hidden_size):
        super(EncoderRNN, self).__init__()
        self.hidden_size = hidden_size

        self.embedding =
            nn.Embedding(input_size, hidden_size)
        self.gru = nn.GRU(hidden_size,
                         hidden_size)

    def forward(self, input, hidden):
        embedded =
            self.embedding(input).view(1, 1, -1)
        output = embedded
        output, hidden = self.gru(output, hidden)
        return output, hidden

    def initHidden(self):
        return torch.zeros(1, 1, self.hidden_size,
                          device=device)
```

Decoder



```
class DecoderRNN(nn.Module):  
    def __init__(self, hidden_size, output_size):  
        super(DecoderRNN, self).__init__()  
        self.hidden_size = hidden_size  
  
        self.embedding =  
            nn.Embedding(output_size, hidden_size)  
        self.gru = nn.GRU(hidden_size,  
                         hidden_size)  
        self.out = nn.Linear(hidden_size,  
                            output_size)  
        self.softmax = nn.LogSoftmax(dim=1)
```

```
def forward(self, input, hidden):  
    output = self.embedding(input).view(1,  
                                       1, -1)  
    output = F.relu(output)  
    output, hidden = self.gru(output, hidden)  
    output = self.softmax(self.out(output[0]))  
    return output, hidden
```

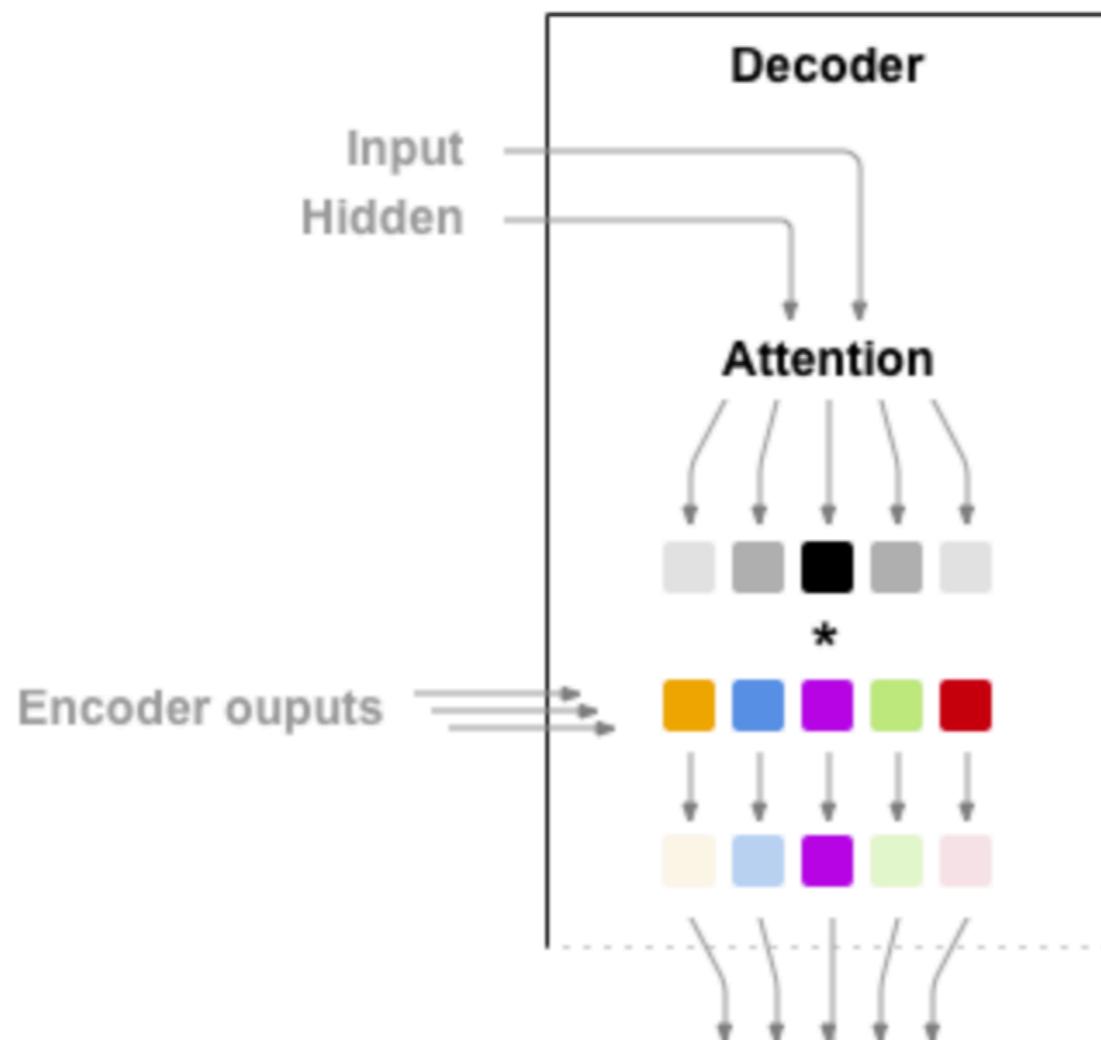
```
def initHidden(self):  
    return torch.zeros(1, 1, self.hidden_size,  
                      device=device)
```

Lab: French to English Translation using RNN

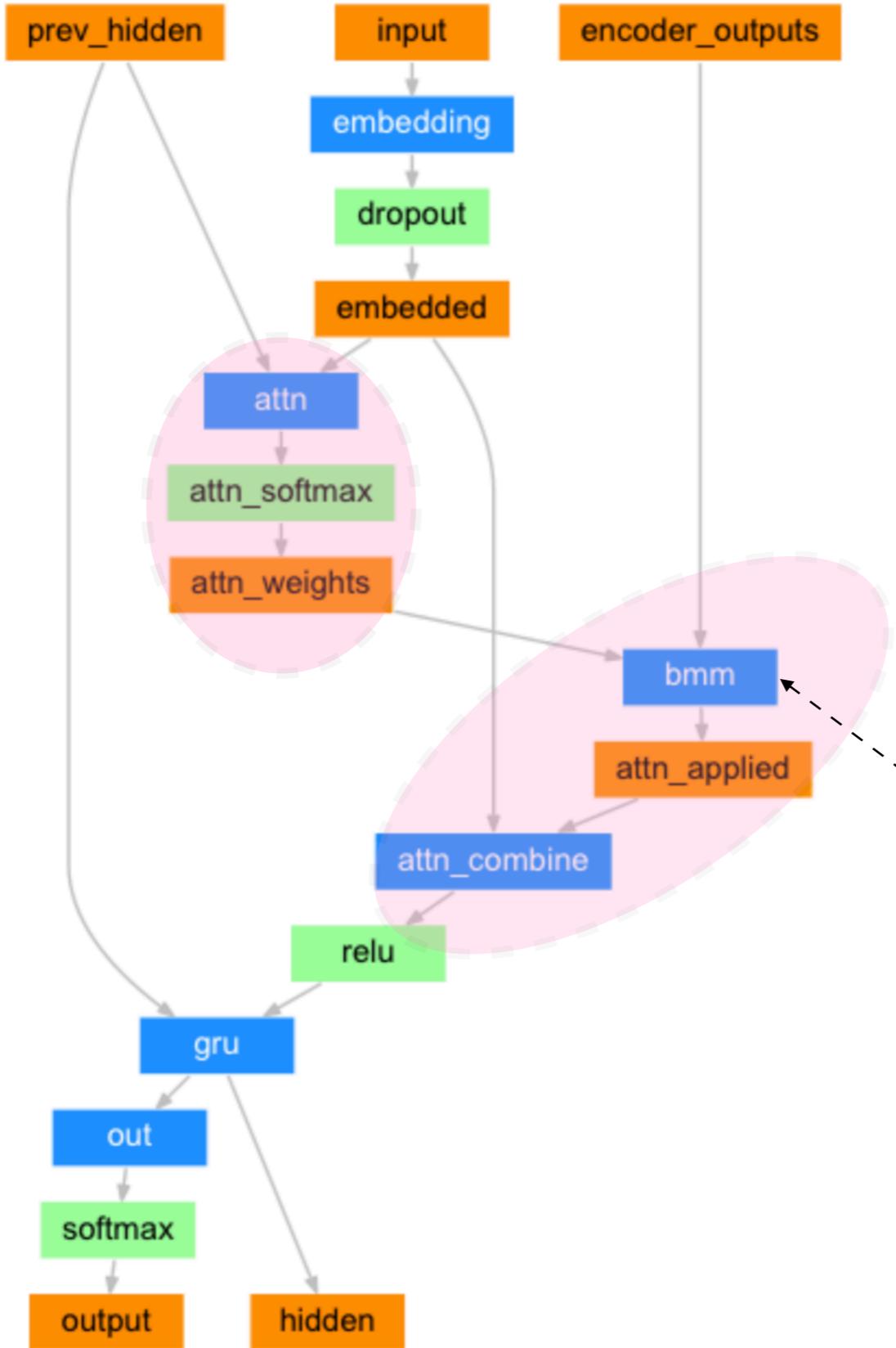
Click this link to begin

[https://colab.research.google.com/github/ravi-ilango/aicamp-mar-2021/blob/main/lab1/
seq2seq translation tutorial.ipynb](https://colab.research.google.com/github/ravi-ilango/aicamp-mar-2021/blob/main/lab1/seq2seq%20translation%20tutorial.ipynb)

Attention Network



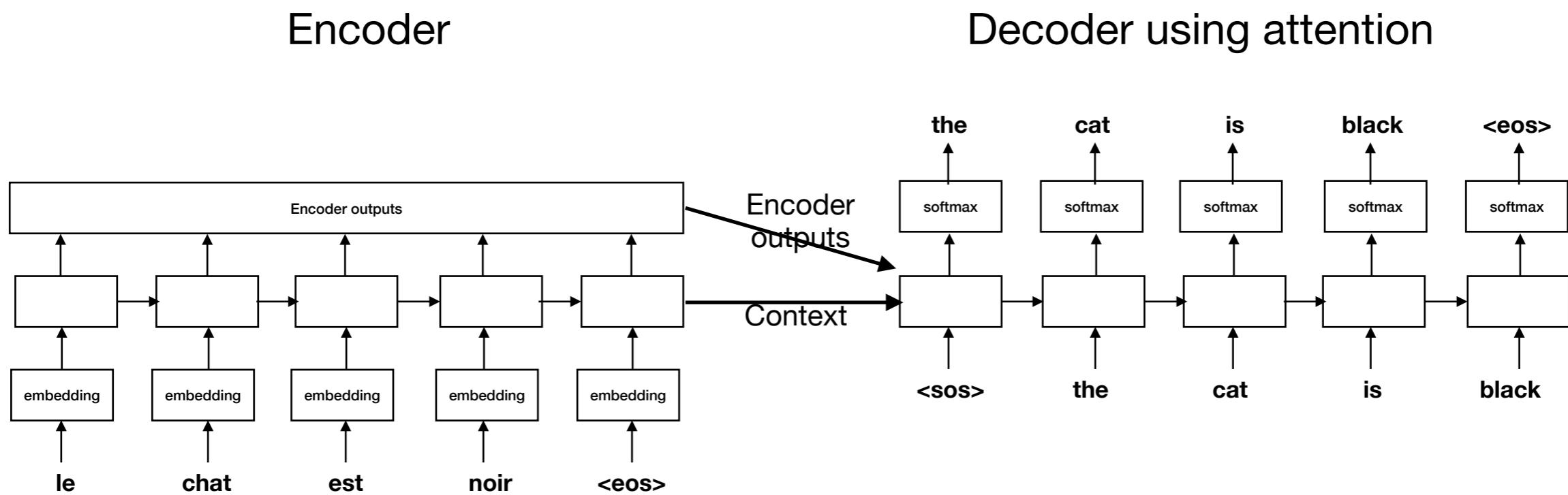
Attention Decoder



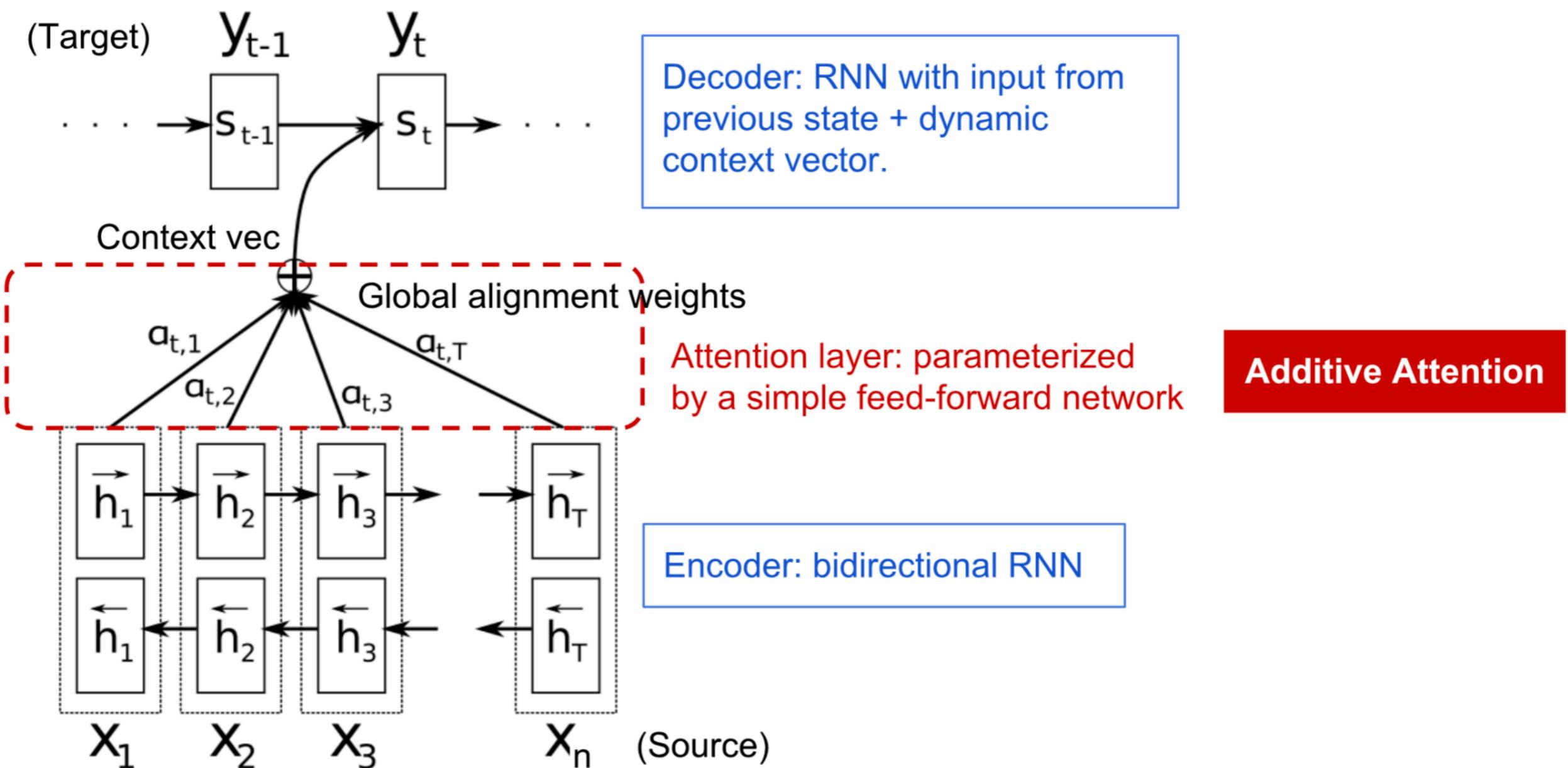
If `input` is a $(b \times n \times m)$ tensor, `mat2` is a $(b \times m \times p)$ tensor, `out` will be a $(b \times n \times p)$ tensor.

$$\text{out}_i = \text{input}_i @ \text{mat2}_i$$

Sequence to Sequence Translation using Attention



The encoder-decoder model with additive attention mechanism in Bahdanau et al., 2015



Attention Decoder

```
class AttnDecoderRNN(nn.Module):
    def __init__(self, hidden_size, output_size, dropout_p=0.1, max_length=MAX_LENGTH):
        super(AttnDecoderRNN, self).__init__()
        self.hidden_size = hidden_size
        self.output_size = output_size
        self.dropout_p = dropout_p
        self.max_length = max_length

        self.embedding = nn.Embedding(self.output_size, self.hidden_size)
        self.attn = nn.Linear(self.hidden_size * 2, self.max_length)
        self.attn_combine = nn.Linear(self.hidden_size * 2, self.hidden_size)
        self.dropout = nn.Dropout(self.dropout_p)
        self.gru = nn.GRU(self.hidden_size, self.hidden_size)
        self.out = nn.Linear(self.hidden_size, self.output_size)

    def forward(self, input, hidden, encoder_outputs):
        embedded = self.embedding(input).view(1, 1, -1)
        embedded = self.dropout(embedded)

        attn_weights = F.softmax(
            self.attn(torch.cat((embedded[0], hidden[0]), 1)), dim=1)
        attn_applied = torch.bmm(attn_weights.unsqueeze(0),
                               encoder_outputs.unsqueeze(0))

        output = torch.cat((embedded[0], attn_applied[0]), 1)
        output = self.attn_combine(output).unsqueeze(0)

        output = F.relu(output)
        output, hidden = self.gru(output, hidden)

        output = F.log_softmax(self.out(output[0]), dim=1)
        return output, hidden, attn_weights

    def initHidden(self):
        return torch.zeros(1, 1, self.hidden_size, device=device)
```

Lab: Spanish to English translation

- Copy the French to English translation notebook (https://colab.research.google.com/github/ravilango/aicamp-mar-2021/blob/main/lab1/seq2seq_translation_tutorial.ipynb)
- Use the Spanish to English translation dataset from <http://www.manythings.org/anki/>
- Use Attention Decoder (instead of RNN). Refer to https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html

Transformers

The Transformer presents a new approach, it proposes to extract information from each position and to apply the mechanism of attention in order to connect two distant words, which can then be parallelized, thus accelerating learning. It applies a mechanism of self-attention.

Transformer relies entirely on self-attention to compute representations of its input and output without using sequence- aligned RNNs or convolution.

Transformers

- **Transformer:** The decoder is stacked directly on top of encoder. Encoder and decoder both are composed of stack of identical layers.
- Each of those stacked layers is composed out of two general types of sub-layers:
 - multi-head self-attention mechanism
 - position-wise fully connected FFN.

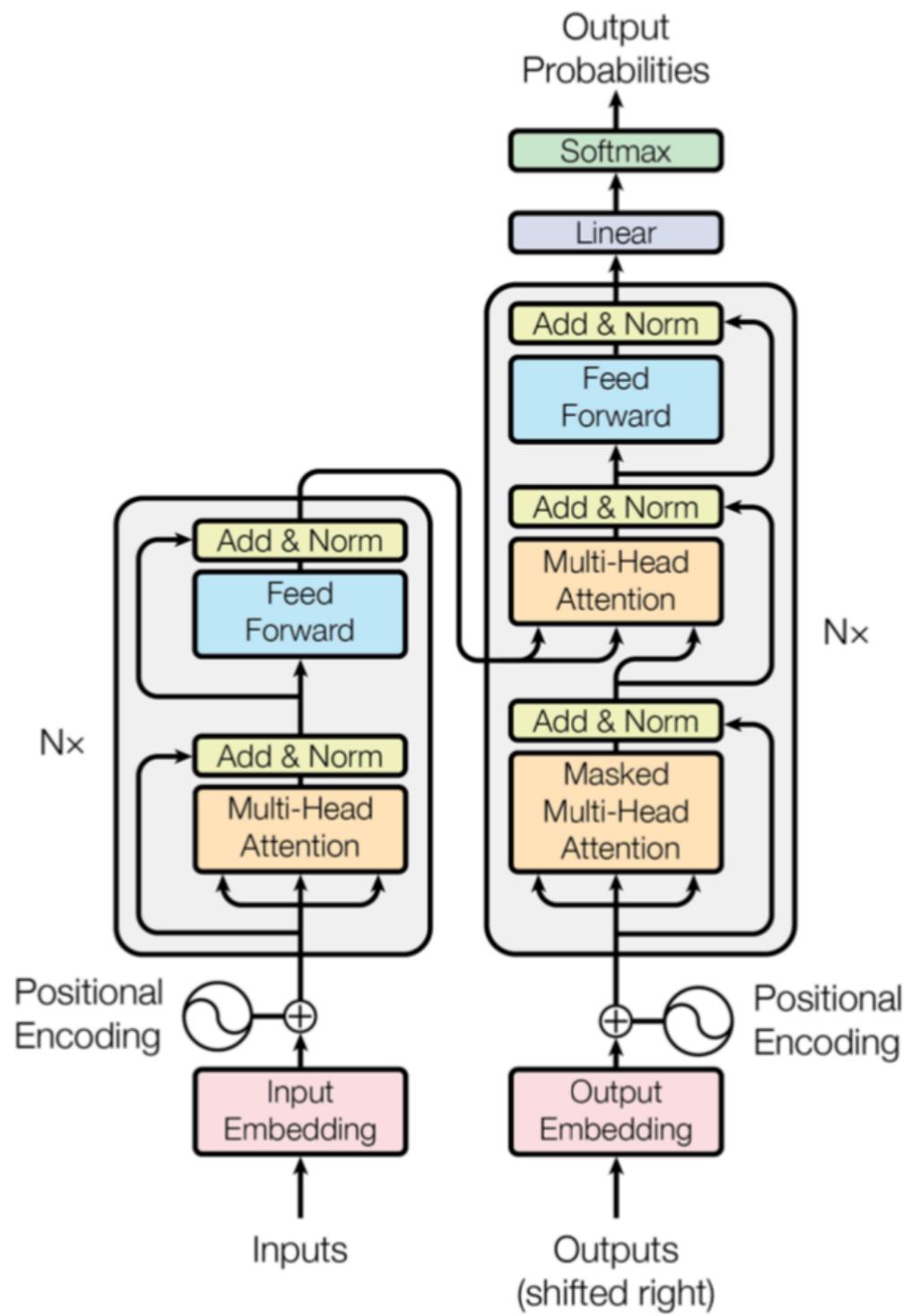
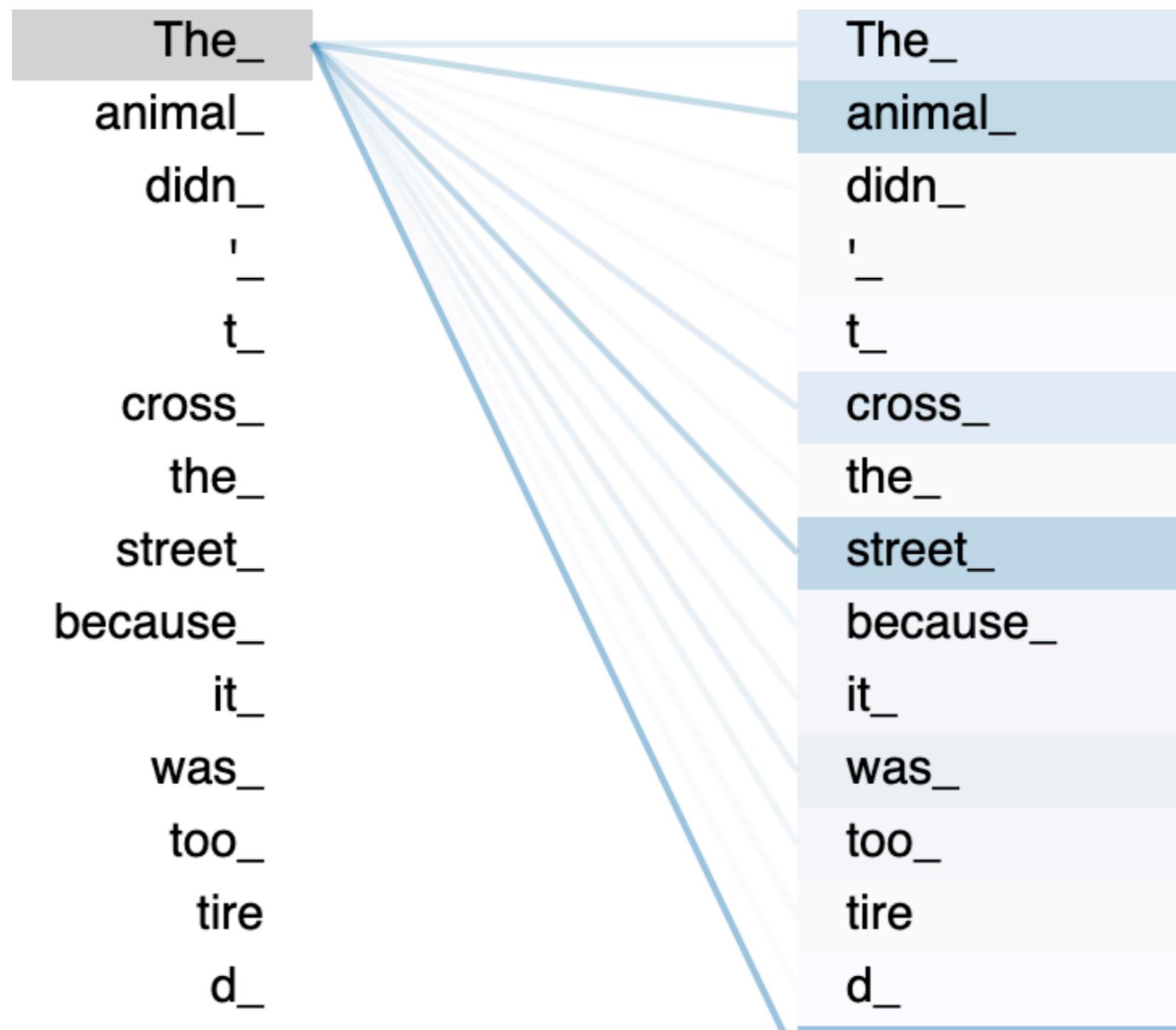


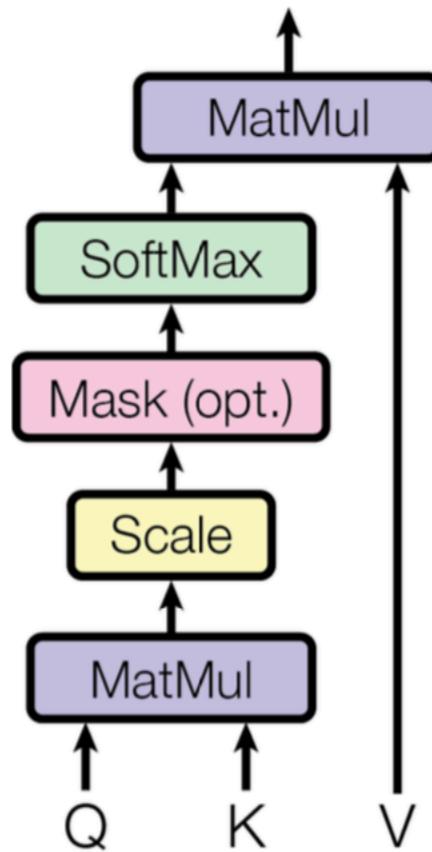
Figure 1: The Transformer - model architecture.

Self Attention



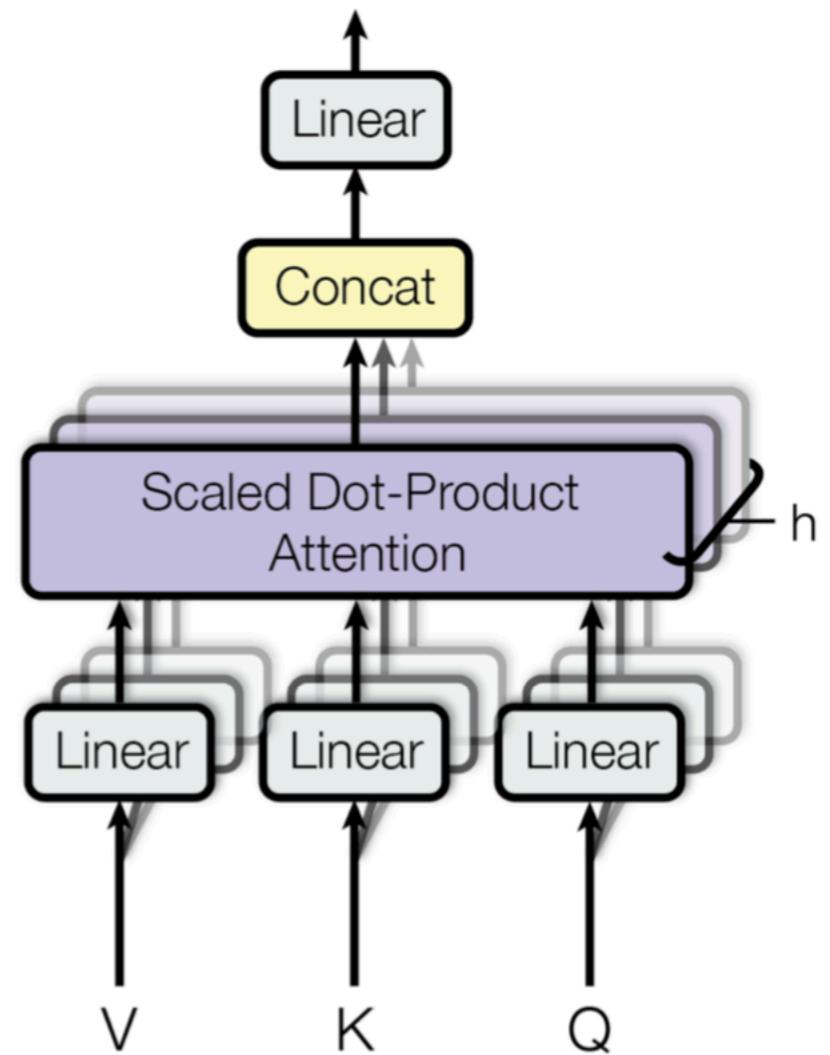
Transformers

Scaled Dot-Product Attention

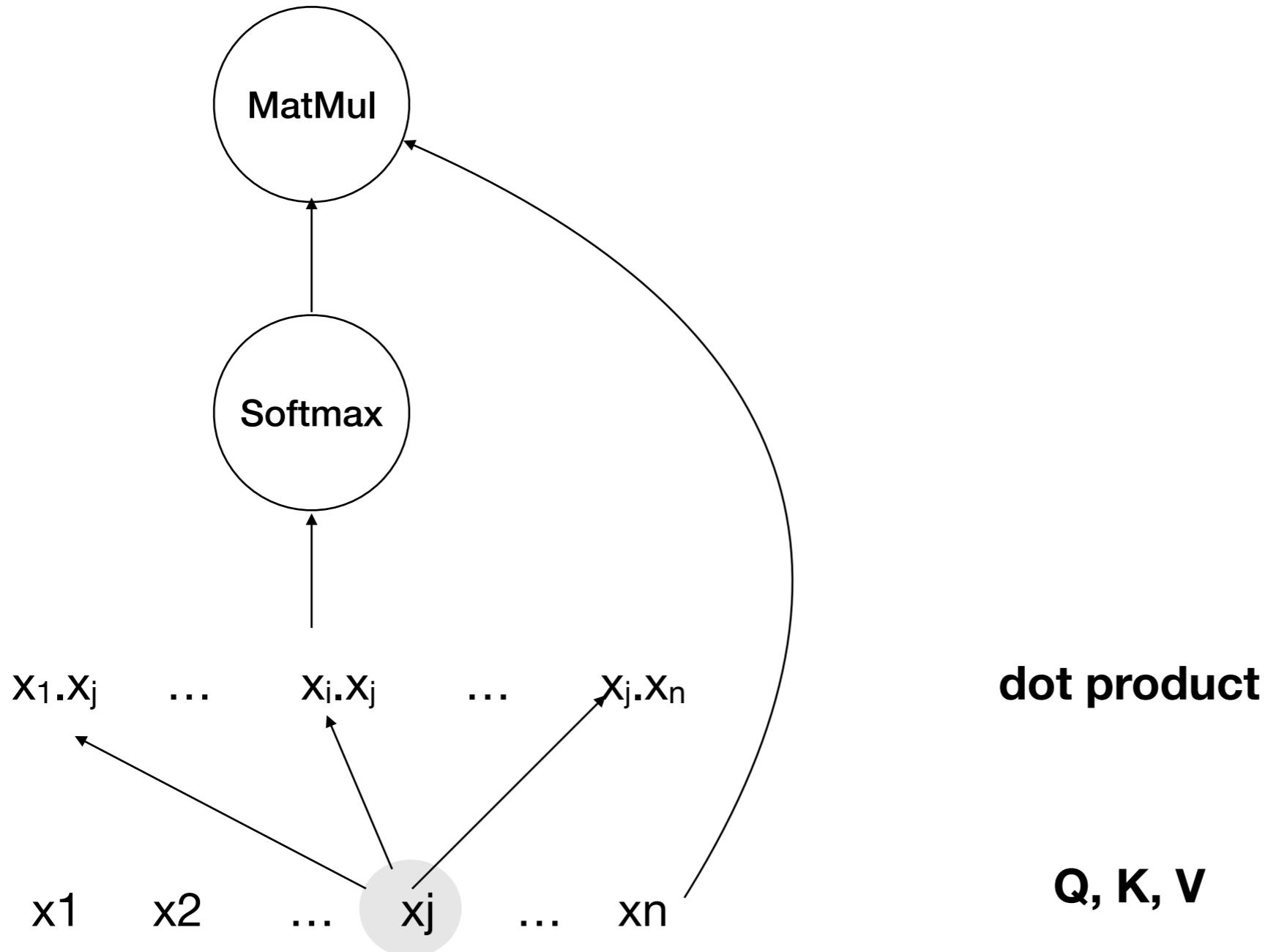


$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Multi-Head Attention



Self Attention



Illustration

[https://mchromiak.github.io/articles/2017/Sep/12/Transformer-
Attention-is-all-you-need/#positional-encoding-pe](https://mchromiak.github.io/articles/2017/Sep/12/Transformer-Attention-is-all-you-need/#positional-encoding-pe)

PyTorch Transformers

- PyTorch Transformers is a library of state-of-the-art pre-trained models for Natural Language Processing (NLP)
- Contains PyTorch implementations, pre-trained model weights for BERT (Google), GPT (OpenAI), Transformer-XL, XLNet, XLM (Facebook) and more ...

Transfer Learning

In practice, very few people train an entire Deep Learning Network from scratch (with random initialization), because it is relatively rare to have a dataset of sufficient size. Instead, it is common to pretrain a ConvNet on a very large dataset (e.g. ImageNet, which contains 1.2 million images with 1000 categories), and then use the ConvNet either as an initialization or a fixed feature extractor for the task of interest

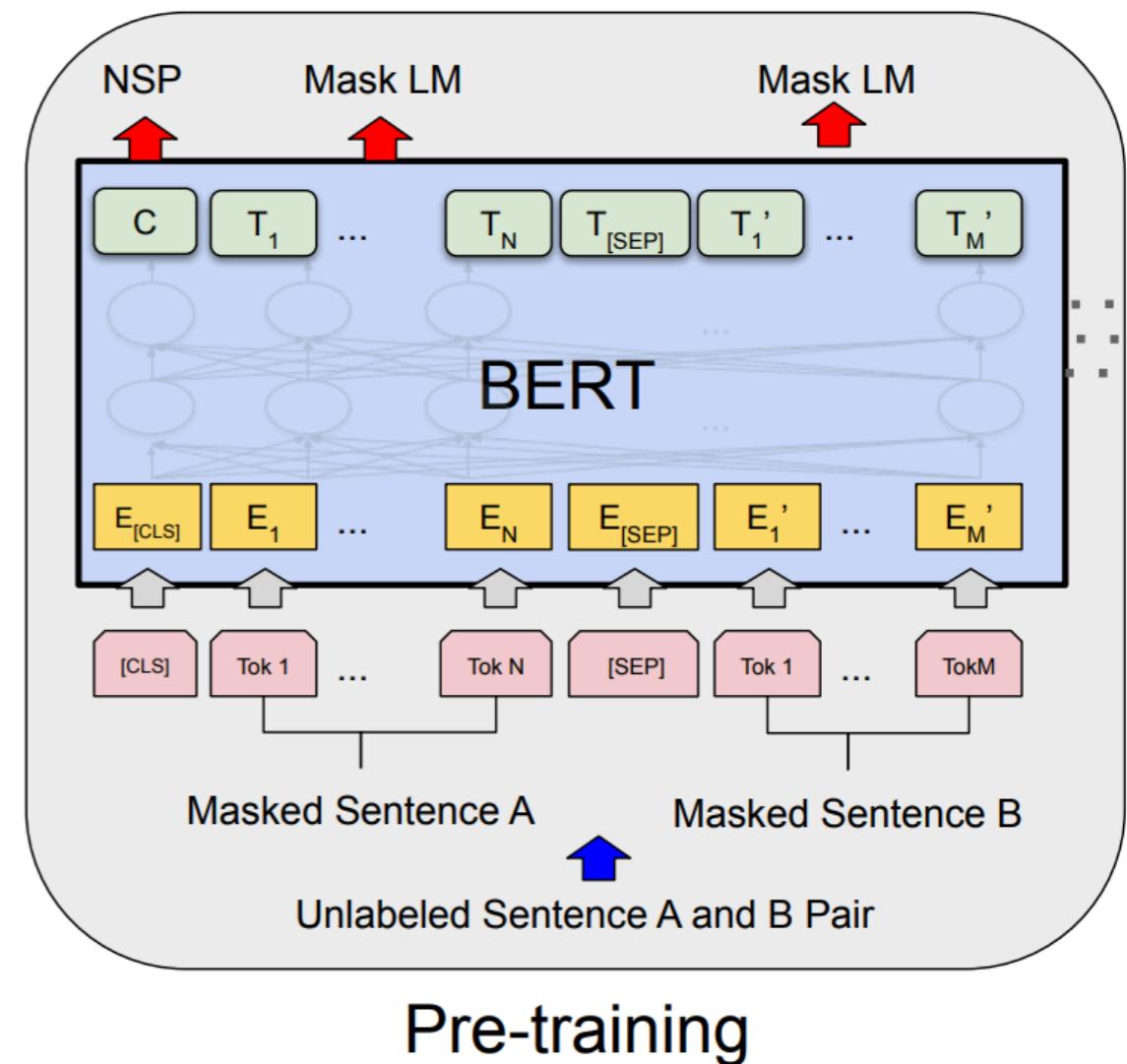
BERT - Bidirectional Encoder Representations from Transformers

Model Details

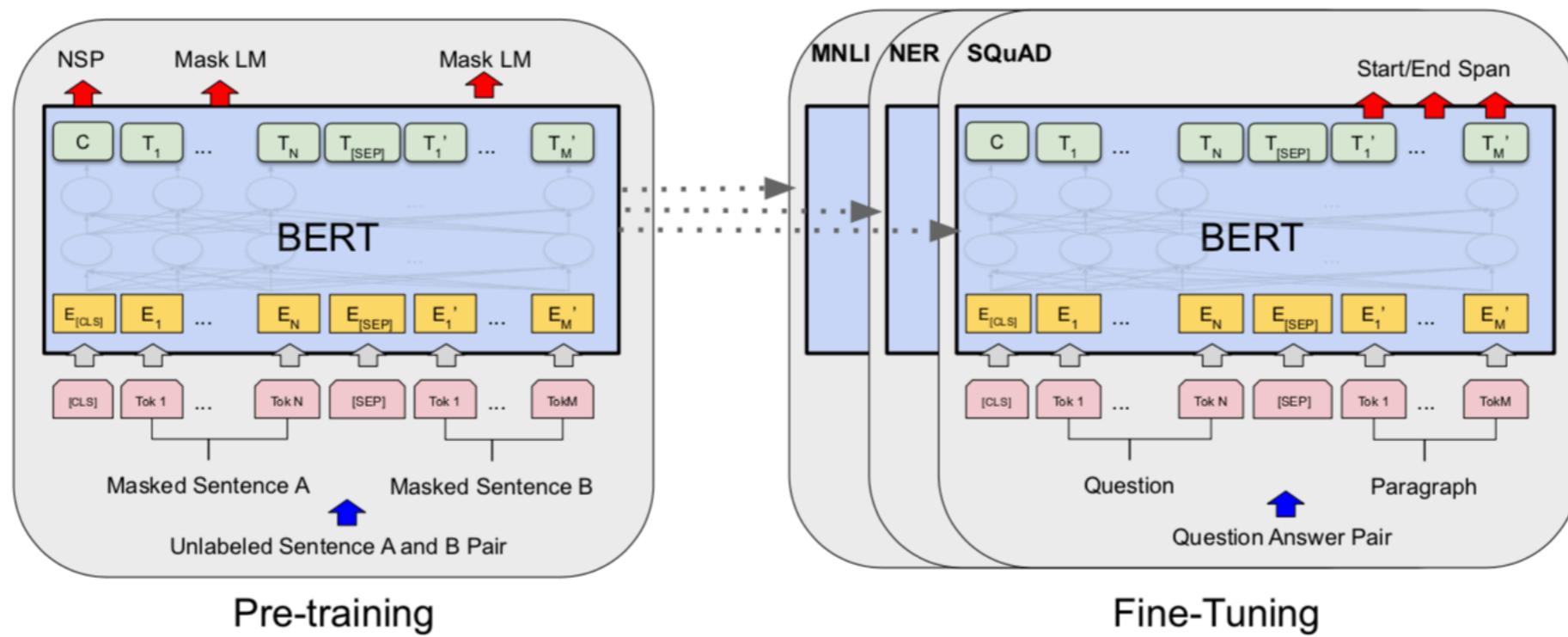
- Data: Wikipedia (2.5B words) + BookCorpus (800M words)
- Batch Size: 131,072 words (1024 sequences * 128 length or 256 sequences * 512 length)
- Training Time: 1M steps (~40 epochs)
- Optimizer: AdamW, 1e-4 learning rate, linear decay
- BERT-Base: 12-layer, 768-hidden, 12-head
- BERT-Large: 24-layer, 1024-hidden, 16-head
- Trained on 4x4 or 8x8 TPU slice for 4 days

BERT - Bidirectional Encoder Representations from Transformers

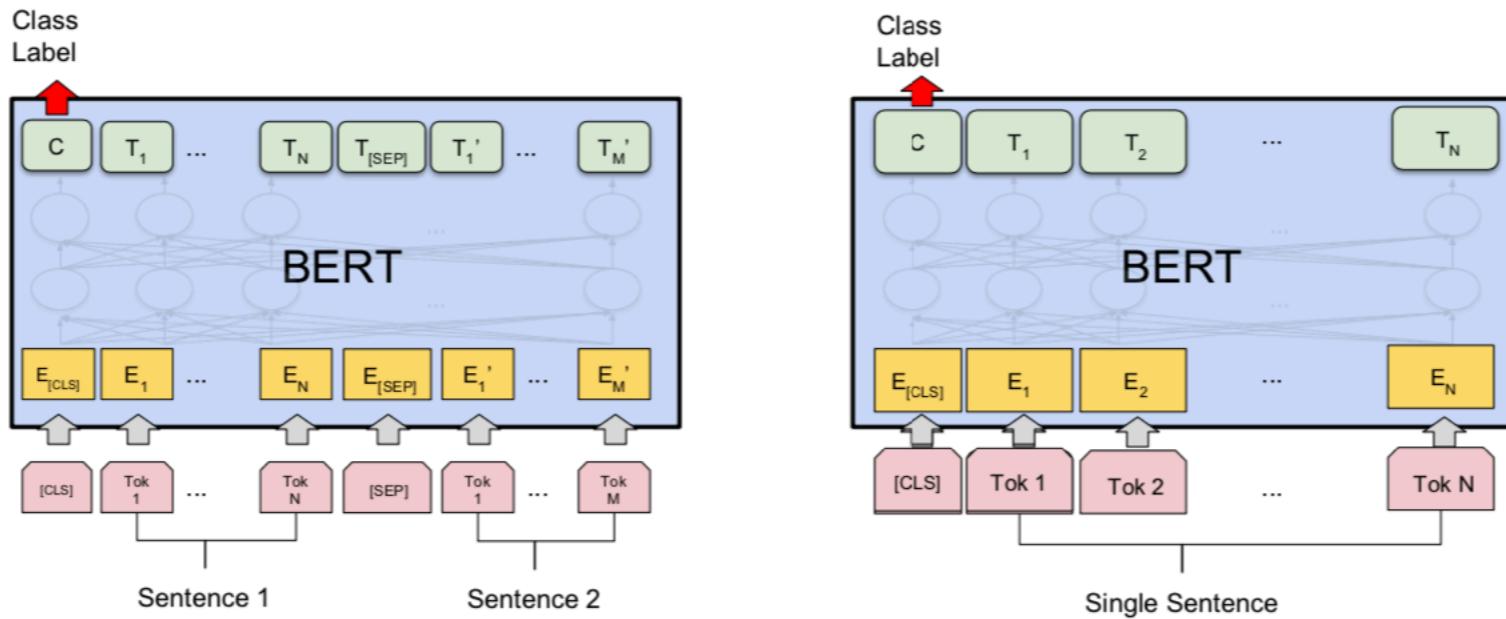
- Pretraining
 - Masked LM
 - Next Sentence Prediction (NSP)
 - Books Corpus (800M words) and English Wikipedia (2,500M words).



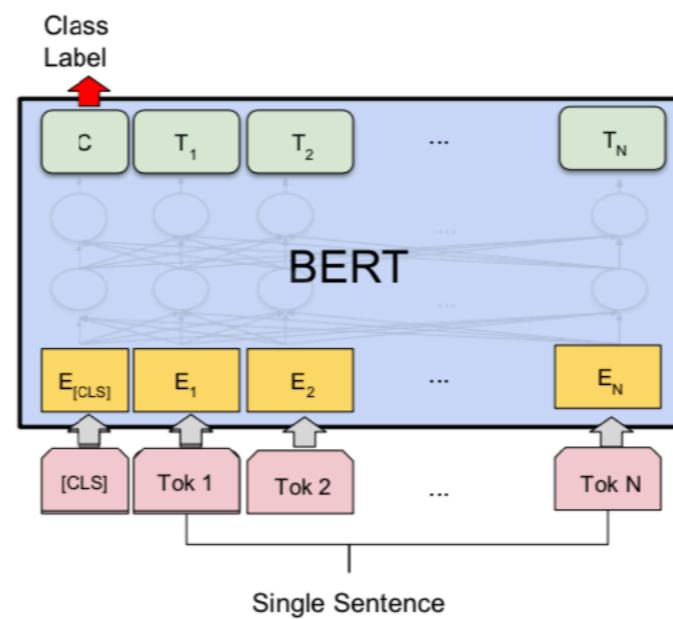
BERT - Pretrained Models



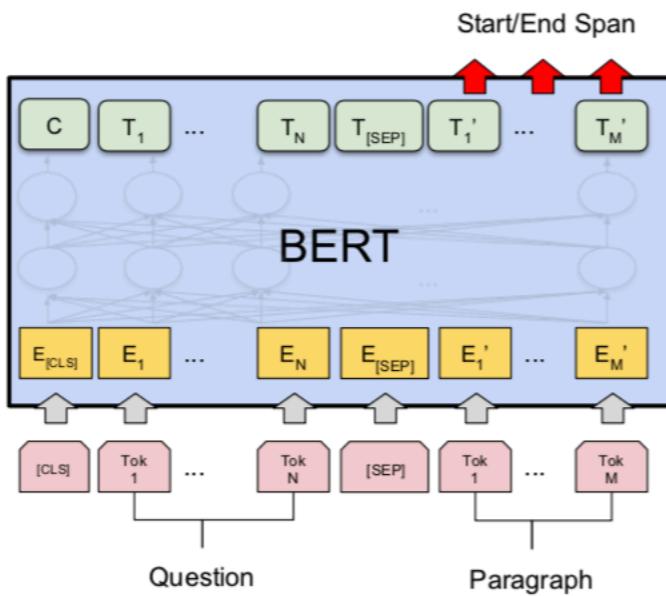
Fine-tuning BERT on Different Tasks



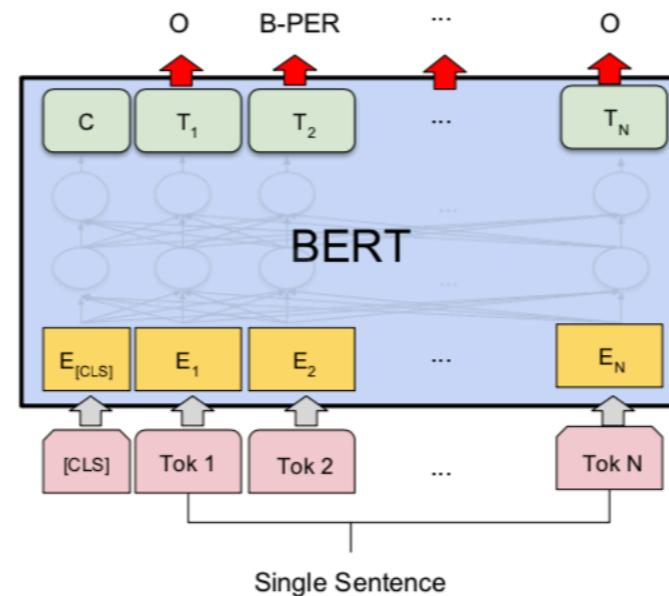
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



(b) Single Sentence Classification Tasks:
SST-2, CoLA



(c) Question Answering Tasks:
SQuAD v1.1



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

Lab: Disaster Detection using BERT

Click this link to begin

[https://colab.research.google.com/github/ravi-ilango/aicamp-mar-2021/blob/main/lab3/
disaster detection bert.ipynb](https://colab.research.google.com/github/ravi-ilango/aicamp-mar-2021/blob/main/lab3/disaster%20detection%20bert.ipynb)

Lab: Sarcasm Detection using BERT

Click this link to begin

[https://colab.research.google.com/github/ravi-ilango/aicamp-mar-2021/blob/main/lab4/
LAB_Sarcasm_Detector.ipynb](https://colab.research.google.com/github/ravi-ilango/aicamp-mar-2021/blob/main/lab4/LAB_Sarcasm_Detector.ipynb)

Lab: Humor detection using XLNet

Click this link to begin

[https://colab.research.google.com/github/ravi-ilango/aicamp-mar-2021/blob/main/lab5/
LAB humor questions.ipynb](https://colab.research.google.com/github/ravi-ilango/aicamp-mar-2021/blob/main/lab5/LAB%20humor%20questions.ipynb)

BLUE

- **BLEU (bilingual evaluation understudy)** is an algorithm for **evaluating** the quality of text which has been **machine-translated** from one **natural language** to another
- Contains PyTorch implementations, pre-trained model weights for BERT (Google), GPT (OpenAI), Transformer-XL, XLNet, XLM (Facebook) and more ...

Attention

3.2.3 Applications of Attention in our Model

The Transformer uses multi-head attention in three different ways:

- In "encoder-decoder attention" layers, the queries come from the previous decoder layer, and the memory keys and values come from the output of the encoder. This allows every position in the decoder to attend over all positions in the input sequence. This mimics the typical encoder-decoder attention mechanisms in sequence-to-sequence models such as [38, 2, 9].
- The encoder contains self-attention layers. In a self-attention layer all of the keys, values and queries come from the same place, in this case, the output of the previous layer in the encoder. Each position in the encoder can attend to all positions in the previous layer of the encoder.
- Similarly, self-attention layers in the decoder allow each position in the decoder to attend to all positions in the decoder up to and including that position. We need to prevent leftward information flow in the decoder to preserve the auto-regressive property. We implement this inside of scaled dot-product attention by masking out (setting to $-\infty$) all values in the input of the softmax which correspond to illegal connections. See Figure 2.

GLUE

MNLI: The Multi-Genre Natural Language Inference

QQA: Quora Question Pairs

QNLI: Question Natural Language Inference