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

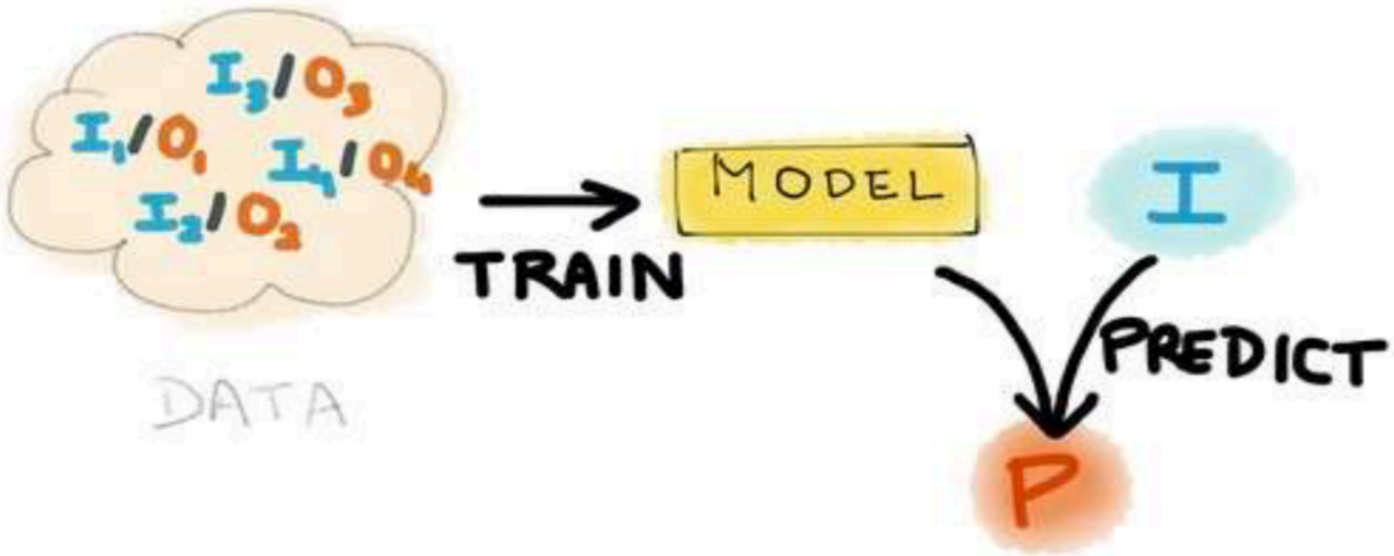

Three lidar systems
A forward facing camera
Radar sensors
Self-driving sensors


PIENSE
THINK
DENKE
PENSER
$24,000
$77,147
$21,600
Who is Stoker?
$1,000
Who is Bram Stoker?
$ 17,973
$5600


ALPHAGO
00:00:49
LEE SEDOL
00:01:00
AlphaGo
Google DeepMind

What is machine learning

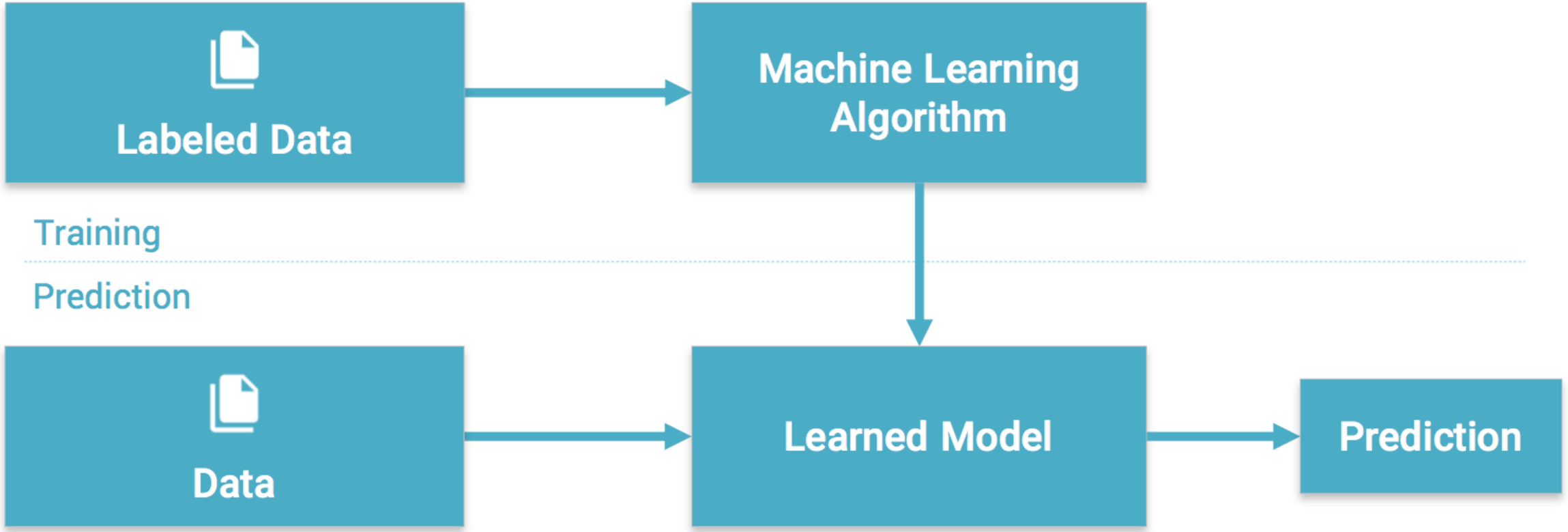
- Subfield of computer science that explores how machines can learn to perform certain task without explicit programming



Machine Learning - Basics

Introduction

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



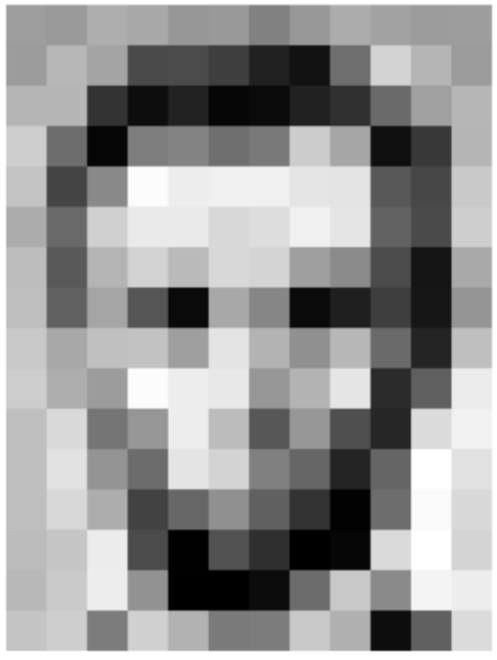
Provides various techniques that can learn from and make predictions on data

Supervised

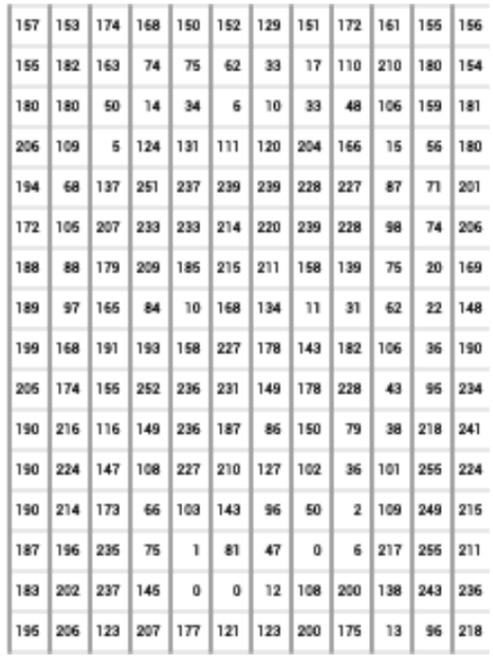

Monday
stock prices
supervised
learning
Tuesday
stock prices


what we
know
supervised
learning
what we want
to know

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	158	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

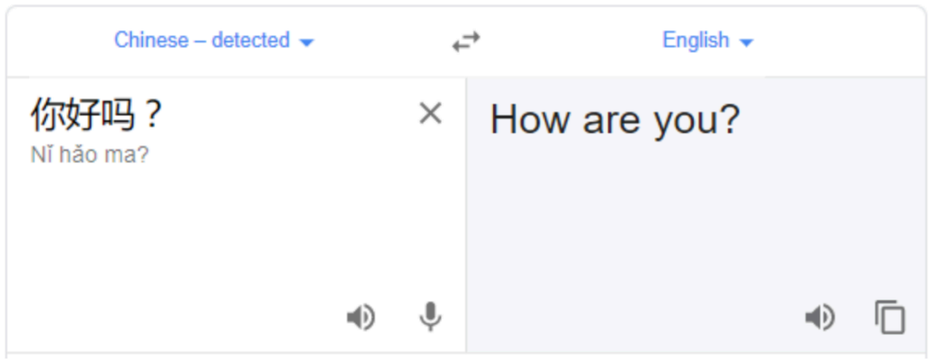
Lincoln	0.8
Washington	0.1
Jefferson	0.05
Obama	0.05

- **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



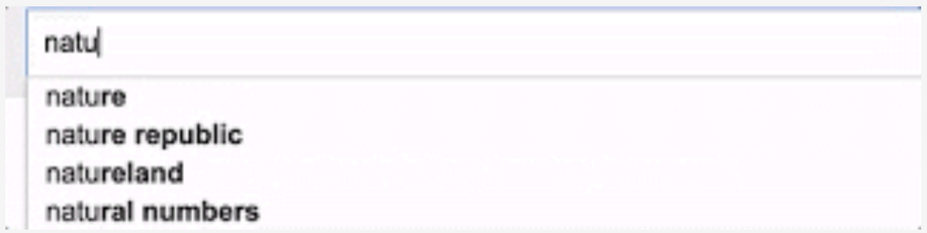
Google Translate from Chinese to English

Chatbot

*Our Financial Consultants use the ask**PRU** chatbot to provide a better service experience to customers.*

Email assistant

Predictive typing suggests the next word in the sentence.



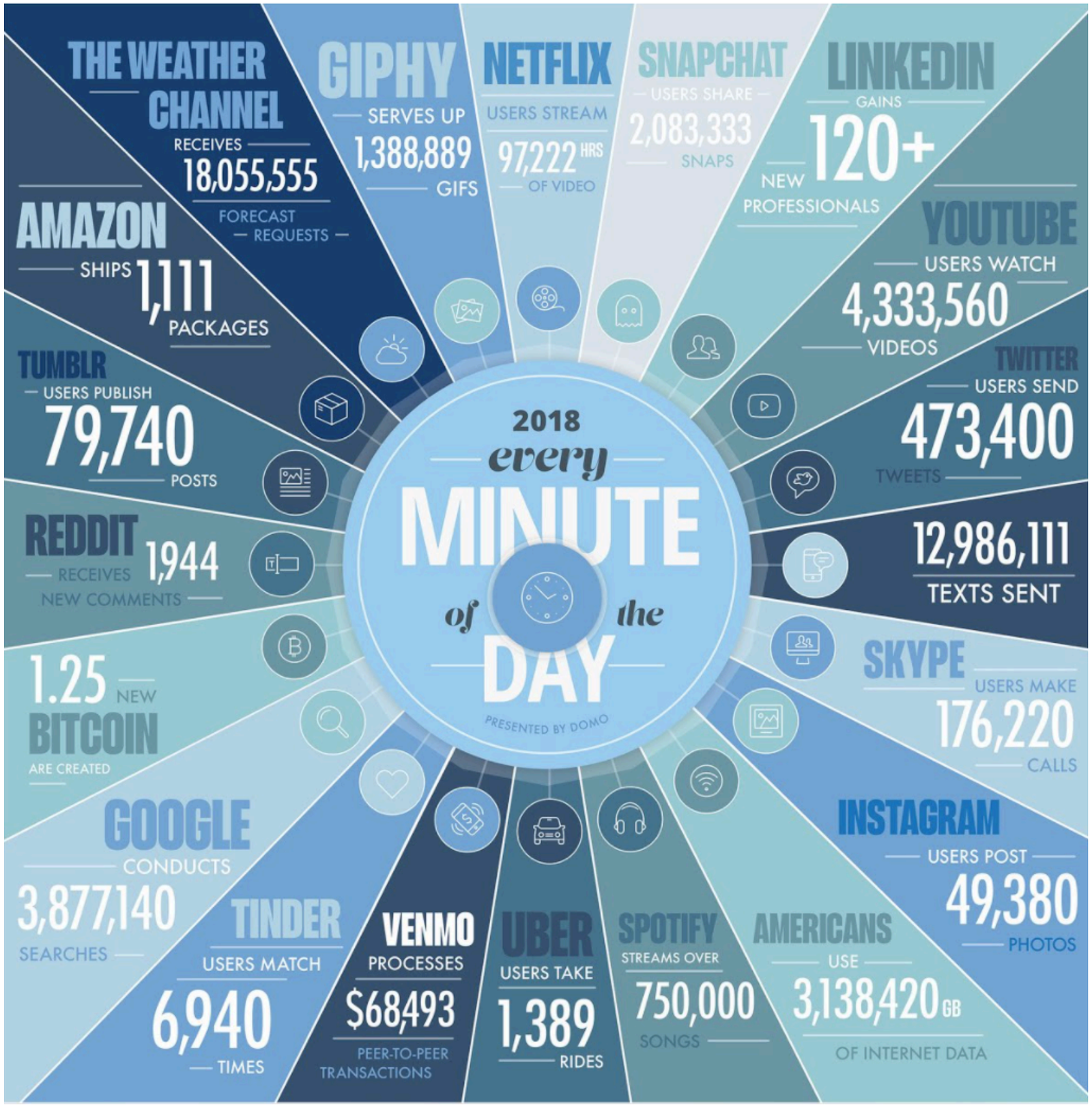
3.5B searches/daily

NLP in business operation

- Achieve efficiencies using machine processing of unstructured tect data

Extract and Summarization

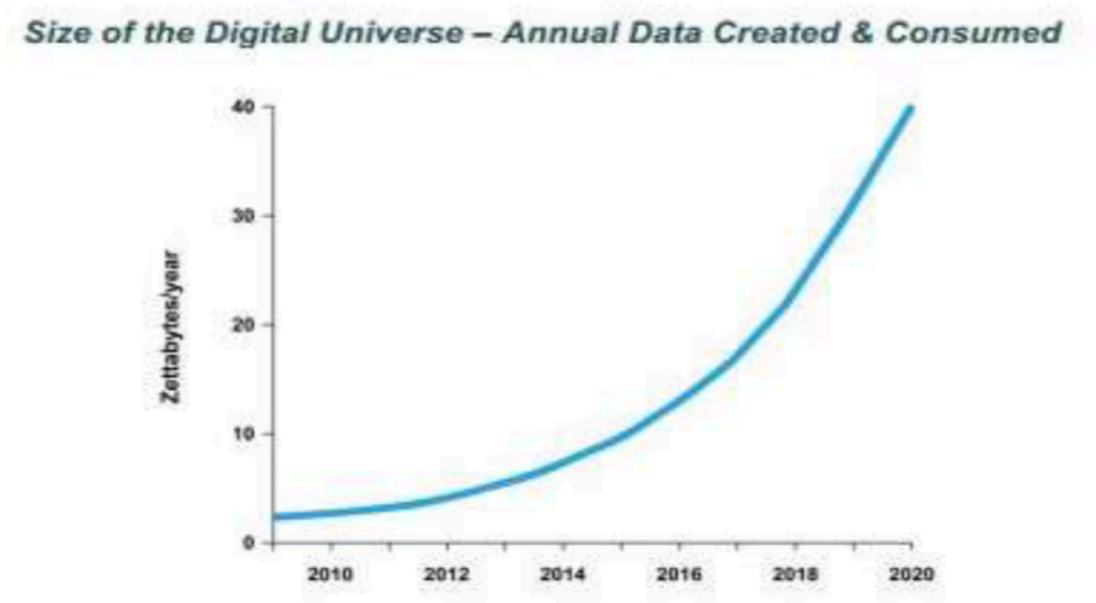
Data production rate



https://www.digitalinformationworld.com/2018/06/infographics-data-never-sleeps-6.html

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 sententces

“I enjoy working in a bank.”

“I enjoy working near a river bank.”

Understanding language is tough

- Syntax - grammer
- 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 possive 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

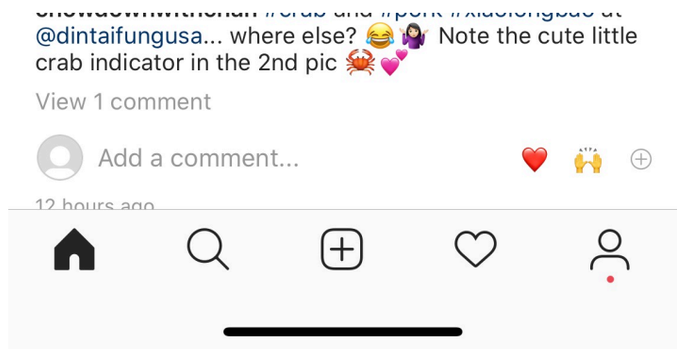
I'm ate her duck

I'm ate or duck

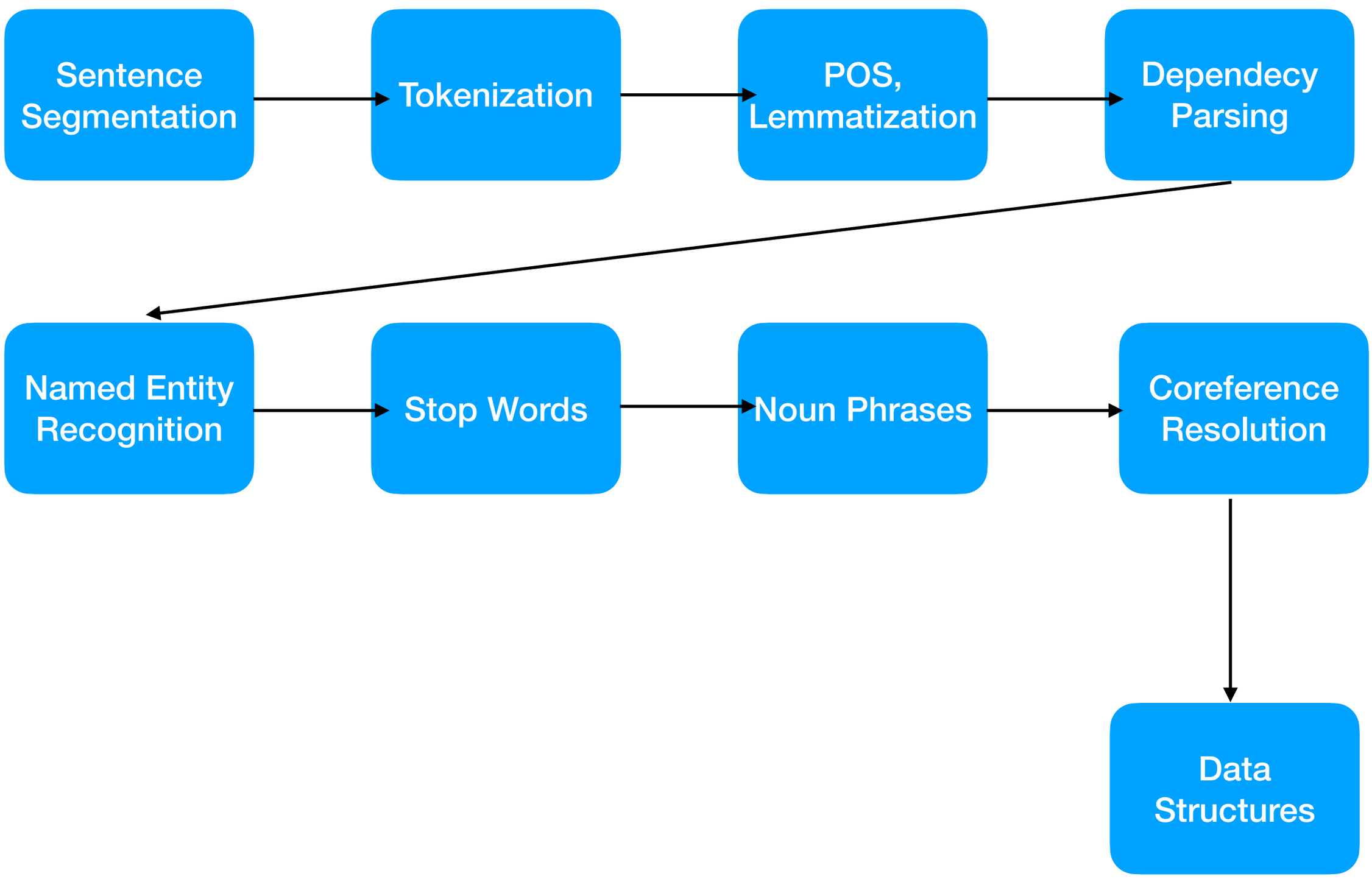
I mate or duck

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

python™

Transformers

PyTorch

scikit learn

NLTK

TensorFlow

spaCy

CoreNLP

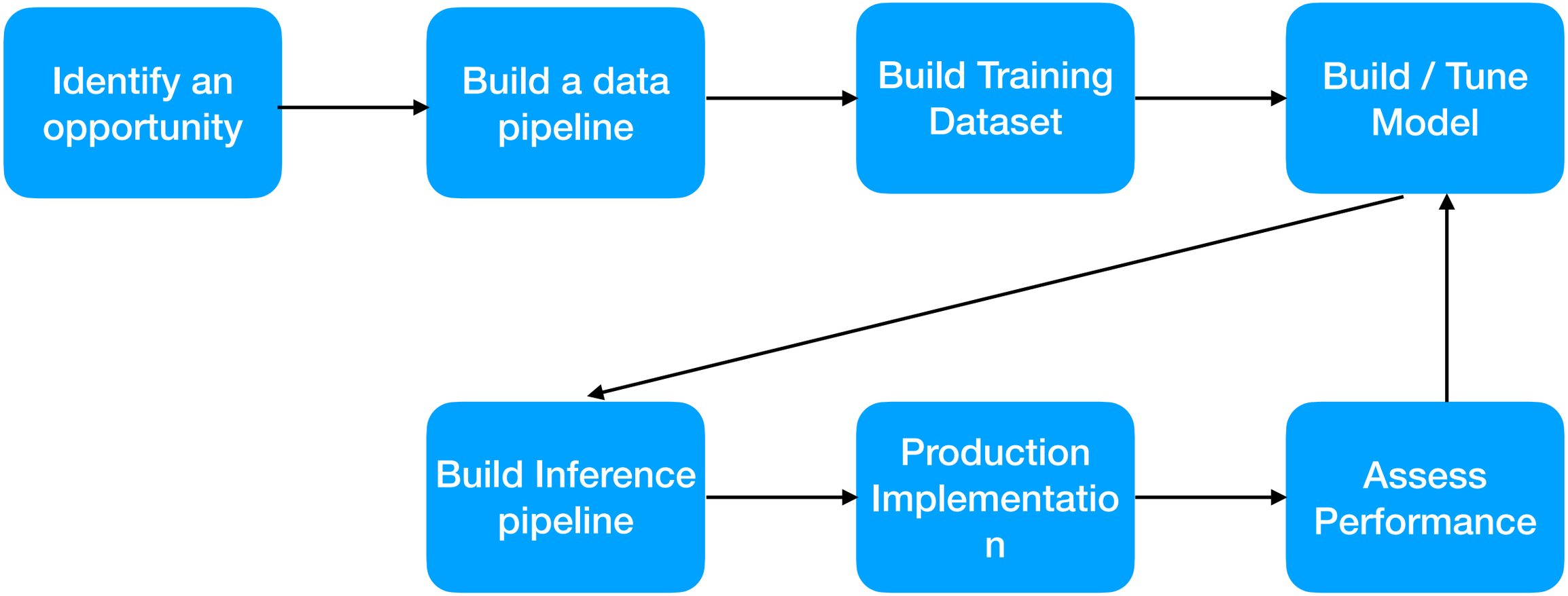
PyText

gensim

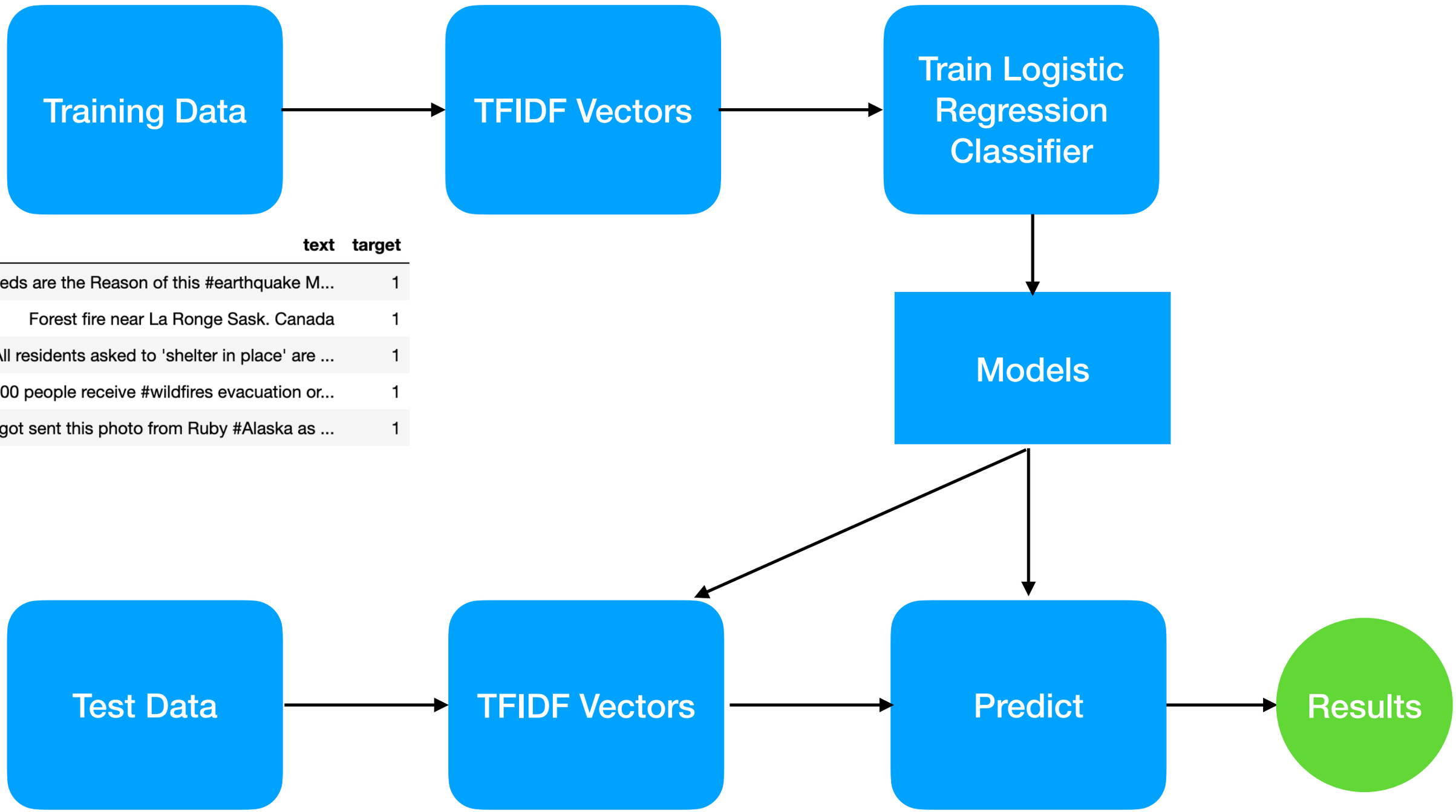
NLP Data Scientist - (some) concepts & techniques

- 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



text	target
Our Deeds are the Reason of this #earthquake M...	1
Forest fire near La Ronge Sask. Canada	1
All residents asked to 'shelter in place' are ...	1
13,000 people receive #wildfires evacuation or...	1
Just got sent this photo from Ruby #Alaska as ...	1

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

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:

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

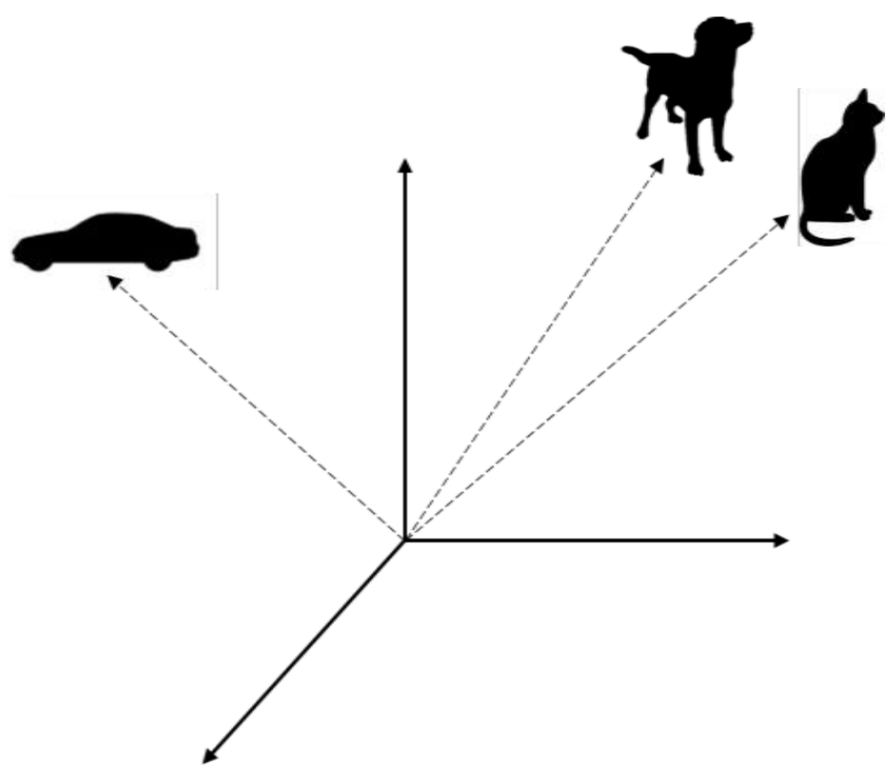
Vector Space Model

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.

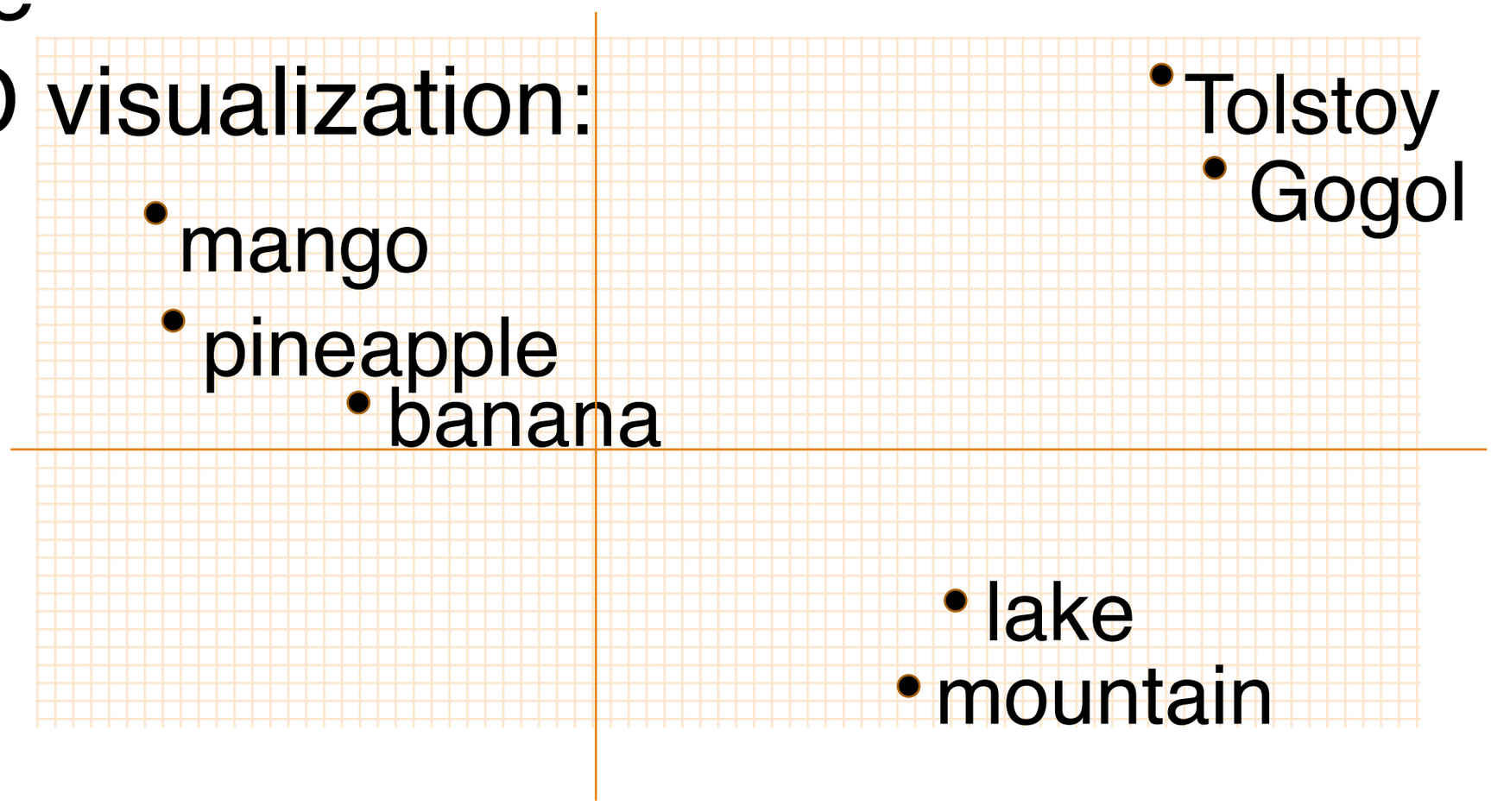
These dense vectors are called embeddings.



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:

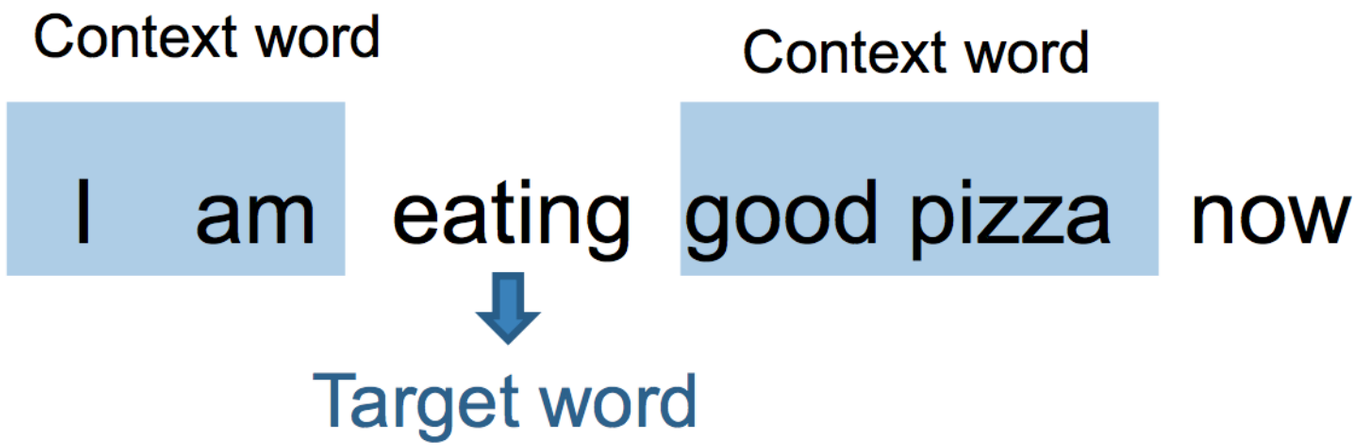
Banana, mango, or pineapple are all delicious…

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

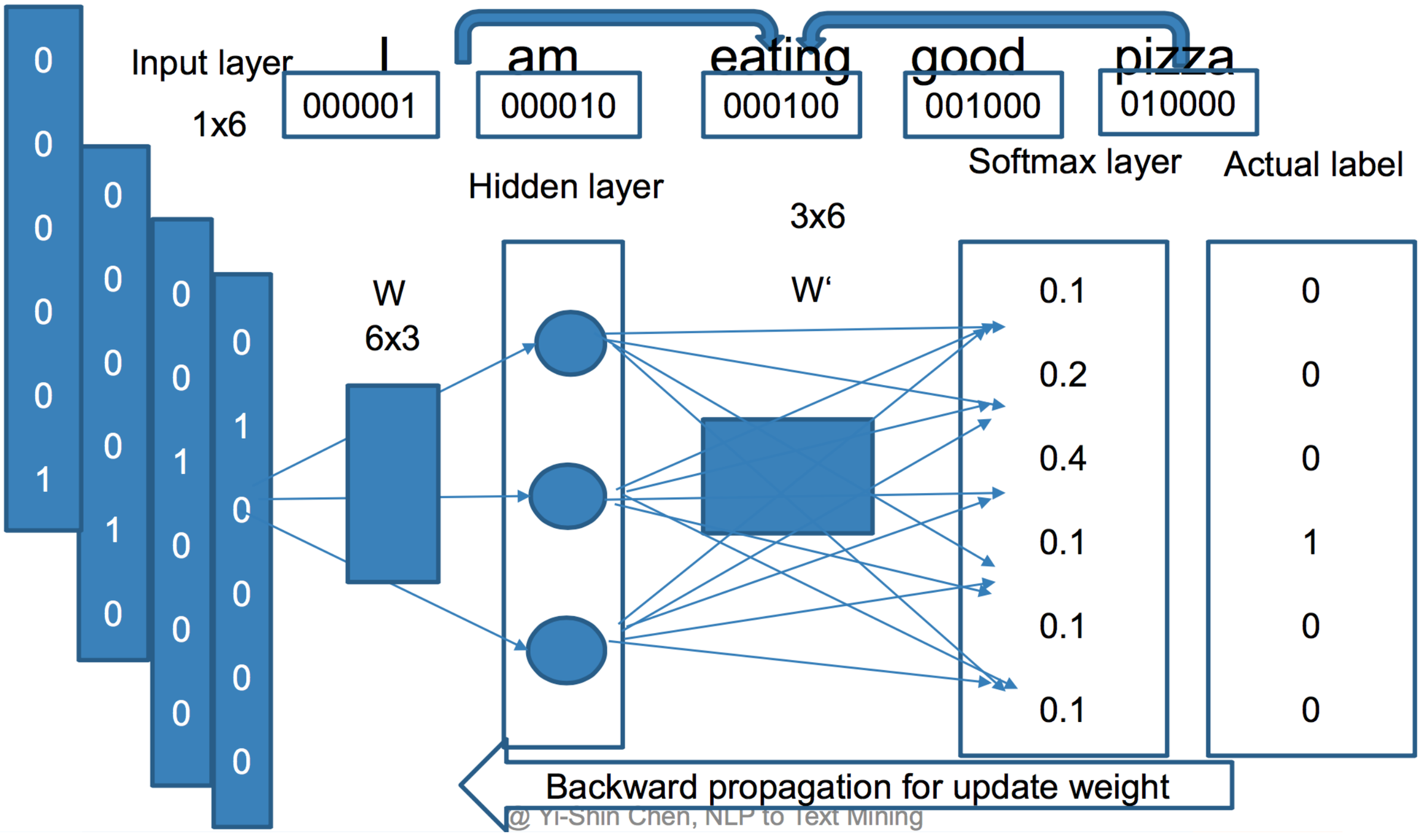


Ex: ([features], label)
([I, am, good, pizza], eating), ([am, eating, pizza, now], good) and so on and so forth.

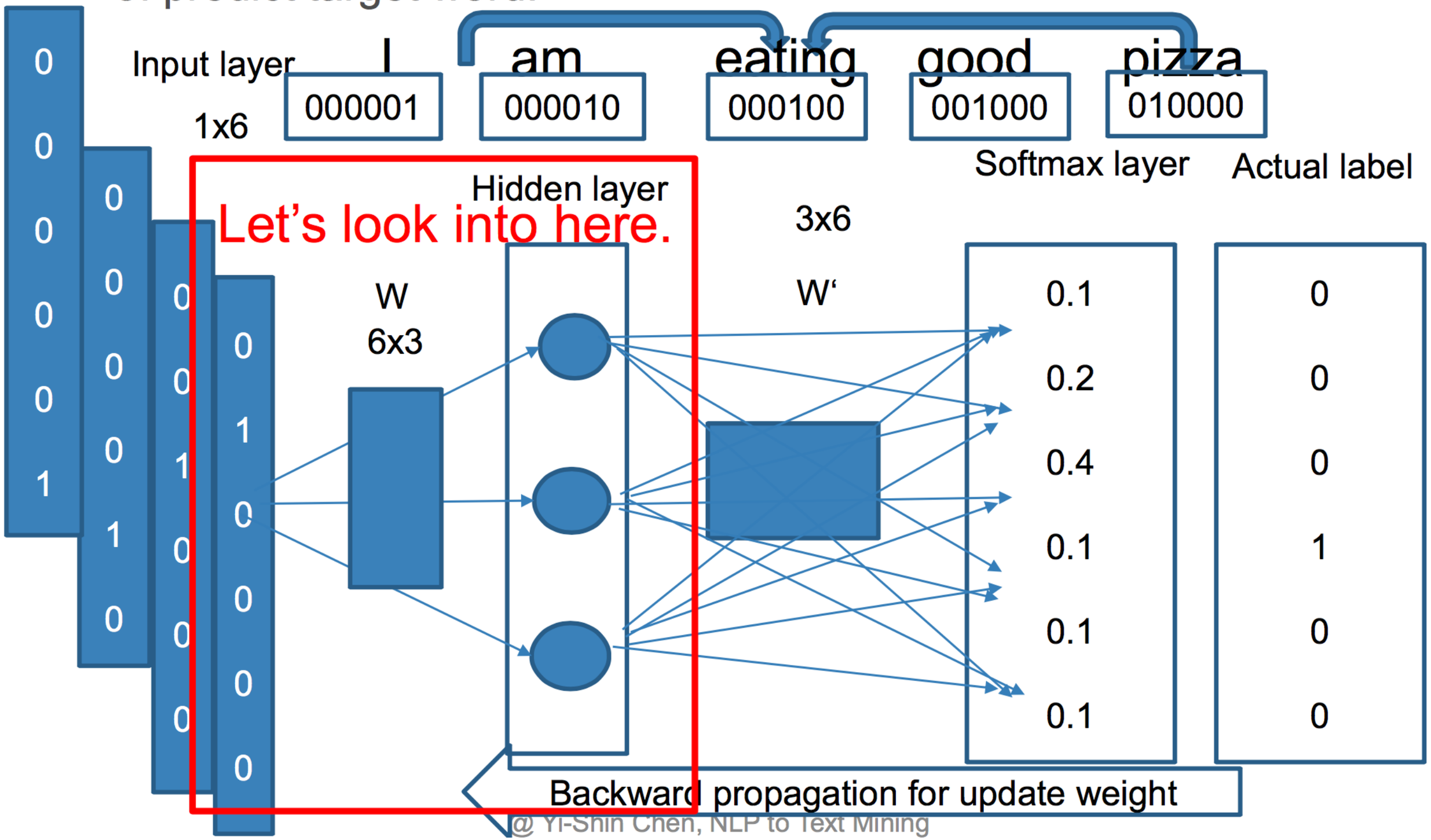
Word Embedding in a Deep Learning network

Continuous Bag-of-Words Model (Contd.)

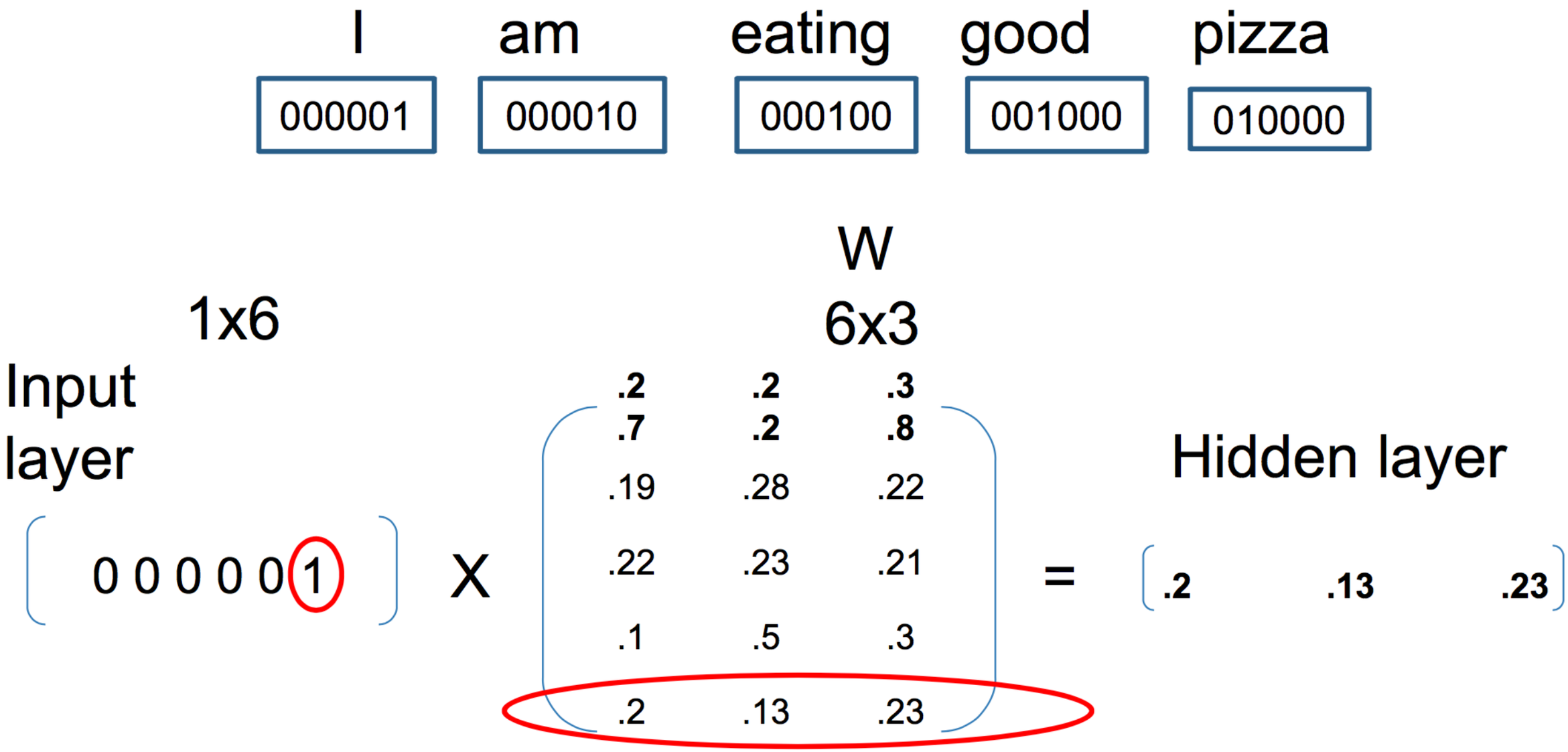
- We train one depth neural network, set hidden layer's width 3.



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

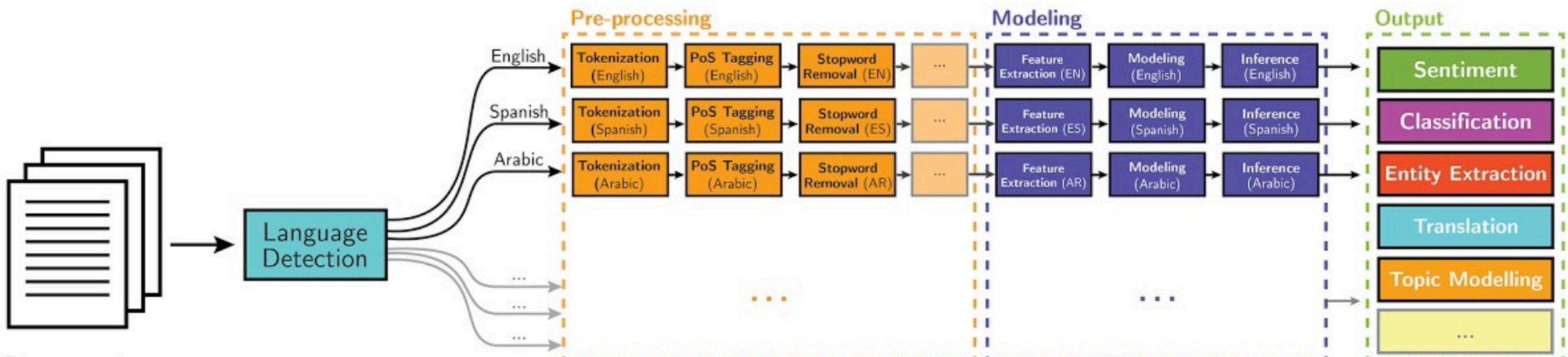


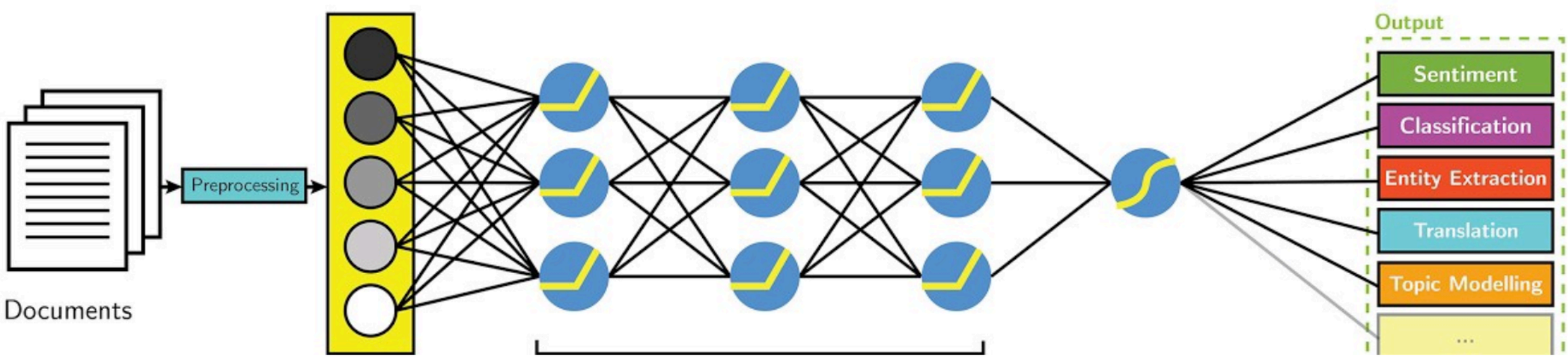
Continuous Bag-of-Words Model (Contd.)



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

NLP


Pre-processing
Modeling
Output
English
Spanish
Arabic
Tokenization (English)
PoS Tagging (English)
Stopword Removal (EN)
Tokenization (Spanish)
PoS Tagging (Spanish)
Stopword Removal (ES)
Tokenization (Arabic)
PoS Tagging (Arabic)
Stopword Removal (AR)
Feature Extraction (EN)
Modeling (English)
Inference (English)
Feature Extraction (ES)
Modeling (Spanish)
Inference (Spanish)
Feature Extraction (AR)
Modeling (Arabic)
Inference (Arabic)
Language Detection
Documents
Sentiment
Classification
Entity Extraction
Translation
Topic Modelling


Deep Learning-based NLP
Preprocessing
Documents
Output
Sentiment
Classification
Entity Extraction
Translation
Topic Modelling

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.

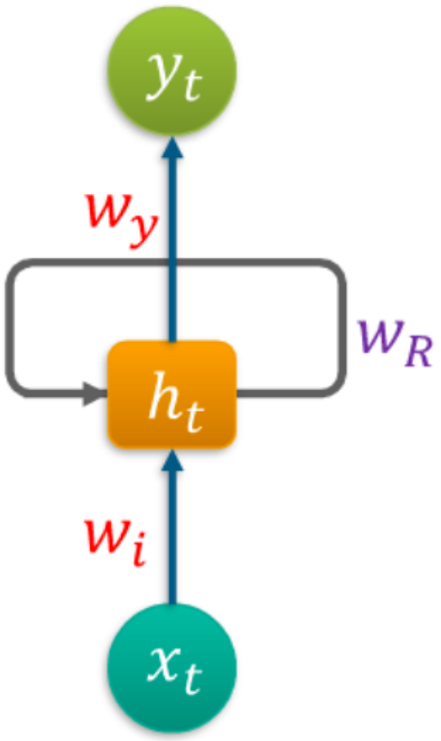
Time Series	Gene Sequences	Sentences	Stock prices

- **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

$$h^{(t)} = g_h\left(w_i x^{(t)} + w_R h^{(t-1)} + b_h\right)$$

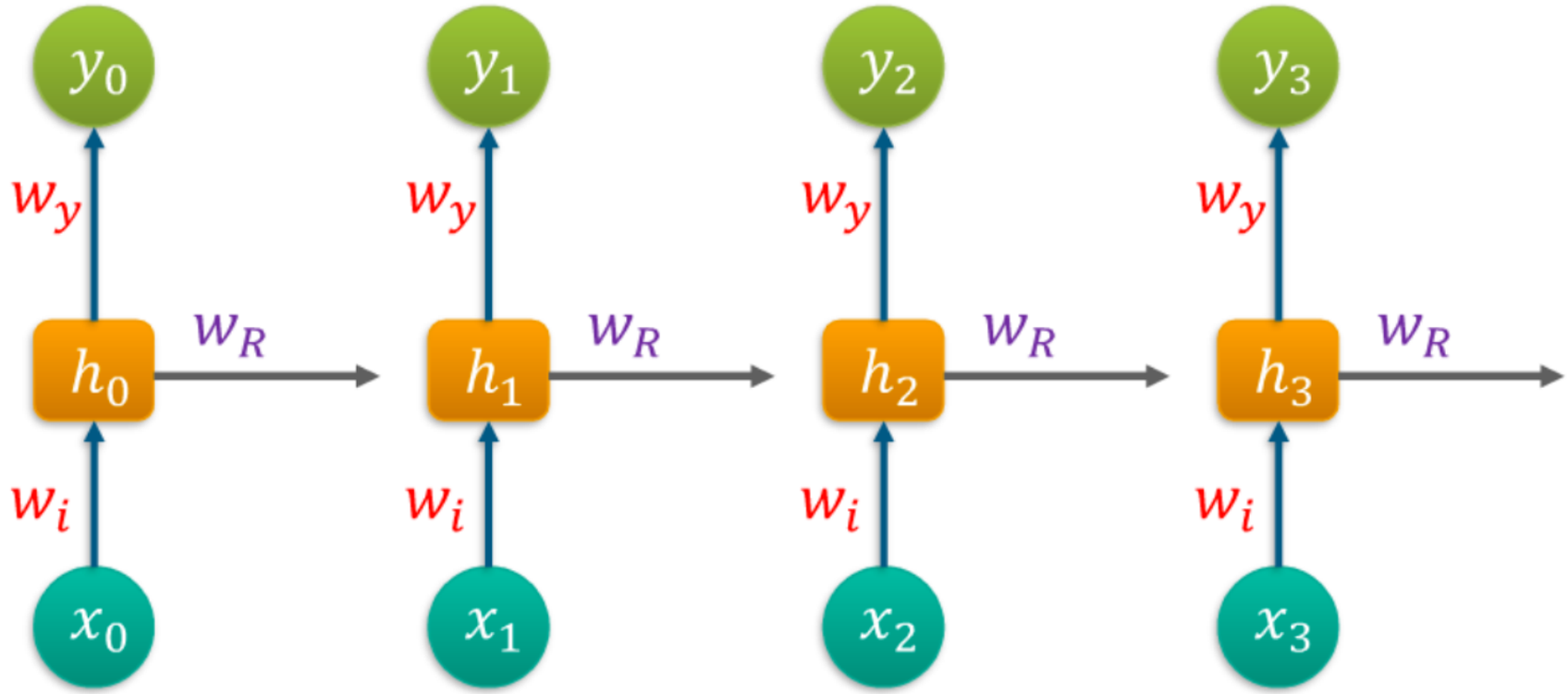
$$y^{(t)} = g_y\left(w_y h^{(t)} + b_y\right)$$

Activity at previous time stamp

Labelling of parameters used

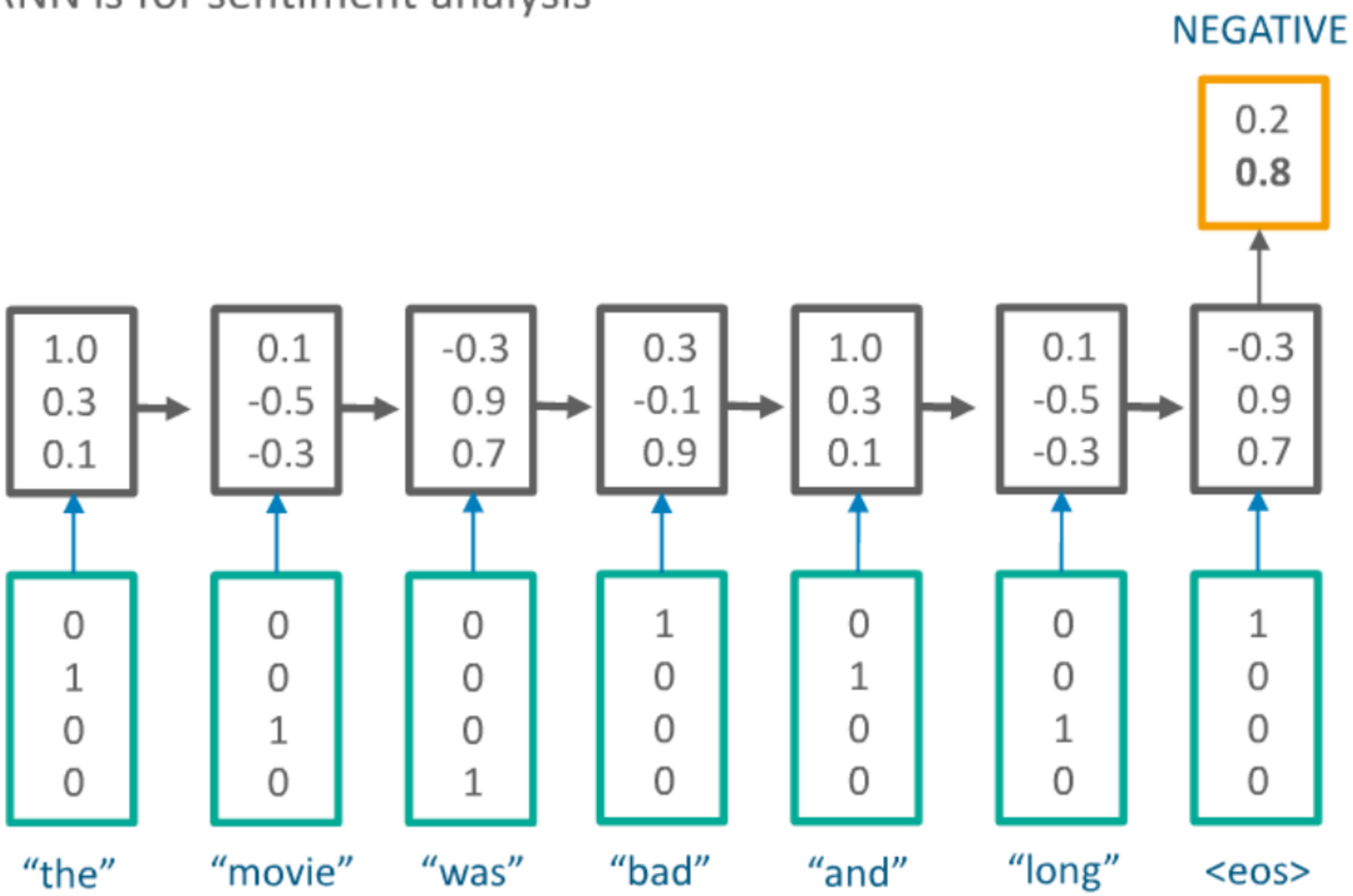
Structure of RNN

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



Example

- Another use of RNN is for sentiment analysis



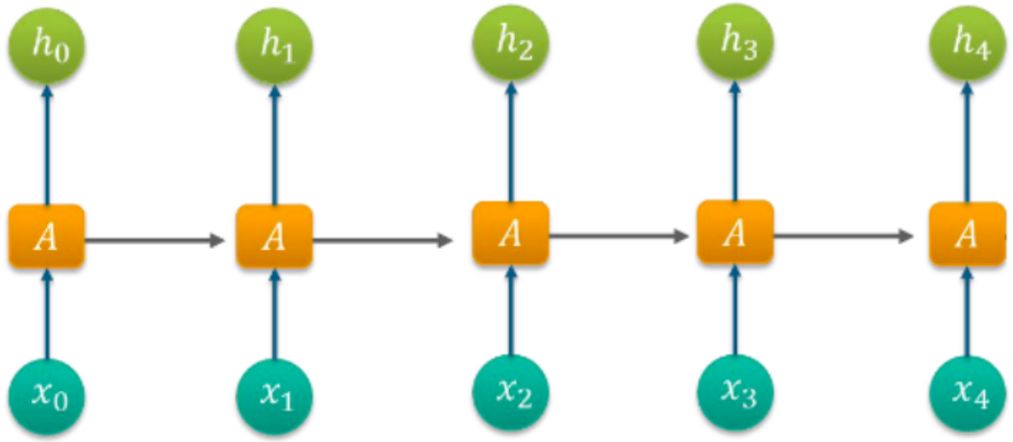
What are LSTMs?

- 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?

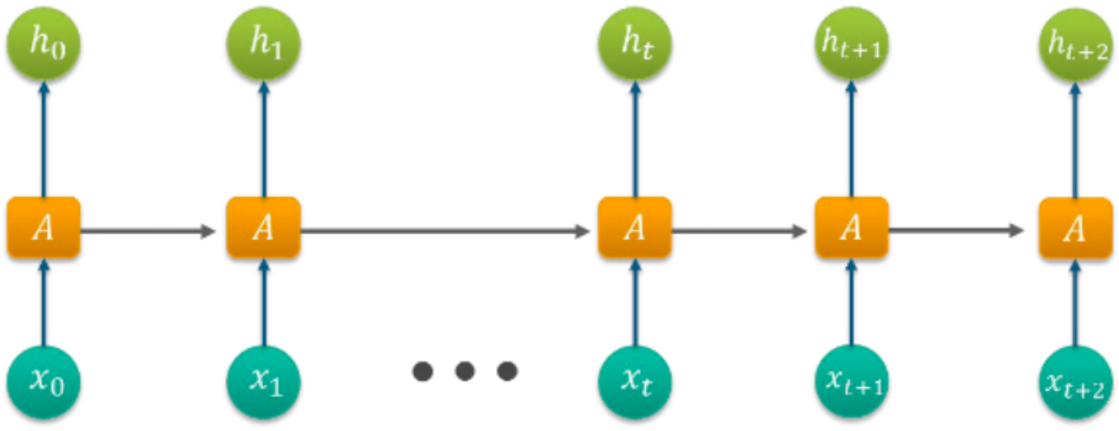
Consider an example where we want to predict the last word of a given text.

- **Short Term Memory**



The clouds are in the ***sky***

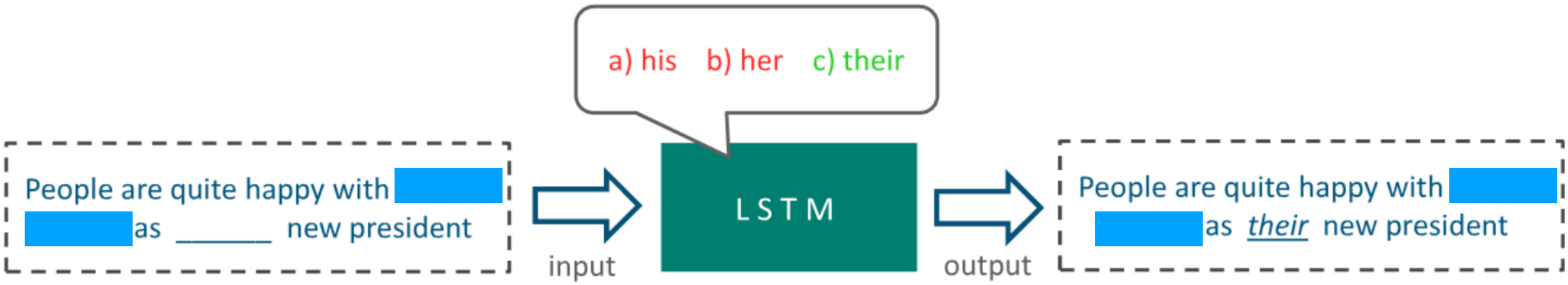
- **Long Term Memory**

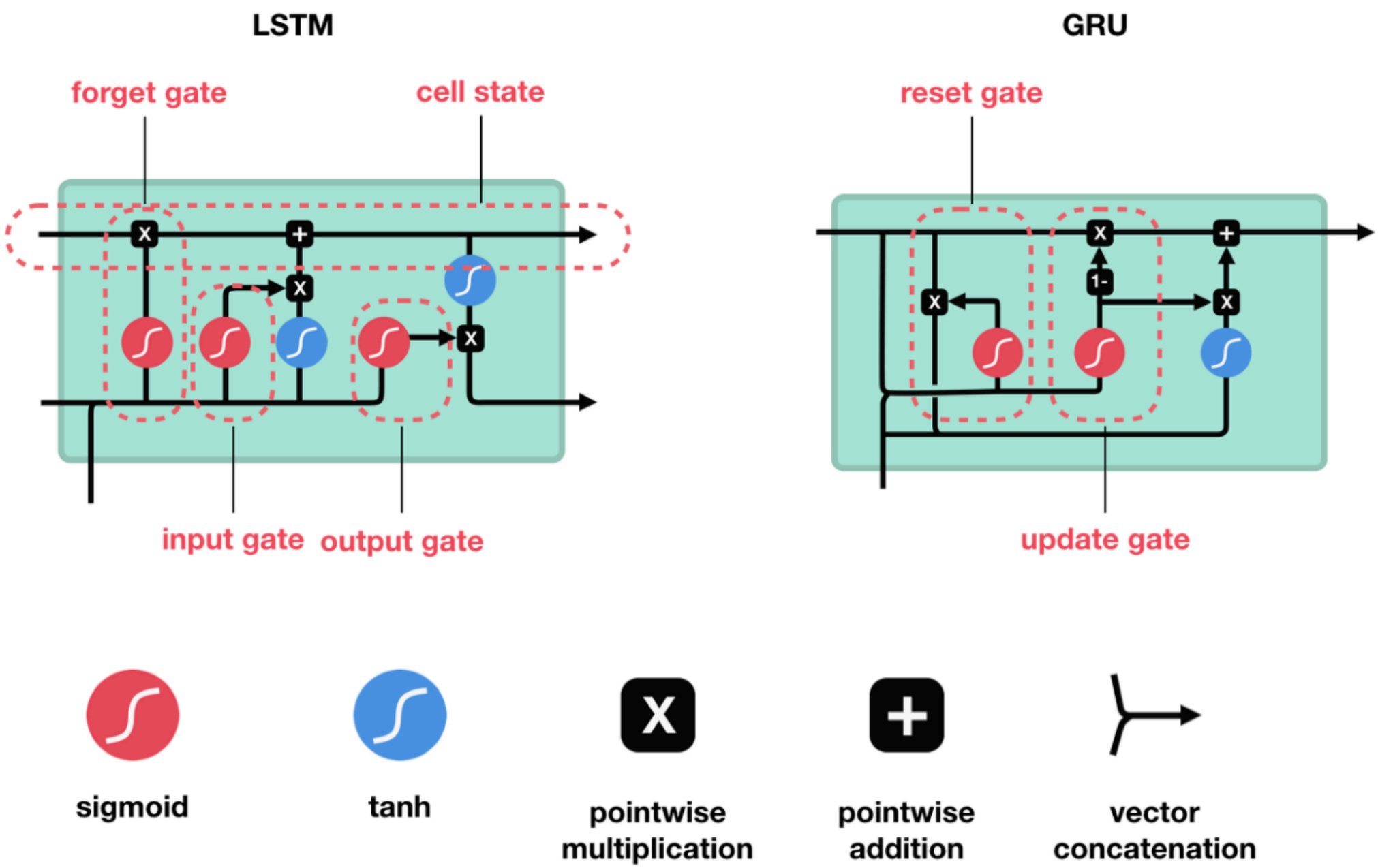


I grew in France… I speak fluent ***French***.

LSTM Working Steps

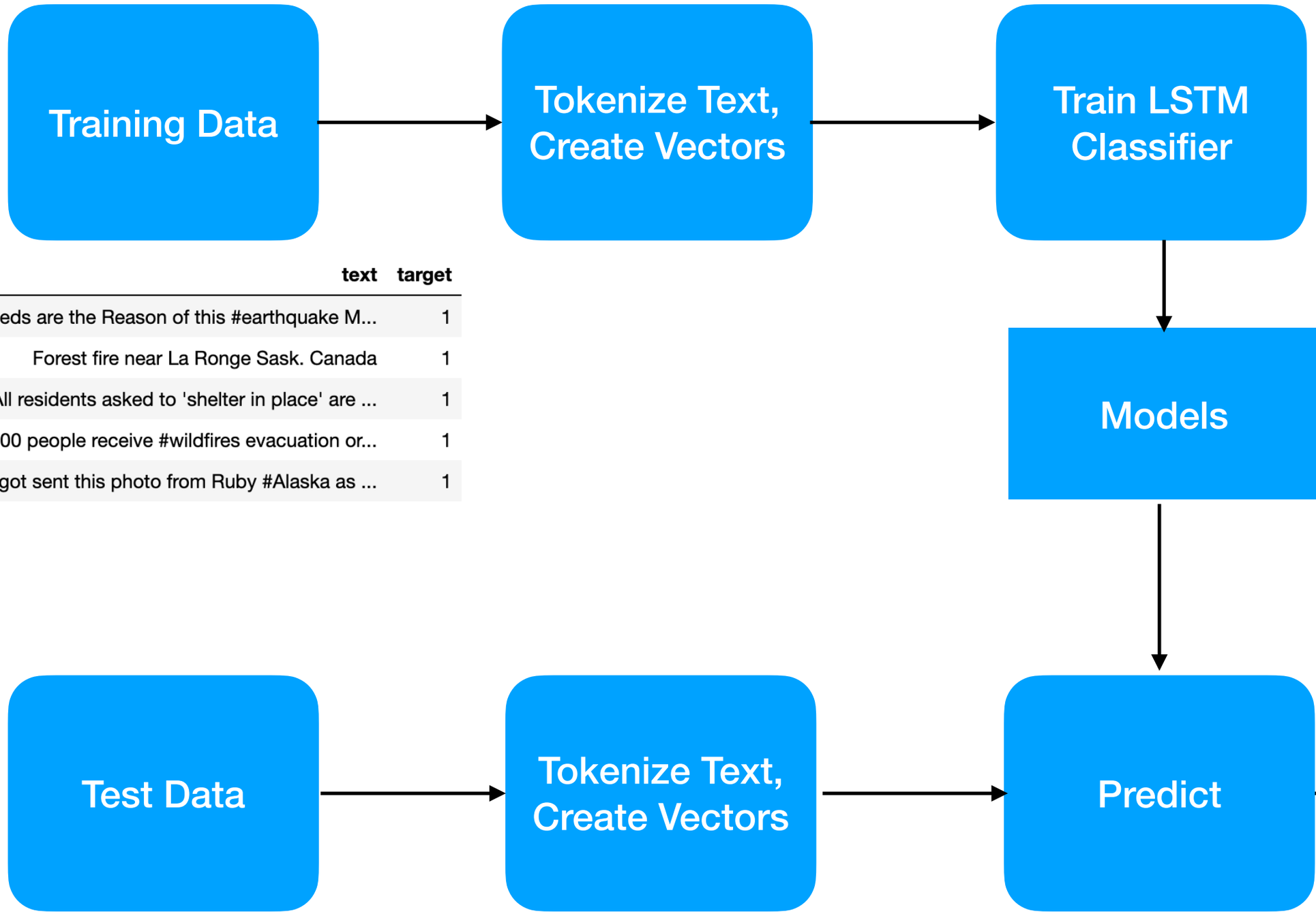
- For the language model example, since it just saw a subject, it might want to output information relevant to a verb, in case that's what is coming next.
- For example, it might output whether the subject is singular or plural, so that we know what form a verb should be conjugated into if that's what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





https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

Lab: Quora Comment Classifier



text	target
Our Deeds are the Reason of this #earthquake M...	1
Forest fire near La Ronge Sask. Canada	1
All residents asked to 'shelter in place' are ...	1
13,000 people receive #wildfires evacuation or...	1
Just got sent this photo from Ruby #Alaska as ...	1

Lab: Quora Insincere Questions Classification using LSTM

Click this link to begin

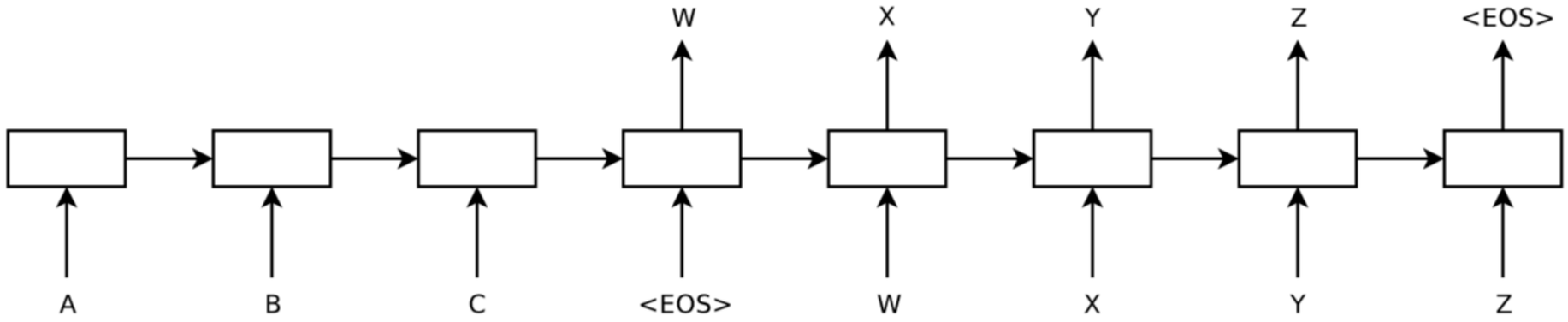
https://colab.research.google.com/github/ravi-ilango/aicamp-mar-2021/blob/main/lab2_1/quora_classifier_lstm.ipynb

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

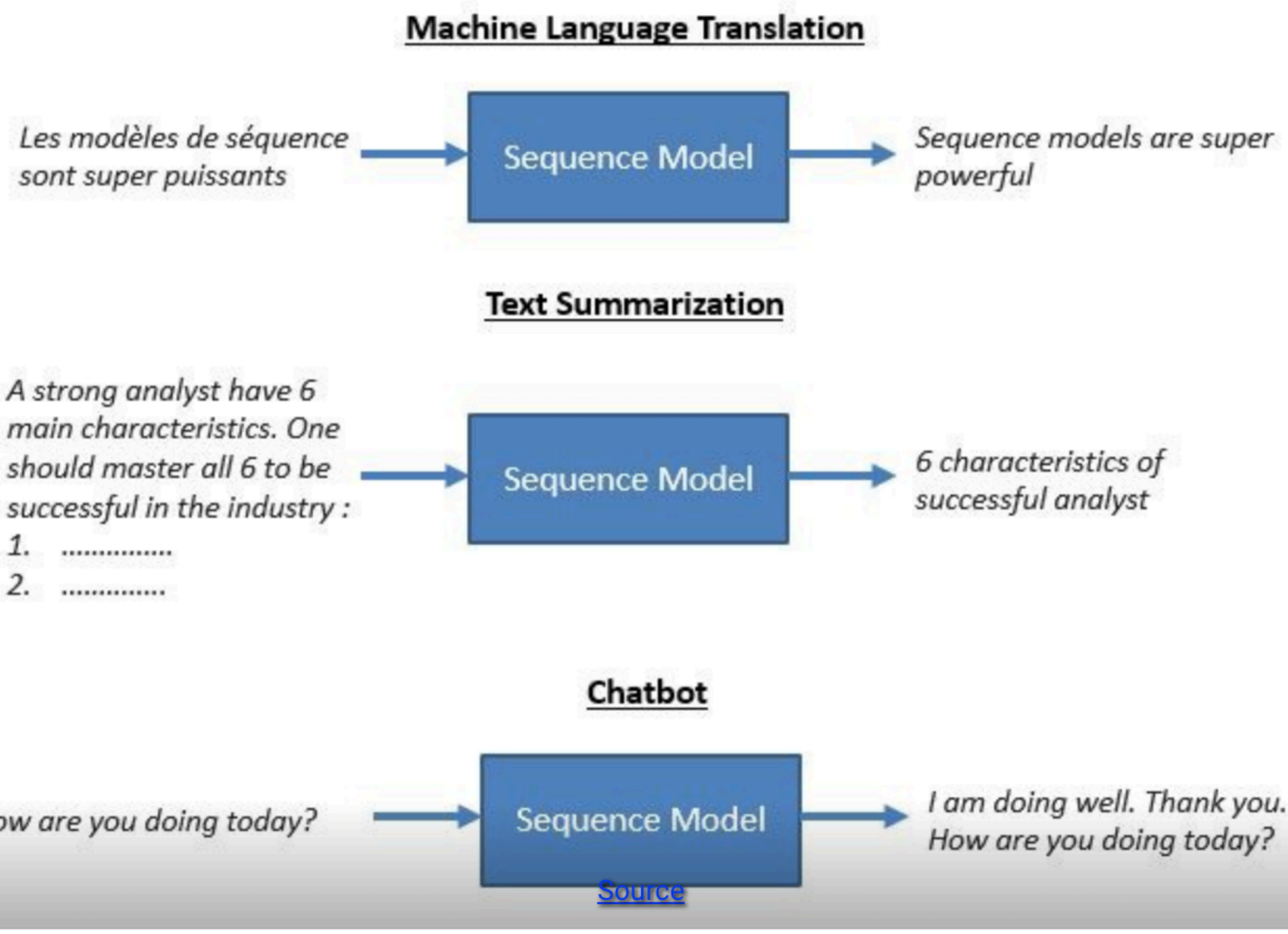
Sequence-to-sequence learning (Seq2Seq) is about training models to convert sequences from one domain (e.g. sentences in English) to sequences in another domain (e.g. the same sentences translated to French)



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

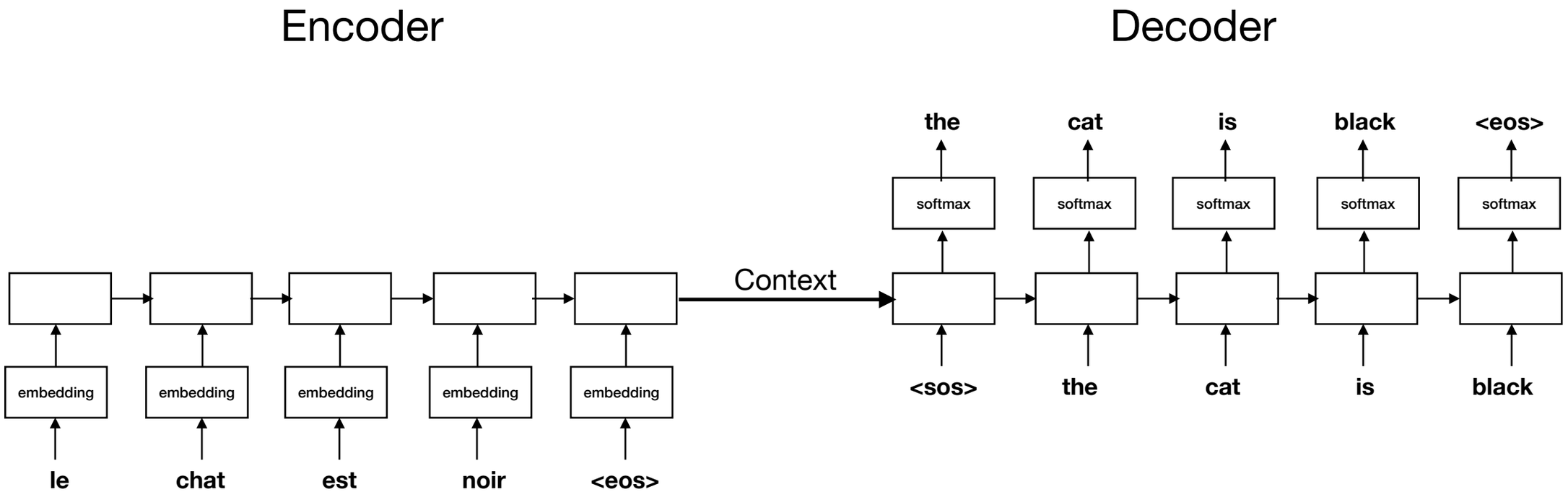
https://arxiv.org/pdf/1409.3215.pdf

Sequence-to-sequence learning - use cases



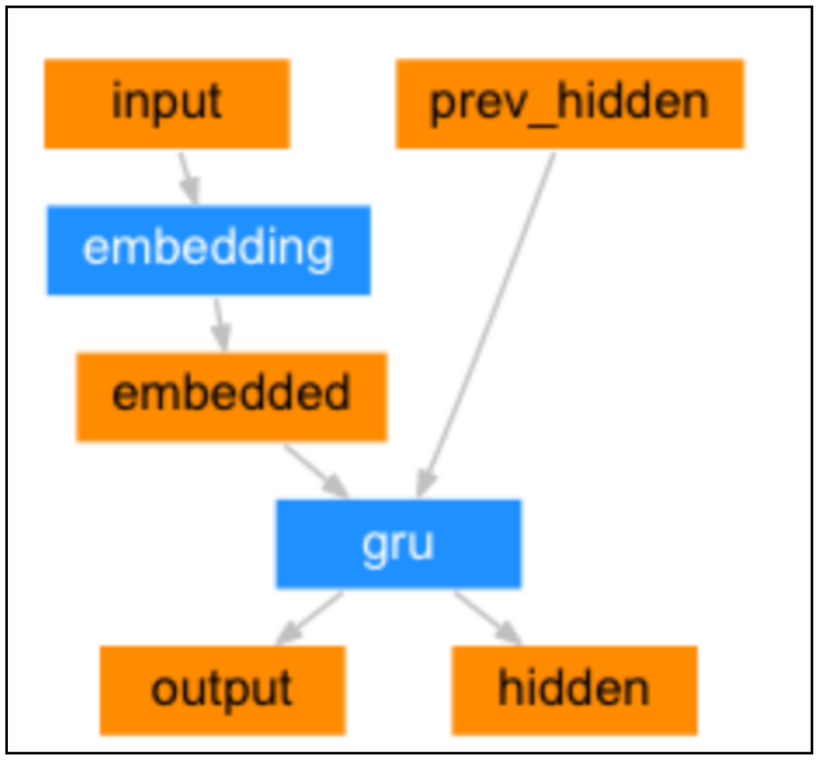
https://www.analyticsvidhya.com/blog/2020/08/a-simple-introduction-to-sequence-to-sequence-models/

French to English Translation



https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html

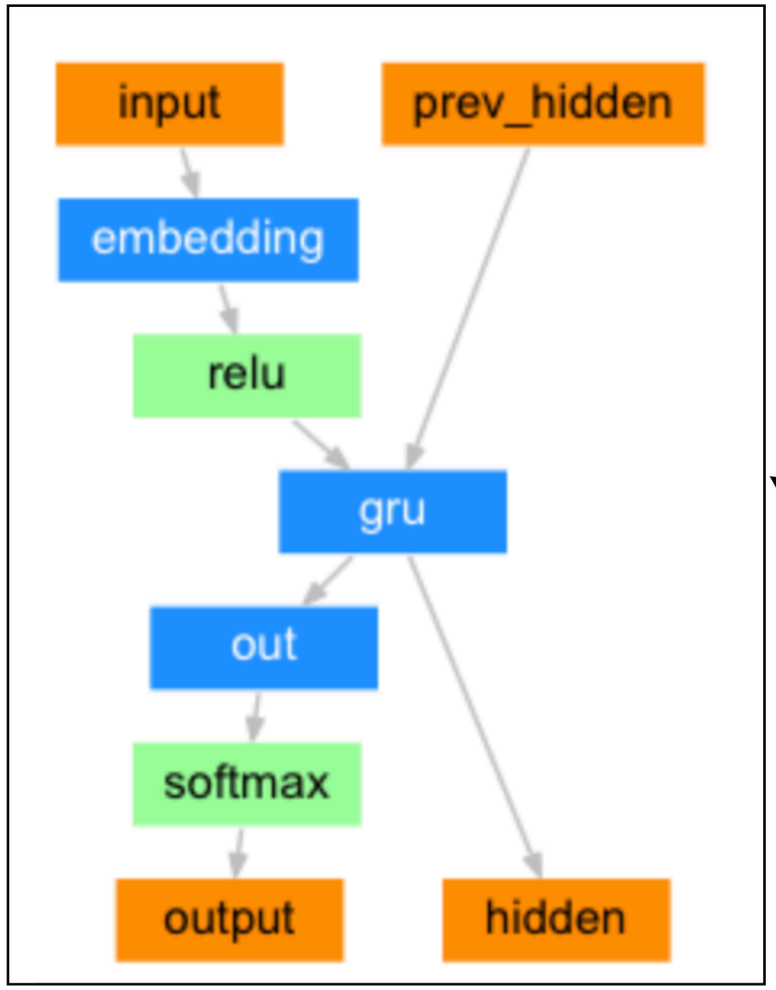
Encoder




```
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)
```

https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html

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)
```

https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html

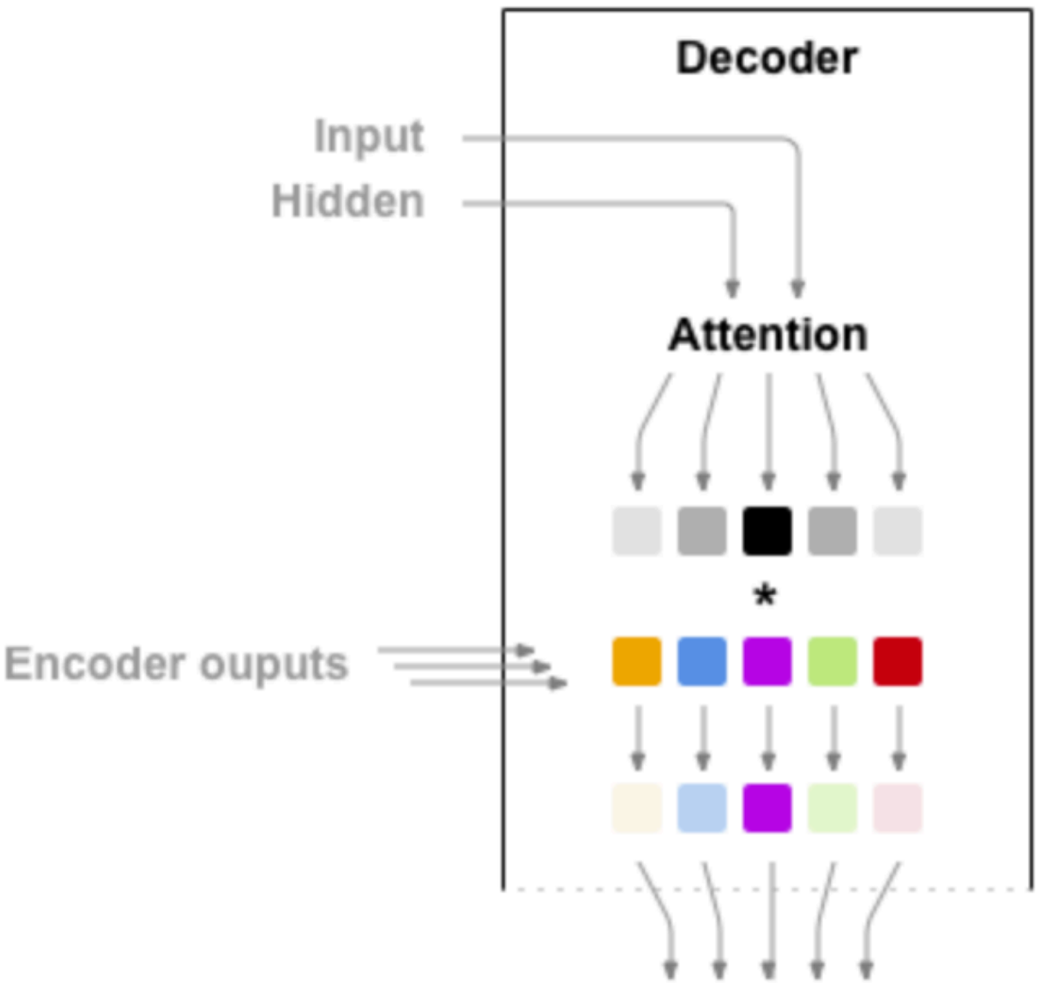
Lab: French to English Translation using RNN

Click this link to begin

https://colab.research.google.com/github/ravi-ilango/aicamp-mar-2021/blob/main/lab1/seq2seq_translation_tutorial.ipynb

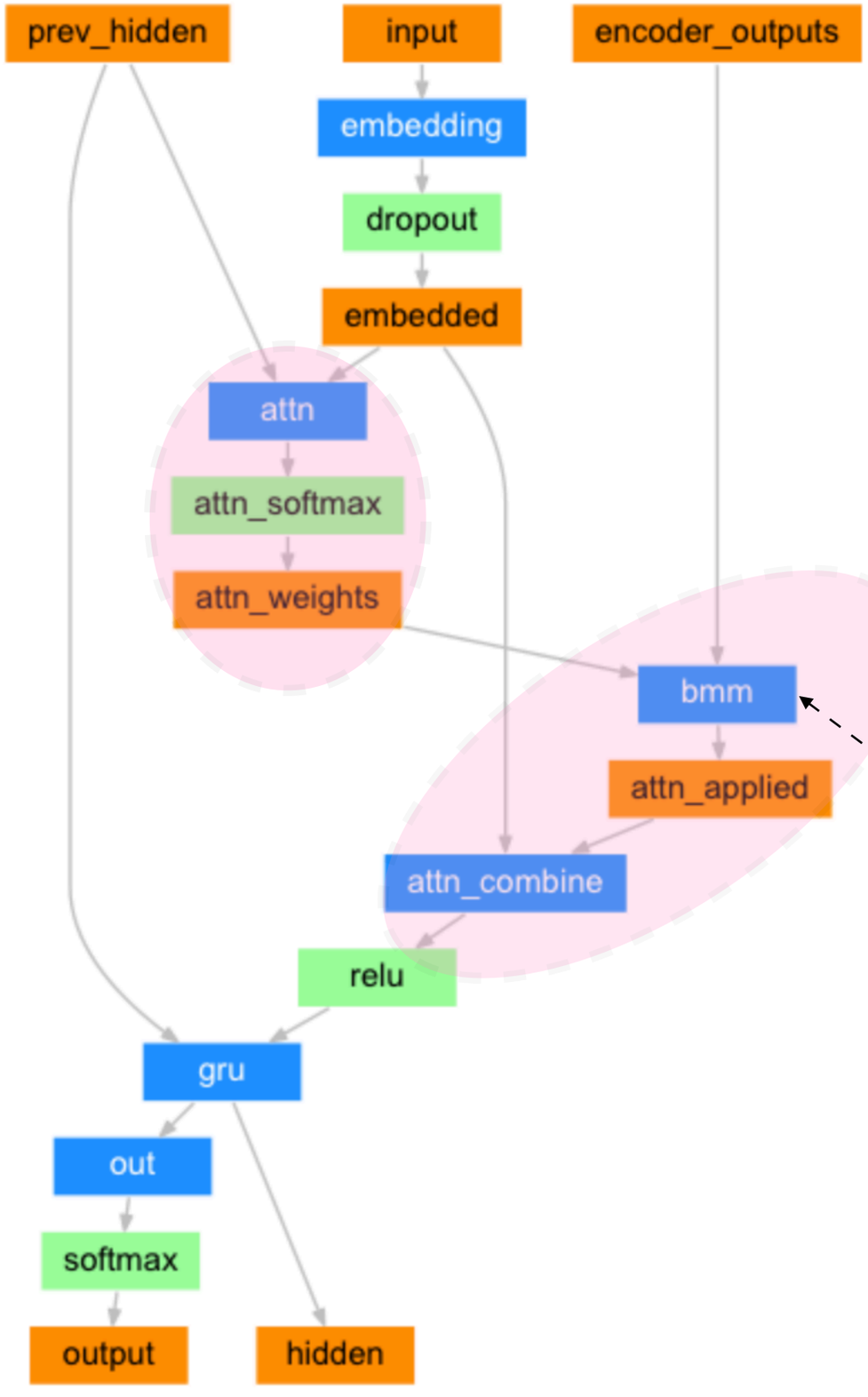
Based on https://pytorch.org/tutorials/

Attention Network



https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html

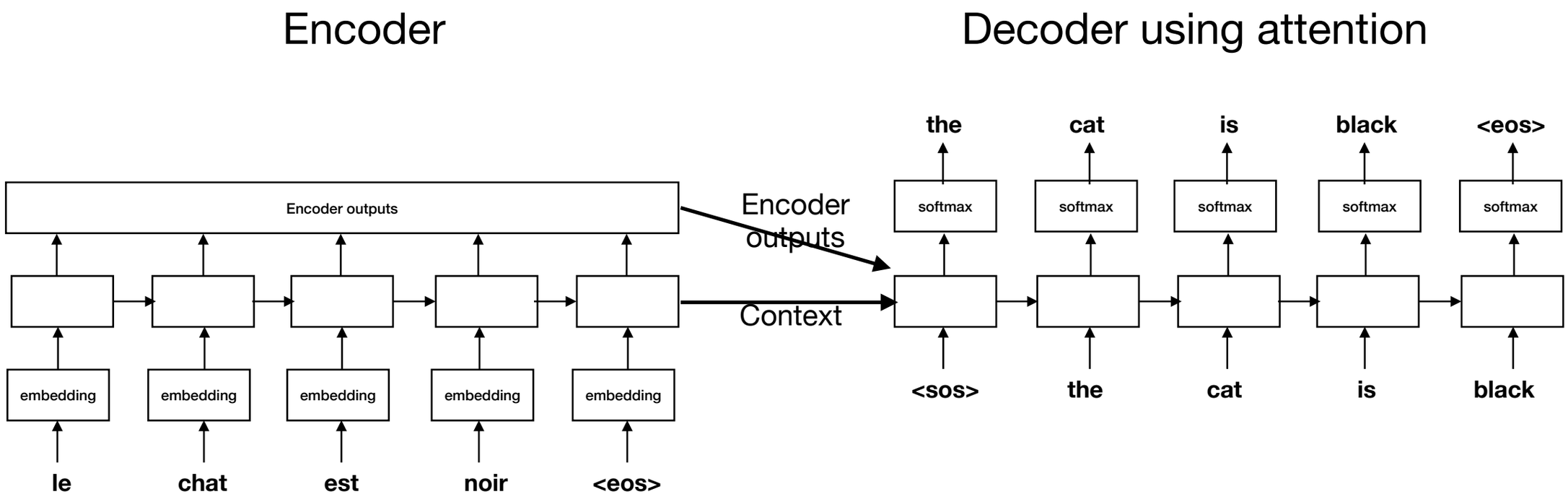
Attention Decoder



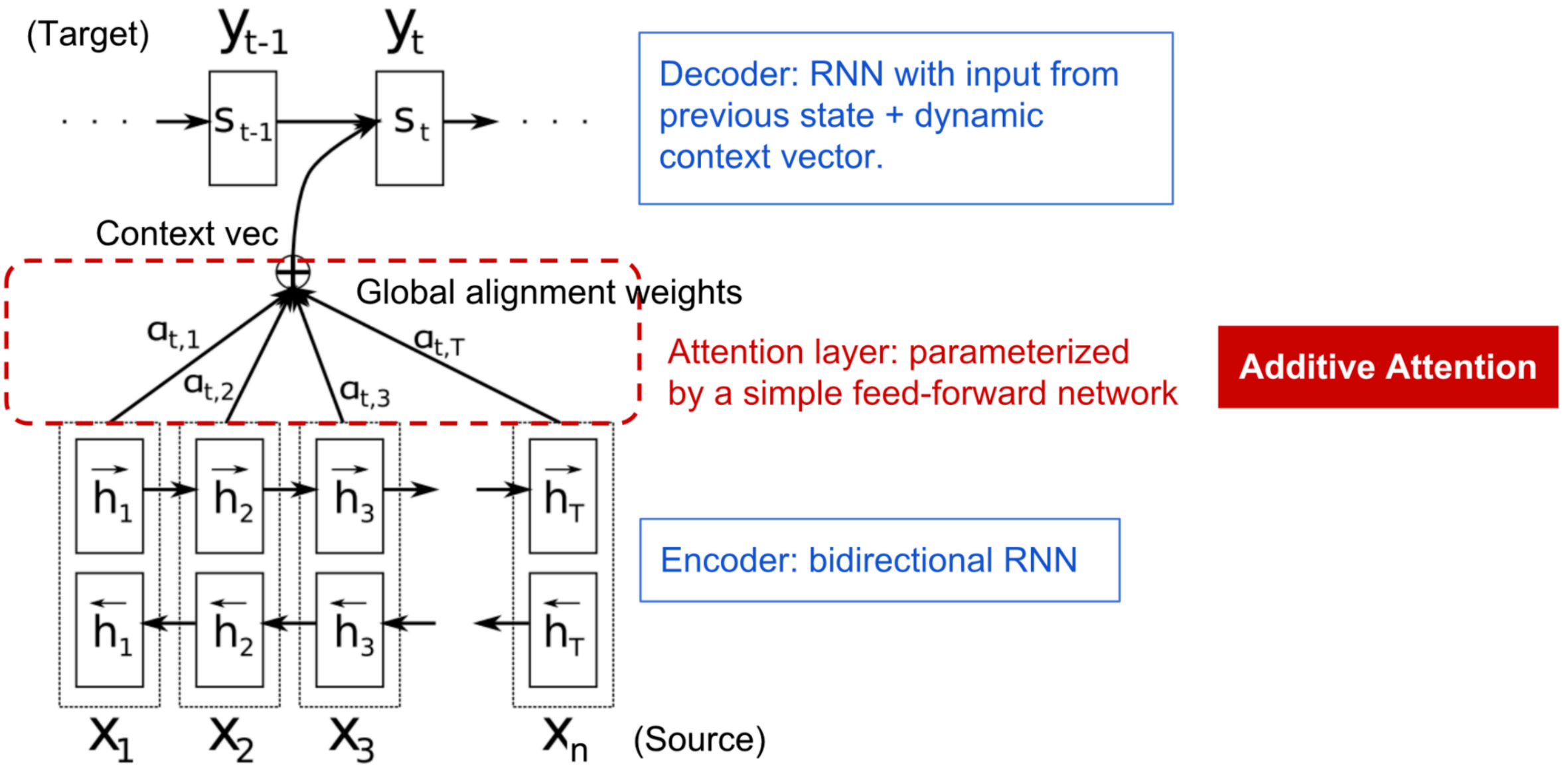
If `input` is a $(b \times n \times m)$ tensor, `mat2` is a $(b \times m \times p)$ tensor, `out` will be a $(b \times n \times p)$ tensor.

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

Sequence to Sequence Translation using Attention



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



https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

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/ravi-ilango/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.

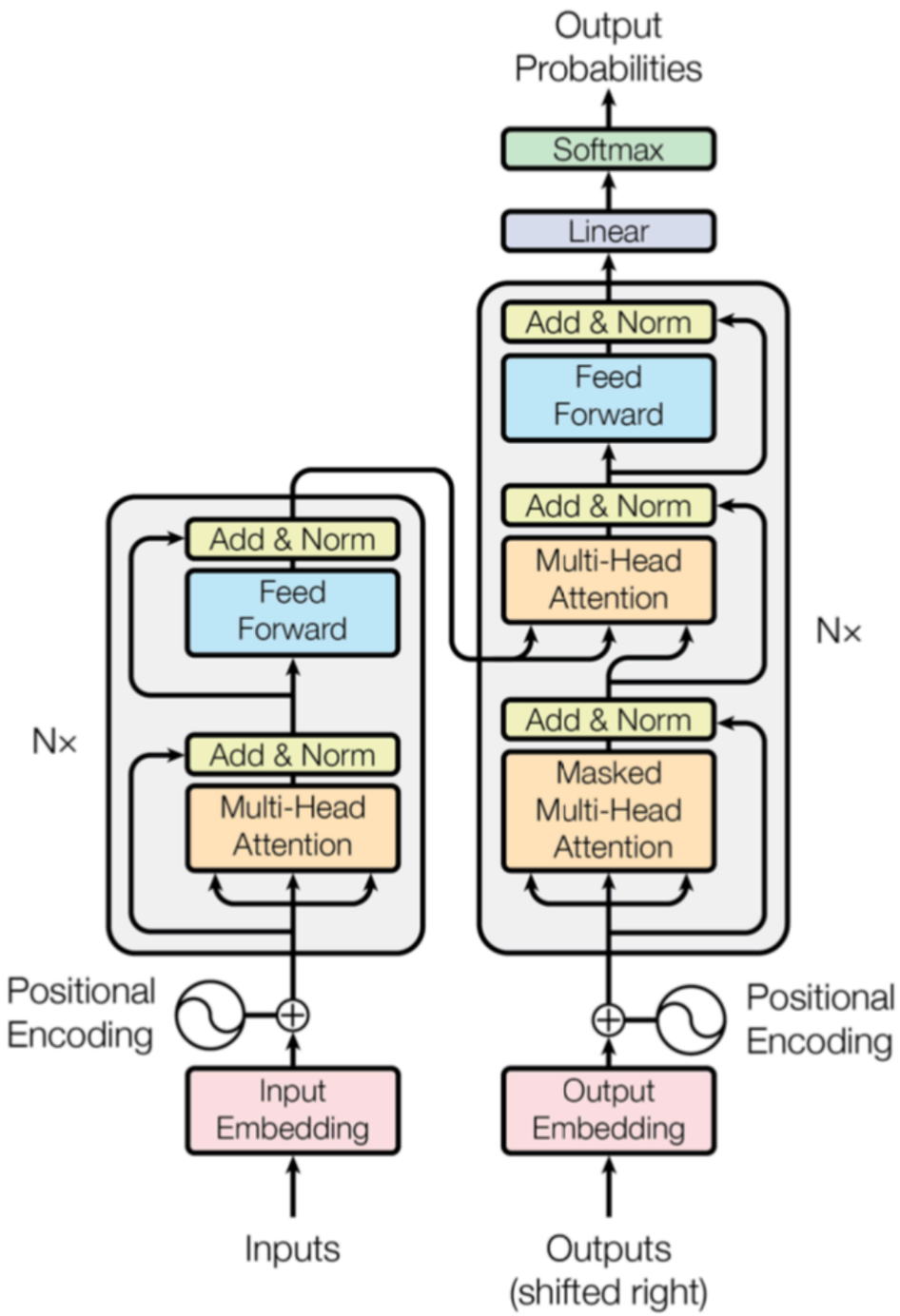
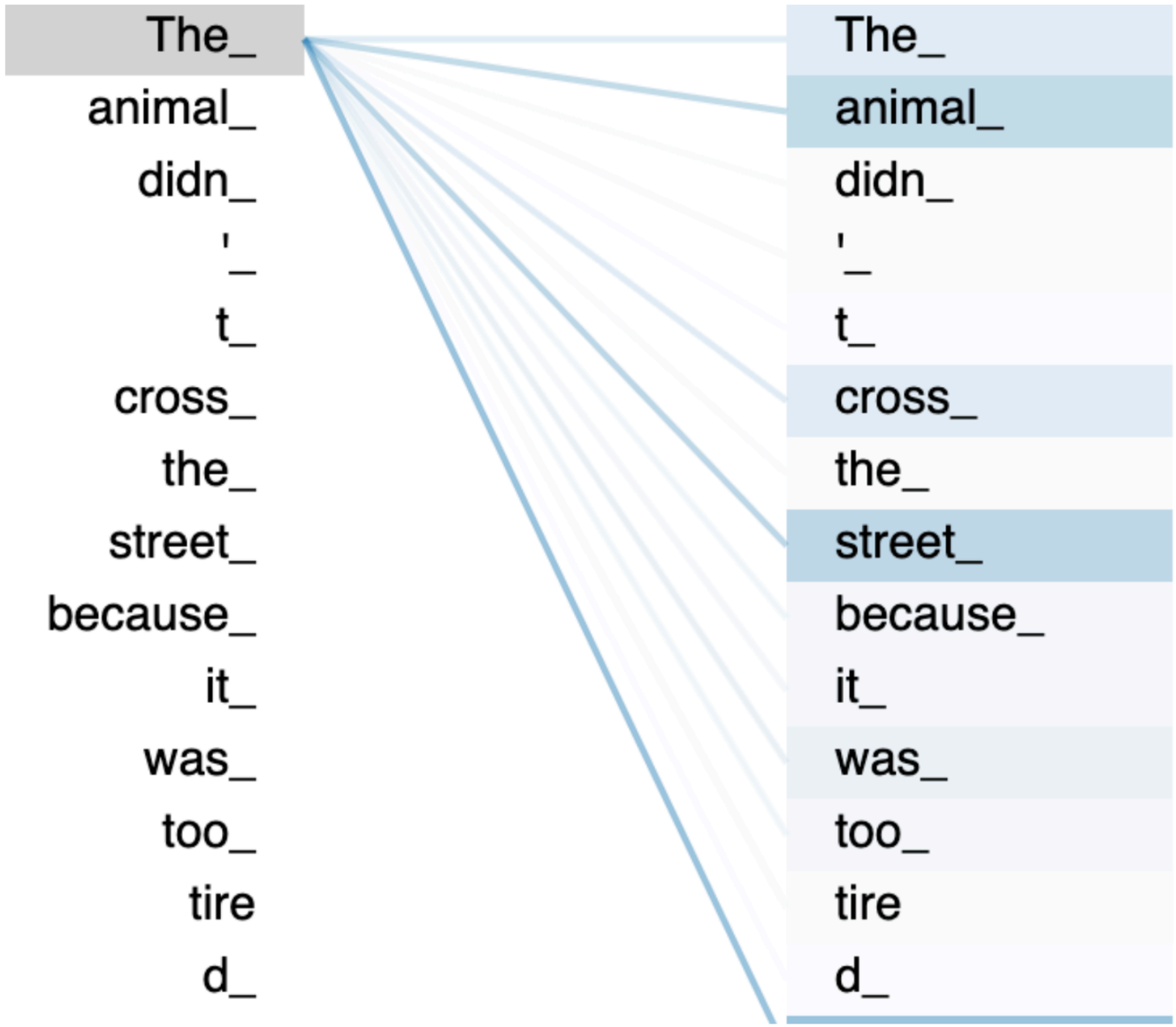


Figure 1: The Transformer - model architecture.

https://arxiv.org/abs/1706.03762

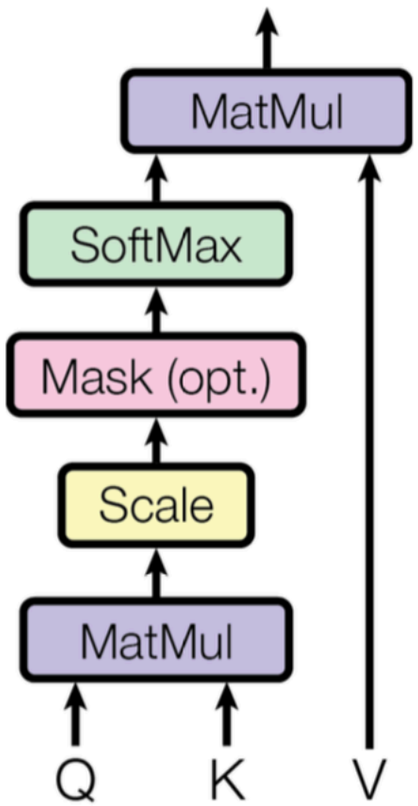
Self Attention



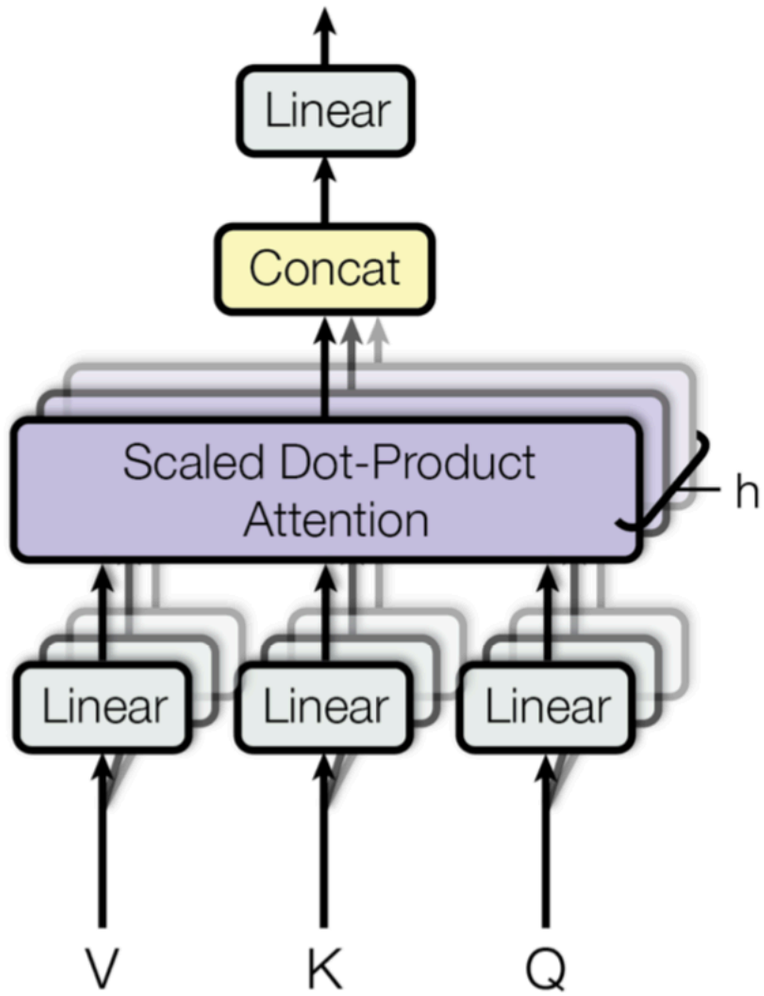
https://towardsdatascience.com/self-attention-5b95ea164f61

Transformers

Scaled Dot-Product Attention



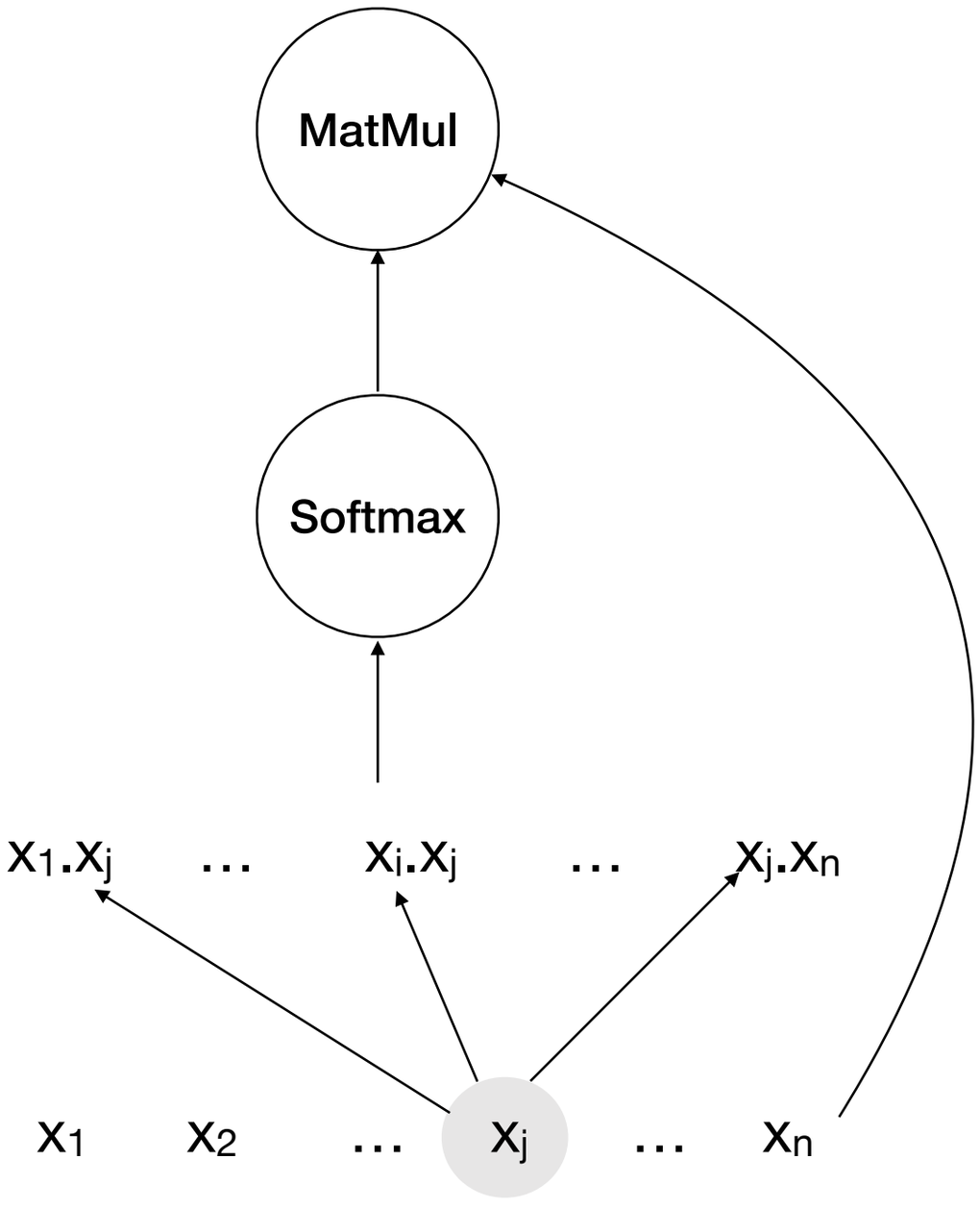
Multi-Head Attention



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

https://arxiv.org/abs/1706.03762

Self Attention



Softmax(Q.K^T). V

Softmax(Q.K^T)

Q.K^T (dot product)

Q, K, V (Query, Keys, Values)

https://towardsdatascience.com/self-attention-5b95ea164f61

Illustration

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, XML (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

http://cs231n.github.io/transfer-learning/

BERT - Bidirectional Encoder Representations from Transformers

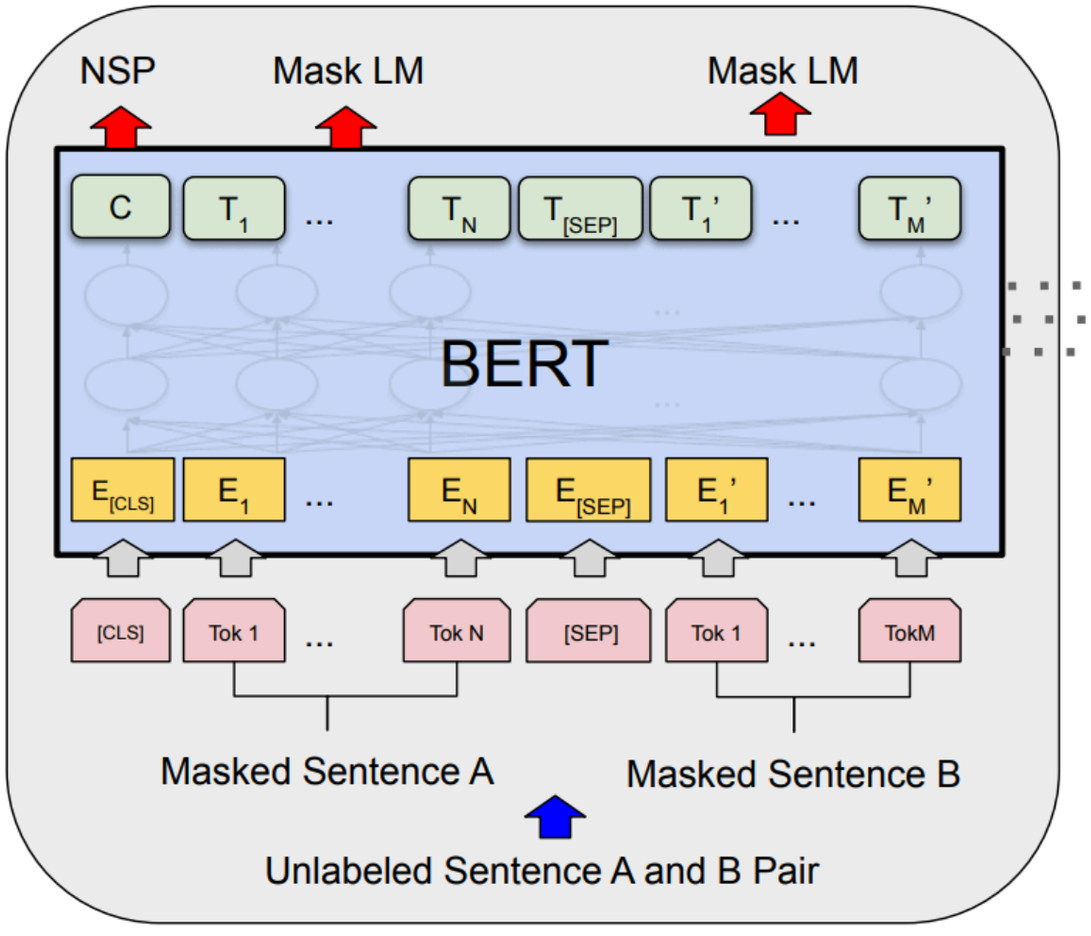
Model Details

- 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

https://arxiv.org/pdf/1810.04805.pdf

BERT - Bidirectional Encoder Representations from Transformers

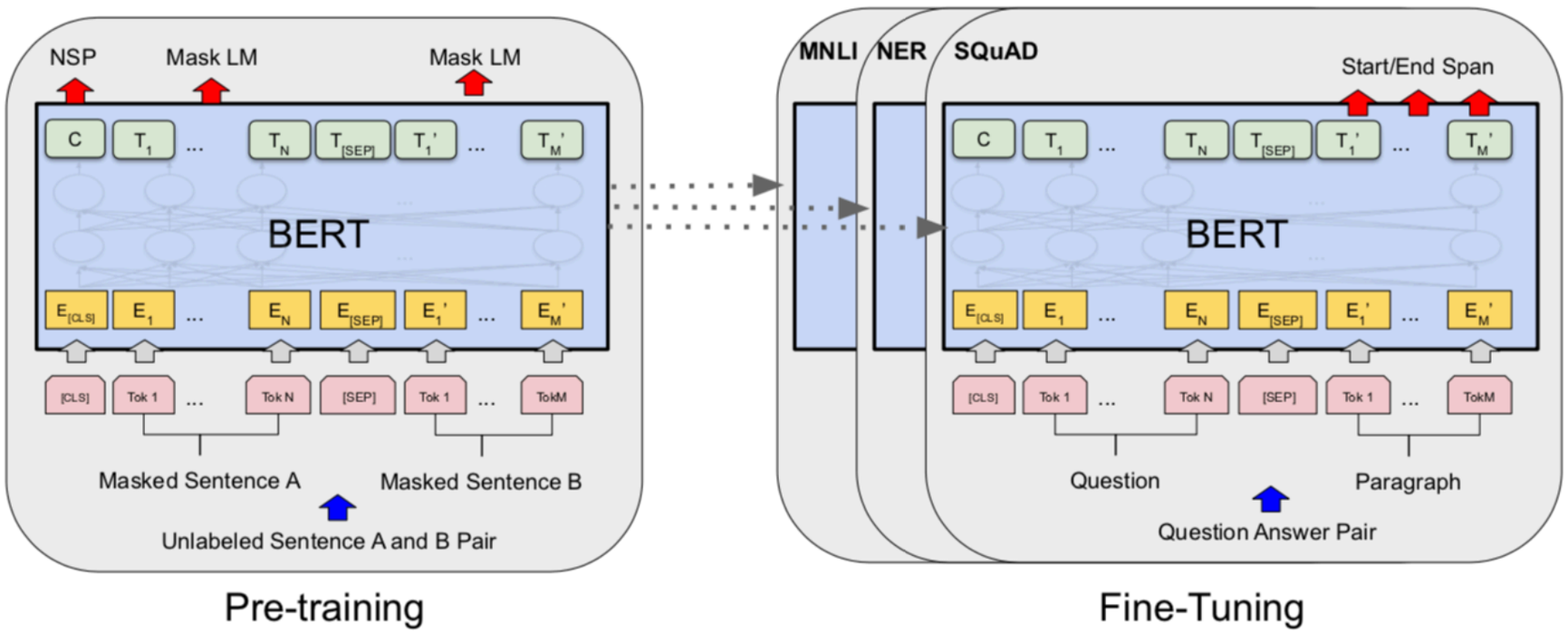
- Pretraining
  - Masked LM
  - Next Sentence Prediction (NSP)
  - Books Corpus (800M words) and English Wikipedia (2,500M words).



Pre-training

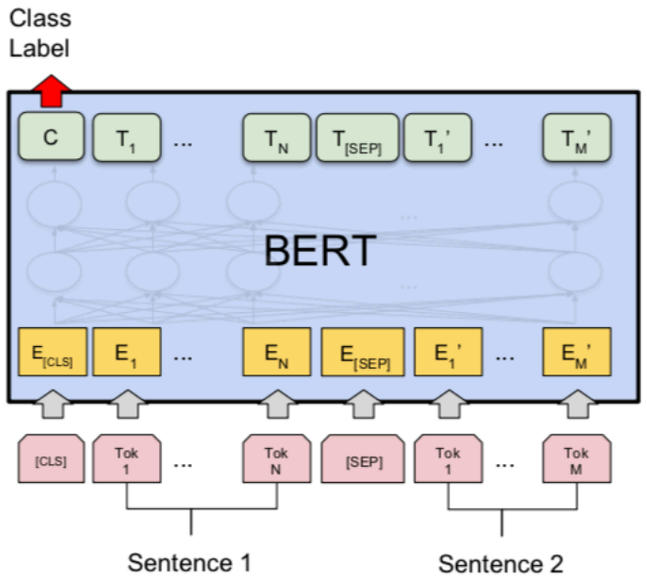
https://arxiv.org/pdf/1810.04805.pdf

BERT - Pretrained Models

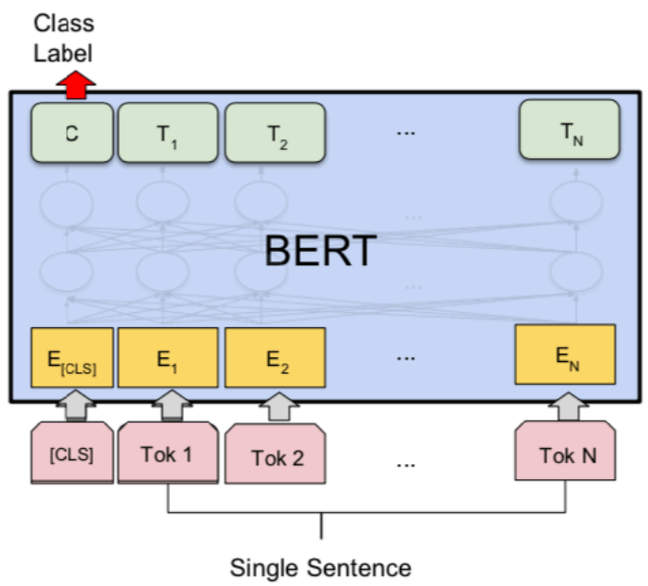


https://arxiv.org/pdf/1810.04805.pdf

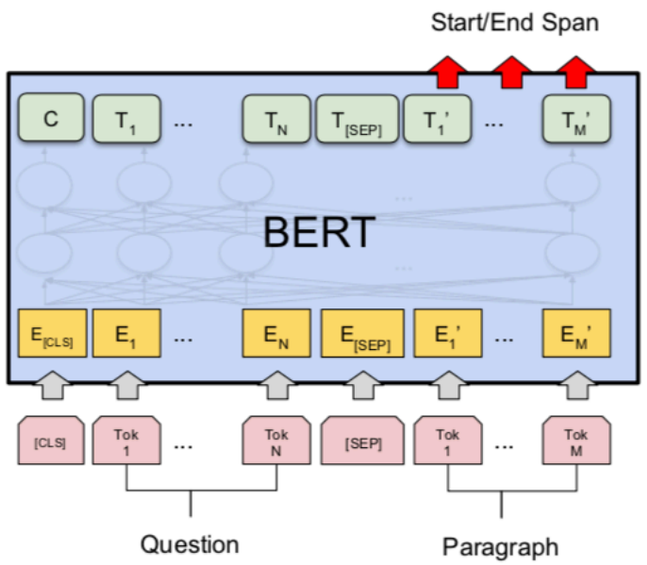
Fine-tuning BERT on Different Tasks



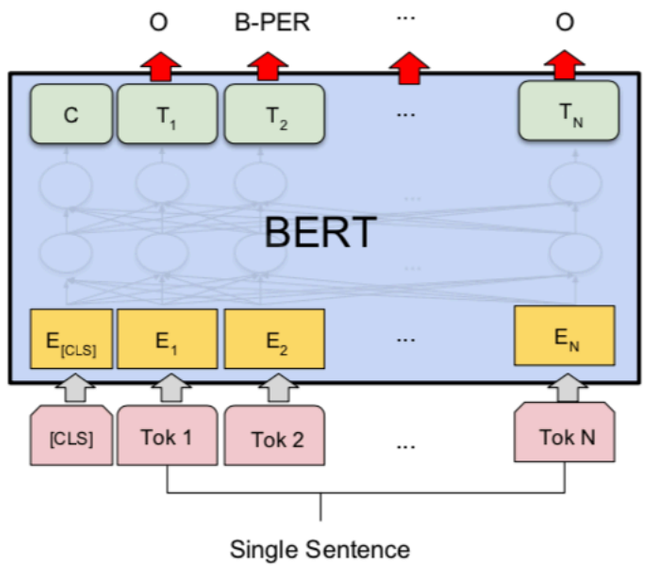
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



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



(c) Question Answering Tasks: SQuAD v1.1



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

https://arxiv.org/pdf/1810.04805.pdf

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

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

XLNet: Generalized Autoregressive Pretraining for Language Understanding

- Uses autoregressive pretraining, avoids using masks, but uses context information from both sides

- Uses Transformer-XL, an efficient learner from long sequences

- Comparison with BERT

Model	SQuAD1.1	SQuAD2.0	RACE	MNLI	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B
BERT-Large (Best of 3)	86.7/92.8	82.8/85.5	75.1	87.3	93.0	91.4	74.0	94.0	88.7	63.7	90.2
XLNet-Large-wikibooks	88.2/94.0	85.1/87.8	77.4	88.4	93.9	91.8	81.2	94.4	90.0	65.2	91.1

Table 1: Fair comparison with BERT. All models are trained using the same data and hyperparameters as in BERT. We use the best of 3 BERT variants for comparison; i.e., the original BERT, BERT with whole word masking, and BERT without next sentence prediction.

https://arxiv.org/pdf/1906.08237.pdf

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

Text Summarization

- Extractive summarization
  - Extract parts of text from the corpus and arrange them to form a summary
  - NLTK Summarizer, Gensim Summarizer
- Abstractive summarization
  - Generate new text using natural language generation

Lab: Text Summarization

https://colab.research.google.com/github/ravi-ilango/aicamp-mar-2021/blob/main/lab6/TextSummarization.ipynb

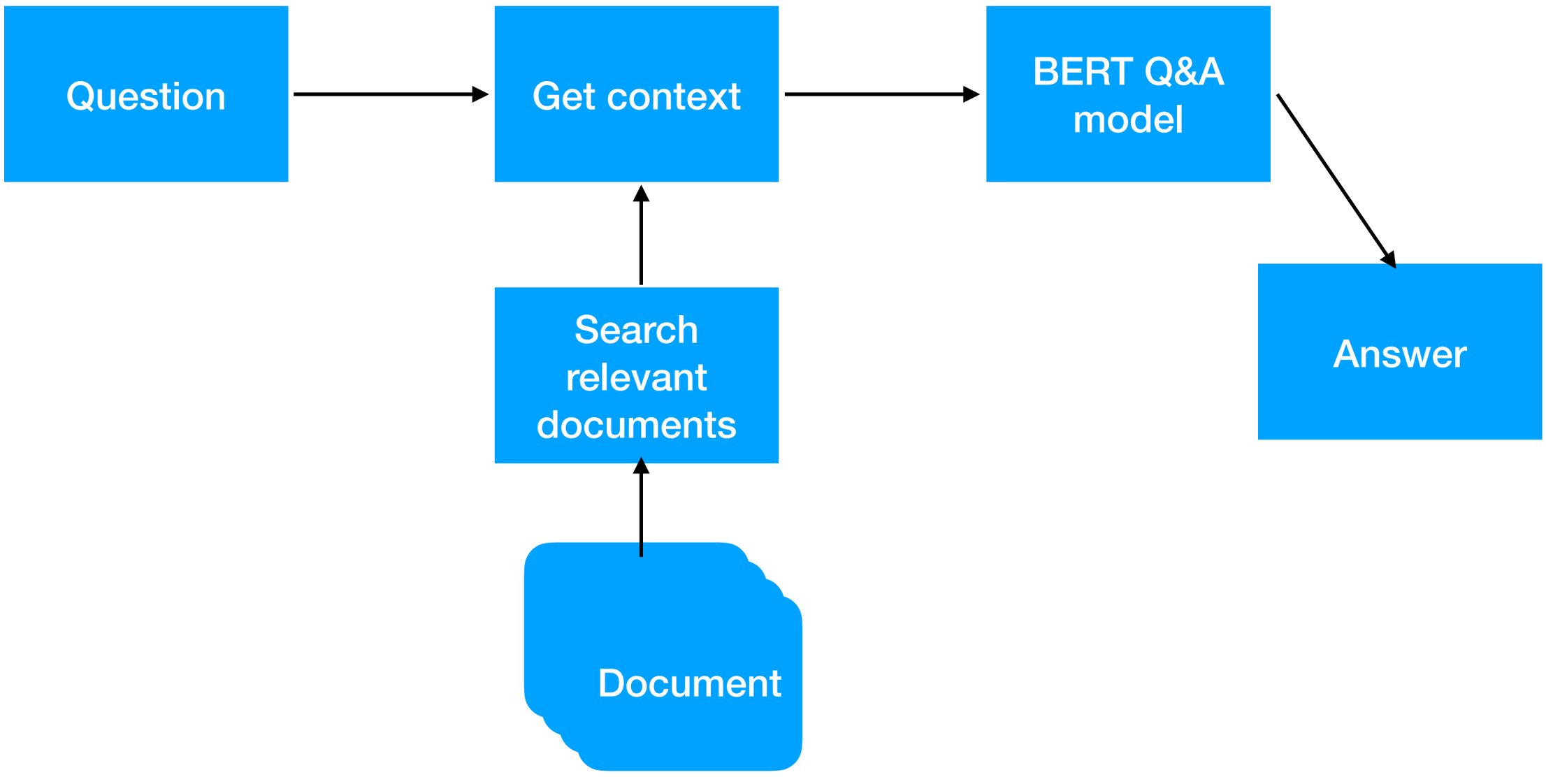
BERT Question and Answering Model

- Pretrained models which can answer questions based on context

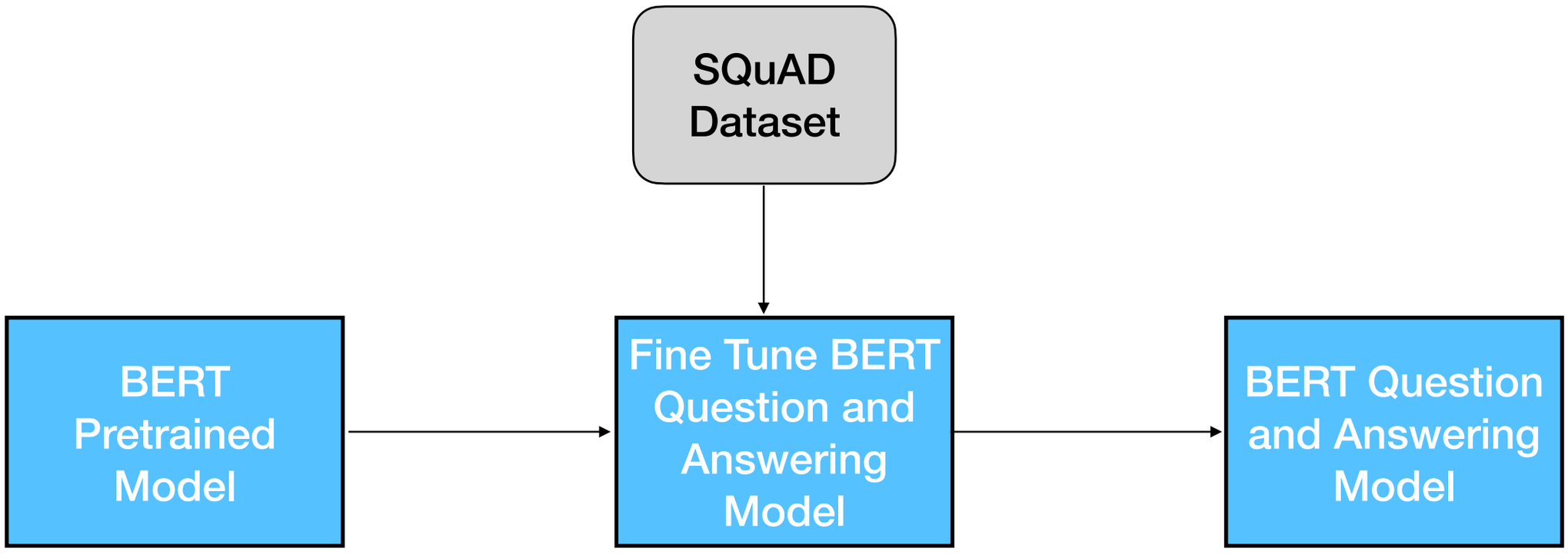
- Trained using Stanford Question and Answer Dataset (SQuAD)

- Can be used to deploy a Question & Annswering solution

BERT Question and Answering Model - Usecase



BERT Question and Answering Model Dev



SQuAD dataset

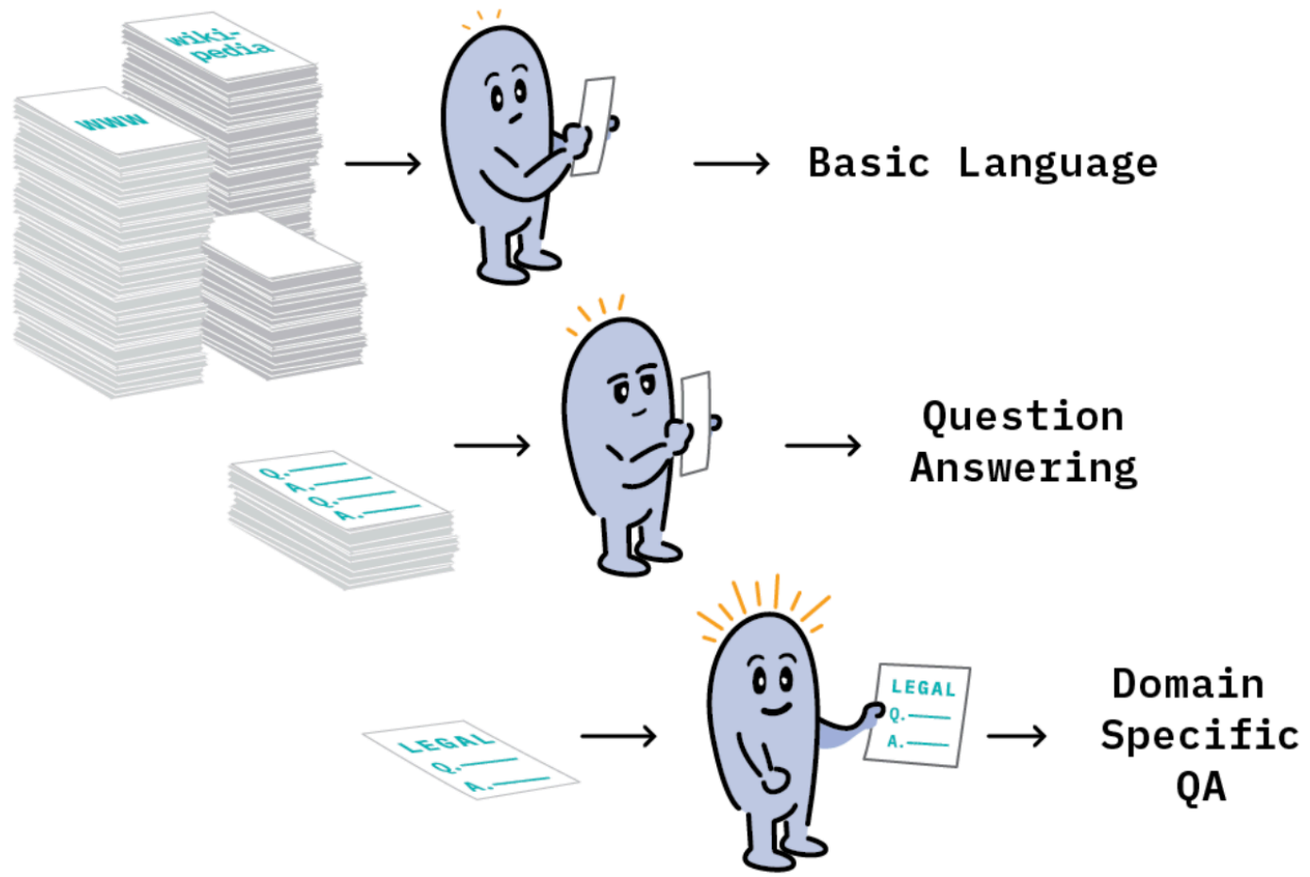
Stanford Question Answering Dataset (source: https://rajpurkar.github.io/SQuAD-explorer/)

SQuAD is a Span-based QA dataset.

```
dataset['validation'][105]
```

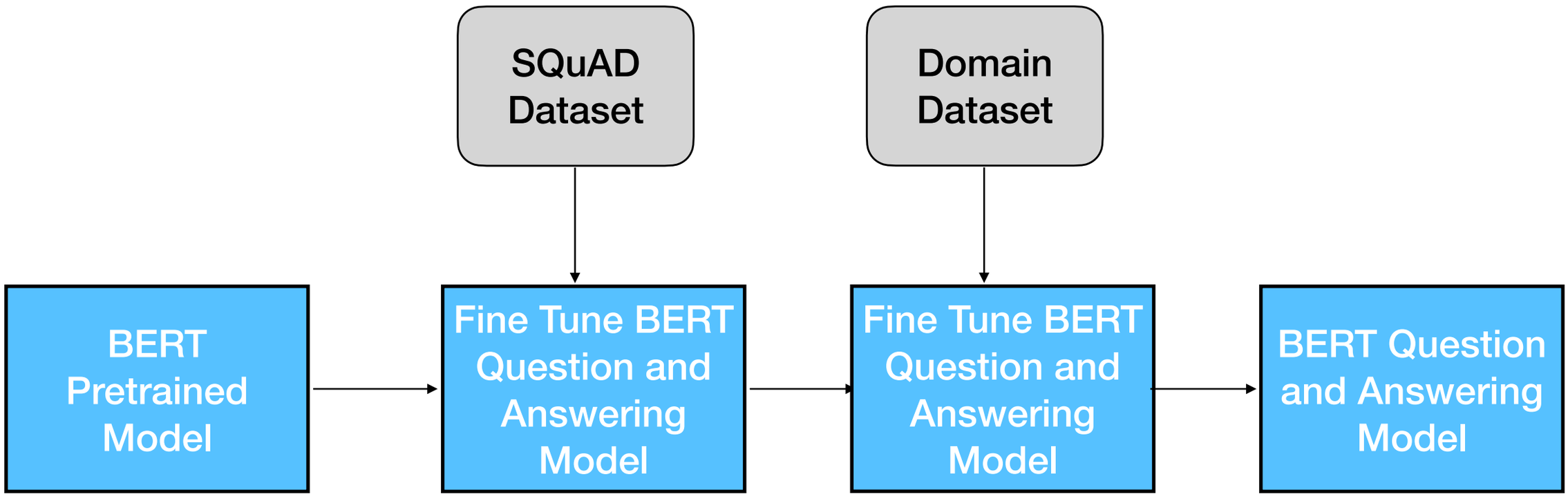
```
{'answers': {'answer_start': [245, 194, 194],
  'text': ['Bruno Mars', 'Coldplay', 'Coldplay']},
 'context': 'CBS broadcast Super Bowl 50 in the U.S., and charged an average of $5 million for a 30-second commercial during the game. The Super Bowl 50 halftime show was headlined by the British rock group Coldplay with special guest performers Beyoncé and Bruno Mars, who headlined the Super Bowl XLVII and Super Bowl XLVIII halftime shows, respectiv ely. It was the third-most watched U.S. broadcast ever.',
 'id': '56d98c53dc89441400fdb548',
 'question': 'What performer lead the Super Bowl XLVIII halftime show?',
 'title': 'Super_Bowl_50'}
```

BERT Question and Answering Model for your domain

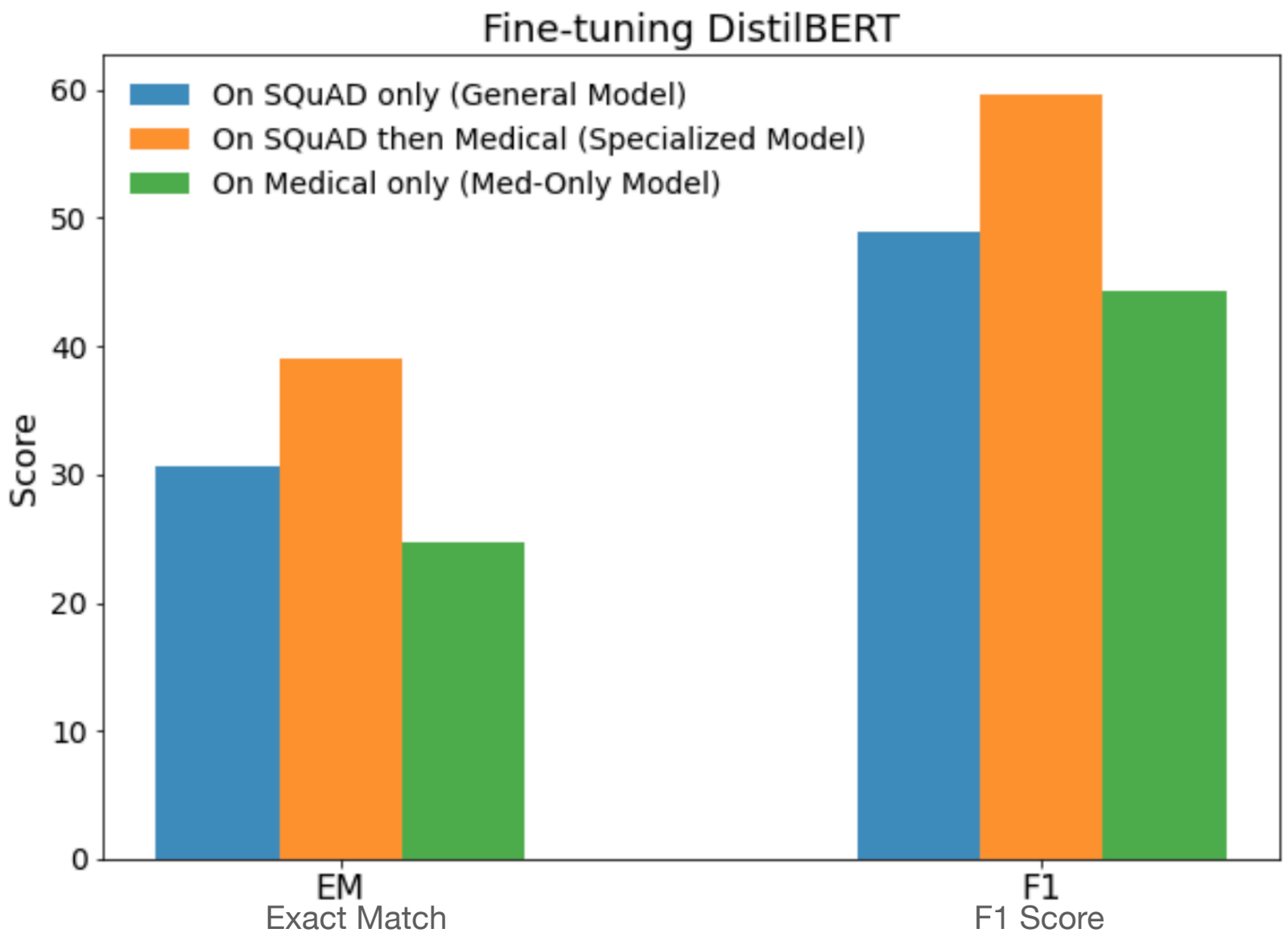


https://qa.fastforwardlabs.com/

BERT Question and Answering Model Dev



Model accuracies



https://qa.fastforwardlabs.com/

Lab: Question and Answer model

Click this link to begin

https://colab.research.google.com/github/ravi-ilango/aicamp-mar-2021/blob/main/lab7/QandA_SQuAD.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, XML (Facebook) and more …

https://www.aclweb.org/anthology/P02-1040.pdf

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.

Q - hi
K - {hi}
V - w ??

- 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.

https://arxiv.org/abs/1706.03762

GLUE

MNLI: The Multi-Genre Natural Language Inference

QQA: Quora Question Pairs

QNLI: Question Natural Language Inference

https://gluebenchmark.com/tasks