

Deep Learning for NLP with PyTorch

Dec 12-13, 2020

Topics

- Sequence to Sequence Model using RNN
- Text Classifiers
 - TFIDF and Logistic Regression
 - Pretrained Models, LSTM
- Text Summarization
- Question and Answering models (Chatbots)
- Tools: PyTorch, Gensim, NLTK

Bio

- **Ravi Ilango**
- Lead Data Scientist, 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

Trust is the new currency in business.

RELYANCE.AI

Trust as a Service
Data Protection Compliance & Legal Ops
Completely Re-Imagined

FUNDING & TEAM

Well-Funded



Unusual
Ventures

Top Notch Talent



ORACLE®

Carnegie
Mellon
University



Massachusetts
Institute of
Technology



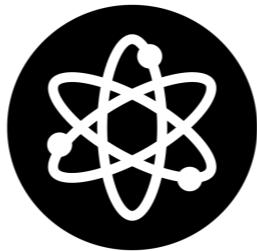
Berkeley
UNIVERSITY OF CALIFORNIA

Google



APPDYNAMICS

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE



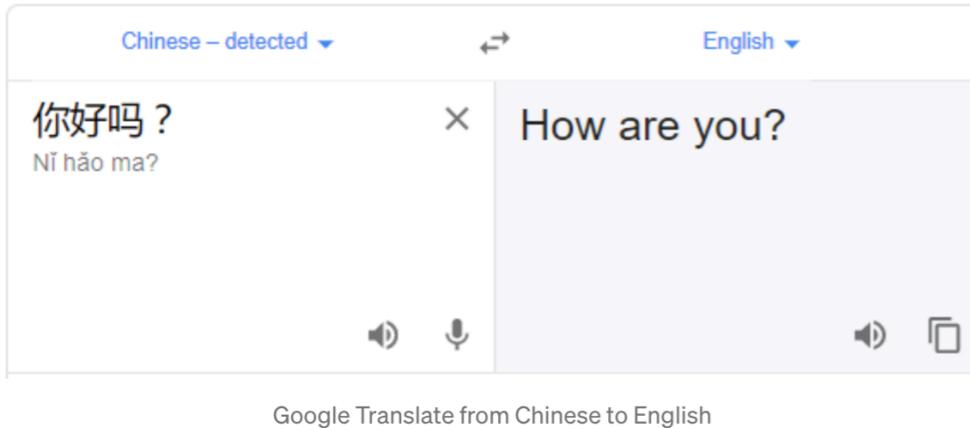
We're hiring:

<https://relyance.ai/careers>

NLP Everyday

Google Translate

used by 500 million people every day



Google Translate from Chinese to English

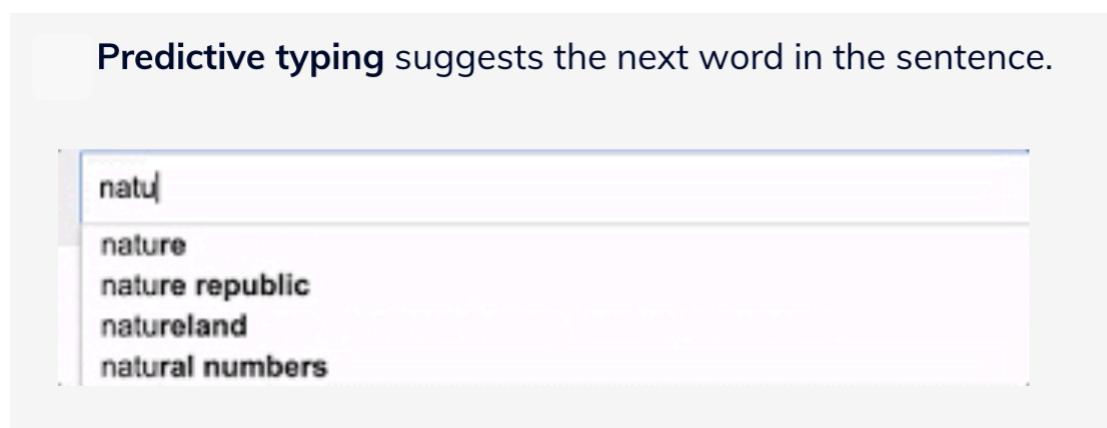
Chatbot



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



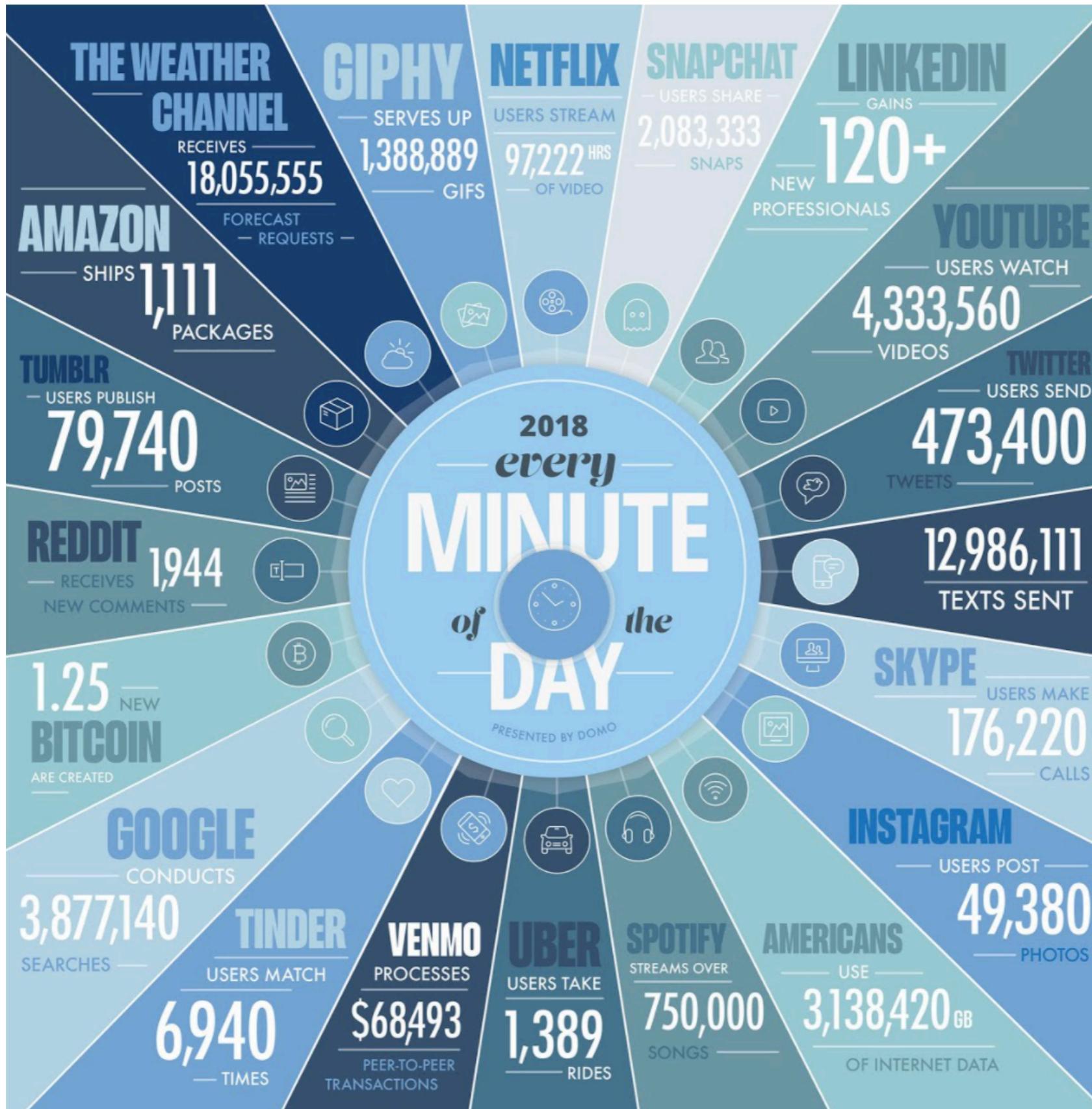
3.5B searches/daily

Extract and Summarization

NLP in business operation

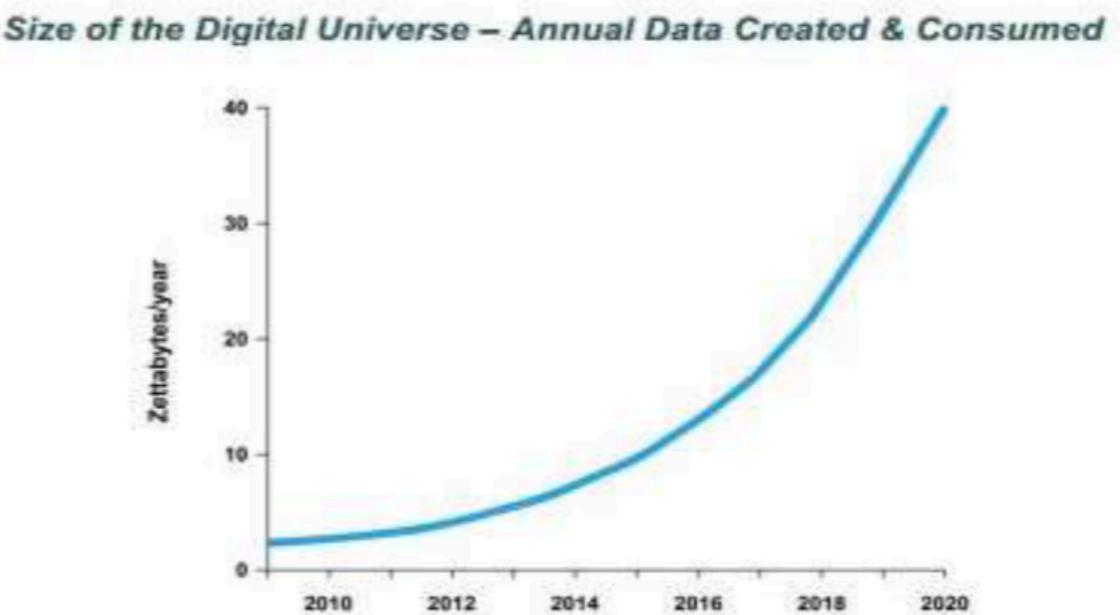
- Achieve efficiencies using machine processing of unstructured text data

Data production rate



Text data

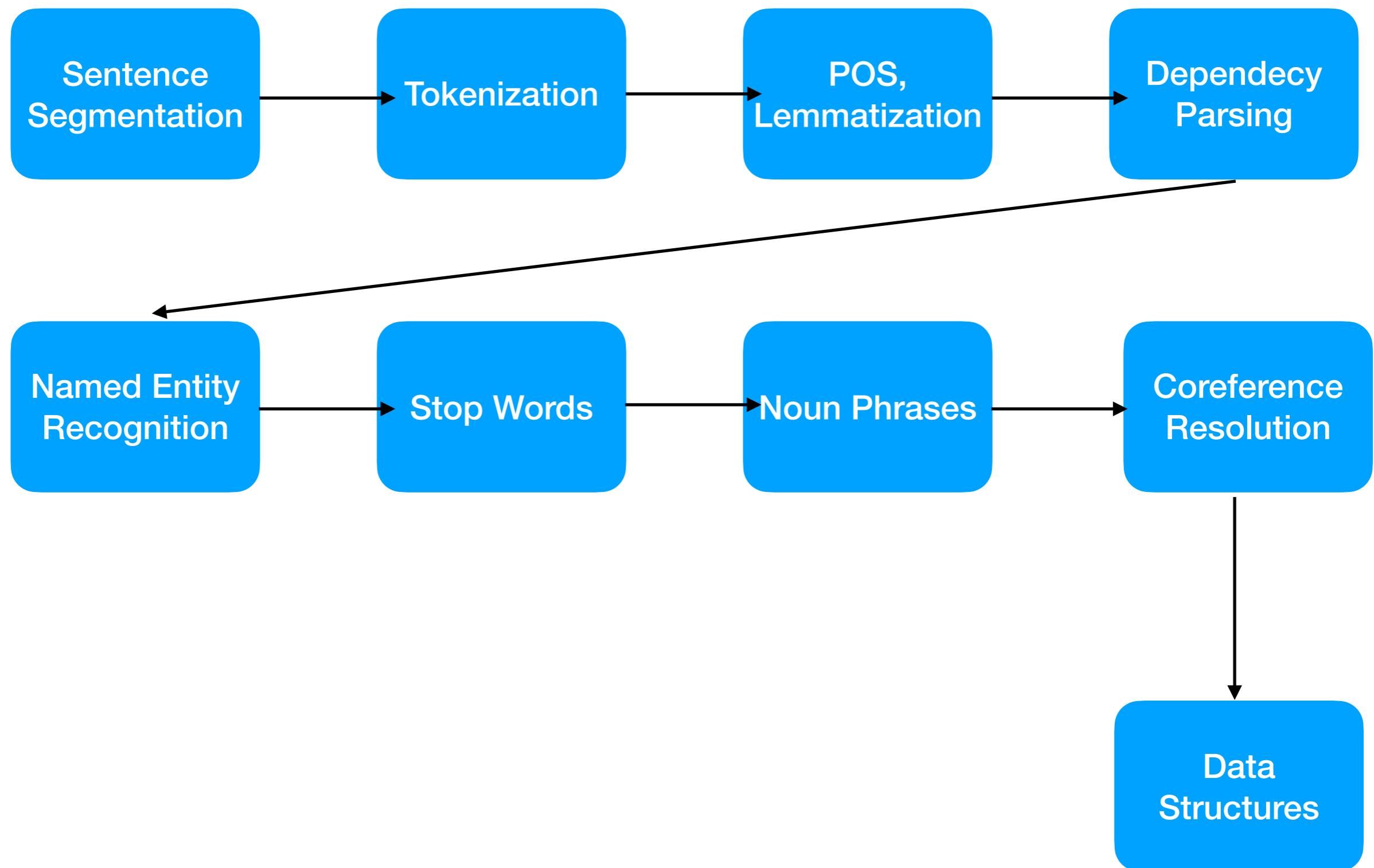
- About 80% of data in organizations are in text format
- Harder to analyse than structured data
- Huge amount of textual documents
 - Only in biomedicine 2200 scientific papers are published every day
- Growing exponentially



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

NLP Pipeline to produce representative data structures



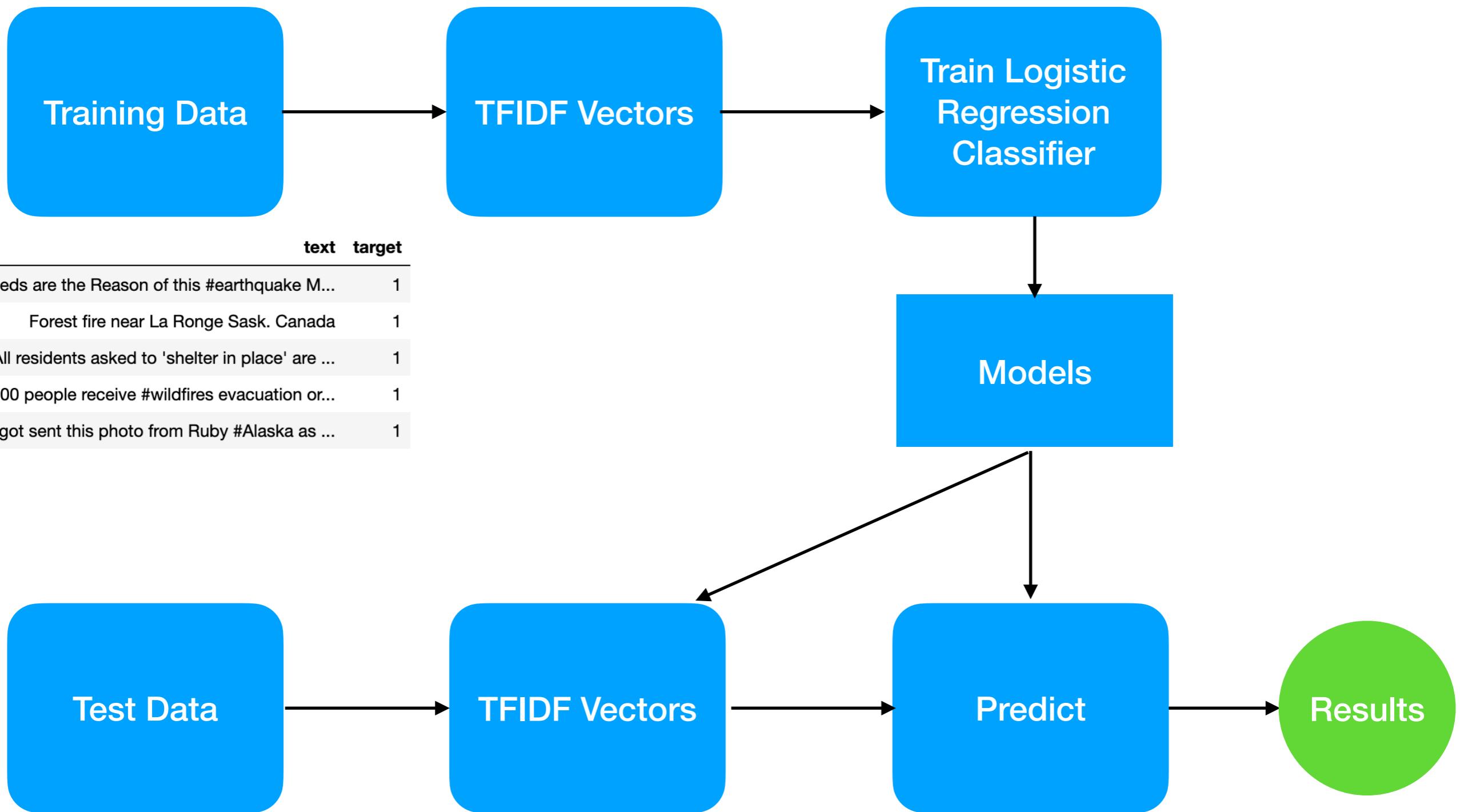
NLP Data Scientist - (some) tools



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

Lab: Disaster Detection



Lab: Disaster Detection using TFIDF and Logistic Regression

Click this link to begin

[https://colab.research.google.com/github/ravi-ilango/acm-dec-2020-nlp/blob/main/lab2/
disaster detection tfidf.ipynb](https://colab.research.google.com/github/ravi-ilango/acm-dec-2020-nlp/blob/main/lab2/disaster%20detection%20tfidf.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:

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

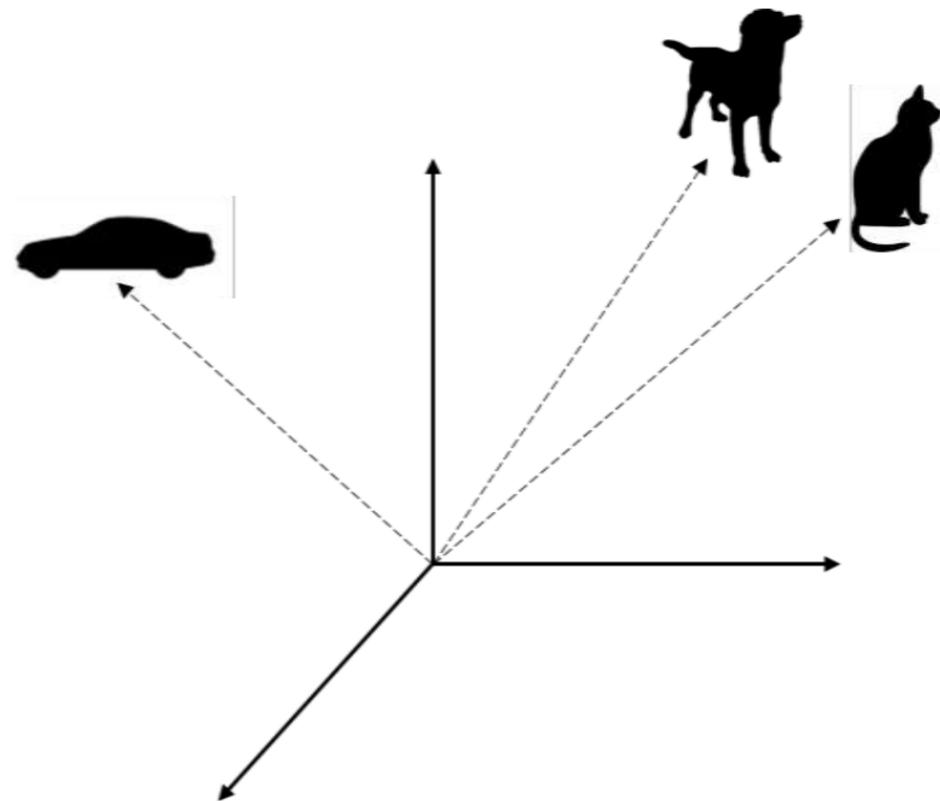
Vector Space Model

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.

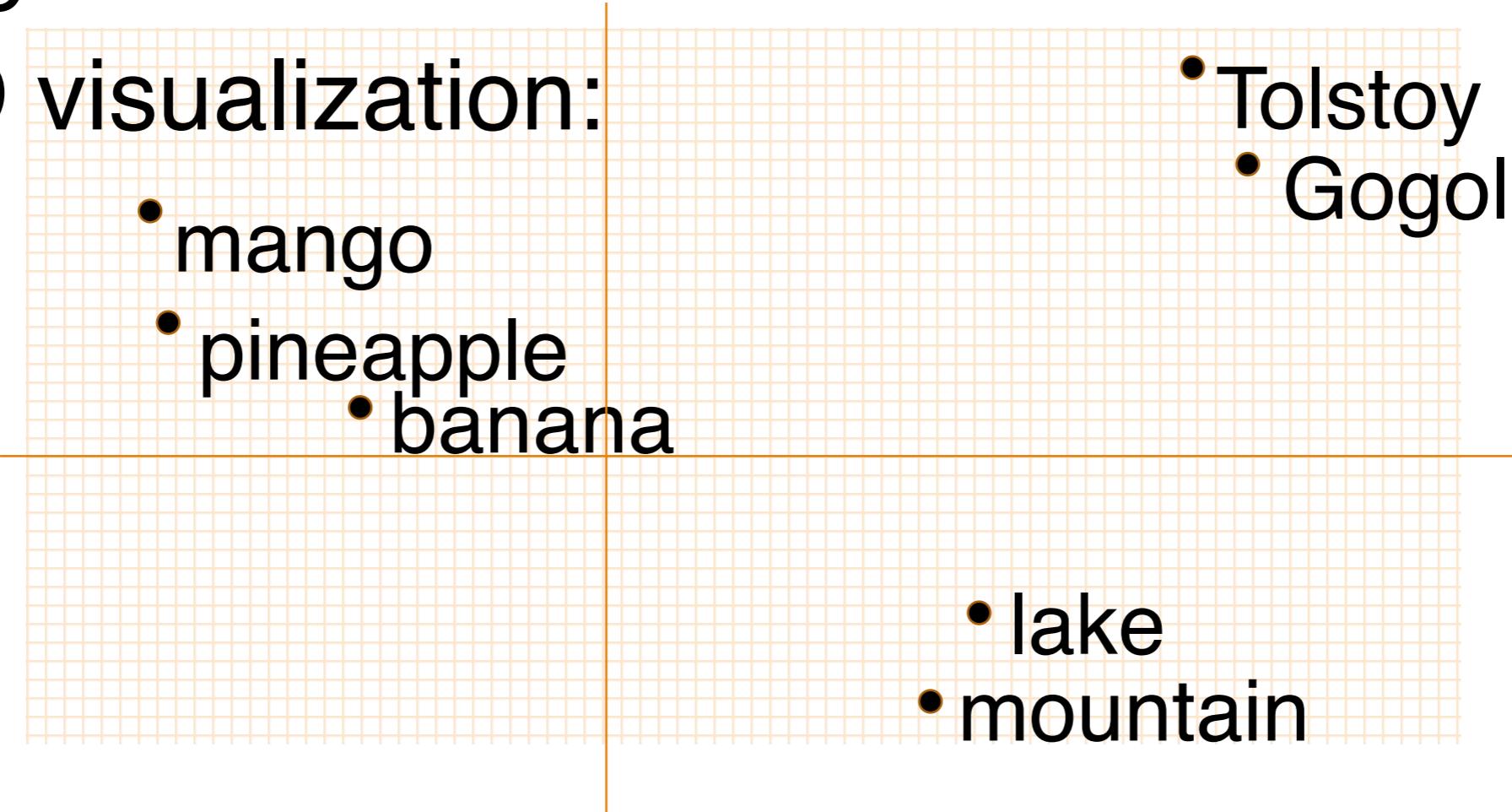
These dense vectors are called embeddings.



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...

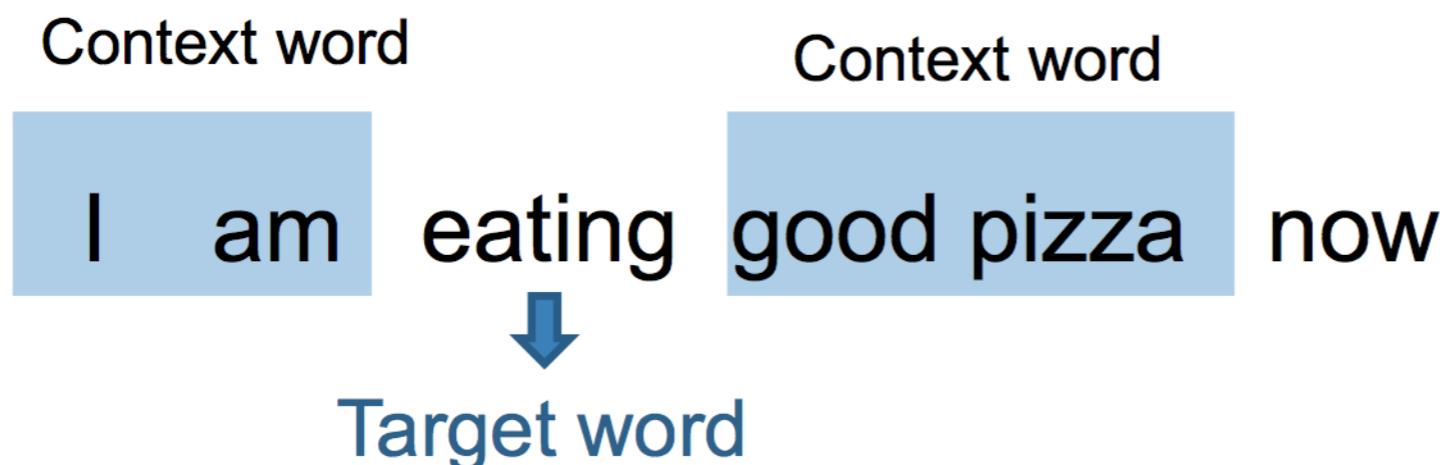
Move banana closer to mango

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



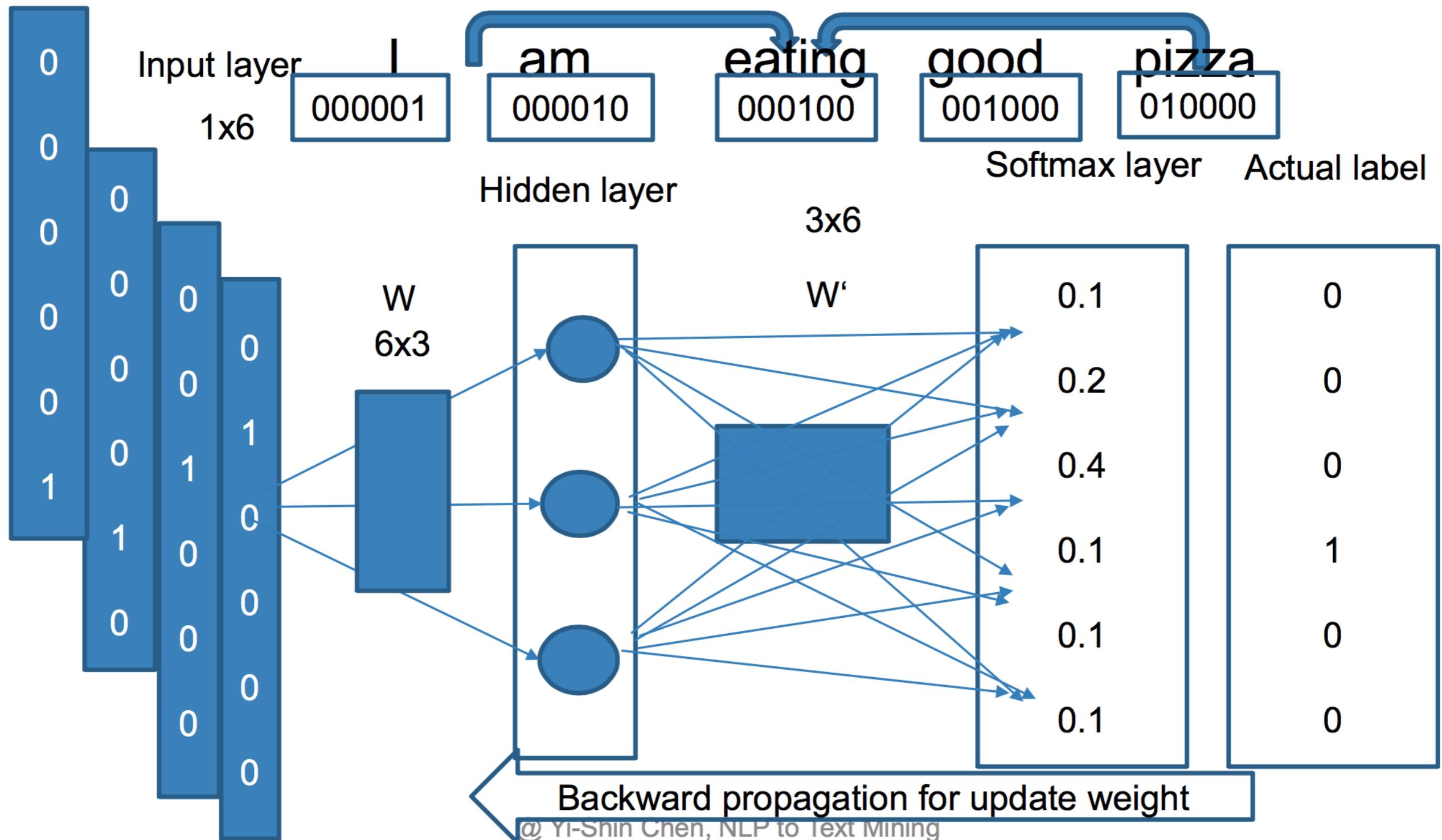
Ex: ([features], label)

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

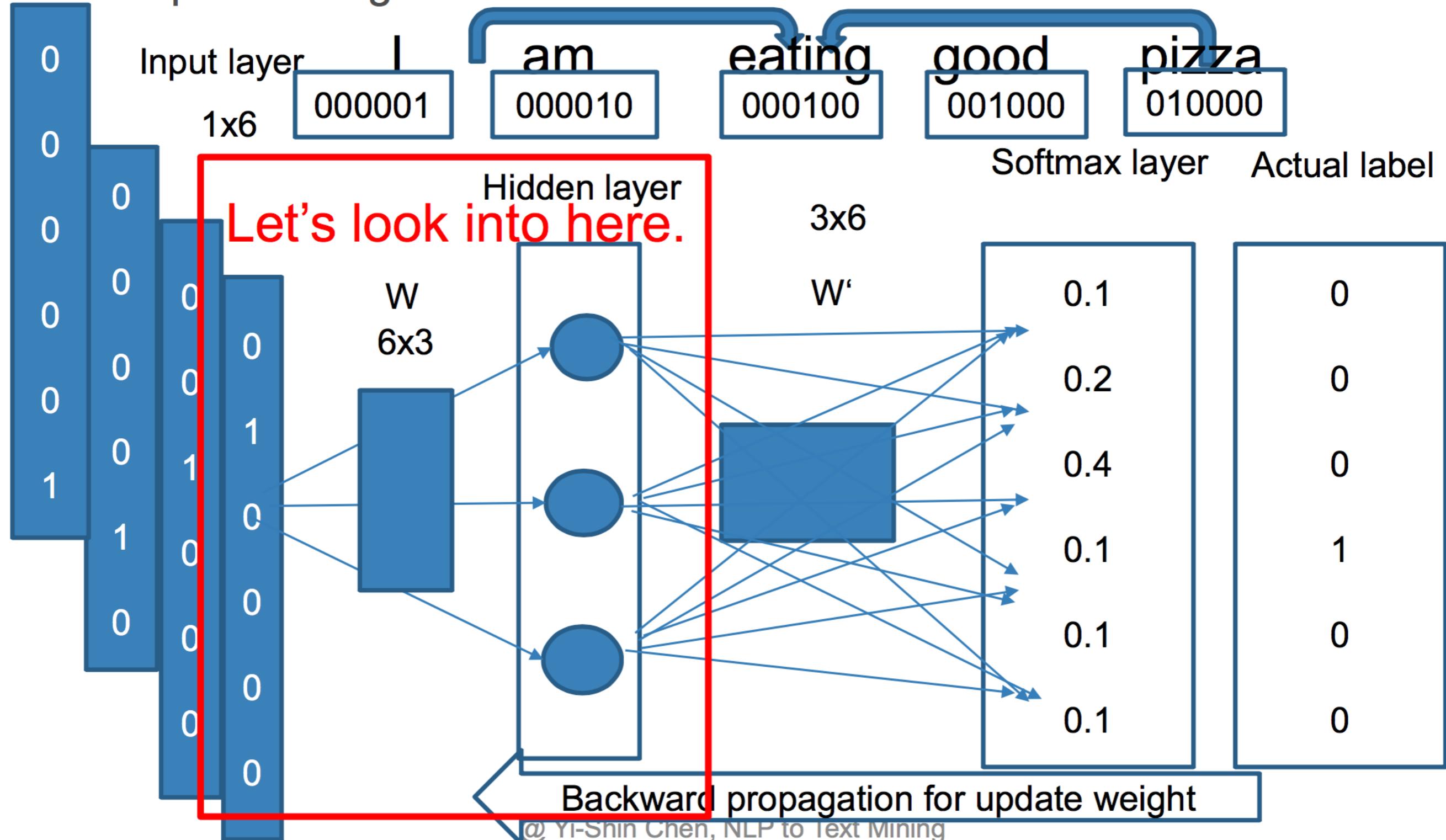
Word Embedding in a Deep Learning network

Continuous Bag-of-Words Model (Contd.)

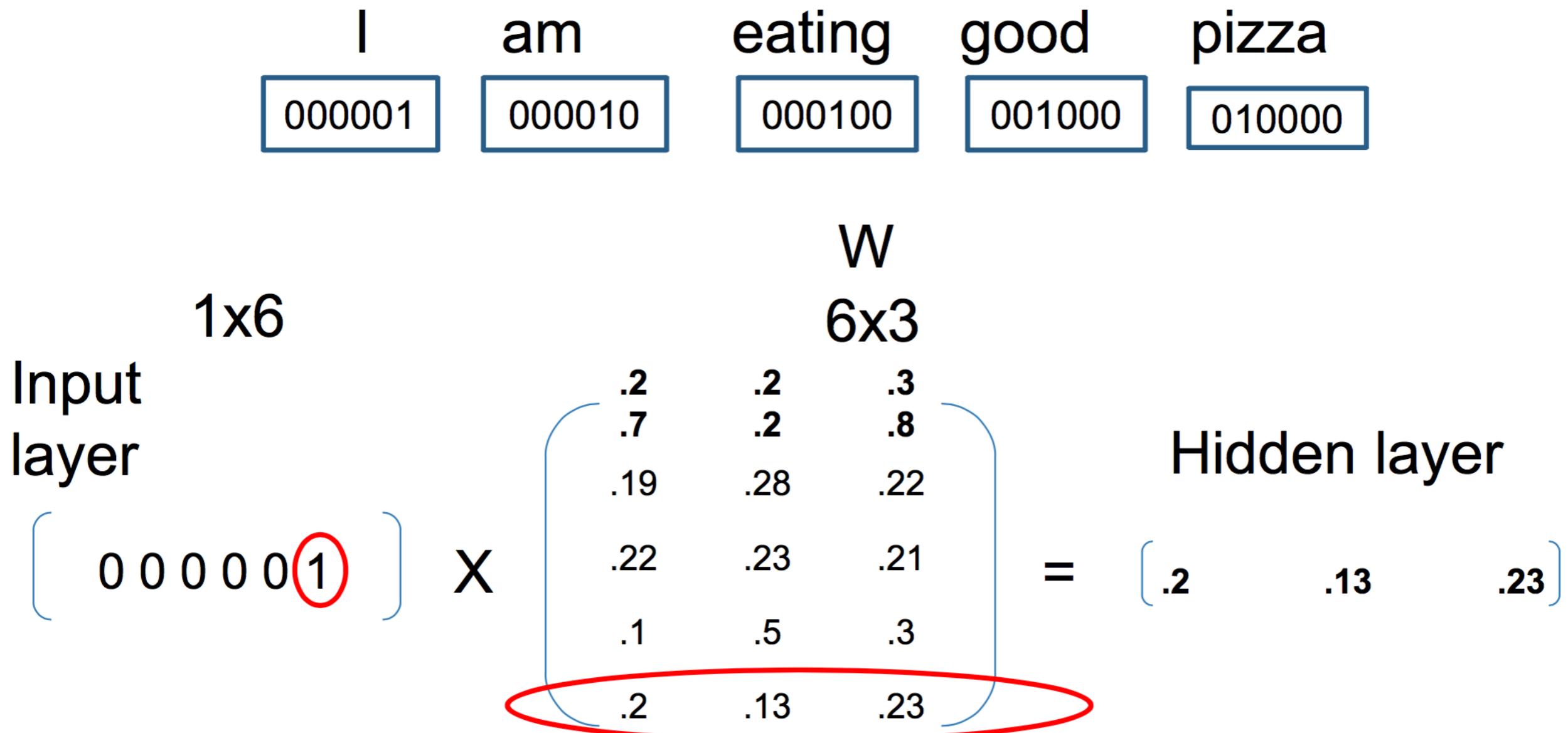
- We train one depth neural network, set hidden layer's width 3.



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

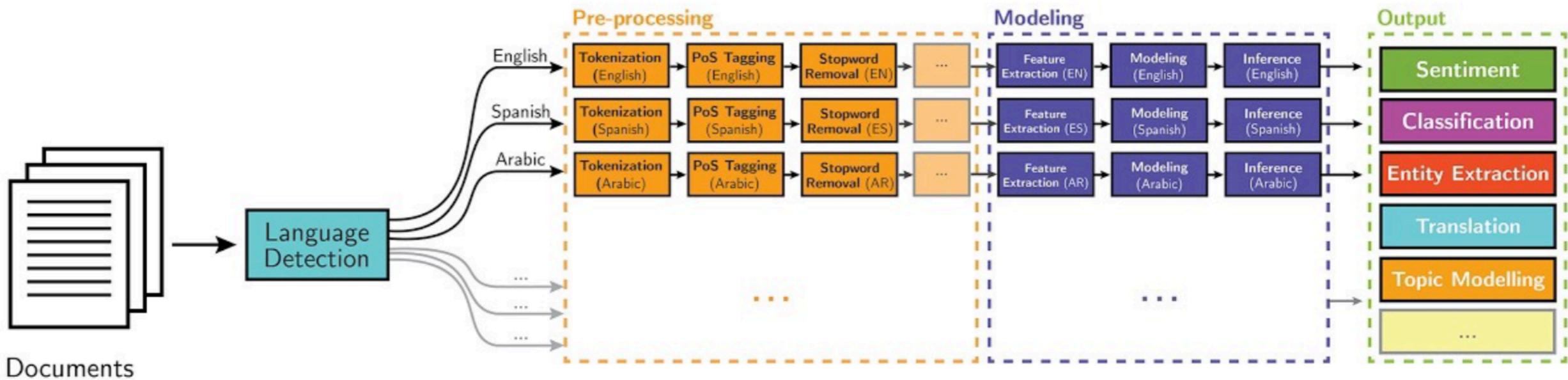


Continuous Bag-of-Words Model (Contd.)

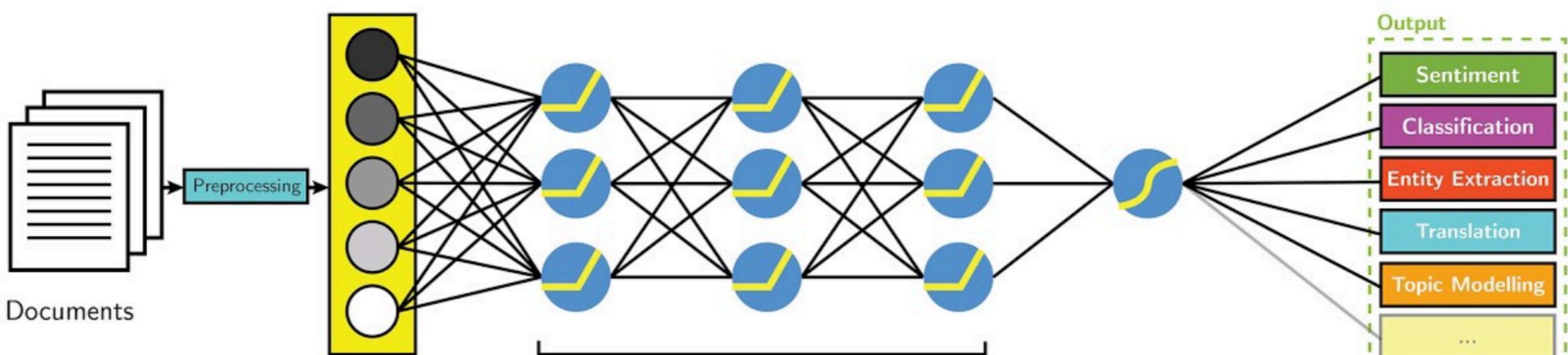


- ▶ The output of the hidden layer is just the “word vector” for the input word

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

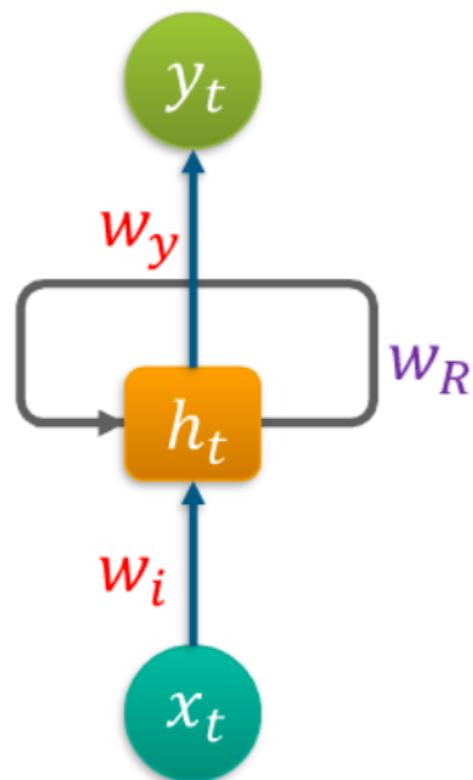
Stock prices

- **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

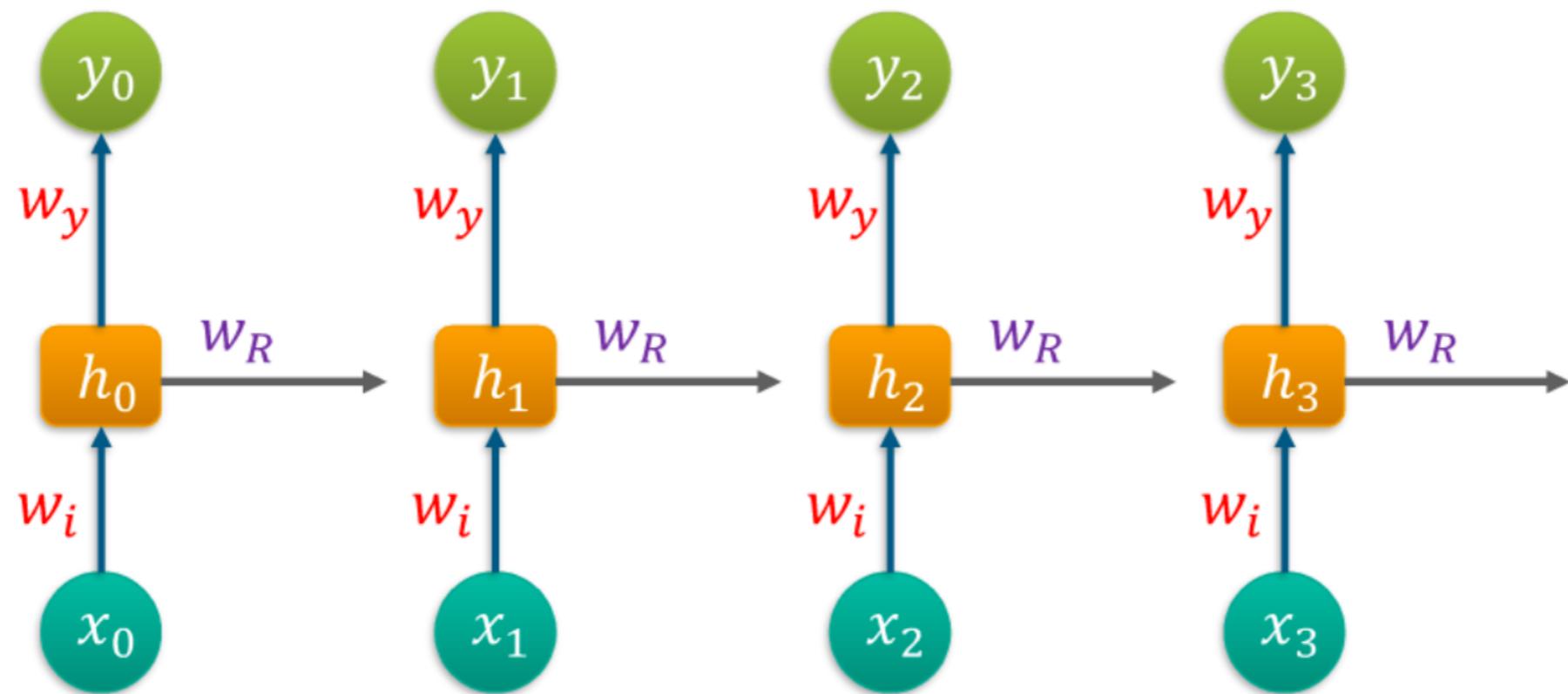
$$h^{(t)} = g_h (w_i x^{(t)} + w_R h^{(t-1)} + b_h)$$
$$y^{(t)} = g_y (w_y h^{(t)} + b_y)$$

Activity at previous
time stamp

Labelling of parameters used

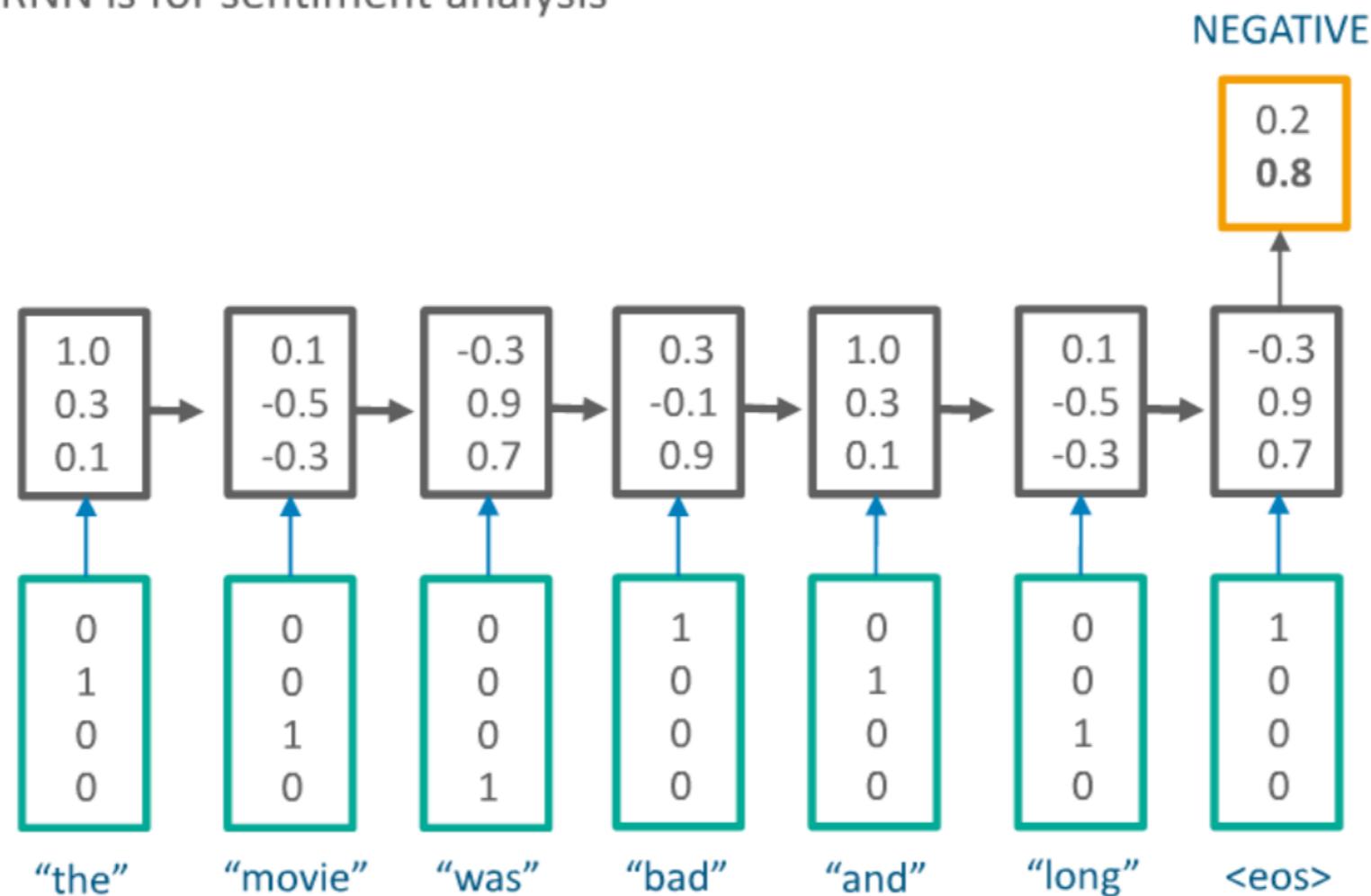
Structure of RNN

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



Example

- Another use of RNN is for sentiment analysis



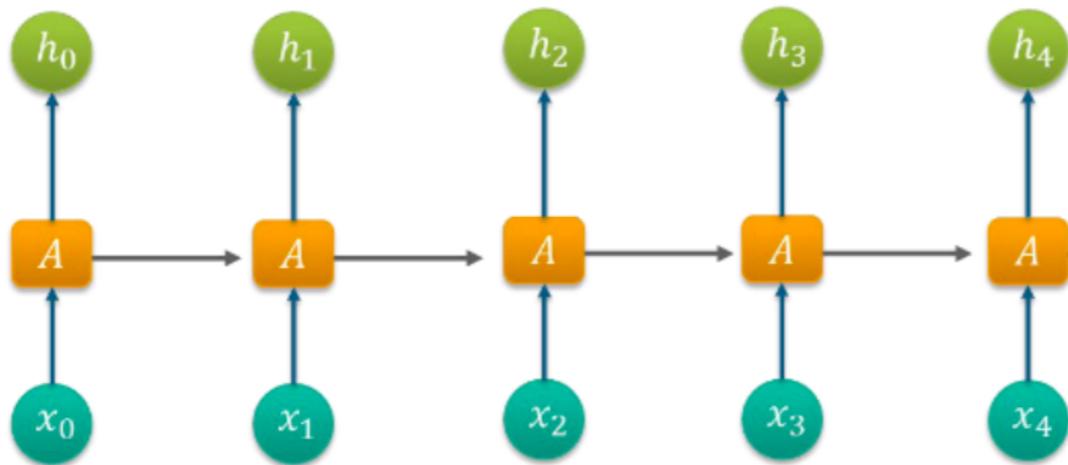
What are LSTMs?

- 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.

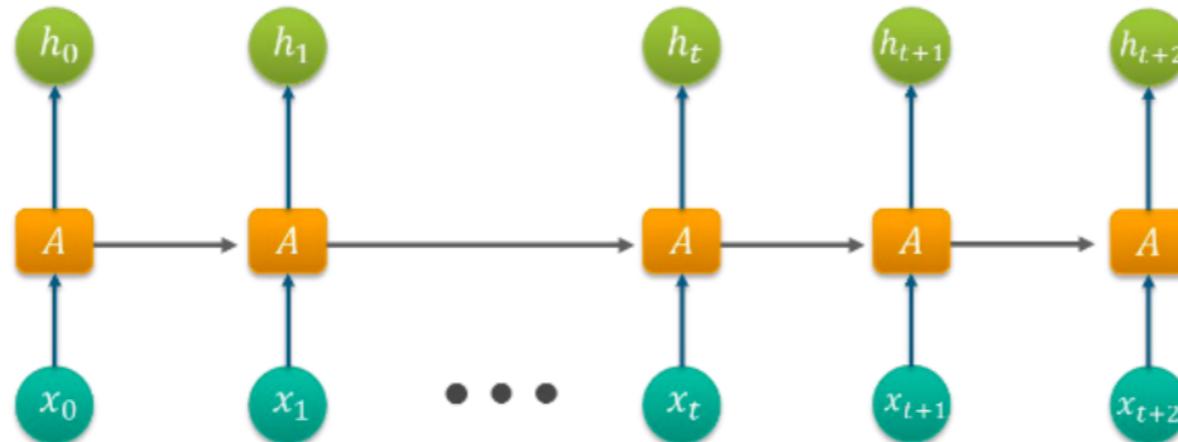
What are LSTMs?

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

- **Short Term Memory**

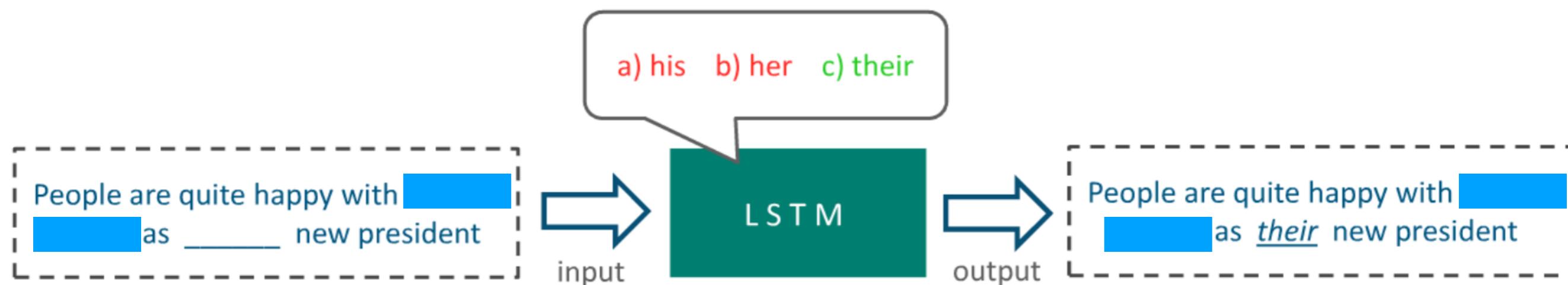


- **Long Term Memory**



LSTM Working Steps

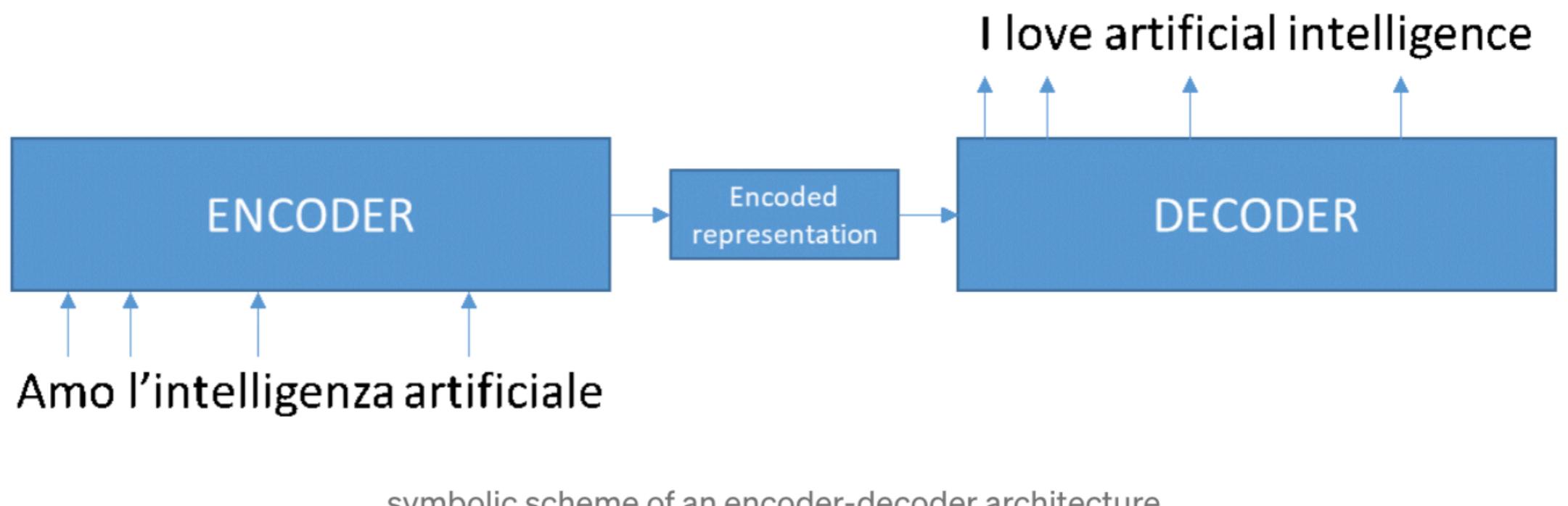
- For the language model example, since it just saw a subject, it might want to output information relevant to a verb, in case that's what is coming next.
- For example, it might output whether the subject is singular or plural, so that we know what form a verb should be conjugated into if that's what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



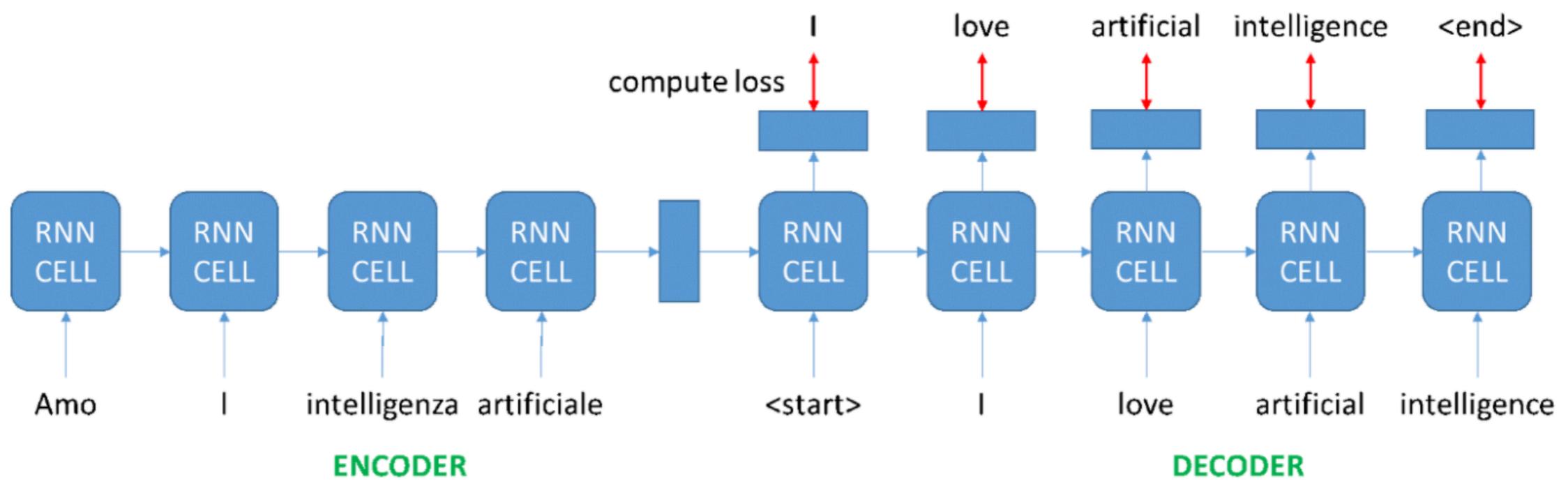
Sequence to Sequence Model using RNN

- 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)

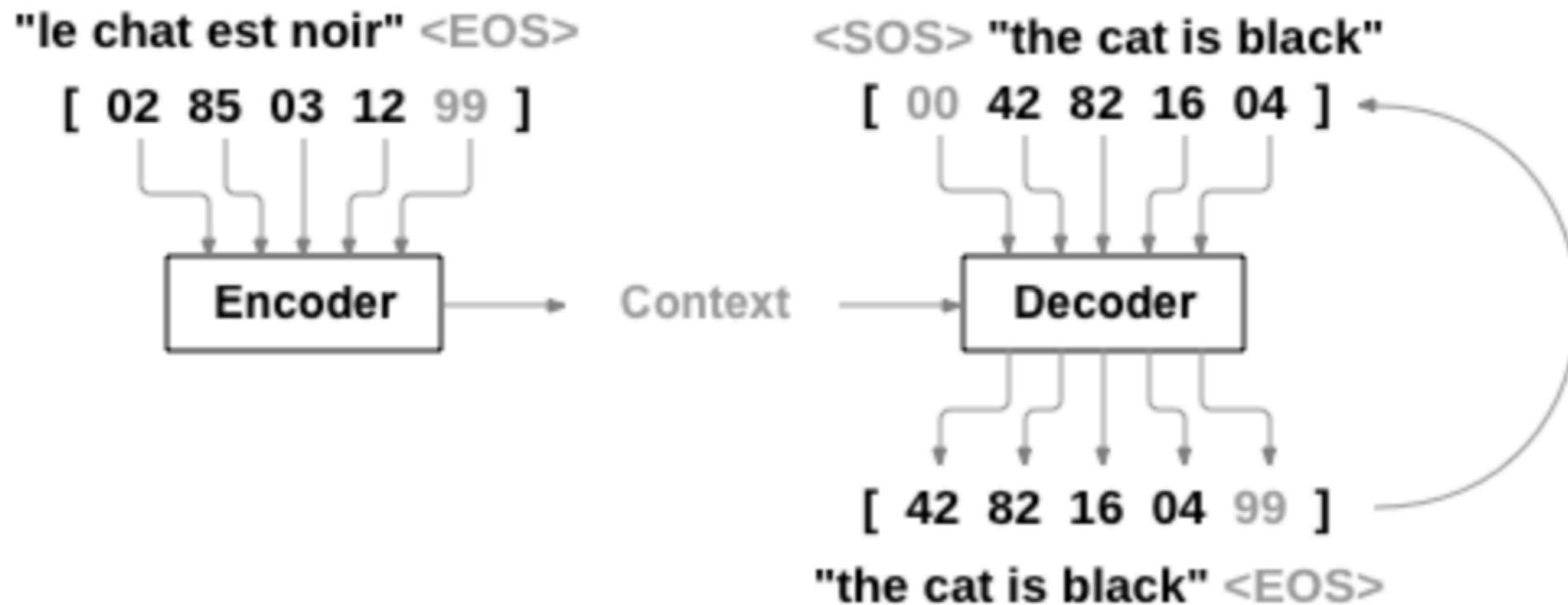
Encoder Decoder Network



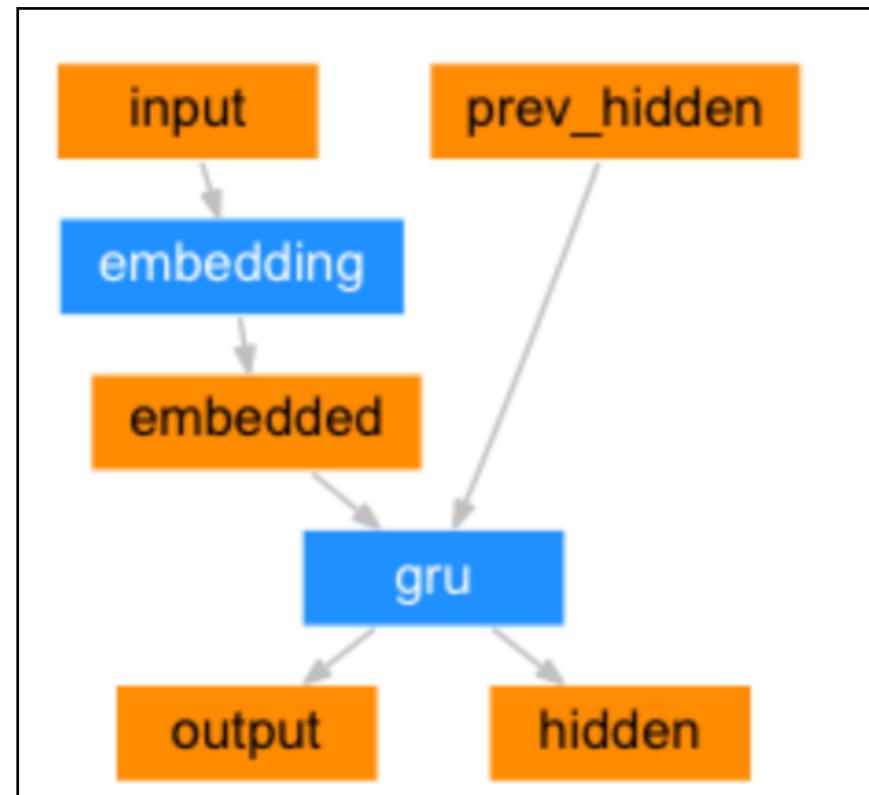
Encoder Decoder Network



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

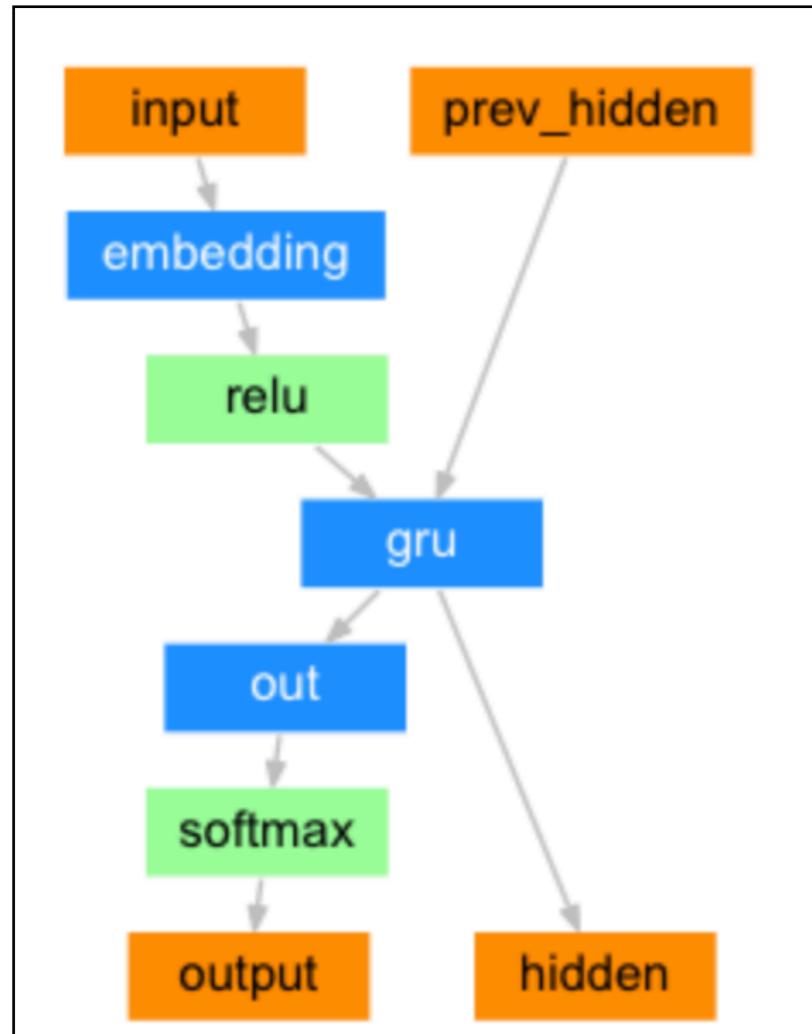
        self.embedding =
            nn.Embedding(input_size, hidden_size)
        self.gru = nn.GRU(hidden_size,
                         hidden_size)

    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)
```

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
```

[s://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html](https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html)

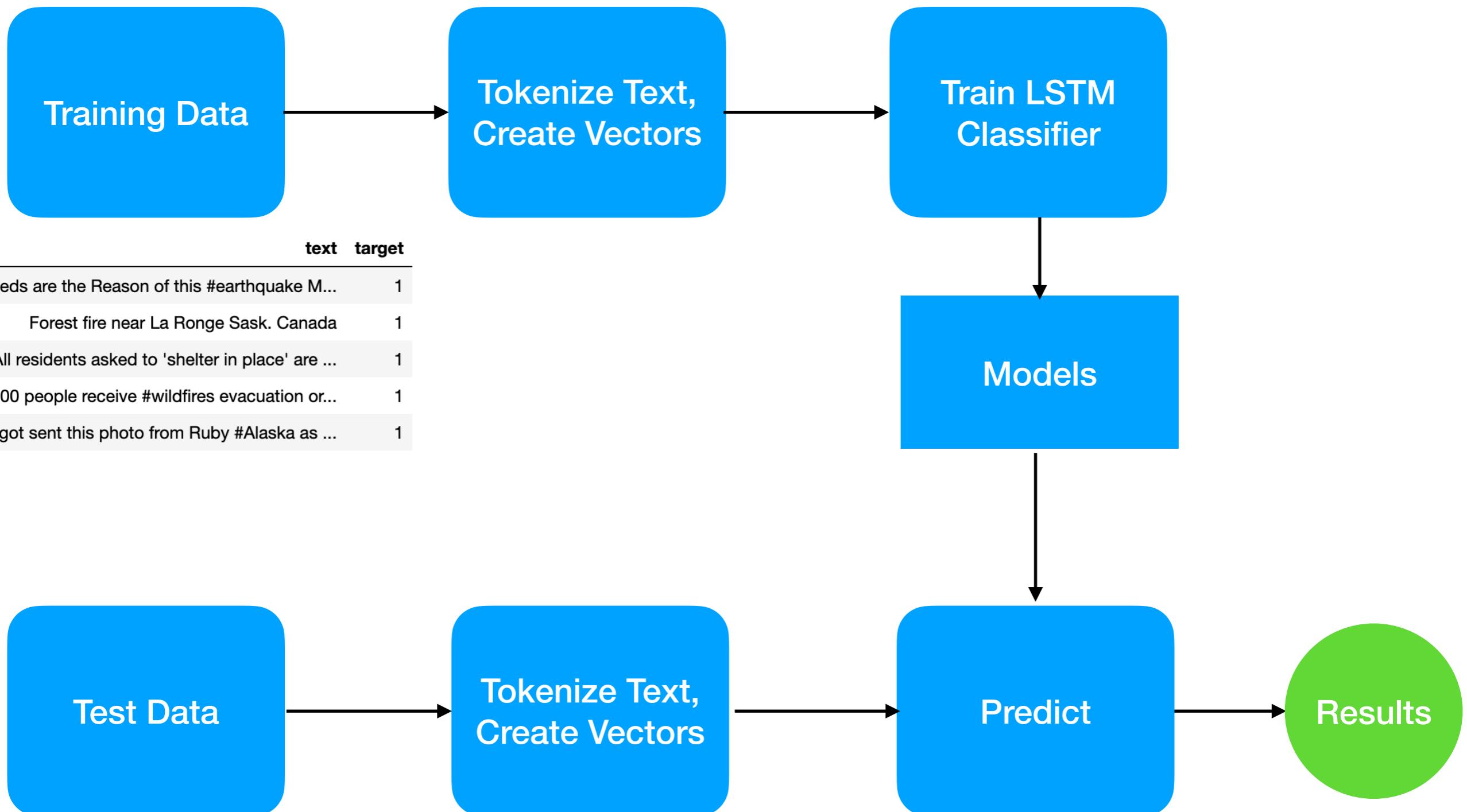
```
def initHidden(self):  
    return torch.zeros(1, 1, self.hidden_size,  
                      device=device)
```

Lab: Sequence to Sequence Model using RNN

Click this link to begin

[https://colab.research.google.com/github/ravi-ilango/acm-dec-2020-nlp/blob/main/lab1/
seq2seq_translation_tutorial.ipynb](https://colab.research.google.com/github/ravi-ilango/acm-dec-2020-nlp/blob/main/lab1/seq2seq_translation_tutorial.ipynb)

Lab: Quora Comment Classifier



Lab: Quora Insincere Questions Classification using LSTM

Click this link to begin

[https://colab.research.google.com/github/ravi-ilango/acm-dec-2020-nlp/blob/main/lab2_1/
quora_classifier_lstm.ipynb](https://colab.research.google.com/github/ravi-ilango/acm-dec-2020-nlp/blob/main/lab2_1/quora_classifier_lstm.ipynb)

Lab: Humor detection using LSTM

Click this link to begin

[https://colab.research.google.com/github/ravi-ilango/acm-dec-2020-nlp/blob/main/lab2_1/
quora_classifier_lstm.ipynb](https://colab.research.google.com/github/ravi-ilango/acm-dec-2020-nlp/blob/main/lab2_1/quora_classifier_lstm.ipynb)

Transformers

- **Transformer:** It applies attention mechanisms to gather information about the relevant context of a given word, and then encode that context in a rich vector that smartly represents the word.

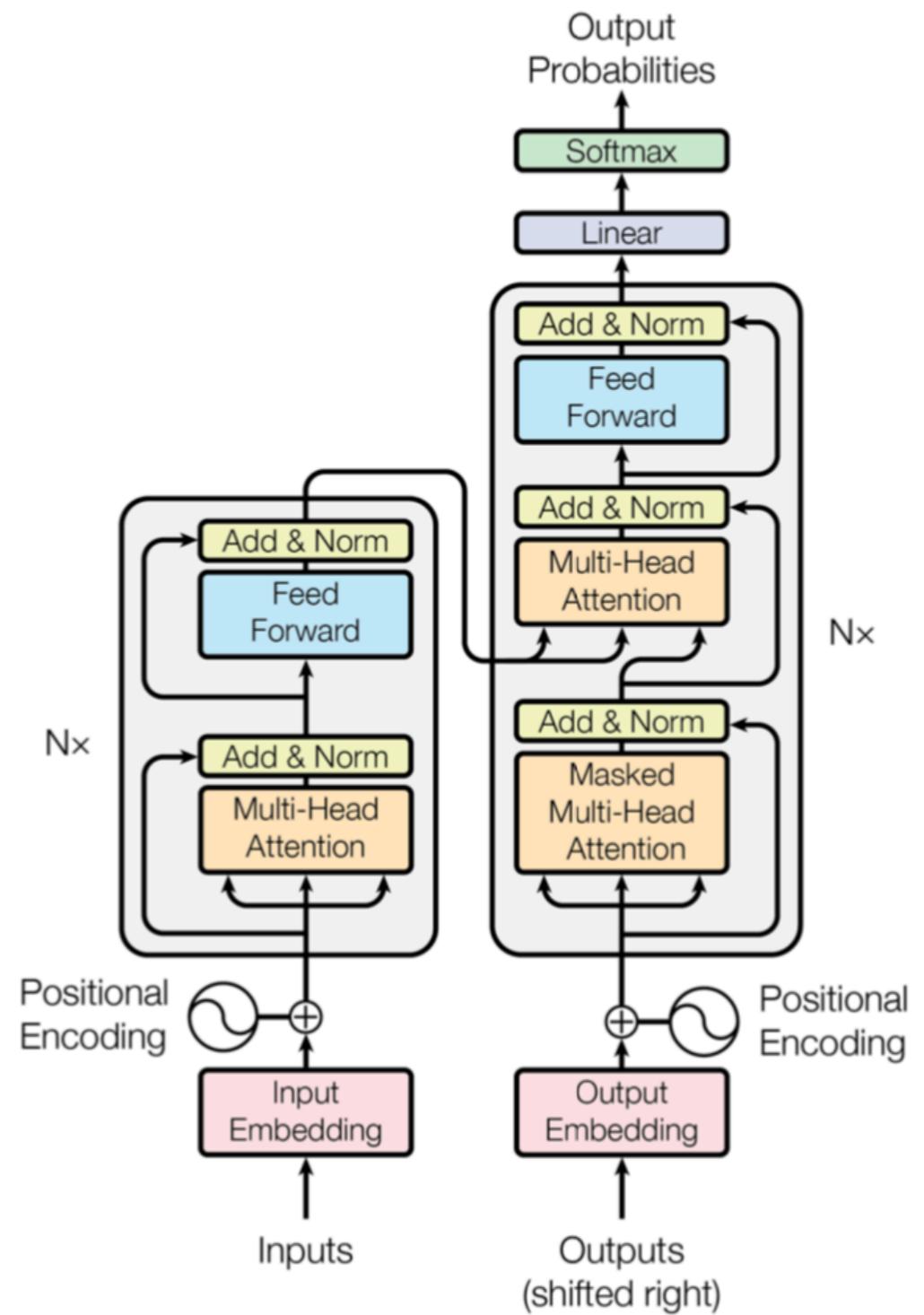


Figure 1: The Transformer - model architecture.

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.

<https://medium.com/analytics-vidhya/transformer-vs-rnn-and-cnn-18eeefa3602b>

<https://arxiv.org/abs/1705.03122>

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

Source: <http://cs231n.github.io/transfer-learning/>

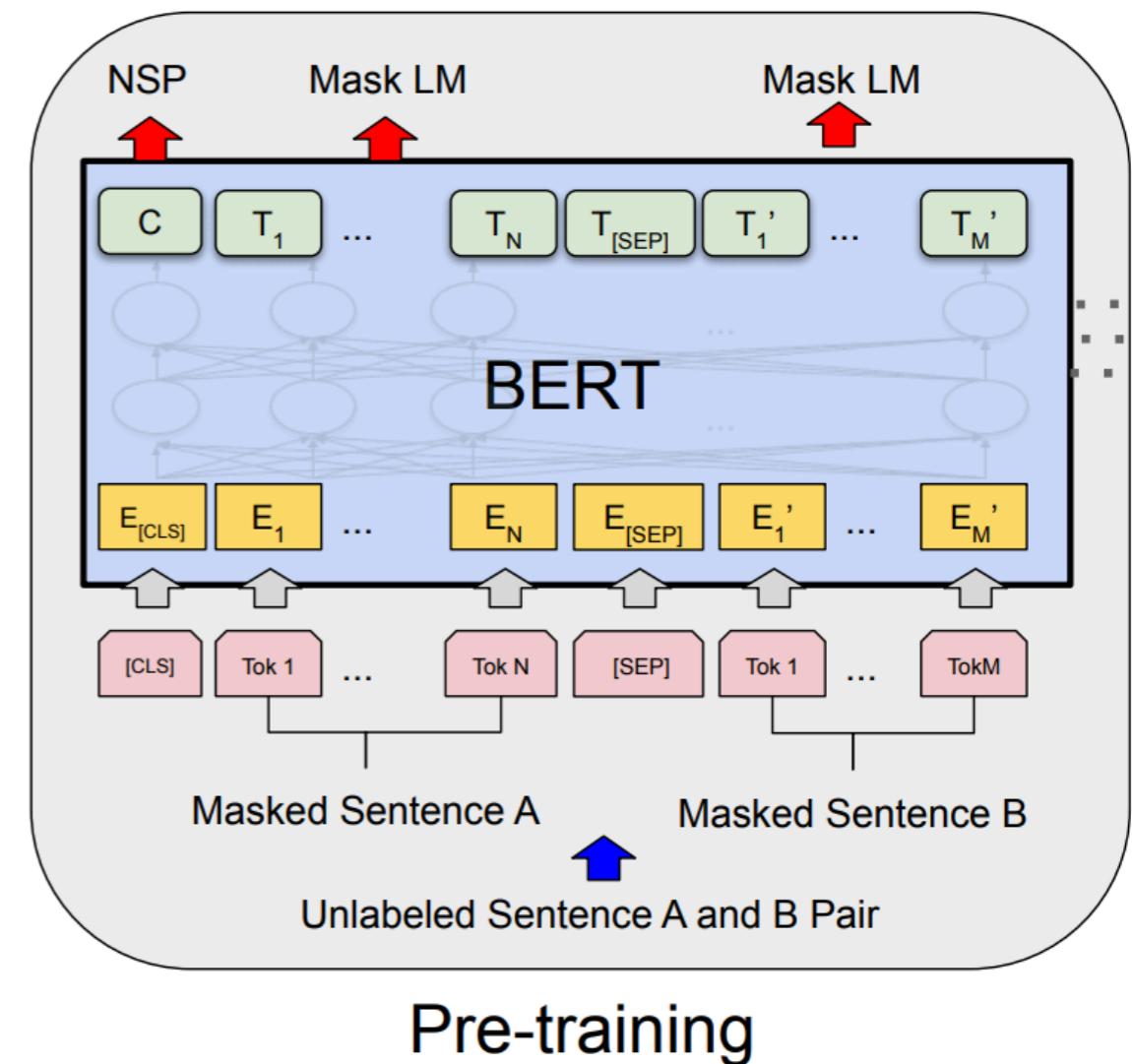
BERT - Bidirectional Encoder Representations from Transformers

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

BERT - Bidirectional Encoder Representations from Transformers

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



HuggingFace

MODELS

- ALBERT
- AutoClasses
- BART
- BERT
- BertGeneration
- Blenderbot
- CamemBERT
- CTRL
- DeBERTa
- DialoGPT
- DistilBERT
- DPR
- ELECTRA
- Encoder Decoder Models
- FlauBERT

SIGN IN MODELS FORUM

Docs » Quick tour CO Open in Colab View page source

Quick tour

Let's have a quick look at the 😊 Transformers library features. The library downloads pretrained models for Natural Language Understanding (NLU) tasks, such as analyzing the sentiment of a text, and Natural Language Generation (NLG), such as completing a prompt with new text or translating in another language.

First we will see how to easily leverage the pipeline API to quickly use those pretrained models at inference. Then, we will dig a little bit more and see how the library gives you access to those models and helps you preprocess your data.

Lab: Disaster Detection using BERT

[https://colab.research.google.com/github/ravi-ilango/acm-dec-2020-nlp/blob/main/lab3/
disaster detection bert.ipynb](https://colab.research.google.com/github/ravi-ilango/acm-dec-2020-nlp/blob/main/lab3/disaster%20detection%20bert.ipynb)

Lab: Sarcasm Detection using BERT

[https://colab.research.google.com/github/ravi-ilango/acm-dec-2020-nlp/blob/main/lab4/
LAB_Sarcasm_Detector.ipynb](https://colab.research.google.com/github/ravi-ilango/acm-dec-2020-nlp/blob/main/lab4/LAB_Sarcasm_Detector.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

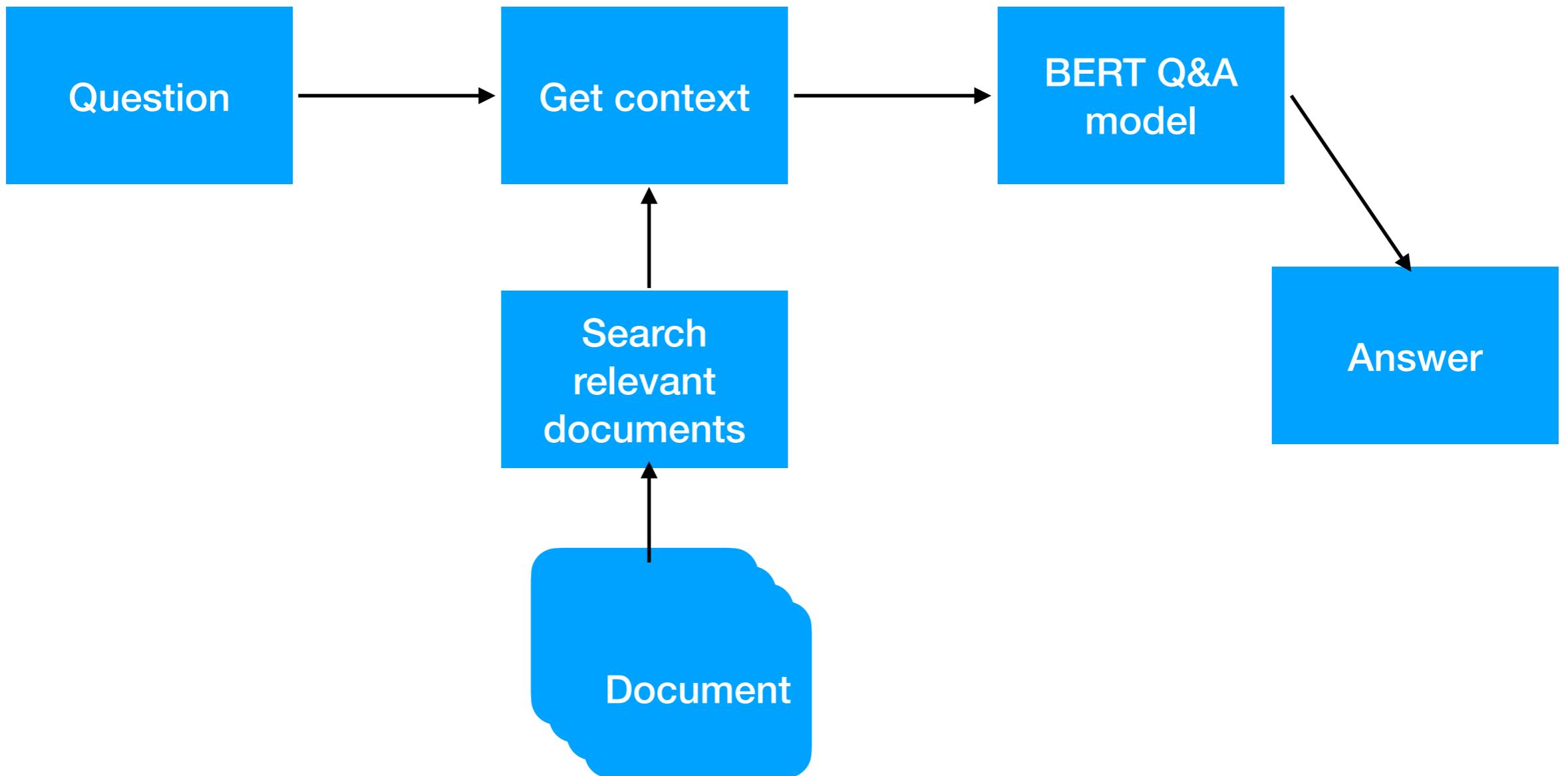
Lab: Text Summarization

[https://colab.research.google.com/github/ravi-ilango/acm-dec-2020-nlp/blob/main/lab6/
TextSummarization.ipynb](https://colab.research.google.com/github/ravi-ilango/acm-dec-2020-nlp/blob/main/lab6/TextSummarization.ipynb)

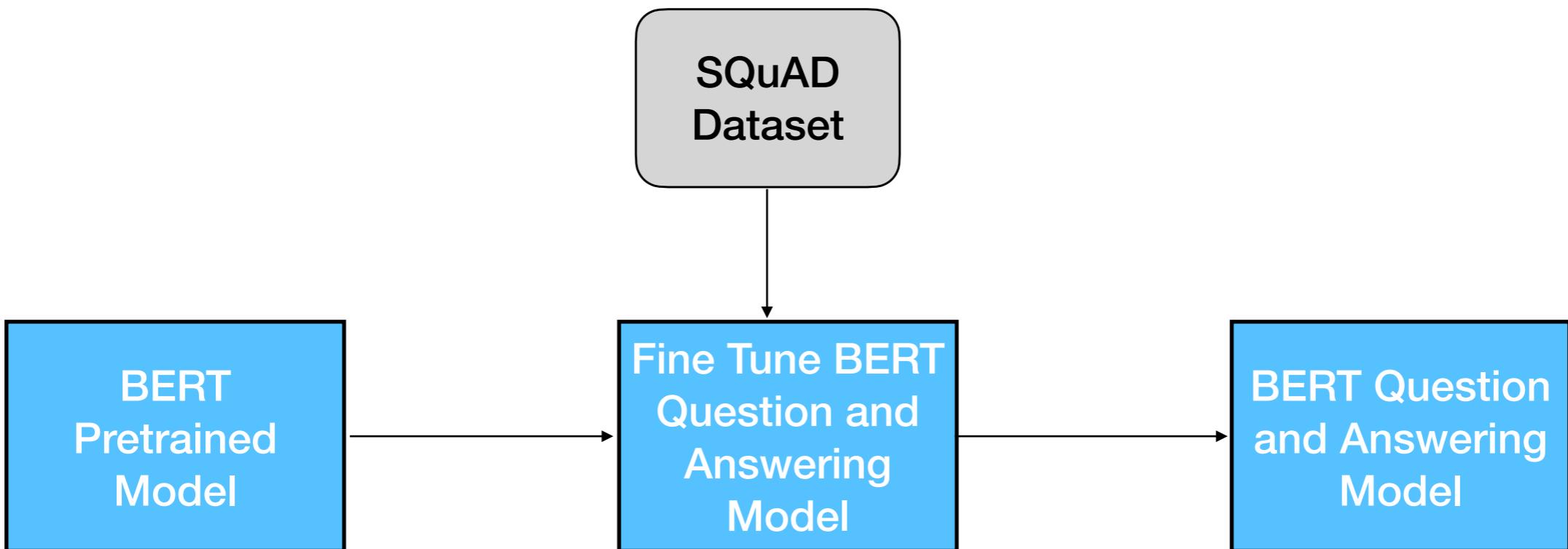
BERT Question and Answering Model

- Pretrained models which can answer questions based on context
- Trained using Stanford Question and Answer Dataset (SQuAD)
- Can be used to deploy a Question & Answering solution

BERT Question and Answering Model - Usecase



BERT Question and Answering Model Dev



SQuAD dataset

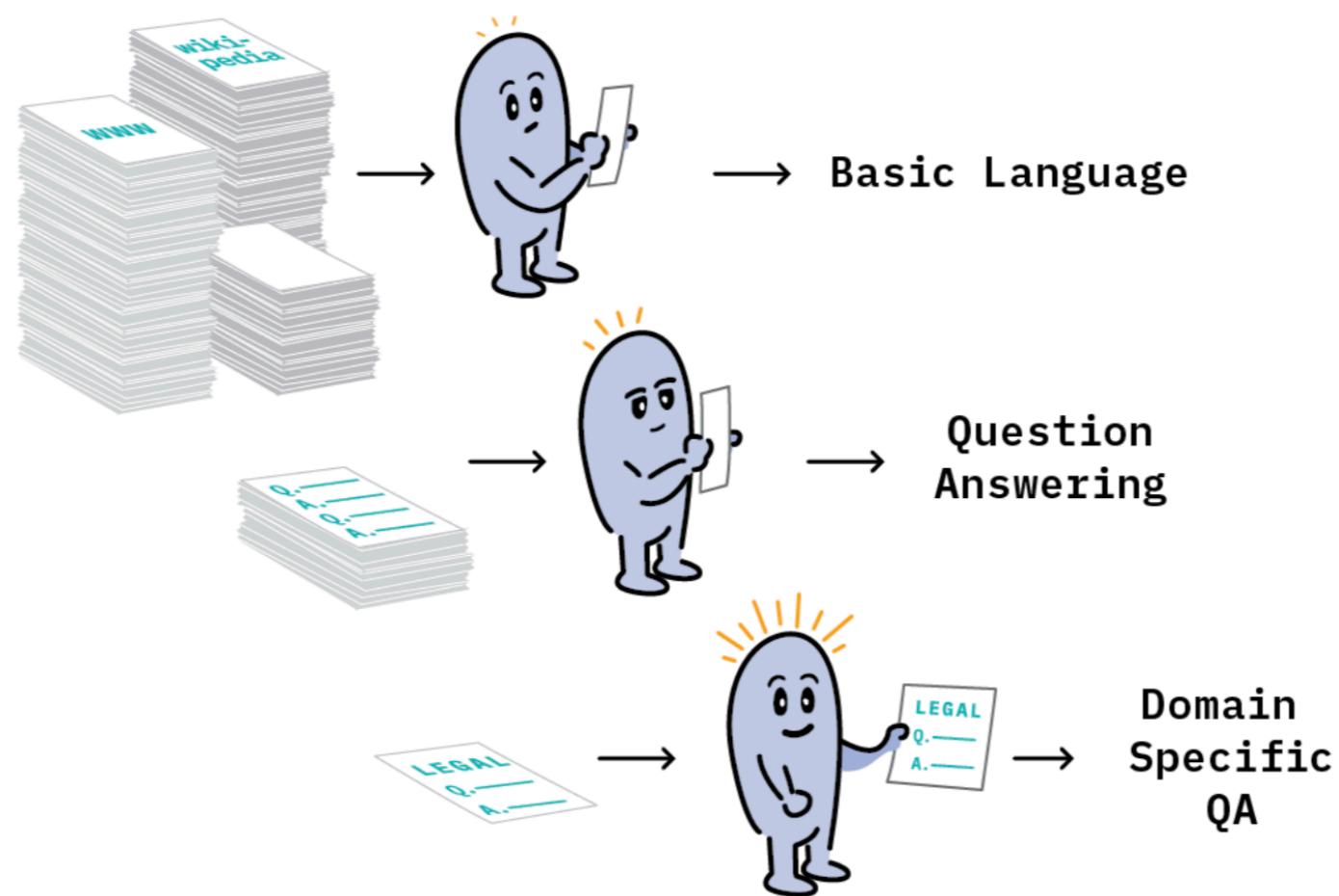
Stanford Question Answering Dataset (source: <https://rajpurkar.github.io/SQuAD-explorer/>)

SQuAD is a Span-based QA dataset.

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

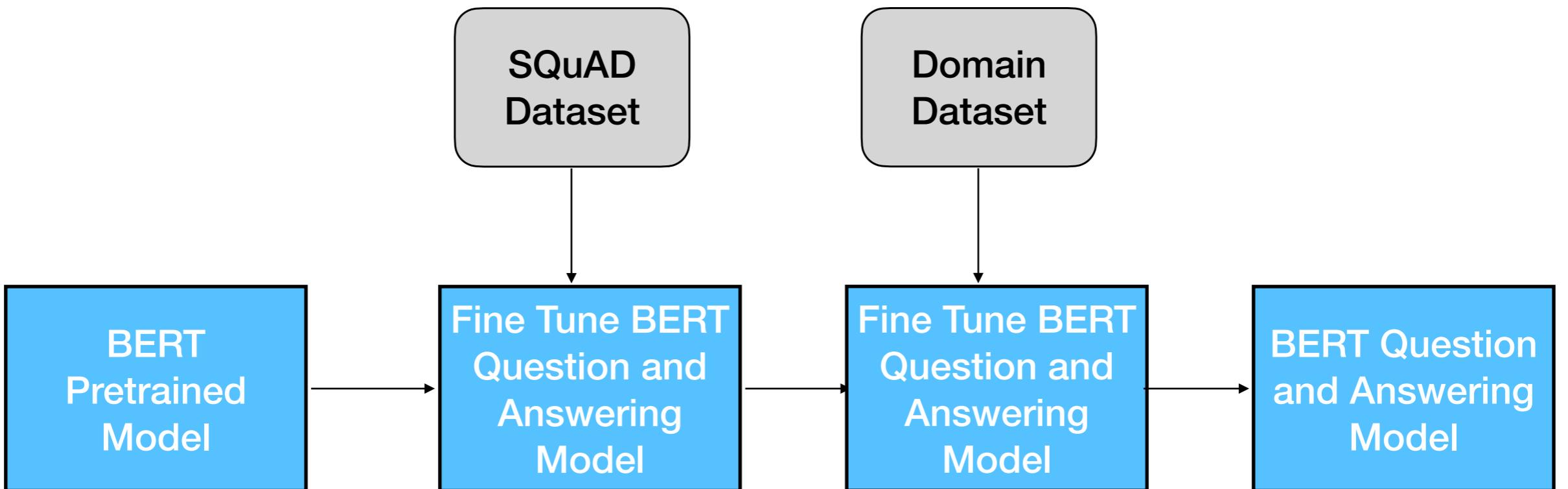
```
{'answers': {'answer_start': [245, 194, 194],
  'text': ['Bruno Mars', 'Coldplay', 'Coldplay']},
 'context': 'CBS broadcast Super Bowl 50 in the U.S., and charged an average of $5 million for a 30-second commercial during the game. The Super Bowl 50 halftime show was headlined by the British rock group Coldplay with special guest performers Beyoncé and Bruno Mars, who headlined the Super Bowl XLVII and Super Bowl XLVIII halftime shows, respectively. 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

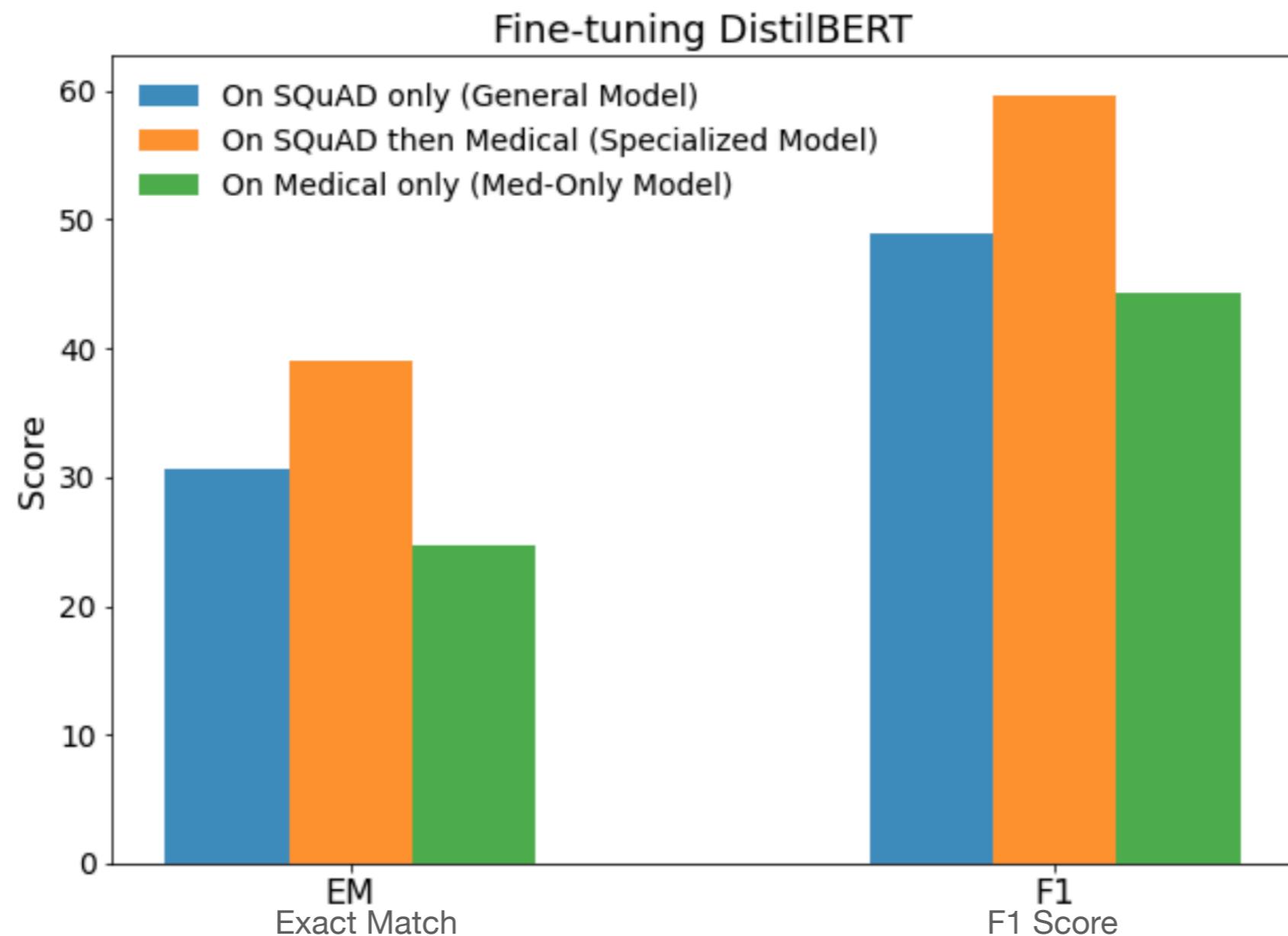


<https://qa.fastforwardlabs.com/>

BERT Question and Answering Model Dev



Model accuracies



Lab: Question and Answer model

Click this link to begin

https://colab.research.google.com/github/ravi-ilango/acm-dec-2020-nlp/blob/main/lab7/QandA_SQuAD.ipynb