Performing Binary Text Classification Using Words

Janani Ravi

CO-FOUNDER, LOONYCORN

www.loonycorn.com

Overview

Input processing as prelude to text classification

Representing text as tensors

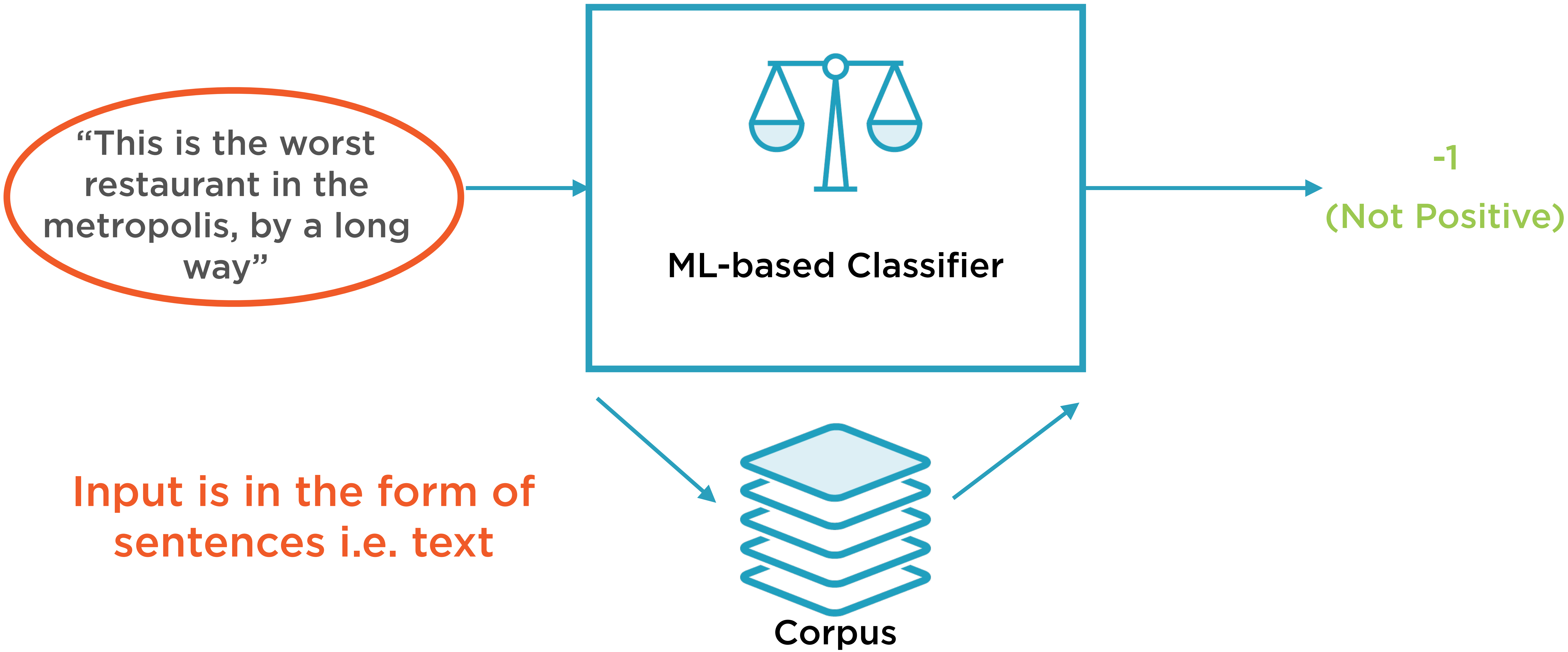
Generating word embeddings from a training corpus

Using basic RNN and LSTM cells

Binary text classification using RNNs

Word Embeddings

Text Classification Using Neural Networks



"This is the worst restaurant in the metropolis, by a long way"

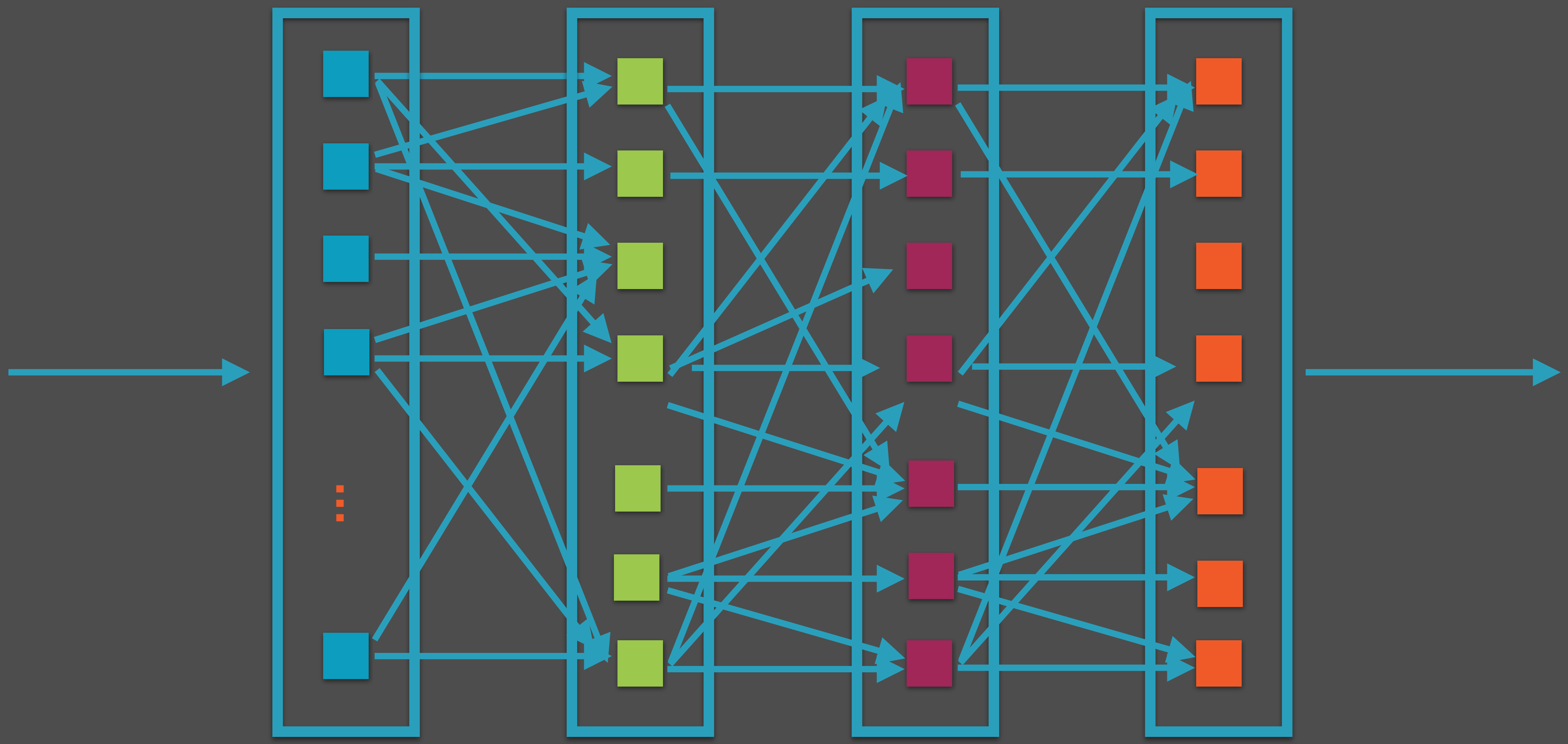
ML-based Classifier

-1

(Not Positive)

Corpus

Input is in the form of sentences i.e. text



Neural networks only process **numeric input**, they don't work with plain text

```
d = “This is not the worst restaurant in the metropolis,
not by a long way”
```

Document as Word Sequence

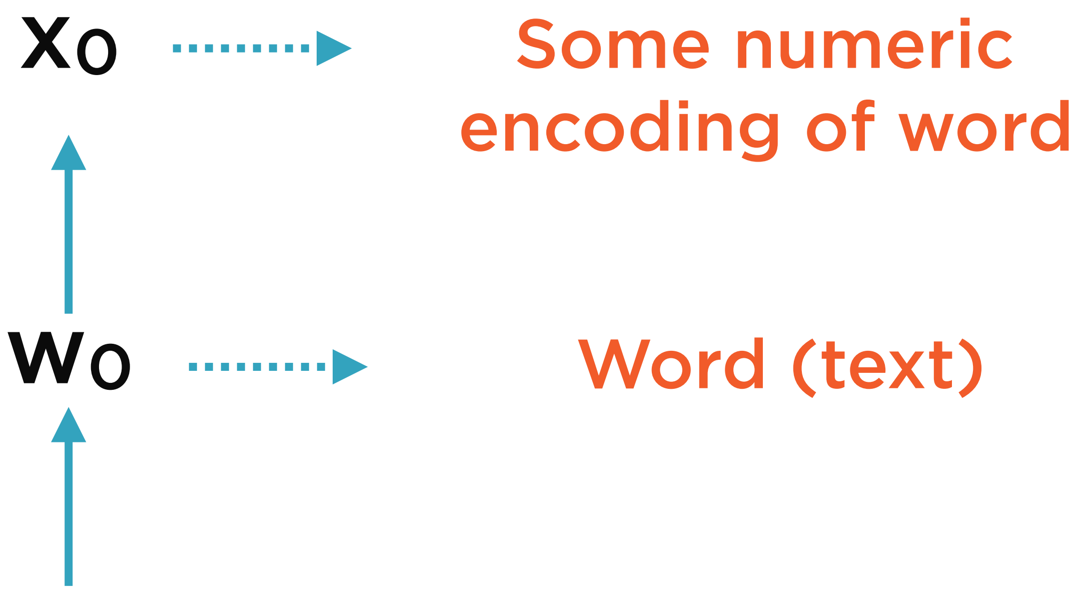
Model a document as an ordered sequence of words

```
d = “This is not the worst restaurant in the metropolis,
not by a long way”
```

```
(“This”, “is”,”not”,”the”,”worst”,”restaurant”,”in”,”the”,
“metropolis”, “not”,”by”,”a”,”long”,”way”)
```

Document as Word Sequence

Tokenise document into individual words

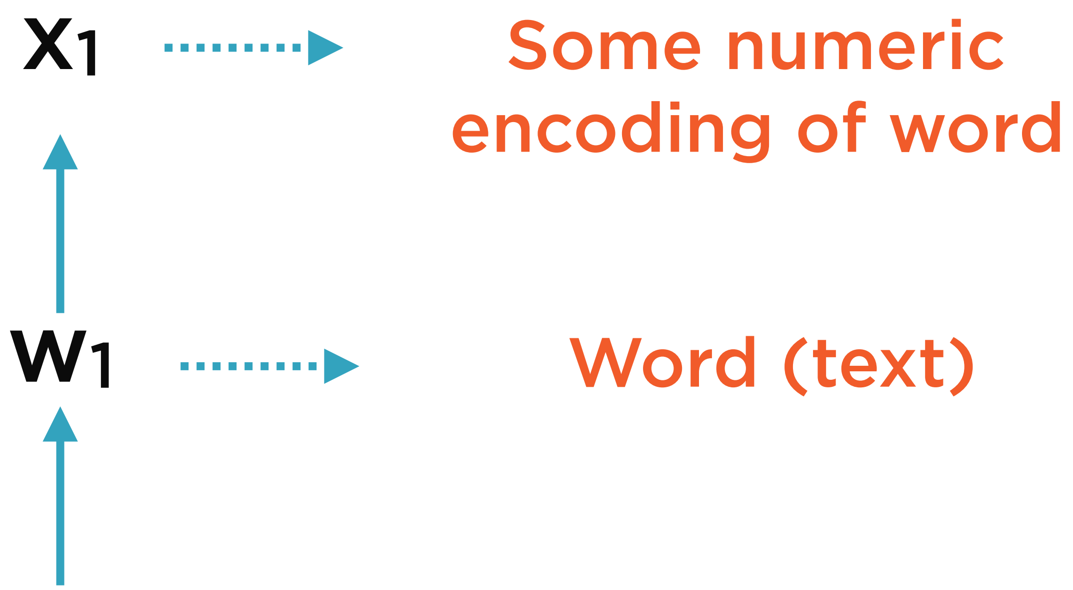


x_0 Some numeric encoding of word

w_0 Word (text)

("This", "is","not","the","worst","restaurant","in","the", "metropolis", "not","by","a","long","way")

Represent Each Word as a Number



x_1

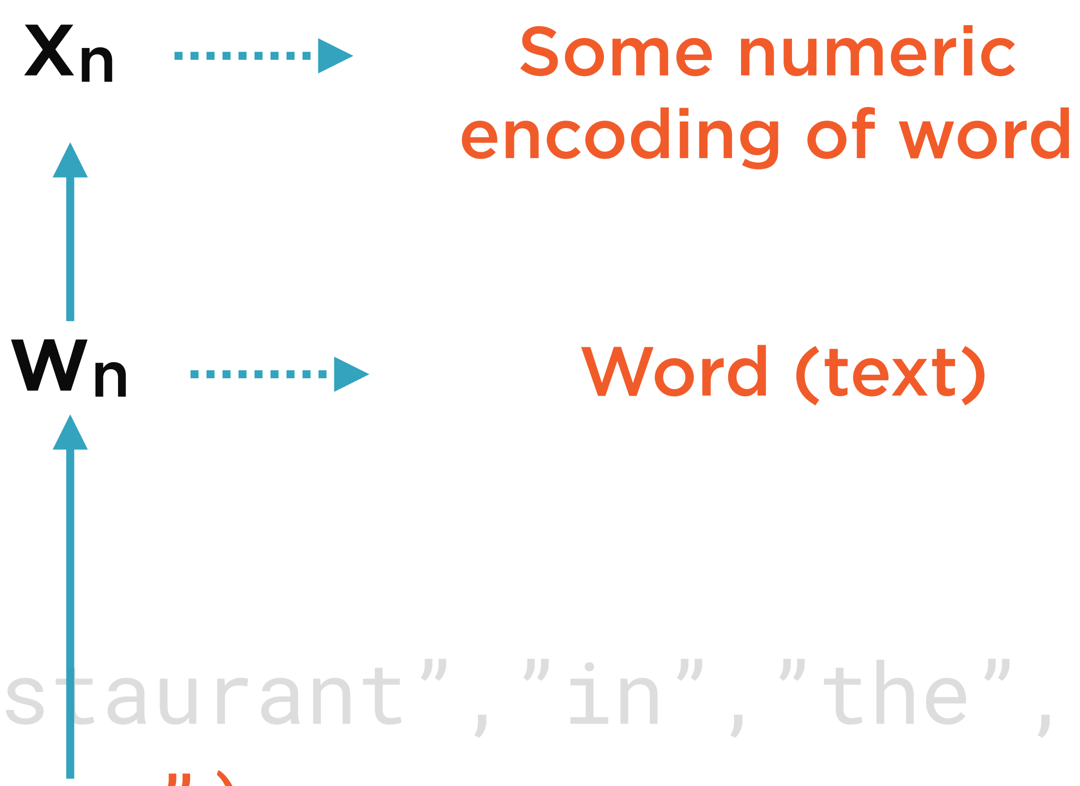
Some numeric encoding of word

w_1

Word (text)

(“This”, “is”,”not”,”the”,”worst”,”restaurant”,”in”,”the”, “metropolis”, “not”,”by”,”a”,”long”,”way”)

Represent Each Word as a Number



x_n

Some numeric encoding of word

w_n

Word (text)

(“This”, “is”,”not”,”the”,”worst”,”restaurant”,”in”,”the”, “metropolis”, “not”,”by”,”a”,”long”,”way”)

Represent Each Word as a Number

$d = [x_0, x_1, \dots x_n]$

Document as Tensor

Represent each word as numeric data, aggregate into tensor

x_i = [?]

The Big Question

How best can words be represented as numeric data?

```
d = [[?], [?], … [?]]
```

The Big Question

How best can words be represented as numeric data?

Word Embeddings

One-hot

Frequency-based

Prediction-based

Word Embeddings

One-hot

Frequency-based

Prediction-based

Word Embeddings

Numerical representations of text which capture meanings and semantic relationships

Pre-trained Word Embeddings

Word2Vec

GLoVe

Word embeddings can also be generated using your text corpus during the training process

Low dimensionality encoding of input words

Preprocessing Text Using torchtext

torchtext

Utility package for processing text in PyTorch, specifically meant for natural language processing

torchtext

torchtext.data

- Fields, iterators, pipelines

torchtext.datasets

- Sentiment analysis
- Sequence tagging
- Question classification

torchtext

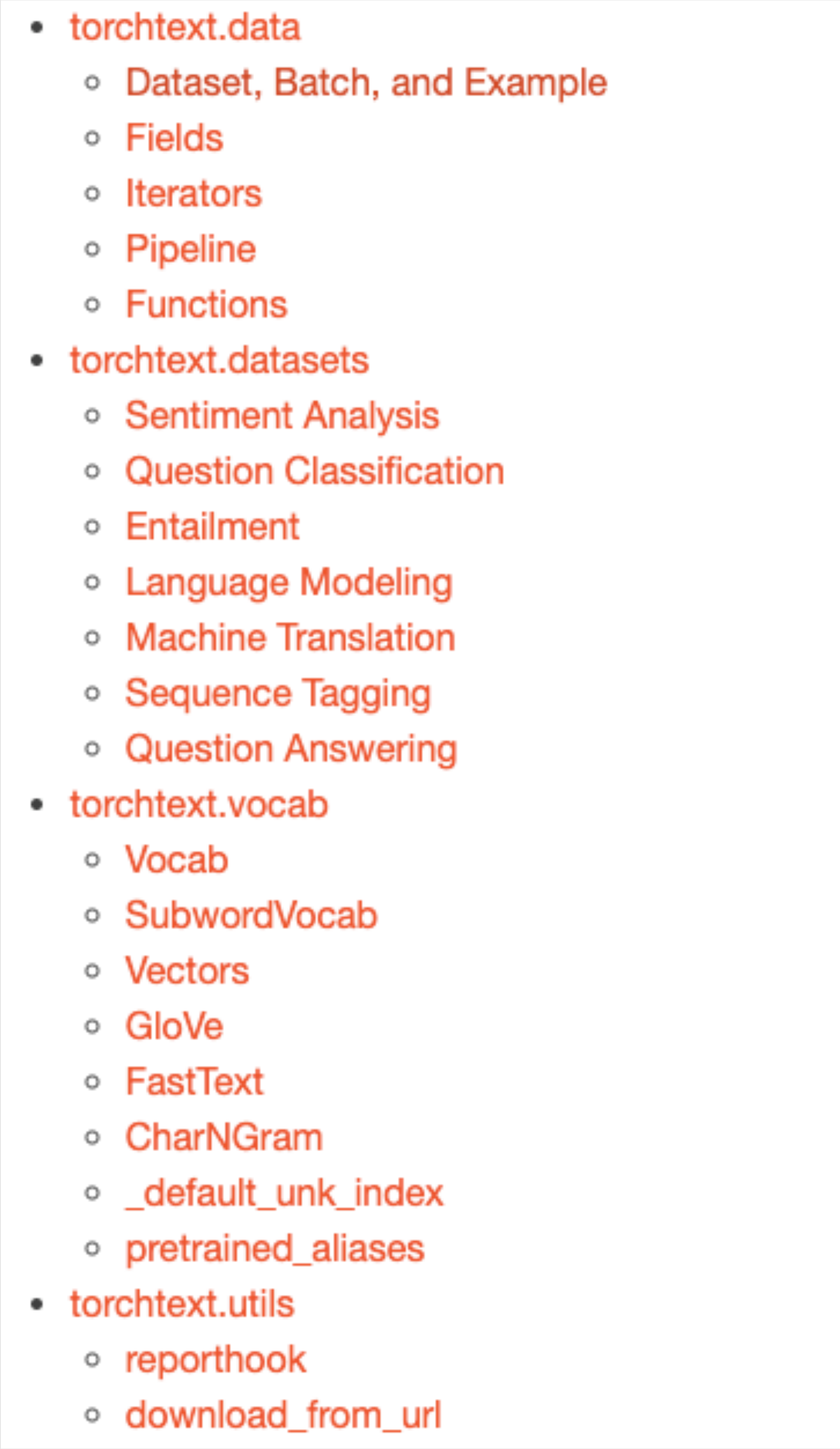
torchtext.vocab

- GloVe
- CharNGram

torchtext.utils

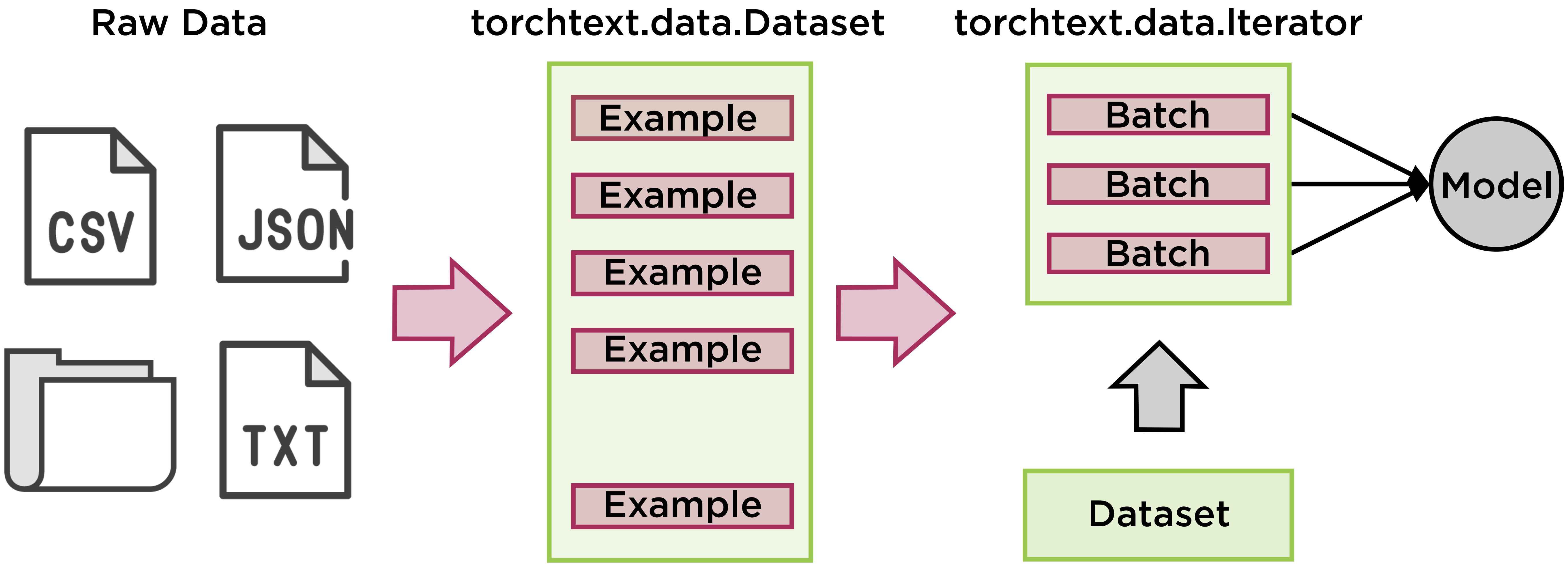
- download_from_url

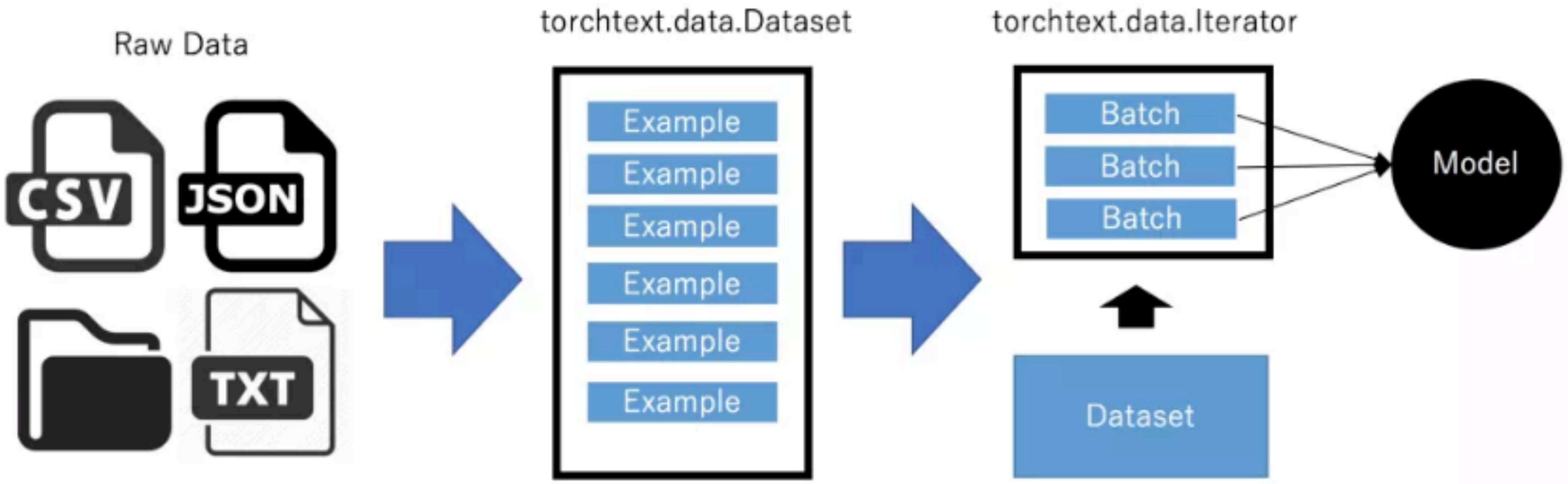
Torchtext



- torchtext.data
  - Dataset, Batch, and Example
  - Fields
  - Iterators
  - Pipeline
  - Functions
- torchtext.datasets
  - Sentiment Analysis
  - Question Classification
  - Entailment
  - Language Modeling
  - Machine Translation
  - Sequence Tagging
  - Question Answering
- torchtext.vocab
  - Vocab
  - SubwordVocab
  - Vectors
  - GloVe
  - FastText
  - CharNGram
  - _default_unk_index
  - pretrained_aliases
- torchtext.utils
  - reporthook
  - download_from_url

Using torchtext





Handles:
- Reading data into memory
- Preprocessing data
- Numericalizing data

Handles:
- Batching and padding data
- Moving data to the GPU

Torchtext takes in raw data in the form of text files, csv/tsv files, json files, and directories (as of now) and converts them to Datasets. Datasets are simply preprocessed blocks of data read into memory with various fields. They are a canonical form of processed data that other data structures can use.

Torchtext then passes the Dataset to an Iterator. Iterators handle numericalizing, batching, packaging, and moving the data to the GPU. Basically, it does all the heavy lifting necessary to pass the data to a neural network.

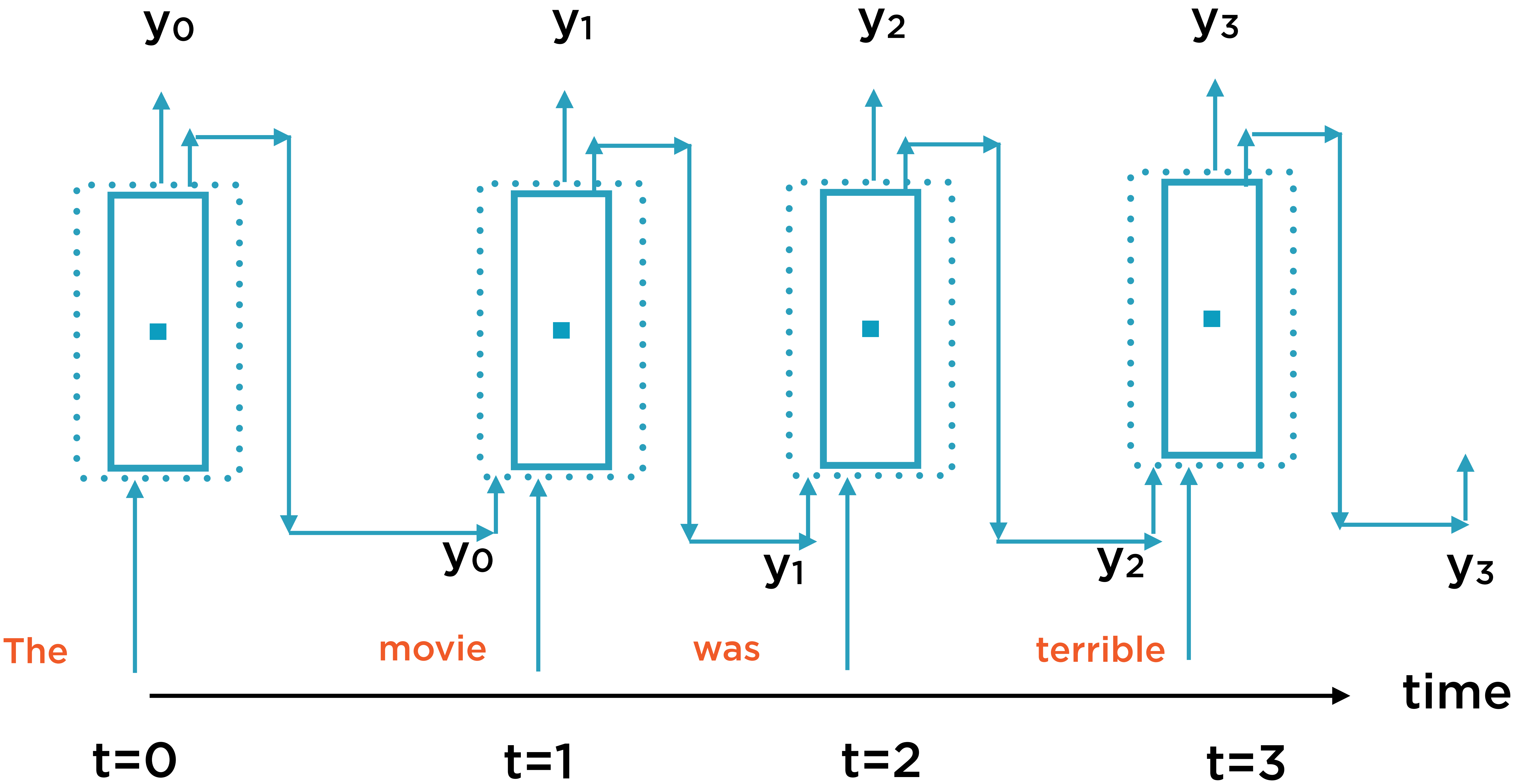
Feeding Text Data to RNNs

```
d = “The movie was terrible. Hated it.”
```

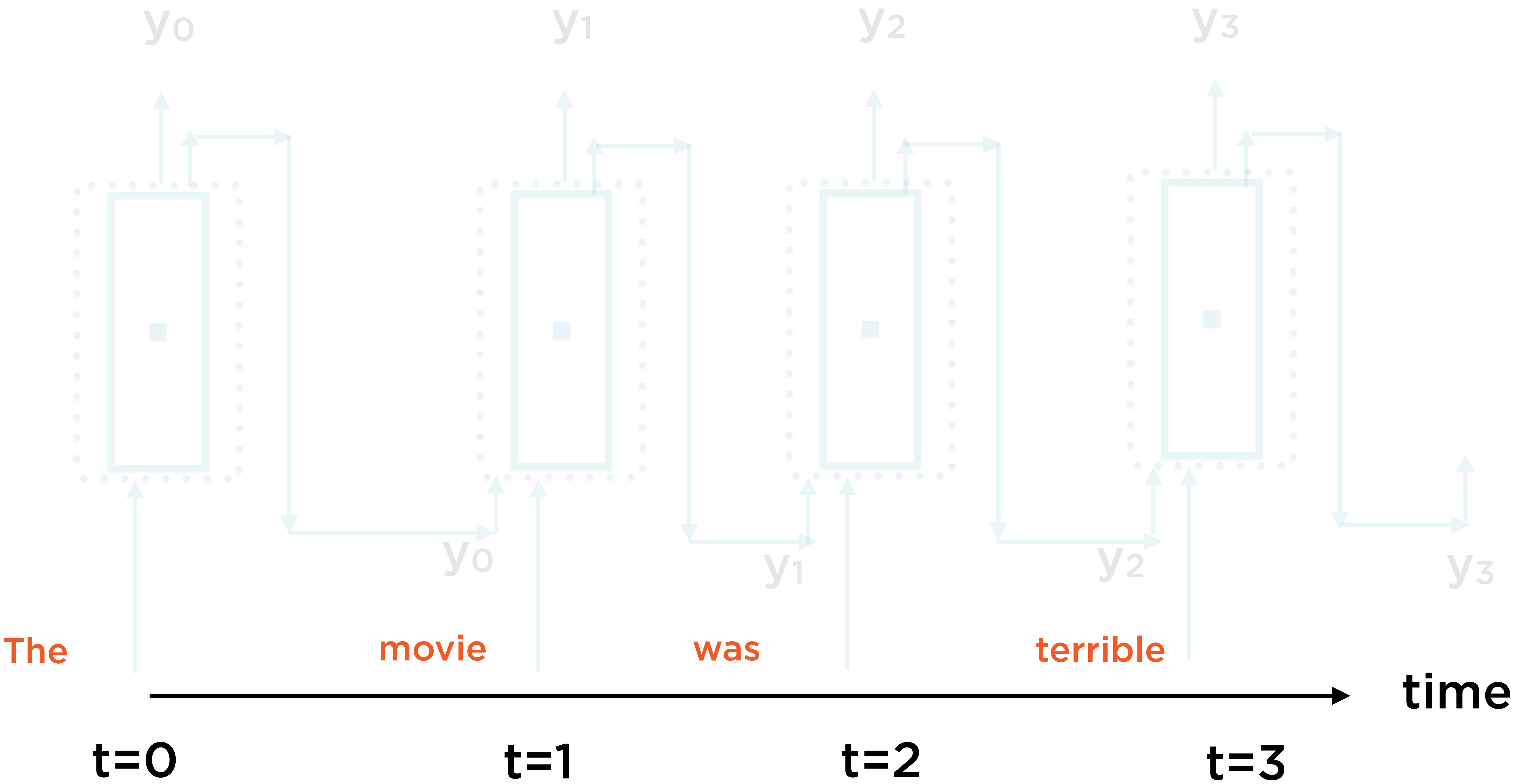
Document as Word Sequence

Model a document as an ordered sequence of words

Words as Input to RNNs



One Word Fed in at a Time Instant



The

movie

was

terrible

time

t=0

t=1

t=2

t=3

Documents and Corpus

Reviews
Amazing!
Worst movie ever
Two thumbs up
Part 2 was bad, 3 the worst
Up there with the greats

Consider all the words in the entire corpus - this is the vocabulary

Split into Words

Reviews
Amazing!
Worst movie ever
Two thumbs up
Part 2 was bad, 3 the worst
Up there with the greats

All Words
amazing
worst
movie
ever
two
thumbs
up
part
was
bad
3
the
there
with
greats

Create a set of all words (all across the corpus)

Assign Unique Identifiers

amazing	1
worst	2
movie	3
ever	4
two	5
thumbs	6
up	7
part	8
was	9
bad	10
3	11
the	12
there	13
with	14
greats	15

Each word has a unique numeric identifier

```
d = “This is not the worst restaurant in the metropolis,
not by a long way”

d = [1, 2, 3, 4, 5, 6, 7, 4, 8, 3, 9, 10, 11, 12]
```

Vector of Unique Identifiers

Reviews All of the Same Length

amazing	1	pad	pad
worst movie ever	2	3	4
two thumbs up	5	unk	7
part 3 was bad, part 2 ok	8	11	9

Reviews are either padded or truncated so all sentences are of the same length

Reviews All of the Same Length

amazing	1	pad	pad
worst movie ever	2	3	4
two thumbs up	5	unk	7
part 3 was bad, part 2 ok	8	11	9

Pad shorter reviews with a special pad token

Reviews All of the Same Length

amazing	1	pad	pad
worst movie ever	2	3	4
two thumbs up	5	unk	7
part 3 was bad, part 2 ok	8	11	9

Truncate longer reviews

Reviews All of the Same Length

amazing	1	pad	pad
worst movie ever	2	3	4
two thumbs up	5	unk	7
part 3 was bad, part 2 ok	8	11	9

Words not in the vocabulary of the training data represented using a special token

All of the numeric identifiers will be converted to word embeddings

Demo

Binary text classification using RNNs

Summary

Input processing as prelude to text classification

Representing text as tensors

Generating word embeddings from a training corpus

Using basic RNN and LSTM cells

Binary text classification using RNNs