



ALMA MATER STUDIORUM  
UNIVERSITÀ DI BOLOGNA  
CAMPUS DI FORLÌ

**91258 / B0385**

## **Natural Language Processing**

### **Lesson 19. LSTM: characters and generation**

Alberto Barrón-Cedeño  
a.barron@unibo.it

01/12/2025

## Previously

- Convolutional neural networks
- Recurrent neural networks
- Bidirectional Recurrent neural networks
- Long short-term memory networks

## Table of Contents

1. Out of Vocabulary
2. Characters
3. Text generation

Chapter 9 of Lane et al. (2019)

## Out of Vocabulary

## The curse of OOV

Out-of-vocabularies cause big trouble

The Mexico City Metro, operated by the Sistema de Transporte Colectivo, is the second largest metro system in North America after the New York City Subway.

The Mexico\_City Metro, operated by the · de · ·, is the second largest metro system in North America after the New\_York City Subway.

### Alternatives

- Replace the unknown with a random word, from the embedding space
- Replace the unknown word with UNK, and produce a random vector
- **Turn into characters**

[https://en.wikipedia.org/wiki/Mexico\\_City\\_Metro](https://en.wikipedia.org/wiki/Mexico_City_Metro) (2021)


## Characters

## Into Characters

Words are *just* a sequence of characters

By modeling the representations at the character level...

- We end up with a small closed vocabulary
- We get rid of OOVs
- We can learn patterns at a lower level
- We reduce the variety of input vectors drastically

 Let us see

## Into Characters: outcome

- The training takes close to 4 minutes (the original implementation from the book takes more than 30)<sup>1</sup>

epoch	seconds	acc	acc <sub>val</sub>
1	24	0.5365	0.5785
2	21	0.6468	0.5827
3	41	0.6859	0.5763
4	21	0.7262	0.5739
5	20	0.7539	0.5731
6	21	0.7766	0.5666
7	19	0.8008	0.5700
8	20	0.8135	0.5719
9	19	0.8342	0.5799
10	21	0.8459	0.5843

<sup>1</sup>Using Google's colab2.5GHz Quad-Core Intel Core i7 with 16GB of RAM

## Into Characters: outcome

- The training accuracy is “promising”:  $\sim 84.40$
- The validation accuracy is terrible:  $\sim 58.40$
- Overfitting

### Reasons/Solutions

- The model might be *memorising* the dataset
- Increase the dropout (try!)
- Add more labeled data (hard!)

A character-level model shines at its best when modeling/generating language

## Text generation

## Predicting the next word

- An LSTM can learn

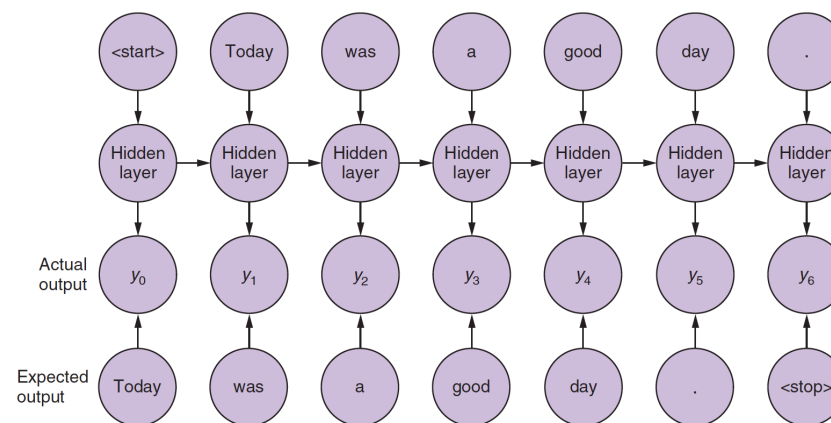
$$p(w_t \mid w_{t-1}, w_{t-2}, \dots, w_{t-n}) \quad (1)$$

- It can do so **with a memory** (full context)
- It can do so at the **character level**

### From classification to *generation*

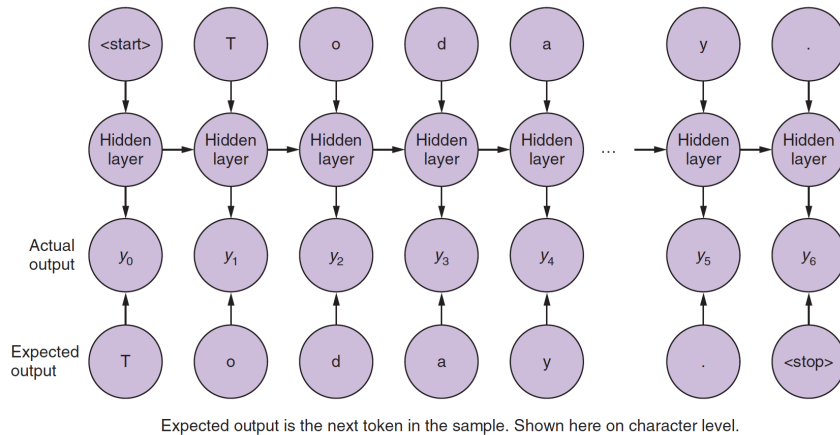
- Now we want to predict the next word ( $\sim$  word2vec?)
- We want to learn a *general* representation of language

## Unrolling the next-word prediction (word 2-grams)



(Lane et al., 2019, 299)

## Unrolling the next-word character prediction

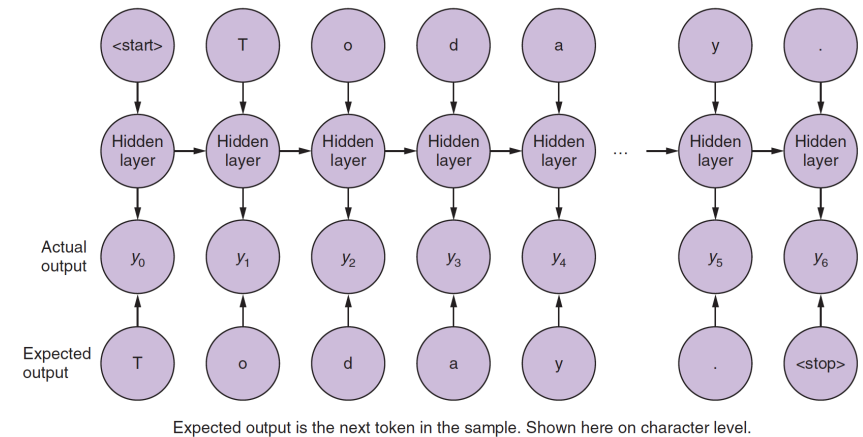


- Now the error is computed for every single output
- We still back-propagate only after passing a full instance

(Lane et al., 2019, 299)

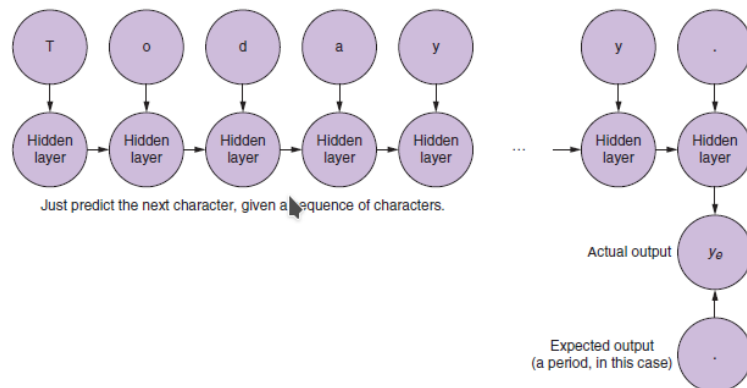
## New target labels

New output: a one-hot encoding (again) of the next character



(Lane et al., 2019, 299)

## Predict after having looked at a sequence



(Lane et al., 2019, 300)

## Generation example

Since we are interested in *style* and in creating a consistent model, we won't use IMDB (multi-authored and small).

Let us try to *mimic* William Shakespeare

Let us see

## Adding Extra Stuff

- Expand the quantity and quality of the corpus
- Expand the complexity of the model (units/layers/LSTMs)
- Better pre-processing:
  - Better case folding
  - Break into sentences
- Post-processing
  - Add filters on grammar, spelling, and tone
  - Generate many more examples than actually shown to users
- Select better seeds (e.g., context, topic)

Most of these strategies apply to any problem you can think about!

---

(Lane et al., 2019, 307)

## References

Lane, H., C. Howard, and H. Hapkem  
2019. *Natural Language Processing in Action*. Shelter Island, NY: Manning  
Publication Co.