ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI FORLÌ

91258 / B0385
Natural Language Processing

Lesson 18. LSTM: characters and generation

Alberto Barrón-Cedeño
a.barron@unibo.it

09/12/2024

Previously

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

Table of Contents

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.

https://en.wikipedia.org/wiki/Mexico_City_Metro (2021)

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.

https://en.wikipedia.org/wiki/Mexico_City_Metro (2021)

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 wit UNK, and produce a random vector

https://en.wikipedia.org/wiki/Mexico_City_Metro (2021)

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 wit UNK, and produce a random vector
- **Turn into characters**

https://en.wikipedia.org/wiki/Mexico_City_Metro (2021)

Characters

Into Characters

Words are *just* a sequence of 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

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

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

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

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 no less than 30 minutes (it took me 36 last time)[1]

[1] 2.5GHz Quad-Core Intel Core i7 with 16GB of RAM

Into Characters: outcome

- The training takes no less than 30 minutes (it took me 36 last time)[1]

epoch	**seconds**	**acc**	**acc_{val}**
1	208	0.5206	0.5934

[1] 2.5GHz Quad-Core Intel Core i7 with 16GB of RAM

Into Characters: outcome

- The training takes no less than 30 minutes (it took me 36 last time)[1]

epoch	**seconds**	**acc**	**acc$_{val}$**
1	208	0.5206	0.5934
2	190	0.6832	0.5900
3	184	0.7534	0.5826
4	183	0.8029	0.5664

[1] 2.5GHz Quad-Core Intel Core i7 with 16GB of RAM

Into Characters: outcome

- The training takes no less than 30 minutes (it took me 36 last time)[1]

epoch	**seconds**	**acc**	**acc$_{val}$**
1	208	0.5206	0.5934
2	190	0.6832	0.5900
3	184	0.7534	0.5826
4	183	0.8029	0.5664
5	182	0.8371	0.5654
6	182	0.8633	0.5652
7	182	0.8908	0.5672
8	179	0.9086	0.5774
9	178	0.9212	0.5744

[1] 2.5GHz Quad-Core Intel Core i7 with 16GB of RAM

Into Characters: outcome

- The training takes no less than 30 minutes (it took me 36 last time)[1]

epoch	**seconds**	**acc**	**acc$_{val}$**
1	208	0.5206	0.5934
2	190	0.6832	0.5900
3	184	0.7534	0.5826
4	183	0.8029	0.5664
5	182	0.8371	0.5654
6	182	0.8633	0.5652
7	182	0.8908	0.5672
8	179	0.9086	0.5774
9	178	0.9212	0.5744
10	179	0.9346	0.5898

[1] 2.5GHz Quad-Core Intel Core i7 with 16GB of RAM

Into Characters: outcome

- The training accuracy is quite promising: ~ 93.00
- The validation accuracy is terrible: ~ 59.00

Into Characters: outcome

- The training accuracy is quite promising: ~ 93.00
- The validation accuracy is terrible: ~ 59.00
- Overfitting

Reasons/Solutions

- The model might be *memorising* the dataset

Into Characters: outcome

- The training accuracy is quite promising: ~ 93.00
- The validation accuracy is terrible: ~ 59.00
- 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

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

Predicting the next word

- An LSTM can learn

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

- It can do so **with a memory** (full context)

Predicting the next word

- An LSTM can learn

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

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

Predicting the next word

- An LSTM can learn

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

- 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?)

Predicting the next word

- An LSTM can learn

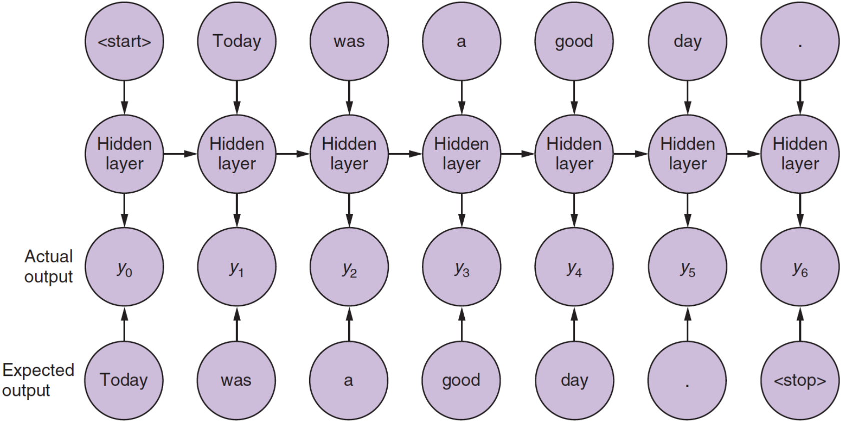
$$p(w_t \mid w_{t-1}, w_{t-2}, \ldots, w_{t-n}) \tag{1}$$

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

From classification to *generation*

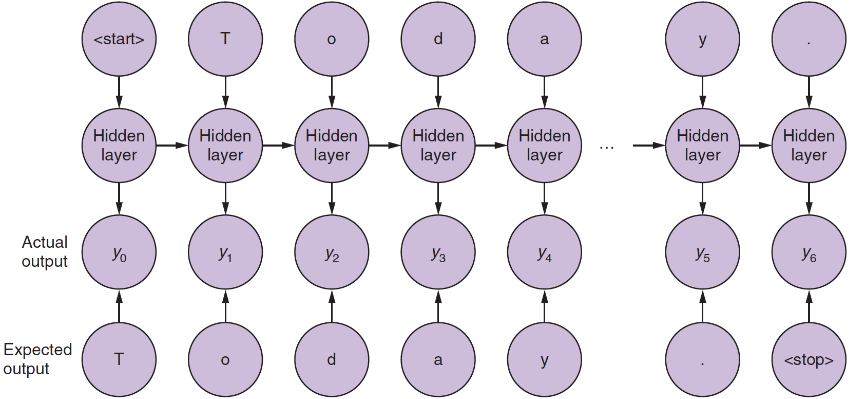
- Now we want to predict the next word ($\sim$ word2vec?)
- We want to learn a *general* representation of language

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



(Lane et al., 2019, 299)

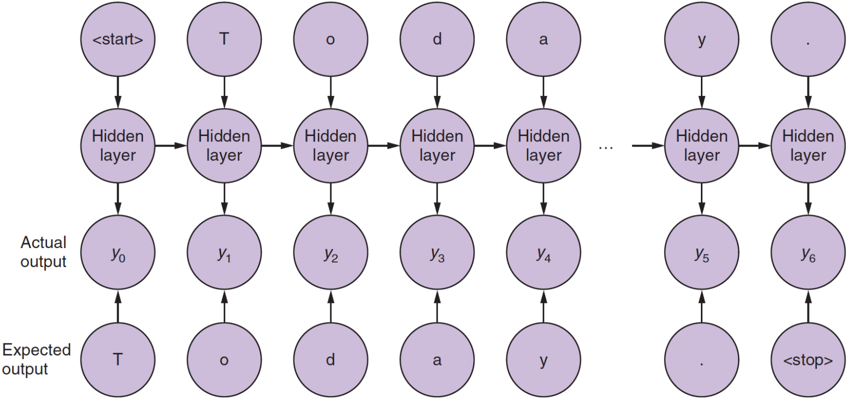
Unrolling the next-~~word~~character prediction



Expected output is the next token in the sample. Shown here on character level.

(Lane et al., 2019, 299)

Unrolling the next-~~word~~character prediction



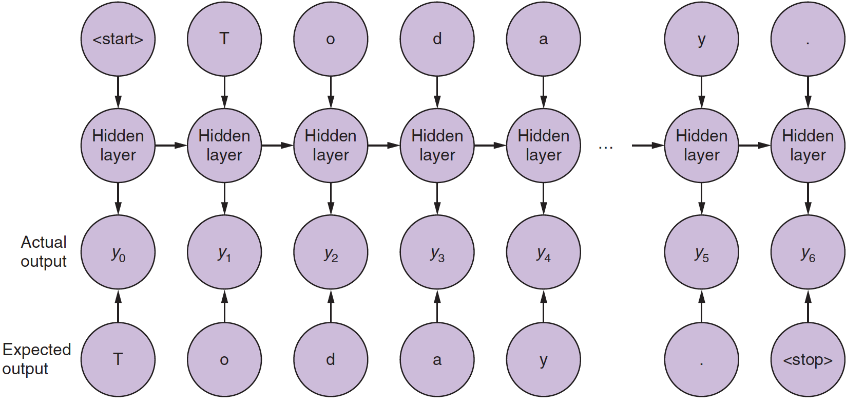
Expected output is the next token in the sample. Shown here on character level.

- 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

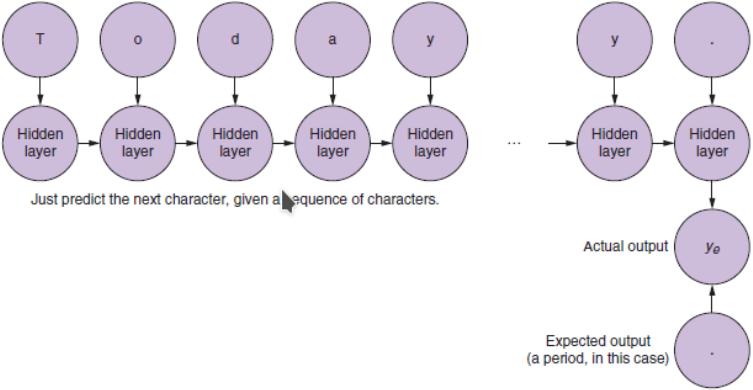
New output: a one-hot encoding (again) of the next character



Expected output is the next token in the sample. Shown here on character level.

(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 wont use IMDB (multi-authored and small).

Generation example

Since we are interested in *style* and in creating a consistent model, we wont 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

(Lane et al., 2019, 307)

Adding Extra Stuff

- Expand the quantity and quality of the corpus
- Expand the complexity of the model (units/layers/LSTMs)

(Lane et al., 2019, 307)

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

(Lane et al., 2019, 307)

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

(Lane et al., 2019, 307)

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)

(Lane et al., 2019, 307)

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.