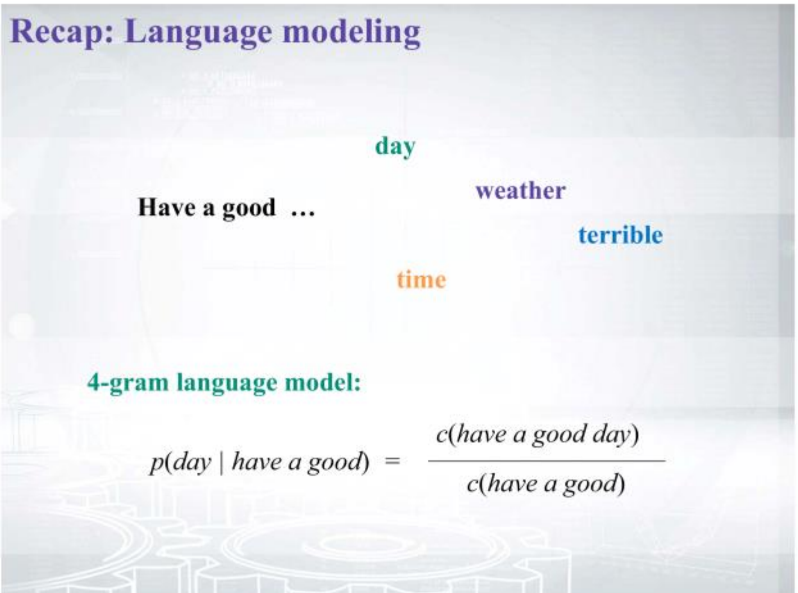
Deep learning for the same tasks

Friday, June 15, 2018 9:51 PM

nlm-dl


Recap: Language modeling
day
weather
Have a good ...
terrible
time
4-gram language model:
$p(day \mid have\ a\ good) = \frac{c(have\ a\ good\ day)}{c(have\ a\ good)}$

Curse of dimensionality

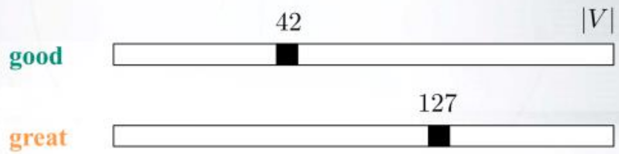
Imagine you have seen the following many times:

- **Have a good day**.

However, you have not seen the following:

- **Have a great day**.

What happens than (even with smoothing)?

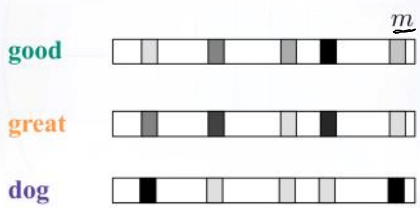


one-hot encoding
not good

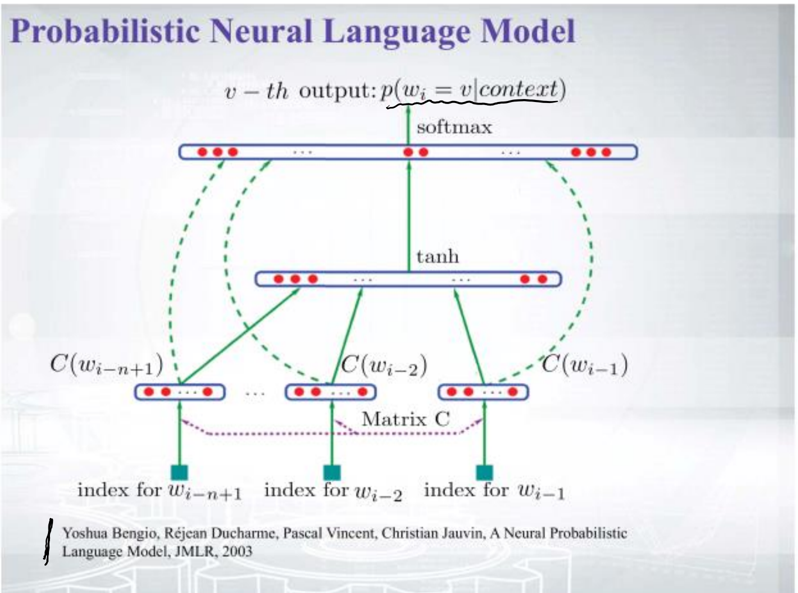
How to generalize better

- Learn distributed representations for words
- Express probabilities of sequences in terms of these distributed representations and learn parameters

distributed representation



$C^{|V| \times m}$ – matrix of distributed word representations.


Probabilistic Neural Language Model
v − th output: p(w_i = v|context)
softmax
tanh
C(w_{i−n+1})
C(w_{i−2})
C(w_{i−1})
Matrix C
index for w_{i−n+1}
index for w_{i−2}
index for w_{i−1}
Yoshua Bengio, Réjean Ducharme, Pascal Vincent, Christian Jauvin, A Neural Probabilistic Language Model, JMLR, 2003

Probabilistic Neural Language Model

$$p(w_i|w_{i-n+1}, \dots w_{i-1}) = \frac{\exp(y_{w_i})}{\sum_{w \in V} \exp(y_w)}$$

$$y = b + Wx + U \tanh(d + Hx)$$

$$x = [C(w_{i-n+1}), \dots C(w_{i-1})]^T$$

Probabilistic Neural Language Model

$$p(w_i|w_{i-n+1}, \dots w_{i-1}) = \frac{\exp(y_{w_i})}{\sum_{w \in V} \exp(y_w)}$$

Softmax* over components of *y

$$y = b + Wx + U \tanh(d + Hx)$$

$$x = [C(w_{i-n+1}), \dots C(w_{i-1})]^T$$

Probabilistic Neural Language Model

$$p(w_i|w_{i-n+1},\dots w_{i-1}) = \frac{\exp(y_{w_i})}{\sum\limits_{w \in V} \exp(y_w)}$$

Softmax* over components of *y

$$y = b + Wx + U\tanh(d + Hx)$$

***Feed-forward NN* with tons of parameters**

$$x = [C(w_{i-n+1}),\dots C(w_{i-1})]^T$$

$|V|$ by $m * (n-1)$

concatenation $\begin{array}{cccc} w_1 & w_2 & & w_{n-1} \\ \left[\begin{array}{c} \\ \\ \end{array} \right. & & \cdots & \left. \begin{array}{c} \\ \\ \end{array} \right] \\ m & m & & m \end{array}$

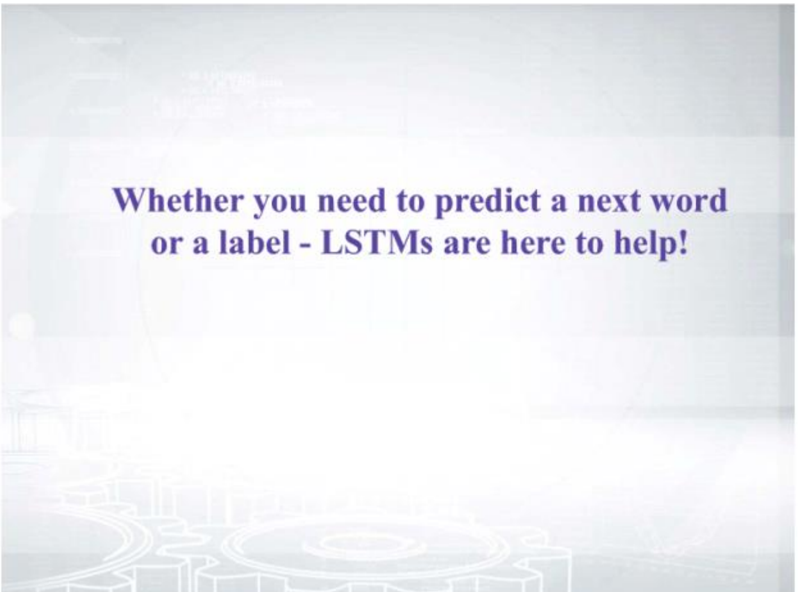
$n \times m$

row of
C matrices.

=

It is not a bag-of-word model.
words close to your product has
higher influence.

lstm


Whether you need to predict a next word
or a label - LSTMs are here to help!

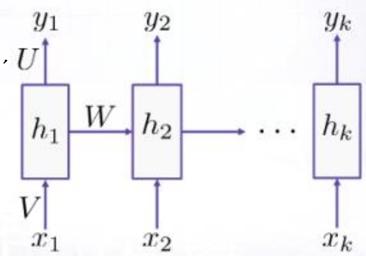
Recap: Recurrent Neural Networks

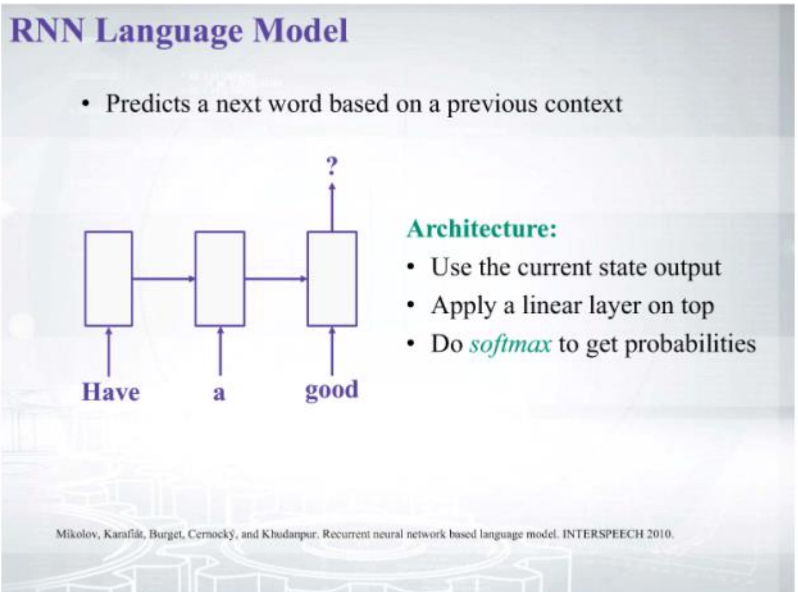
- Extremely popular architecture for any sequential data:

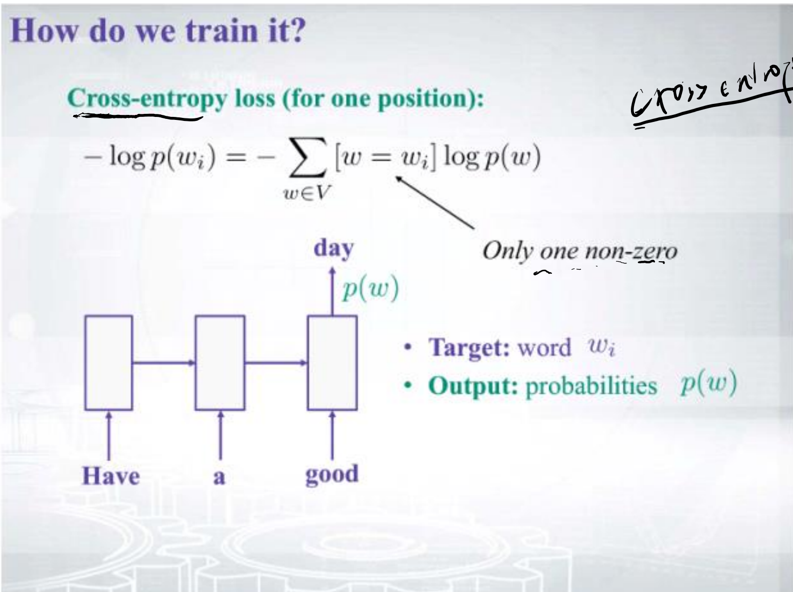
$$h_i = f(Wh_{i-1} + Vx_i + b)$$

$$y_i = Uh_i + \tilde{b}$$

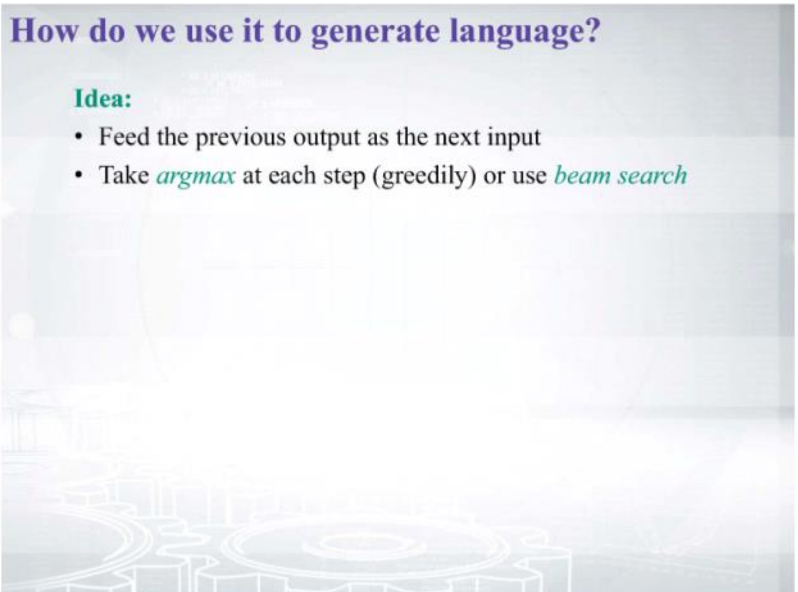
hidden layer
x size of vocab
dim.




RNN Language Model
Predicts a next word based on a previous context
?
Have
a
good
Architecture:
Use the current state output
Apply a linear layer on top
Do softmax to get probabilities
Mikolov, Karafiát, Burget, Cernocký, and Khudanpur. Recurrent neural network based language model. INTERSPEECH 2010.



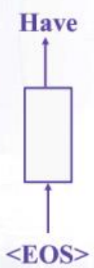
cross entropy loss.


How do we use it to generate language?
Idea:
• Feed the previous output as the next input
• Take *argmax* at each step (greedily) or use *beam search*

How do we use it to generate language?

Idea:

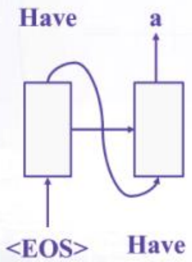
- Feed the previous output as the next input
- Take *argmax* at each step (greedily) or use *beam search*



How do we use it to generate language?

Idea:

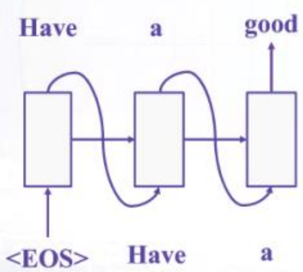
- Feed the previous output as the next input
- Take *argmax* at each step (greedily) or use *beam search*



How do we use it to generate language?

Idea:

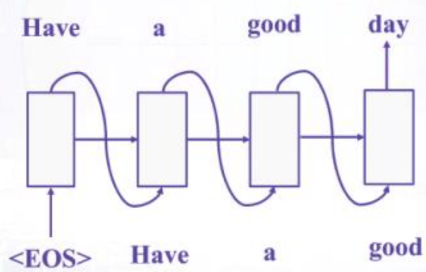
- Feed the previous output as the next input
- Take *argmax* at each step (greedily) or use *beam search*



How do we use it to generate language?

Idea:

- Feed the previous output as the next input
- Take *argmax* at each step (greedily) or use *beam search*



RNN Language Model

- RNN-LM has lower *perplexity* and *word error rate* than 5-gram model with Knesser-Ney smoothing.
- The experiment is held on Wall Street Journal corpus:

Model	# words	PPL	WER
KN5 LM	200K	336	16.4
KN5 LM + RNN 90/2	200K	271	15.4
KN5 LM	1M	287	15.1
KN5 LM + RNN 90/2	1M	225	14.0
KN5 LM	6.4M	221	13.5
KN5 LM + RNN 250/5	6.4M	156	11.7

- Later experiments: char-level RNNs can be very effective!

Mikolov, Karafiát, Burget, Cernocký, and Khudanpur. Recurrent neural network based language model. INTERSPEECH 2010.

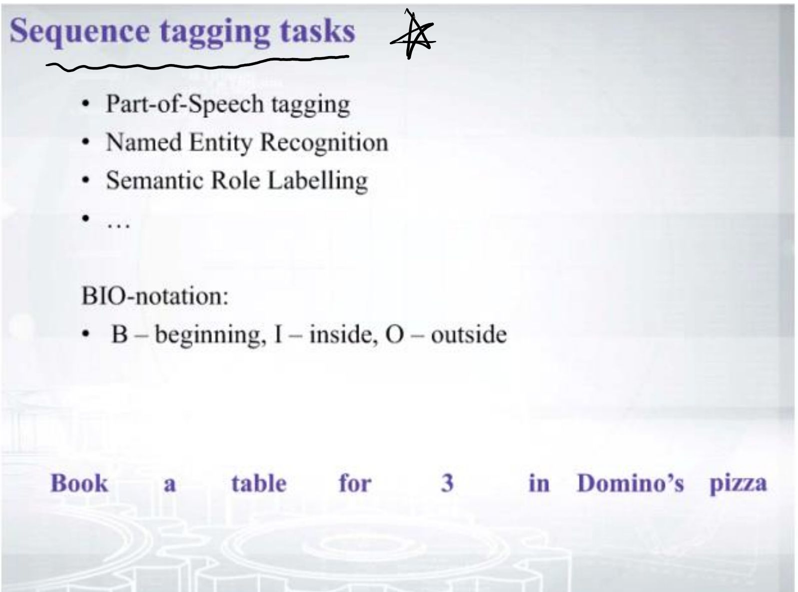
Character-level RNN: Shakespeare example

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.
```

Andrej Karpathy, http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Cook your own Language Model

- Use LSTMs or GRUs, and gradient clipping
  https://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Start with one layer, then stack 3-4, use skip connections
- Use dropout for regularization:
  Zaremba, Sutskever, Vinyals. Recurrent Neural Network Regularization, 2014.
- Have a look into TF tutorial for a working model:
  https://www.tensorflow.org/tutorials/recurrent
- Tune learning rate schedule in SGD or use Adam
- Explore state-of-the-art improvements:
  - **July 2017:** On the State of the Art of Evaluation in Neural Language Models.
  - **August 2017:** Regularizing and Optimizing LSTM Language Models.


Sequence tagging tasks
Part-of-Speech tagging
Named Entity Recognition
Semantic Role Labelling
…
BIO-notation:
B – beginning, I – inside, O – outside
Book a table for 3 in Domino's pizza

Sequence tagging tasks

- Part-of-Speech tagging
- Named Entity Recognition
- Semantic Role Labelling
- …

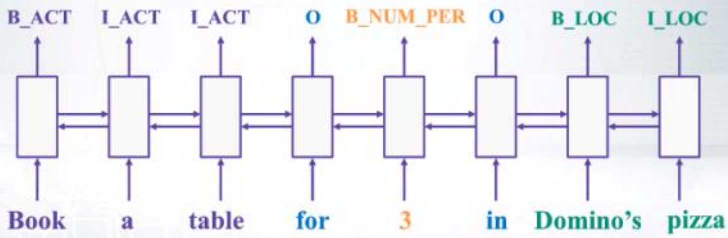
BIO-notation:

- B – beginning, I – inside, O – outside

B_ACT	I_ACT	I_ACT	O	B_NUM_PER	O	B_LOC	I_LOC
Book	**a**	**table**	**for**	**3**	**in**	**Domino's**	**pizza**

Bi-directional LSTM

- Universal approach for sequence tagging
- You can stack several layers + add linear layers on top
- Trained by cross-entropy loss coming from each position



Sequence tagging

① CRF. older

② Bidirectional LSTM.

③ mixed
Bidirectional LSTM generate features
-> fit in CRF.