MxNet Gluon

Basics, Computer Vision, NLP (and even more NLP)
Part V (Word Embeddings & Language Model)

Leonard Lausen
Haibin Lin
Alex Smola

aws

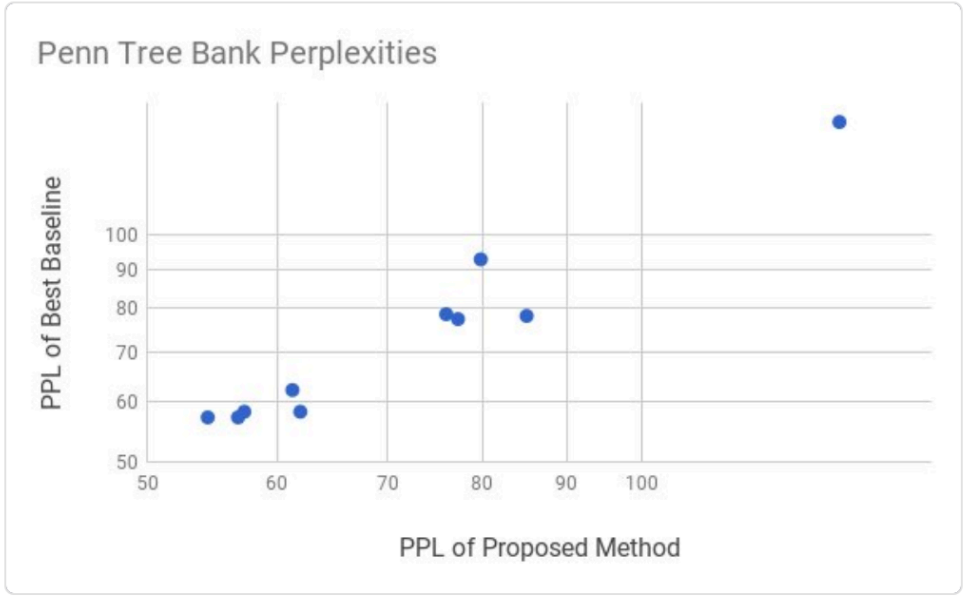
Outline

8:30-9:15	Installation and Basics (NDArray, AutoGrad, Libraries)
9:15-9:30	Neural Networks 101 (MLP, ConvNet, LSTM, Loss, SGD) - Part I
9:30-10:00	Break
10:00-10:30	Neural Networks 101 (MLP, ConvNet, LSTM, Loss, SGD) - Part II
10:30-11:00	Computer Vision 101 (Gluon CV)
11:00-11:30	Parallel and distributed training
11:30-12:00	Data I/O in NLP (and iterators)
12:00-13:30	Break
13:30-14:15	Embeddings
14:15-15:00	Language models (LM)
15:00-15:30	Sequence Generation from LM
15:30-16:00	Break
16:00-16:15	Sentiment analysis
16:15-17:00	Transformer Models & machine translation
17:00-17:30	Questions

Aaron Jaech
@AaronJaech

Follow

Updated graph of 10 #iclr2018 lang modeling papers comparing ppl of proposed model vs. best baseline on penn tree bank.

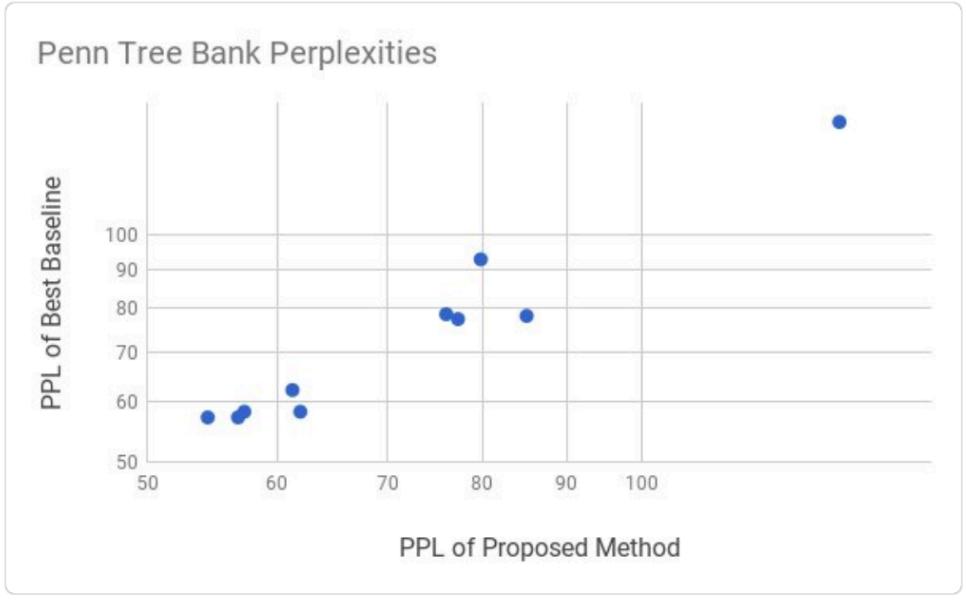


9:47 AM - 29 Oct 2017

Aaron Jaech
@AaronJaech

Follow

Updated graph of 10 #iclr2018 lang modeling papers comparing ppl of proposed model vs. best baseline on penn tree bank.



9:47 AM - 29 Oct 2017

Baselines are hard.

GluonNLP isn't.

aws

GluonNLP Goals

- Prototype code is not reusable without copying.
- Carefully designed API for versatile needs.

- Code may break due to API changes.
- Integrated testing for examples.

- Setting up baseline for NLP tasks is hard.
- Implementation for state-of-the-art models.

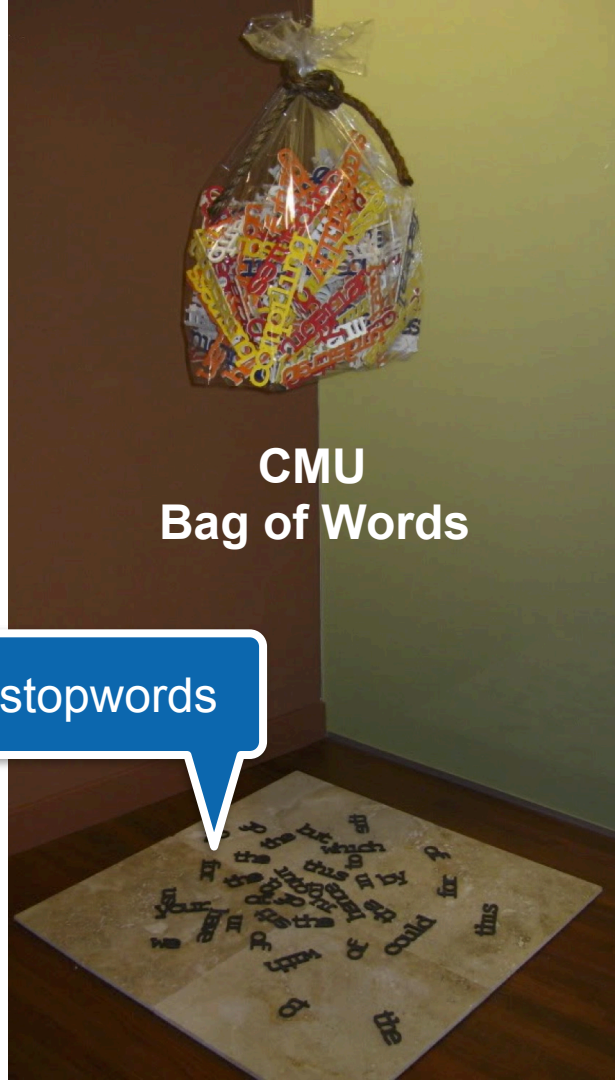
Word embeddings

Bag of words

- Map text to sparse vectors
  'I think therefore I am' = (2,0,1,0,1,1,…)
- Use linear model (Salton & McGill '93)
- Maybe some feature engineering
  (string kernels etc.)

$$f(x) = \langle w, x \rangle = \sum_{\text{word} \in x} w_{\text{word}} n_{\text{word}}[x]$$

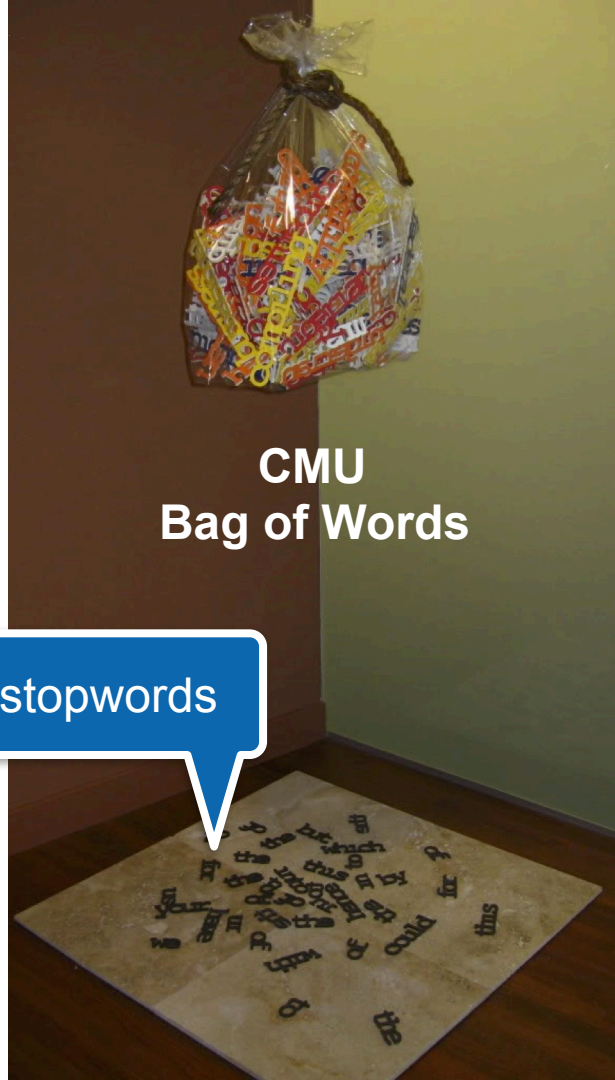
not much progress for 2 decades …



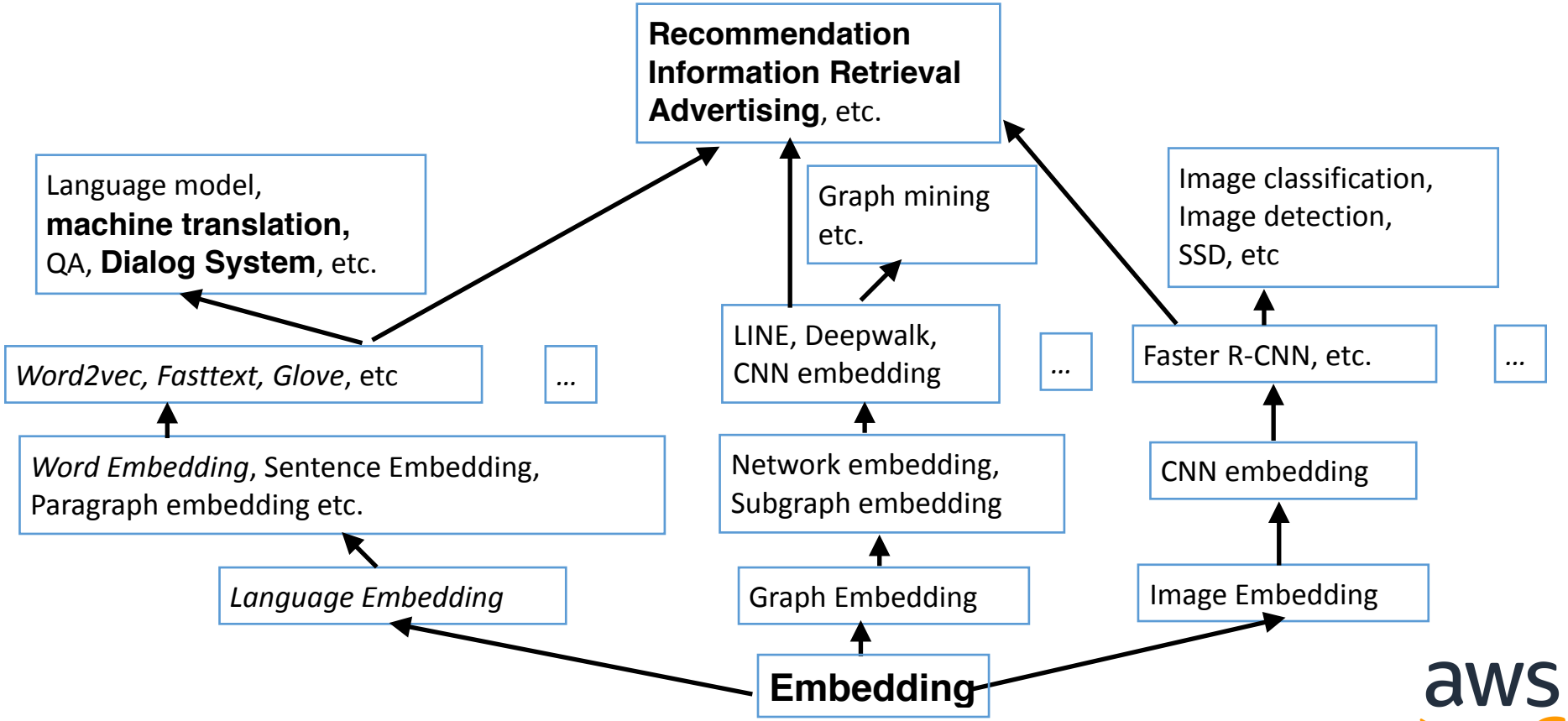
Bag of words

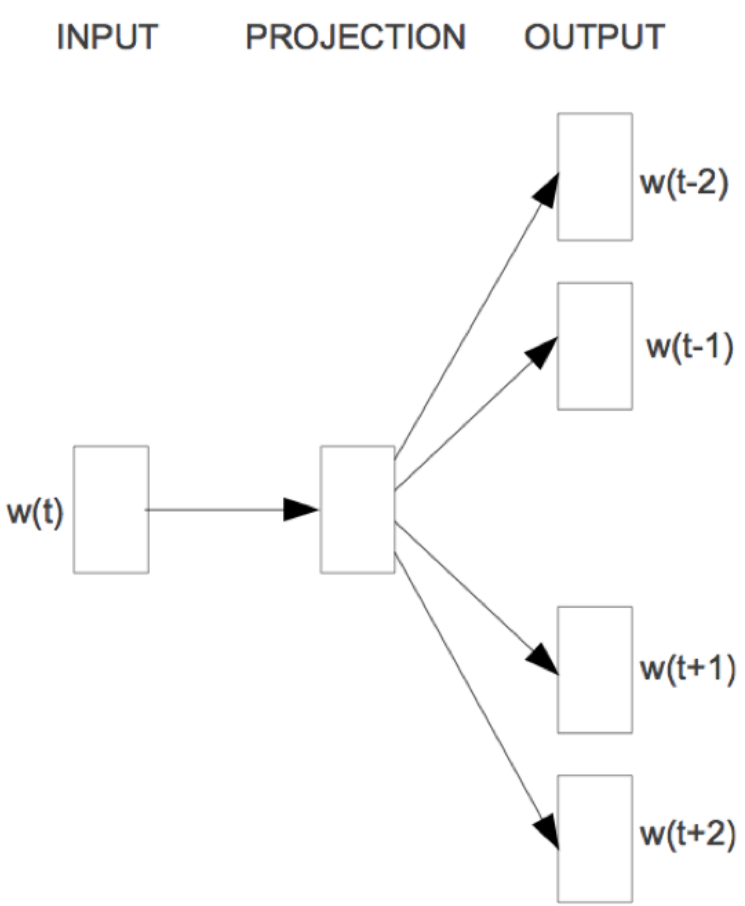
- Map text to sparse vectors
  'I think therefore I am' = (2,0,1,0,1,1,…)
- Ignores important details
  - '**cake**' and '**layercake**' are probably related
  - Neighbors of words are meaningful (Pantel & Lin, 2000)
    - 'Haibin drives a red Ford'
    - 'Haibin drives a red Toyota'

Ford ~ Toyota



Why embeddings?





Word2vec

- Learn distributional similarity by trying to estimate which words co-occur, e.g. 5-grams

$$p(w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}|w)$$

- This is expensive. Hack it

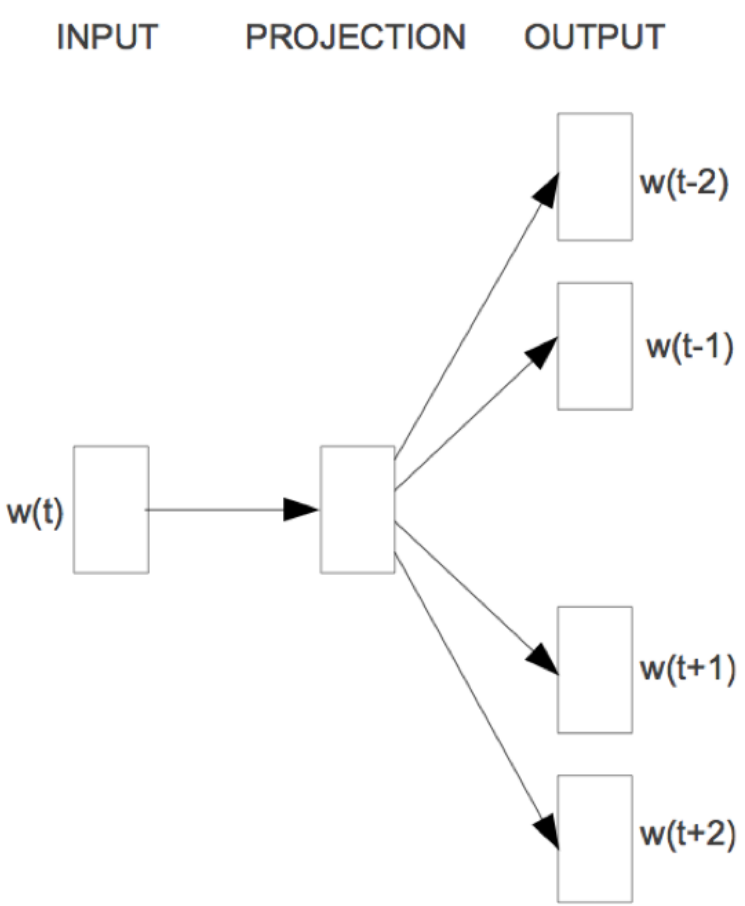
1. Factorization

$$\approx \prod_{j \in \{-2,-1,1,2\}} p(w_{t+j}|w)$$

2. Classification problem

$$\frac{p(w_{t+j}|w)}{p(w_{t+j}|w) + p_{\text{background}}(w)}$$

"Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. ICLR Workshop , 2013."



Word2vec

- Sample anchor word w
- Sample neighboring words w'|w
- Sample k unrelated words v_i from background distribution

$$\frac{p(w_{t+j}|w)}{p(w_{t+j}|w) + k \cdot p_{\text{background}}(w)}$$

- Overweight tail in background

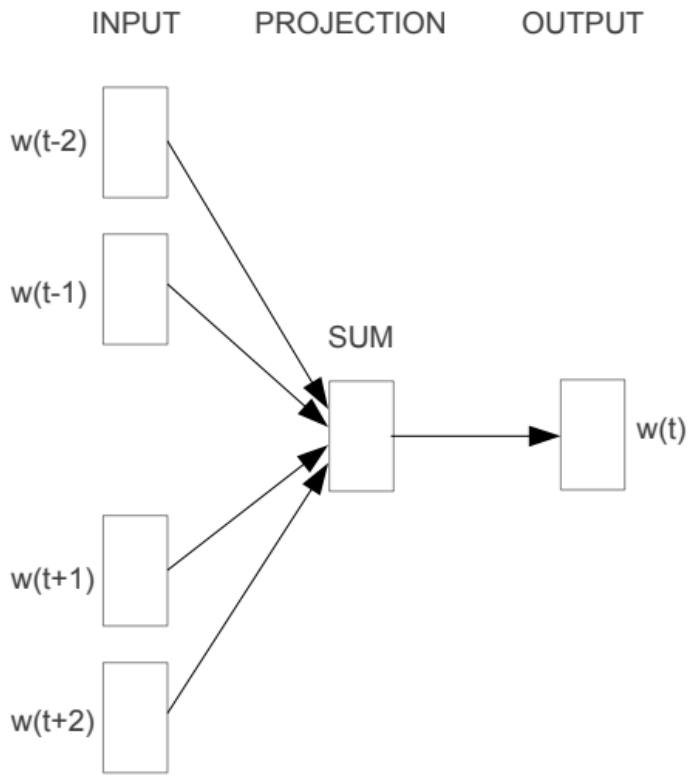
$$p_{\text{background}}(w) \propto p(w)^{0.75}$$

- Train classifier

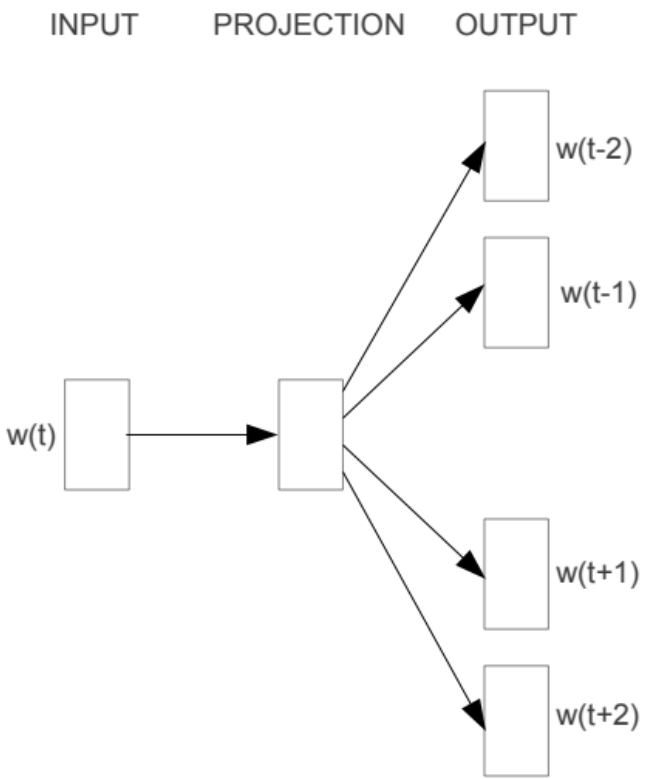
"Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. ICLR Workshop , 2013."

aws

Alternatives



CBOW



Skip-gram

About that classifier …

- Mostly a means to an end - **embeddings** for words
- **Affinity between w and v** via (some variant of)

$$e_w^\top M e_v$$

  A. Encoder and decoder embeddings different
  B. They are the same
- **Computing the embedding**
  A. Dictionary lookup
  B. Character-LSTM
  C. Convolution

Analogies in vector space (king:man = queen:woman)

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Are the embeddings any good?

- Use them for other purposes and see if they improve (usually they do, but can we quantify better)
- **Similarity**
  - Distance in vector space $\|e_w - e_{w'}\|^2$
  - Cosine similarity $\langle e_w, e_{w'} \rangle$ or normalized $\frac{\langle e_w, e_{w'} \rangle}{\|e_w\| \, \|e_{w'}\|}$
  - Both are equivalent for unit vectors

$$\|e_w - e_{w'}\|^2 = \|e_w\|^2 + \|e_{w'}\|^2 - 2 \langle e_w, e_{w'} \rangle$$

Are the embeddings any good?

- **Analogy**
  - Translation in vector space

$$e_{\text{king}} - e_{\text{man}} \approx e_{\text{queen}} - e_{\text{woman}}$$

  - Use this to find e.g. minimum distance in that space

$$\underset{b^* \notin \{a, a^*, b, b^*\}}{\operatorname{argmin}} \left\| a^* - a + b - b^* \right\|^2$$

  - Normalized vectors work even better (aka cosine)

$$\underset{b^* \notin \{a, a^*, b, b^*\}}{\operatorname{argmin}} \left\| e_{a^*} - e_a + e_b - e_{b^*} \right\|^2$$

$$\underset{b^* \notin \{a, a^*, b, b^*\}}{\operatorname{argmax}} \left\langle e_{b^*}, e_{a^*} \right\rangle - \left\langle e_{b^*}, e_a \right\rangle + \left\langle e_{b^*}, e_b \right\rangle$$

Are the embeddings any good?

- **Analogy**
  - Sums of normalized vectors work well (aka cosine)
  - Products even better (Goldberg et al., 2014)

$$\underset{b^* \notin \{a, a^*, b, b^*\}}{\operatorname{argmax}} \quad \langle e_{b^*}, e_{a^*} \rangle - \langle e_{b^*}, e_a \rangle + \langle e_{b^*}, e_b \rangle$$

$$\underset{b^* \notin \{a, a^*, b, b^*\}}{\operatorname{argmax}} \quad \frac{\langle e_{b^*}, e_{a^*} \rangle \langle e_{b^*}, e_b \rangle}{\langle e_{b^*}, e_a \rangle}$$

You can hack this result by taking logarithms. Works better.
And nobody knows why …

FastText (Bojanovski et al., 2017)

- **Syntax helps (sometimes)**
  - ‘cake’ vs. ‘layercake’
  - ‘desert’ vs. ‘dessert’
- **Extract all substrings of length range and embed them**
  {laye,ayer,yerc,erca,rcak,cake} vs. {cake}
  {cake,cak,ake,ca,ak,ke,c,a,k,e}
- **Extra embedding for entire word**
- Add them all and optimize …

Language models, now for real

Language models

- Skip-grams don't really model language, just context
- Chain rule of probability yields

$$p(\text{text}) = \prod_t p(w_t | [w_{t-1} \dots w_1])$$

- Tilted exponential families

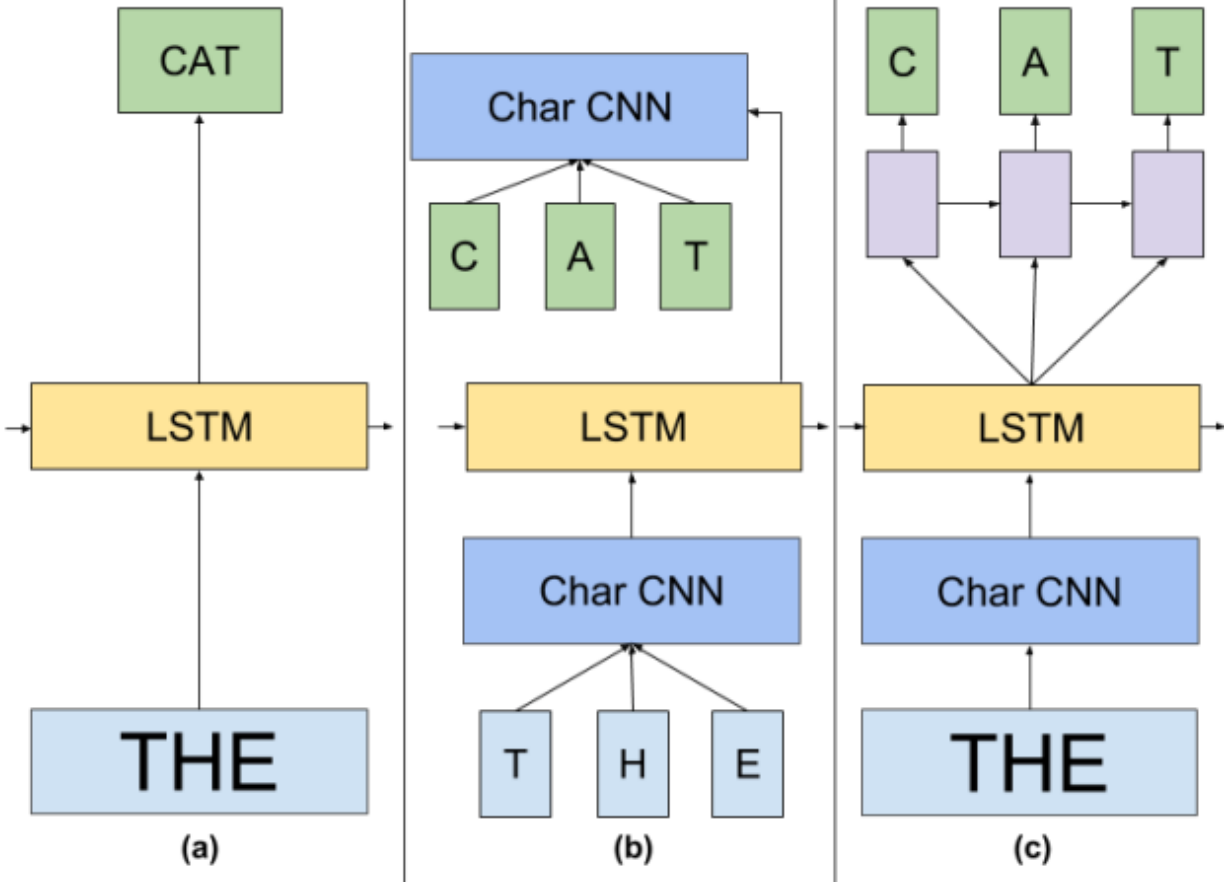
$$p(w_t|\text{past}) \propto p(w_t) \exp(f(w_t|\text{past}))$$

  need to normalize over all other words … too expensive
- **Hack** - Sample from p(w) and then compute

$$\text{softmax}(f(w_t|\text{past}), f(v_1|\text{past}), \dots f(v_k|\text{past}))$$

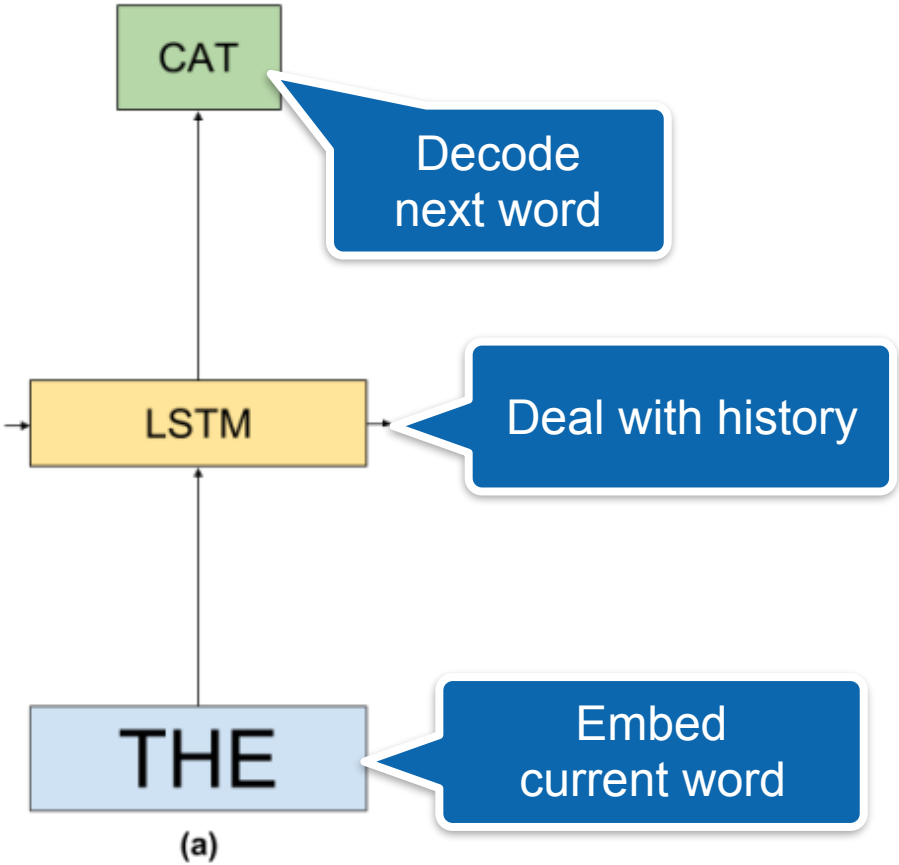
Solving the large vocabulary size problem

Jozefowicz et al., 2016



Solving the large vocabulary size problem

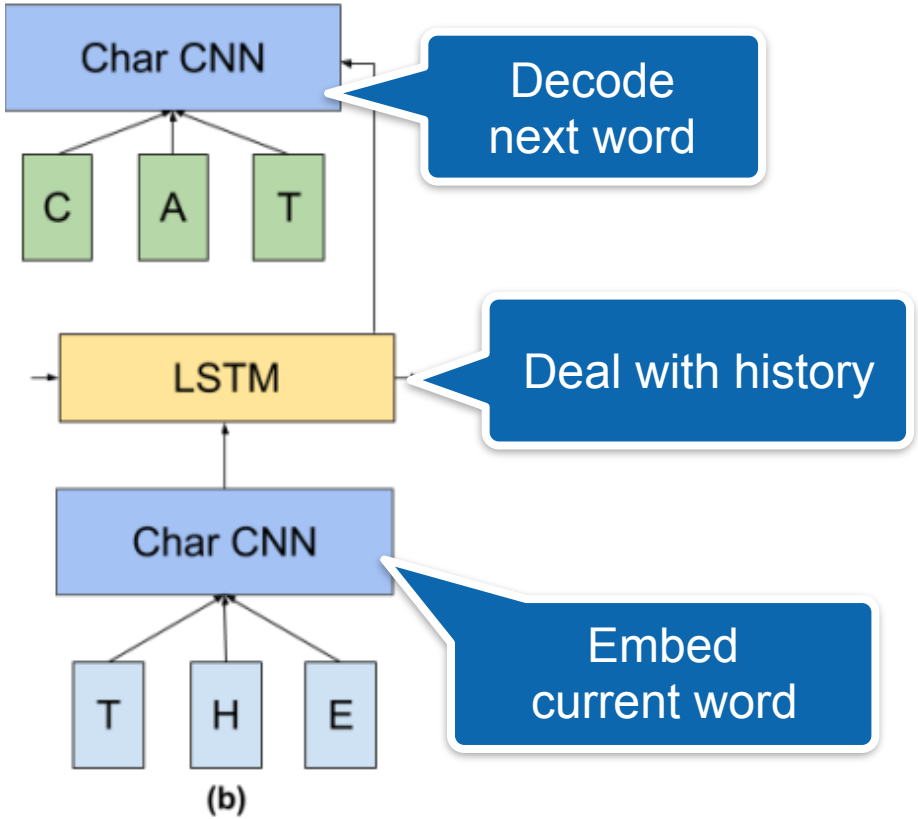
Jozefowicz et al., 2016



- Lots of memory for encoder and decoder
- Lots of computation for decoder

Solving the large vocabulary size problem

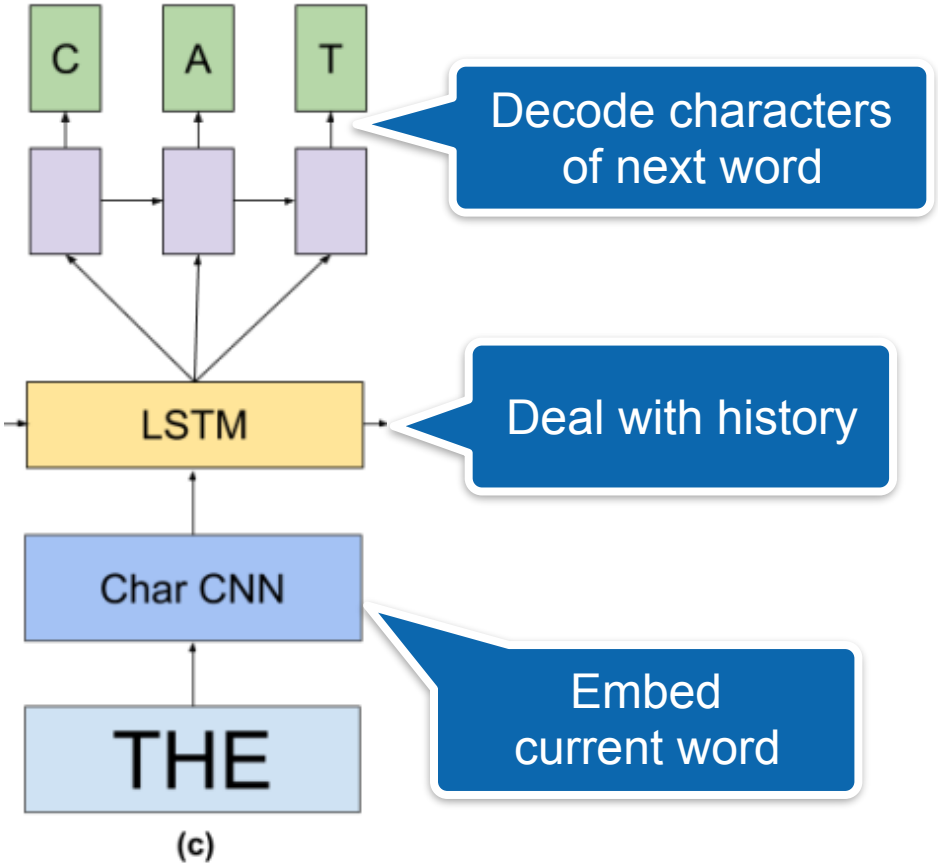
Jozefowicz et al., 2016



- Solves memory problem
- Not so great at sequence decoding (due to convolution)

Solving the large vocabulary size problem

Jozefowicz et al., 2016



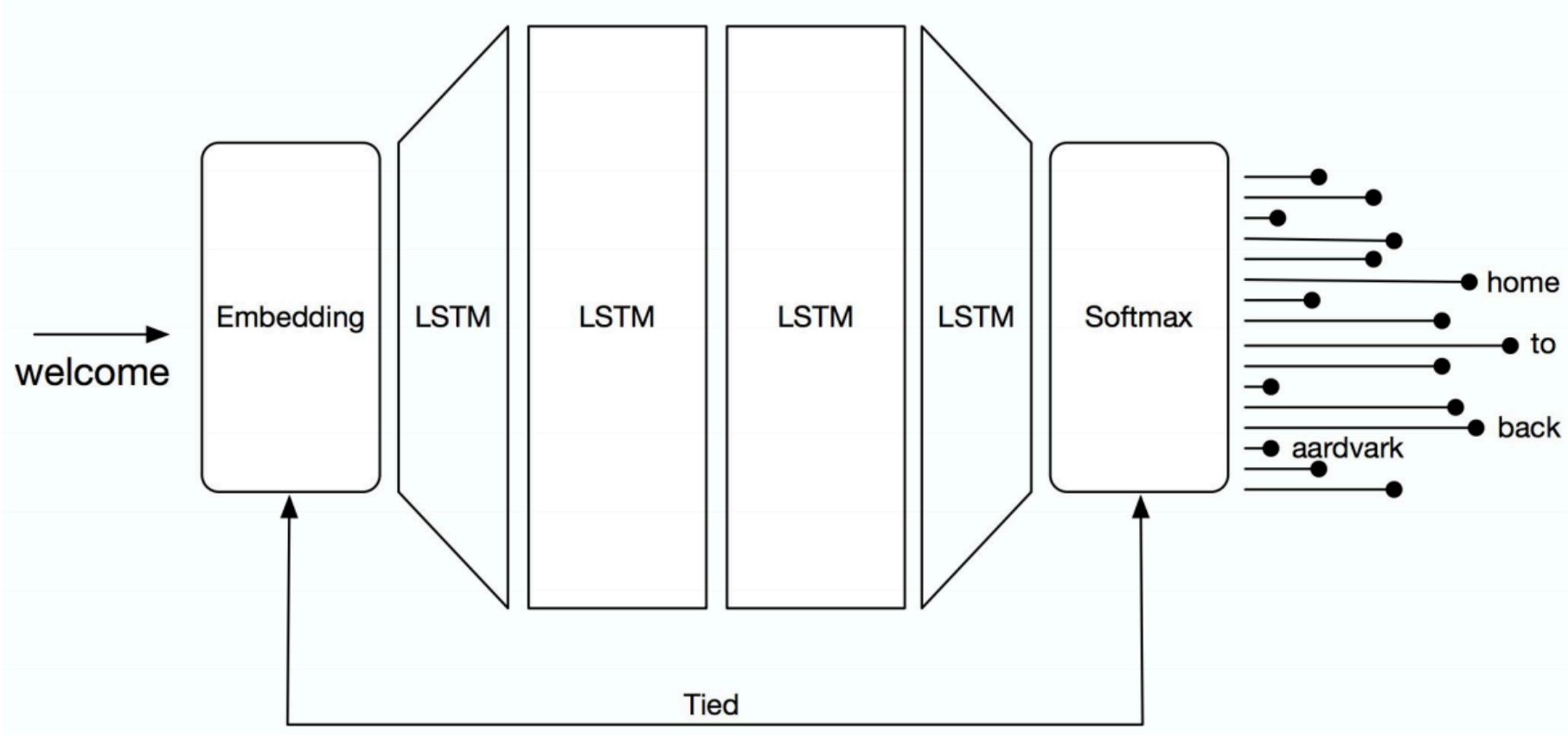
- Solves memory problem
- Good at decoding characters in sequence

Coming soon …

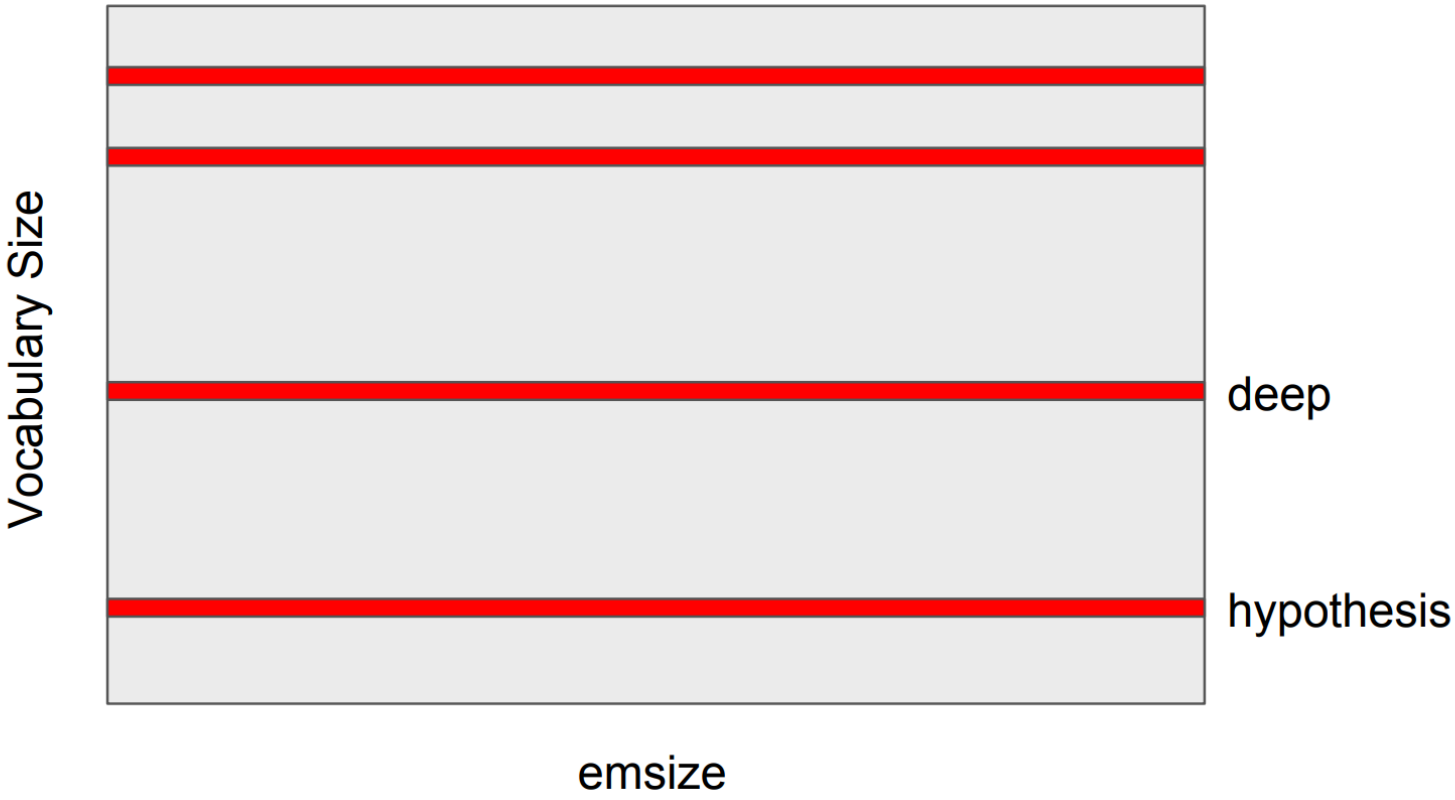
Bag of tricks for LSTMs (Merity et al. 2017)

- DropConnect for LSTM (hidden state)
  Important to fix DropConnect matrix for entire pass
- Parameter Averaging in LSTM
  Keep on training once converged but average weights
- Dropout of embedding weights
  Skip them entirely from embedding matrix
- Tie encoder and decoder
  Halves the bulk of parameters
- Activation regularization (magnitude and speed of change)

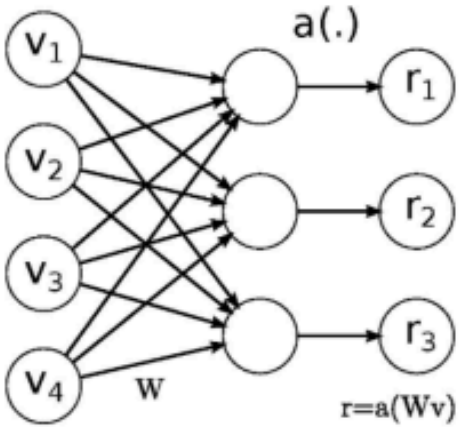
Parameter Tying



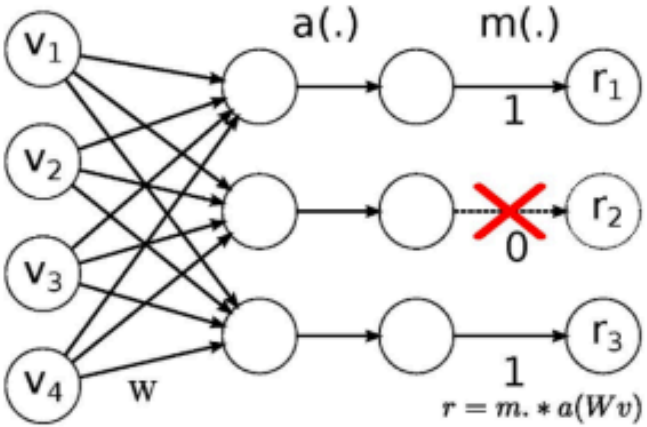
Embedding Dropout



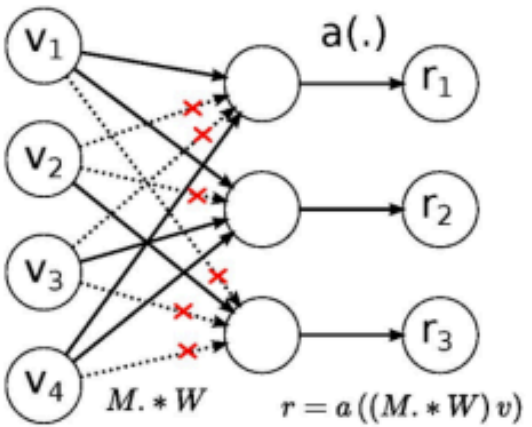
DropConnect on the weights of the LSTM



No-Drop Network



DropOut Network



DropConnect Network