MIDDLE EAST TECHNICAL UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

CENG 403
Introduction to Deep Learning

Week 13a

Sinan Kalkan

Feedforward through Vanilla RNN

Previously on CENG403

The Vanilla RNN Model

First time-step ($t = 1$):

$$\mathbf{h}_1 = tanh\left(W^{xh} \cdot \mathbf{x}_1 + W^{hh} \cdot \mathbf{h}_0\right)$$

$$\hat{\mathbf{y}}_1 = softmax\left(W^{hy} \cdot \mathbf{h}_1\right)$$

$$\mathcal{L}_1 = CE(\hat{\mathbf{y}}_1, \mathbf{y}_1)$$

In general:

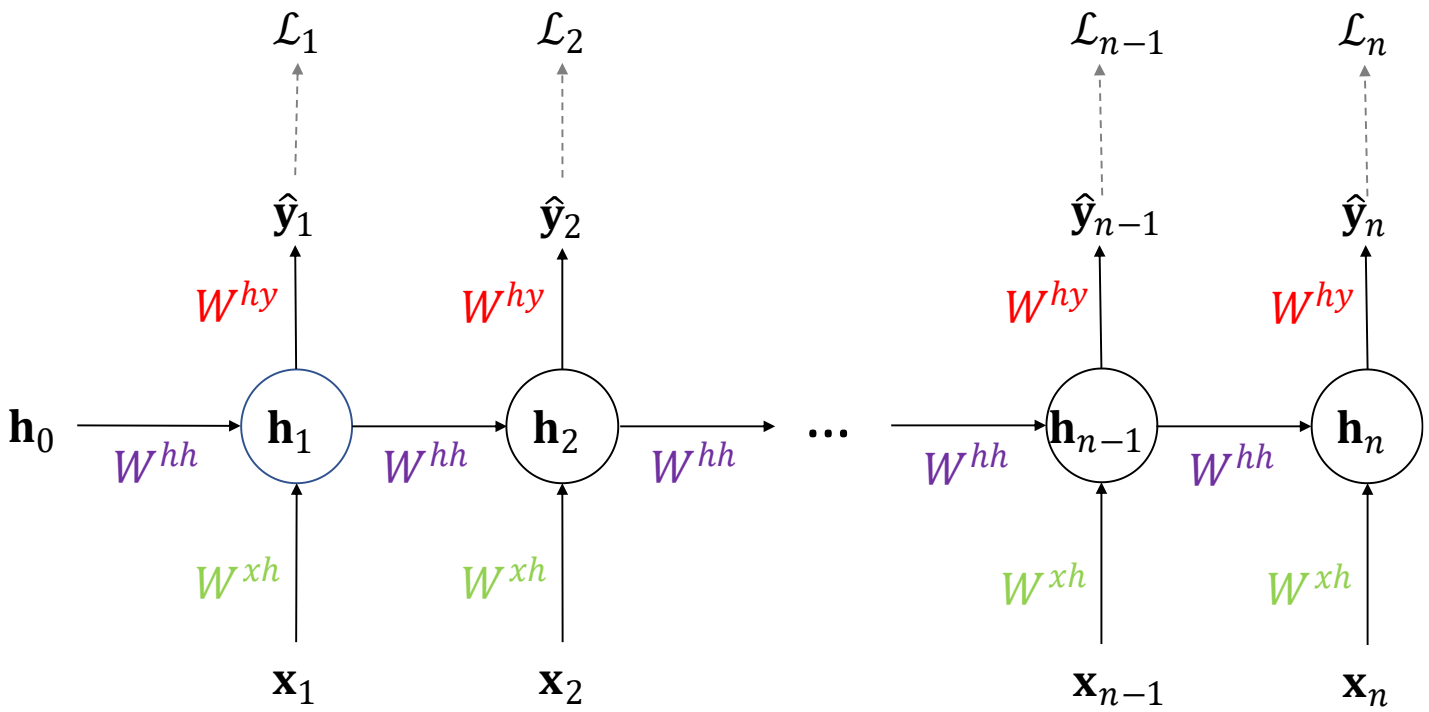
$$\mathbf{h}_t = tanh\left(W^{xh} \cdot \mathbf{x}_t + W^{hh} \cdot \mathbf{h}_{t-1}\right)$$

$$\hat{\mathbf{y}}_t = softmax\left(W^{hy} \cdot \mathbf{h}_t\right)$$

$$\mathcal{L}_t = CE(\hat{\mathbf{y}}_t, \mathbf{y}_t)$$

In total:

$$\mathcal{L} = \sum_t \mathcal{L}_t$$



Backpropagation through Vanilla RNN

Previously on CENG403

The Vanilla RNN Model

In general:

$$\mathbf{h}_t = tanh\left(W^{xh} \cdot \mathbf{x}_t + W^{hh} \cdot \mathbf{h}_{t-1}\right)$$

$$\hat{\mathbf{y}}_t = softmax\left(W^{hy} \cdot \mathbf{h}_t\right)$$

$$\mathcal{L}_t = CE(\hat{\mathbf{y}}_t, \mathbf{y}_t)$$

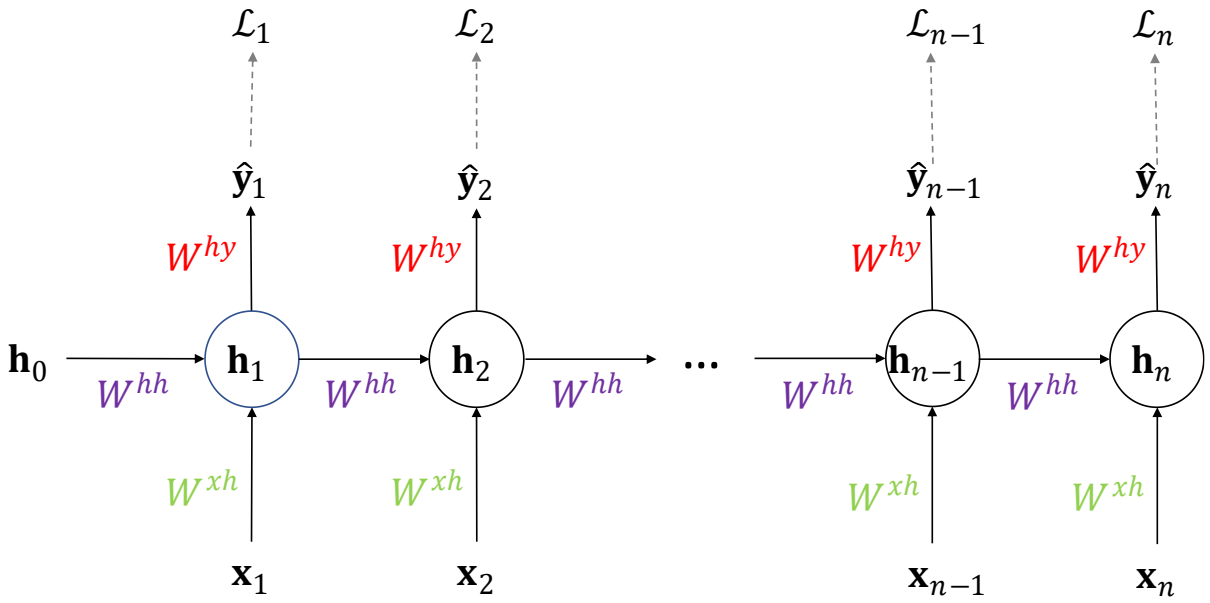
In total:

$$\mathcal{L} = \sum_t \mathcal{L}_t$$

$$\frac{\partial \mathcal{L}}{\partial W^{hy}} = ?$$

$$= \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{y}}_n}\frac{\partial \hat{\mathbf{y}}_n}{\partial W^{hy}} + \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{y}}_{n-1}}\frac{\partial \hat{\mathbf{y}}_{n-1}}{\partial W^{hy}} + \cdots + \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{y}}_1}\frac{\partial \hat{\mathbf{y}}_1}{\partial W^{hy}}$$

$$= \sum_{t=1..n} \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{y}}_t}\frac{\partial \hat{\mathbf{y}}_t}{\partial W^{hy}}$$



Backpropagation through Vanilla RNN

Previously on CENG403

$$\frac{\partial \mathcal{L}}{\partial W^{hh}} = ?$$

$$= \frac{\partial \mathcal{L}}{\partial \mathbf{h}_n}\frac{\partial \mathbf{h}_n}{\partial W^{hh}} + \frac{\partial \mathcal{L}}{\partial \mathbf{h}_{n-1}}\frac{\partial \mathbf{h}_{n-1}}{\partial W^{hh}} + \cdots + \frac{\partial \mathcal{L}}{\partial \mathbf{h}_1}\frac{\partial \mathbf{h}_1}{\partial W^{hh}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}_t} = \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{y}}_t}\frac{\partial \hat{\mathbf{y}}_t}{\partial \mathbf{h}_t} + \frac{\partial \mathcal{L}}{\partial \mathbf{h}_{t+1}}\frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{h}_t}$$

The Vanilla RNN Model

In general:

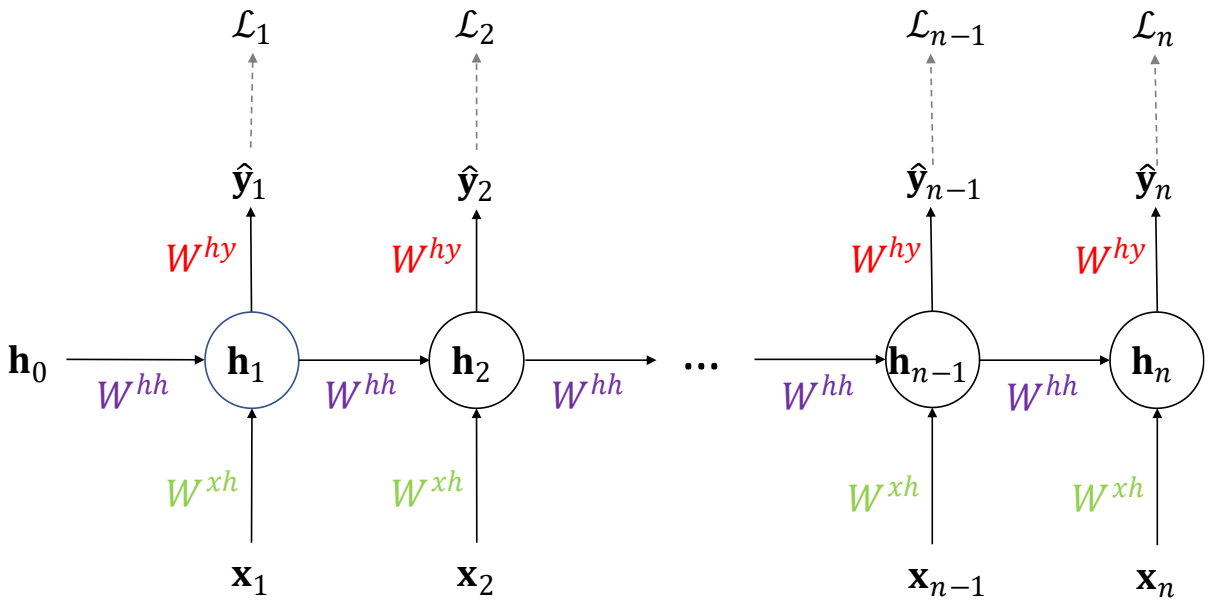
$$\mathbf{h}_t = tanh\left(W^{xh} \cdot \mathbf{x}_t + W^{hh} \cdot \mathbf{h}_{t-1}\right)$$

$$\hat{\mathbf{y}}_t = softmax\left(W^{hy} \cdot \mathbf{h}_t\right)$$

$$\mathcal{L}_t = CE(\hat{\mathbf{y}}_t, \mathbf{y}_t)$$

In total:

$$\mathcal{L} = \sum_t \mathcal{L}_t$$



Backpropagation through Vanilla RNN

Previously on CENG403

$$\frac{\partial \mathcal{L}}{\partial W^{xh}} = ?$$

$$= \frac{\partial \mathcal{L}}{\partial \mathbf{h}_n}\frac{\partial \mathbf{h}_n}{\partial W^{xh}} + \frac{\partial \mathcal{L}}{\partial \mathbf{h}_{n-1}}\frac{\partial \mathbf{h}_{n-1}}{\partial W^{xh}} + \cdots + \frac{\partial \mathcal{L}}{\partial \mathbf{h}_1}\frac{\partial \mathbf{h}_1}{\partial W^{xh}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}_t} = \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{y}}_t}\frac{\partial \hat{\mathbf{y}}_t}{\partial \mathbf{h}_t} + \frac{\partial \mathcal{L}}{\partial \mathbf{h}_{t+1}}\frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{h}_t}$$

(calculated before)

The Vanilla RNN Model

In general:

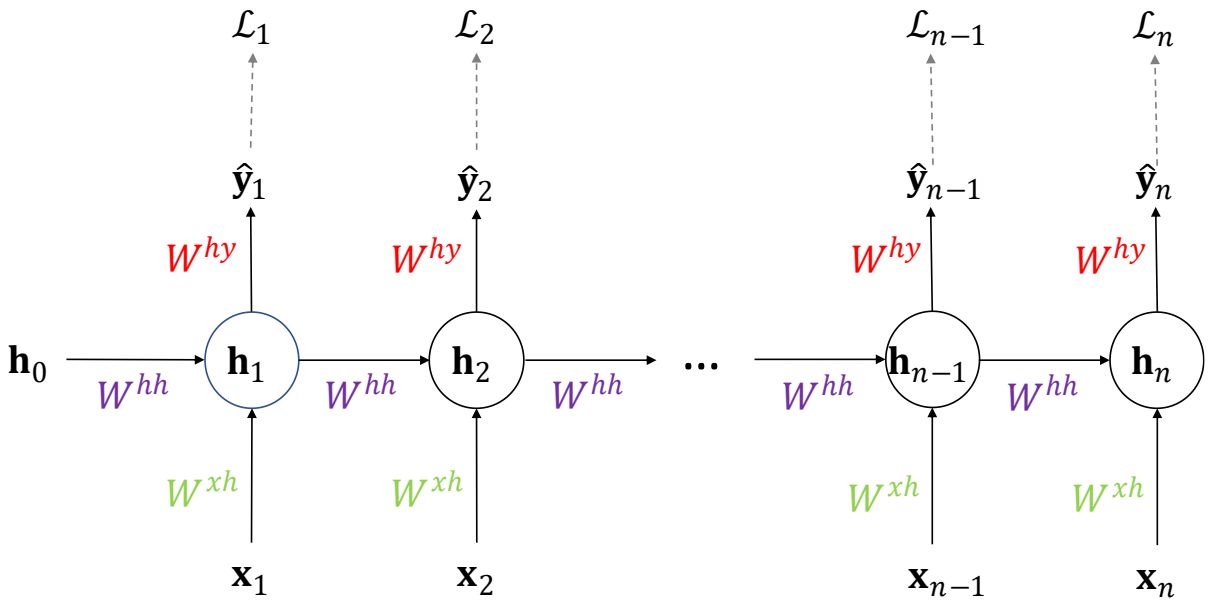
$$\mathbf{h}_t = tanh\left(W^{xh} \cdot \mathbf{x}_t + W^{hh} \cdot \mathbf{h}_{t-1}\right)$$

$$\hat{\mathbf{y}}_t = softmax\left(W^{hy} \cdot \mathbf{h}_t\right)$$

$$\mathcal{L}_t = CE(\hat{\mathbf{y}}_t, \mathbf{y}_t)$$

In total:

$$\mathcal{L} = \sum_t \mathcal{L}_t$$



Previously on CENG403

Initial hidden state

- We need to specify the initial activity state of all the hidden units.
- We could just fix these initial states to have some default value like 0.5.
- But it is better to treat the initial states as learned parameters.
- We learn them in the same way as we learn the weights.
  - Start off with an initial random guess for the initial states.
  - At the end of each training sequence, backpropagate through time all the way to the initial states to get the gradient of the error function with respect to each initial state.
  - Adjust the initial states by following the negative gradient.

Slide: Hinton

Previously on CENG403

Initializing parameters

- Since an unfolded RNN is a deep MLP, we can use Xavier initialization.

Exploding and vanishing gradients problem

Previously on CENG403

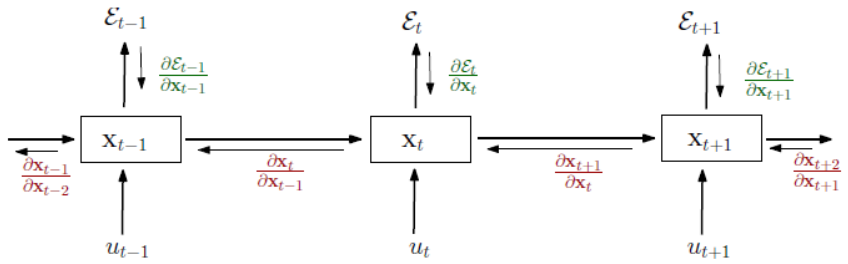
- Solution 1: Gradient clipping for exploding gradients:

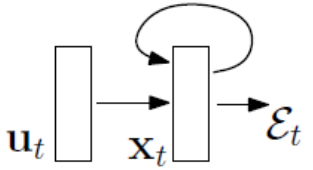
Algorithm 1 Pseudo-code for norm clipping

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|}\hat{\mathbf{g}}$
end if

- For vanishing gradients: Regularization term that penalizes changes in the magnitudes of back-propagated gradients

$$\Omega = \sum_k \Omega_k = \sum_k \left(\frac{\left\| \frac{\partial \mathcal{E}}{\partial \mathbf{x}_{k+1}} \frac{\partial \mathbf{x}_{k+1}}{\partial \mathbf{x}_k} \right\|}{\left\| \frac{\partial \mathcal{E}}{\partial \mathbf{x}_{k+1}} \right\|} - 1 \right)^2$$





On the difficulty of training recurrent neural networks

Razvan Pascanu PASCANUR@IRO.UMONTREAL.CA
Université de Montréal, 2920, chemin de la Tour, Montréal, Québec, Canada, H3T 1J8

Tomas Mikolov T.MIKOLOV@GMAIL.COM
Speech@FIT, Brno University of Technology, Brno, Czech Republic

Yoshua Bengio YOSHUA.BENGIO@UMONTREAL.CA
Université de Montréal, 2920, chemin de la Tour, Montréal, Québec, Canada, H3T 1J8

2012

Previously on CENG403

LSTM in detail

- We first compute an activation vector, a:

$$a = W_x x_t + W_h h_{t-1} + b$$

- Split this into four vectors of the same size:

$$a_f, a_i, a_g, a_o \leftarrow a$$

- We then compute the values of the gates:

$$f = \sigma(a_f) \quad i = \sigma(a_i) \quad g = \tanh(a_g) \quad o = \sigma(a_o)$$

where σ is the sigmoid.

- The next cell state c_t and the hidden state h_t:

$$c_t = f \odot c_{t-1} + i \odot g$$
$$h_t = o \odot \tanh(c_t)$$

where $\odot$ is the element-wise product of vectors

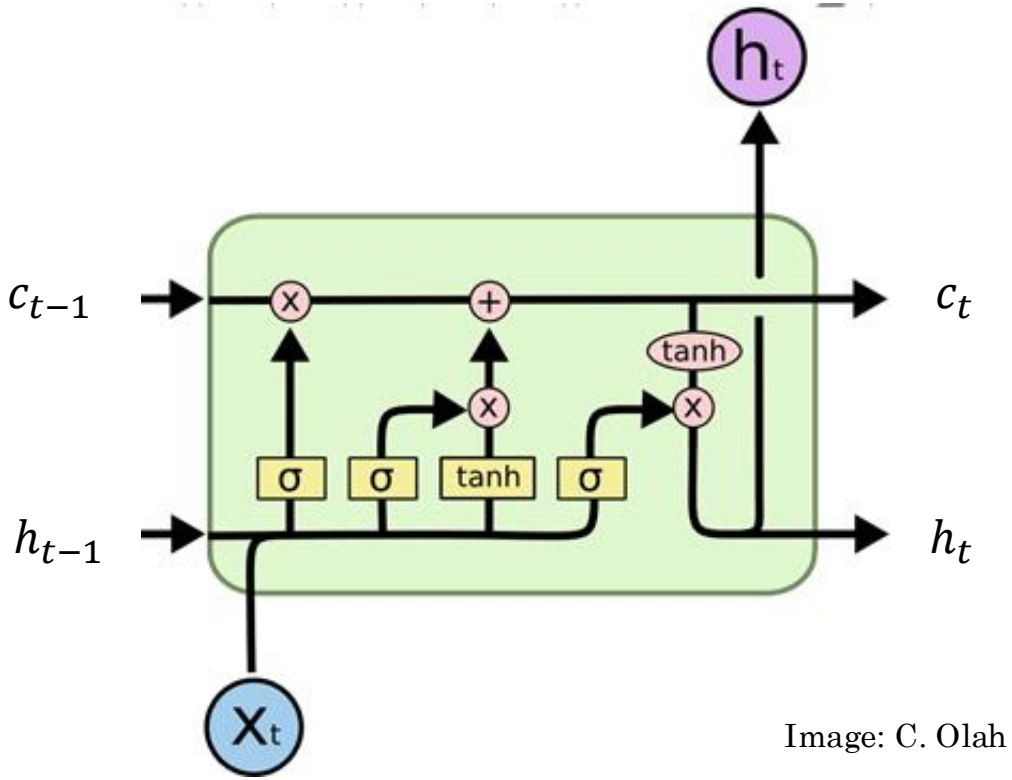


Image: C. Olah

Alternative formulation:

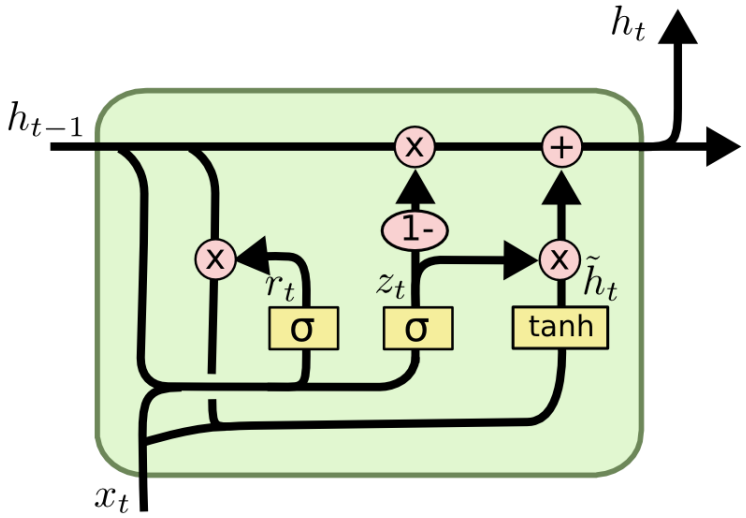
$$i_t = g(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$
$$f_t = g(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$
$$o_t = g(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$

Eqs: Karpathy

Previously on CENG403

LSTM Variants #3: Gated Recurrent Units

- Changes:
  - No explicit memory; memory = hidden output
  - Z = memorize new and forget old



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

Previously on CENG403

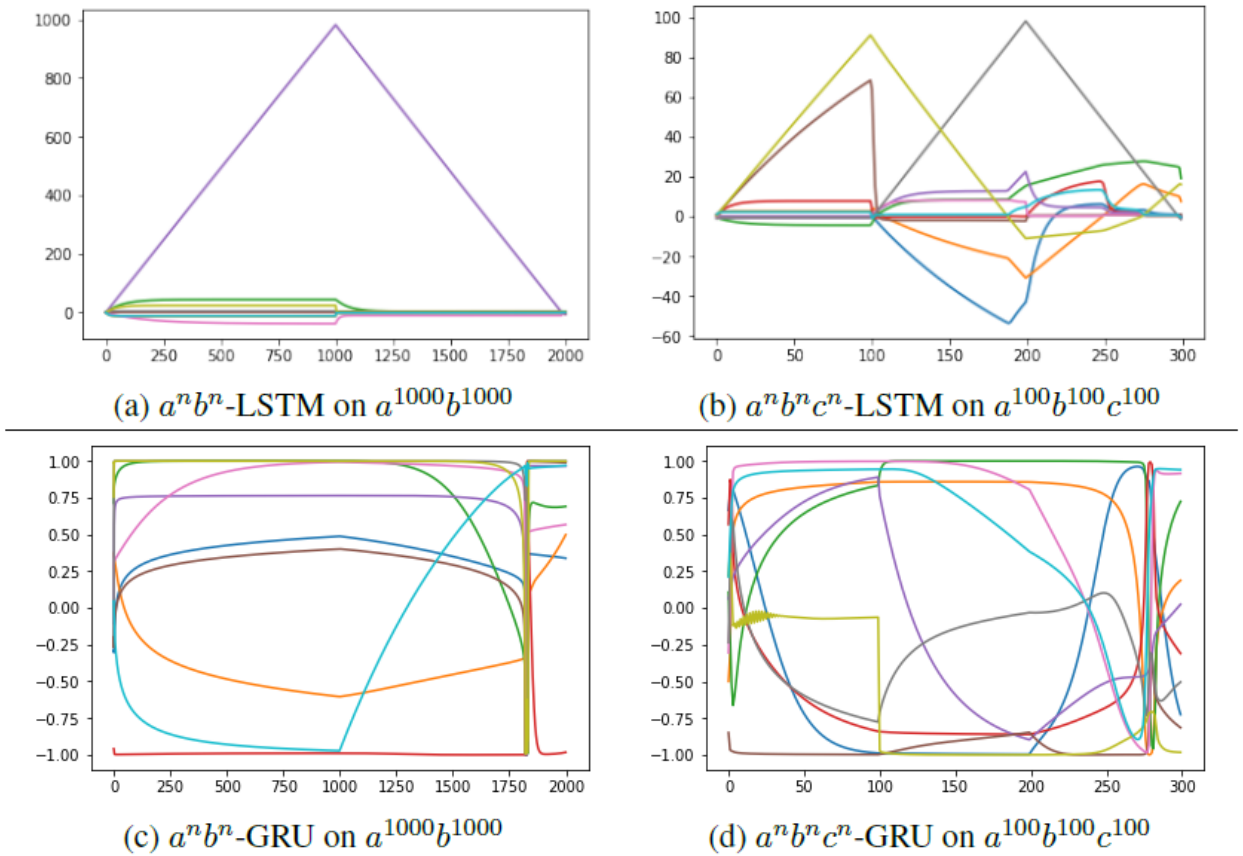
LSTM vs. GRU

Accepted as a short paper in ACL 2018

On the Practical Computational Power of Finite Precision RNNs for Language Recognition

Gail Weiss Technion, Israel
Yoav Goldberg Bar-Ilan University, Israel
Eran Yahav Technion, Israel

{sgailw,yahave}@cs.technion.ac.il
yogo@cs.biu.ac.il



(a) $a^n b^n$-LSTM on $a^{1000} b^{1000}$

(b) $a^n b^n c^n$-LSTM on $a^{100} b^{100} c^{100}$

(c) $a^n b^n$-GRU on $a^{1000} b^{1000}$

(d) $a^n b^n c^n$-GRU on $a^{100} b^{100} c^{100}$

Figure 1: Activations — c for LSTM and h for GRU — for networks trained on $a^n b^n$ and $a^n b^n c^n$. The LSTM has clearly learned to use an explicit counting mechanism, in contrast with the GRU.

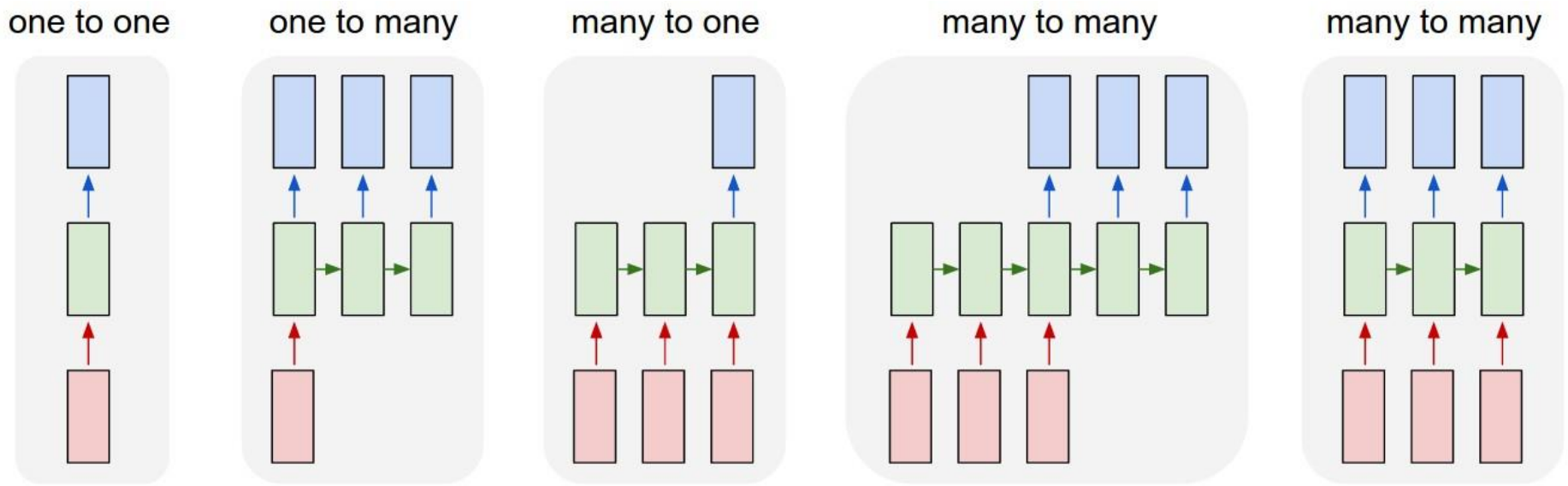
Today

- Recurrent Neural Networks (RNNs)
  - Char-level language modeling
  - Word-level language modeling
  - Image captioning
  - Machine translation

Example: Character-level Text Modeling

Character-level Text Modeling

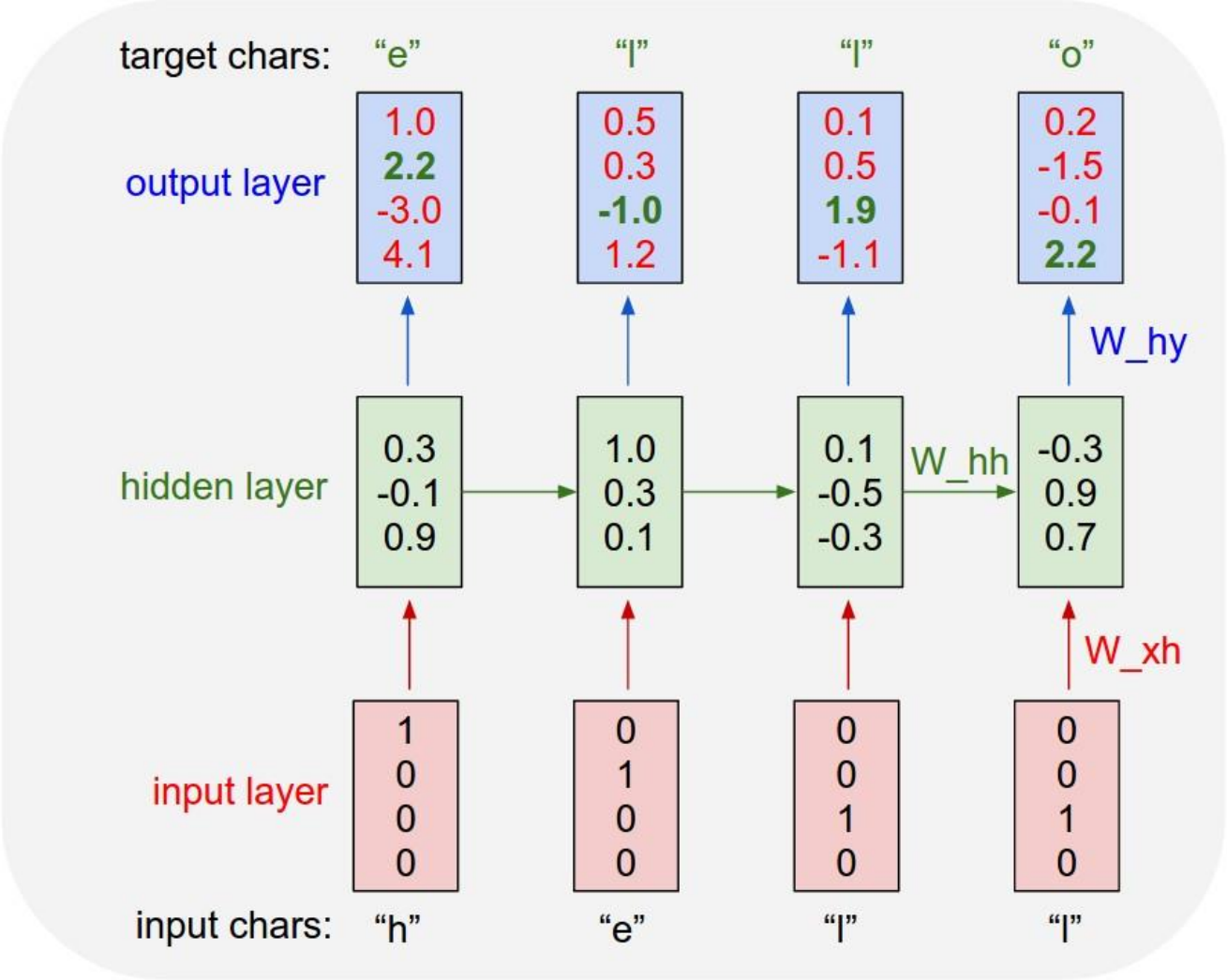
- Problem definition: Find c_{n+1} given $c_1, c_2, \ldots, c_n$.
- Modelling (autoregressive language modeling):
$$p(c_{n+1} \mid c_n, \ldots, c_1)$$
- In general, we just take the last N characters:
$$p(c_{n+1} \mid c_n, \ldots, c_{n-(N-1)})$$
- Learn $p(c_{n+1} = 'a' \mid 'Ankar')$ from data such that
$$p(c_{n+1} = 'a' \mid 'Ankar') > p(c_{n+1} = 'o' \mid 'Ankar')$$



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

A simple scenario

- Alphabet: h, e, l, o
- Text to train to predict: “hello”



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

Sampling: Greedy

- Greedy sampling: Take the most likely word at each step

```
from numpy import array
from numpy import argmax

# greedy decoder
def greedy_decoder(data):
    # index for largest probability each row
    return [argmax(s) for s in data]

# define a sequence of 10 words over a vocab of 5 words
data = [[0.1, 0.2, 0.3, 0.4, 0.5],
        [0.5, 0.4, 0.3, 0.2, 0.1],
        [0.1, 0.2, 0.3, 0.4, 0.5],
        [0.5, 0.4, 0.3, 0.2, 0.1],
        [0.1, 0.2, 0.3, 0.4, 0.5],
        [0.5, 0.4, 0.3, 0.2, 0.1],
        [0.1, 0.2, 0.3, 0.4, 0.5],
        [0.5, 0.4, 0.3, 0.2, 0.1],
        [0.1, 0.2, 0.3, 0.4, 0.5],
        [0.5, 0.4, 0.3, 0.2, 0.1]]
data = array(data)
# decode sequence
result = greedy_decoder(data)
print(result)
```

Running the example outputs a sequence of integers that could then be mapped back to words in the vocabulary.

```
[4, 0, 4, 0, 4, 0, 4, 0, 4, 0]
```

Code: https://machinelearningmastery.com/beam-search-decoder-natural-language-processing/

Sampling: Beam Search

- What happens if we want *k* most likely sequences instead of one?
- Beam search: Consider *k* most likely words at each step, and expand search.

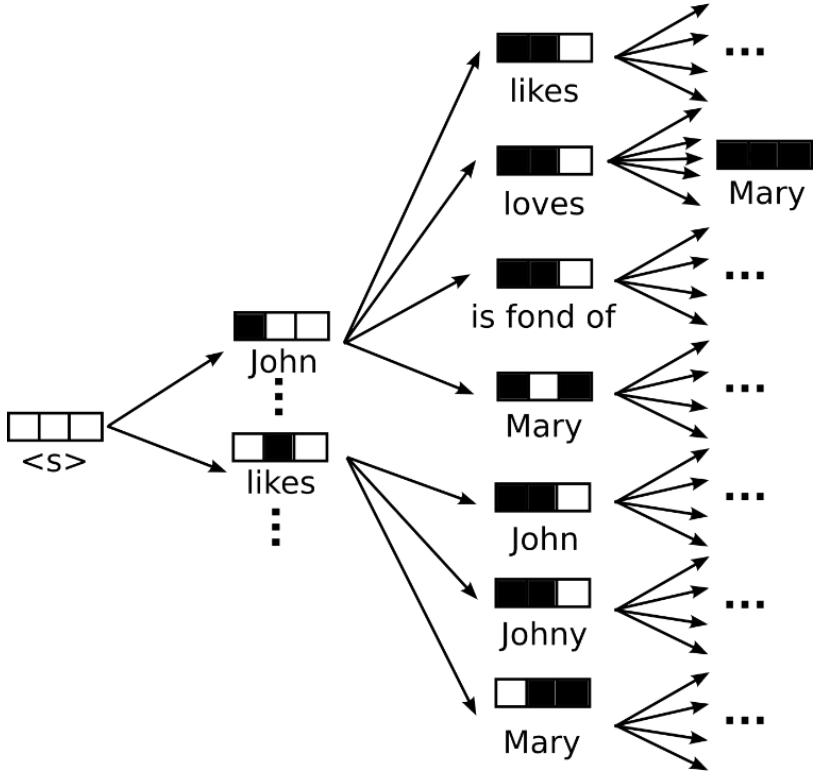


Figure: http://mttalks.ufal.ms.mff.cuni.cz/index.php

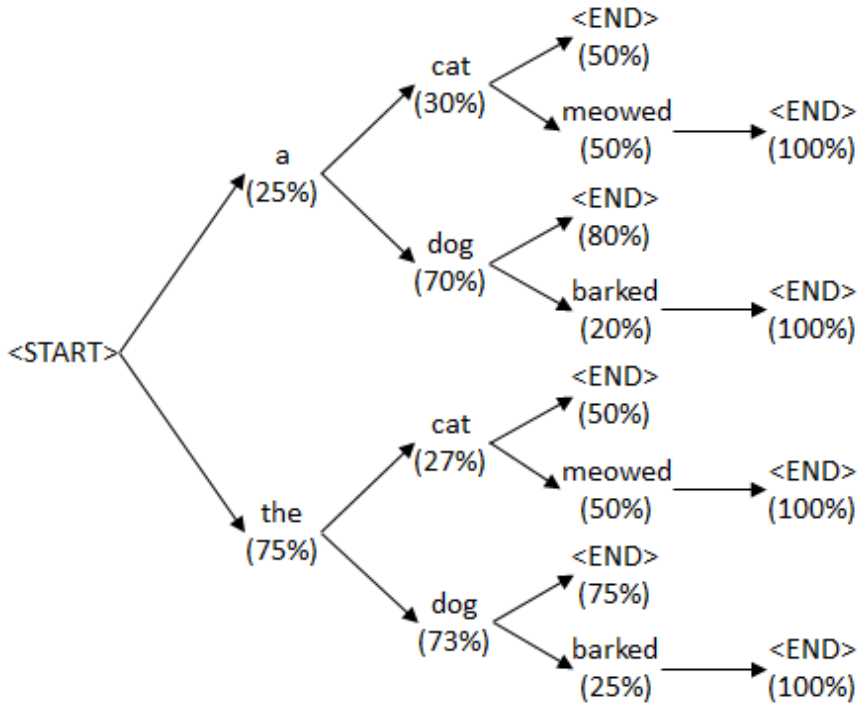


Figure: https://geekyisawesome.blogspot.com.tr/2016/10/using-beam-search-to-generate-most.html

Sampling: Beam Search

- Beam search: Consider *k* most likely words at each step, and expand search.

  (take log for numerical stability; take –log() for minimizing the score)

```python
from math import log
from numpy import array
from numpy import argmax

# beam search
def beam_search_decoder(data, k):
    sequences = [[list(), 0.0]]
    # walk over each step in sequence
    for row in data:
        all_candidates = list()
        # expand each current candidate
        for i in range(len(sequences)):
            seq, score = sequences[i]
            for j in range(len(row)):
                candidate = [seq + [j], score - log(row[j])]
                all_candidates.append(candidate)
        # order all candidates by score
        ordered = sorted(all_candidates, key=lambda tup:tup[1])
        # select k best
        sequences = ordered[:k]
    return sequences
```

```python
# define a sequence of 10 words over a vocab of 5 words
data = [[0.1, 0.2, 0.3, 0.4, 0.5],
        [0.5, 0.4, 0.3, 0.2, 0.1],
        [0.1, 0.2, 0.3, 0.4, 0.5],
        [0.5, 0.4, 0.3, 0.2, 0.1],
        [0.1, 0.2, 0.3, 0.4, 0.5],
        [0.5, 0.4, 0.3, 0.2, 0.1],
        [0.1, 0.2, 0.3, 0.4, 0.5],
        [0.5, 0.4, 0.3, 0.2, 0.1],
        [0.1, 0.2, 0.3, 0.4, 0.5],
        [0.5, 0.4, 0.3, 0.2, 0.1]]
data = array(data)
# decode sequence
result = beam_search_decoder(data, 3)
# print result
for seq in result:
    print(seq)
```

```
[[4, 0, 4, 0, 4, 0, 4, 0, 4, 0], 6.931471805599453]
[[4, 0, 4, 0, 4, 0, 4, 0, 4, 1], 7.154615356913663]
[[4, 0, 4, 0, 4, 0, 4, 0, 3, 0], 7.154615356913663]
```

Code: https://machinelearningmastery.com/beam-search-decoder-natural-language-processing/

More on beam search

- Beam search is applied during inference.
- With modifications on the training procedure, it is possible to use it during training as well.

Sequence-to-Sequence Learning as Beam-Search Optimization

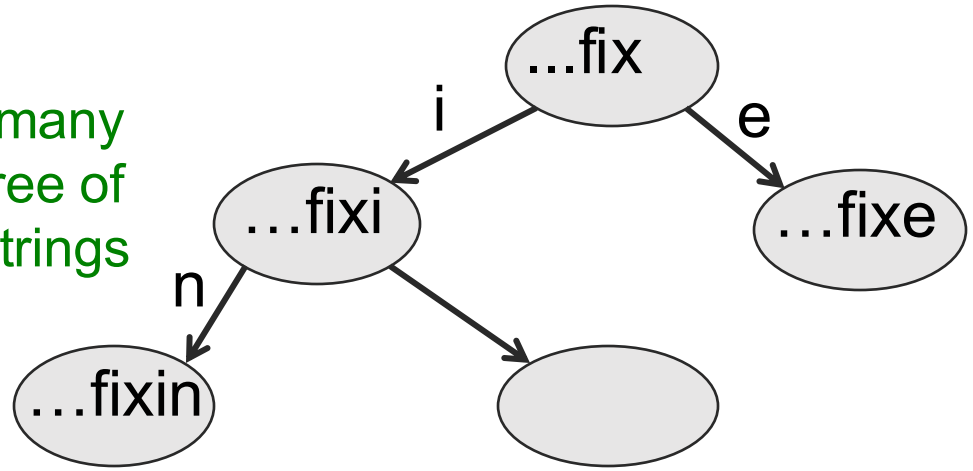
Sam Wiseman and **Alexander M. Rush**
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA, USA
{swiseman,srush}@seas.harvard.edu

2016

https://arxiv.org/abs/1606.02960

A sub-tree in the tree of all character strings

There are exponentially many nodes in the tree of all character strings of length N.



In an RNN, each node is a hidden state vector. The next character must transform this to a new node.

- If the nodes are implemented as hidden states in an RNN, different nodes can share structure because they use distributed representations.
- The next hidden representation needs to depend on the conjunction of the current character and the current hidden representation.

Slide: Hinton

Modeling text: Advantages of working with characters

- The web is composed of character strings.
- Any learning method powerful enough to understand the world by reading the web ought to find it trivial to learn which strings make words (this turns out to be true, as we shall see).
- Pre-processing text to get words is a big hassle
  - What about morphemes (prefixes, suffixes etc)
  - What about subtle effects like “sn” words?
  - What about New York?
  - What about Finnish?

Slide: Hinton

Sample predictions
(when trained on the works of Shakespeare):

- 3-level RNN with 512 hidden nodes in each layer

```
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.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.
```


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

Sample predictions
(when trained on Wikipedia):

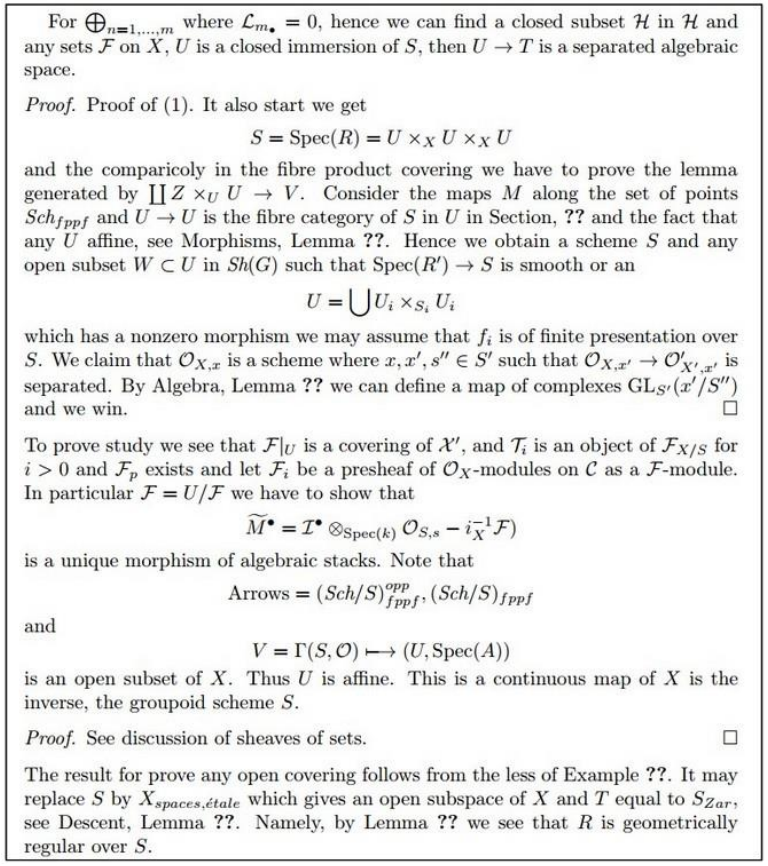
- Using LSTM

```
Naturalism and decision for the majority of Arab countries' capitalide was grounded
by the Irish language by [[John Clair]], [[An Imperial Japanese Revolt]], associated
with Guangzham's sovereignty. His generals were the powerful ruler of the Portugal
in the [[Protestant Immineners]], which could be said to be directly in Cantonese
Communication, which followed a ceremony and set inspired prison, training. The
emperor travelled back to [[Antioch, Perth, October 25|21]] to note, the Kingdom
of Costa Rica, unsuccessful fashioned the [[Thrales]], [[Cynth's Dajoard]], known
in western [[Scotland]], near Italy to the conquest of India with the conflict.
Copyright was the succession of independence in the slop of Syrian influence that
was a famous German movement based on a more popular servicious, non-doctrinal
and sexual power post. Many governments recognize the military housing of the
[[Civil Liberalization and Infantry Resolution 265 National Party in Hungary]],
that is sympathetic to be to the [[Punjab Resolution]]
(PJS)[http://www.humah.yahoo.com/guardian.
cfm/7754800786d17551963s89.htm Official economics Adjoint for the Nazism, Montgomery
was swear to advance to the resources for those Socialism's rule,
was starting to signing a major tripad of aid exile.]]
```


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

Sample predictions
(when trained on Latex documents):

- Using multi-layer LSTM



For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the compariciol y in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, ??? and the fact that any U affine, see Morphisms, Lemma ??. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example ??. It may replace S by $X_{spaces,étale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma ??. Namely, by Lemma ?? we see that R is geometrically regular over S.

Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ *(by the formal open covering* X *and a single map* $\underline{Proj}_X(\mathcal{A}) = \mathrm{Spec}(B)$ *over* U *compatible with the complex*

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ *is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If* T *is surjective we may assume that* T *is connected with residue fields of* S*. Moreover there exists a closed subspace* $Z \subset X$ *of* X *where* U *in* X' *is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem*

(1) f *is locally of finite type. Since* $S = \mathrm{Spec}(R)$ *and* $Y = \mathrm{Spec}(R)$.

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\ldots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{\mathcal{X},\ldots,0}$.

Lemma 0.2. *Let* X *be a locally Noetherian scheme over* S, $E = \mathcal{F}_{X/S}$. *Set* $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. *Since* $\mathcal{I}^n \subset \mathcal{I}^n$ *are nonzero over* $i_0 \leq \mathfrak{p}$ *is a subset of* $\mathcal{J}_{n,0} \circ \overline{A}_2$ *works.*

Lemma 0.3. *In Situation ??. Hence we may assume* $\mathfrak{q}' = 0$.

Proof. We will use the property we see that $\mathfrak{p}$ is the mext functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F-algebra where δ_{n+1} is a scheme over S. □

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

From Ilya Sutskever (using a variant of character-level RNN)

He was elected President during the Revolutionary War and forgave Opus Paul at Rome. The regime of his crew of England, is now Arab women's icons in and the demons that use something between the characters‘ sisters in lower coil trains were always operated on the line of the ephemerable street, respectively, the graphic or other facility for deformation of a given proportion of large segments at RTUS). The B every chord was a "strongly cold internal palette pour even the white blade.”

Some completions produced by the model

- Sheila thrunges (most frequent)
- People thrunge (most frequent next character is space)
- Shiela, Thrungelini del Rey (first try)
- The meaning of life is literary recognition. (6^{th} try)

- The meaning of life is the tradition of the ancient human reproduction: it is less favorable to the good boy for when to remove her bigger.
  (one of the first 10 tries for a model trained for longer).

What does it know?

- It knows a huge number of words and a lot about proper names, dates, and numbers.
- It is good at balancing quotes and brackets.
  - It can count brackets: none, one, many
- It knows a lot about syntax but its very hard to pin down exactly what form this knowledge has.
  - Its syntactic knowledge is not modular.
- It knows a lot of weak semantic associations
  - E.g. it knows Plato is associated with Wittgenstein and cabbage is associated with vegetable.

Example: Word-level Text Modeling

Word-level Text Modeling

- Problem definition: Find ω_{n+1} given $\omega_1, \omega_2, \ldots, \omega_n$.
- Modelling:

$$p(\omega_{n+1} \mid \omega_n, \ldots, \omega_1)$$

- In general, we just take the last N words:

$$p(\omega_{n+1} \mid \omega_n, \ldots, \omega_{n-(N-1)})$$

- Learn $p(\omega_{n+1} = 'Turkey' \mid 'Ankara\ is\ the\ capital\ of\ ')$ from data such that:

$$p(\omega_{n+1} = 'Turkey' \mid 'Ankara\ is\ the\ capital\ of\ ') \quad > \quad p(\omega_{n+1} = 'UK' \mid 'Ankara\ is\ the\ capital\ of\ ')$$

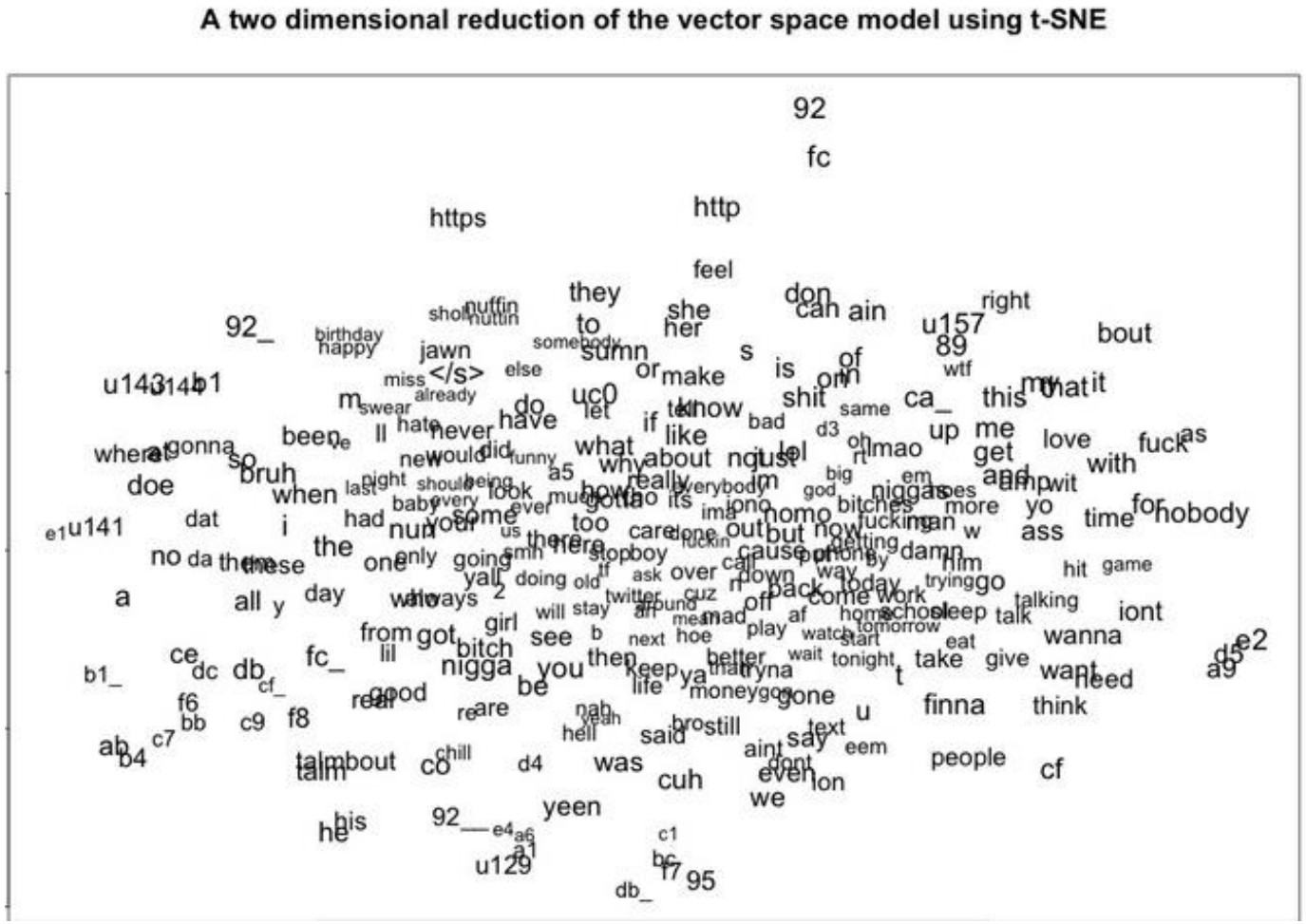
A handicap

- The number of characters is low enough to handle without doing anything extra.
  - English has 26 characters.

- The situation is very different for words.
  - English has ~ 170,000 different words!

- This increases dimensionality and makes it difficult to capture “semantics”.

- Solution: Map words to a lower dimensional space, a.k.a. word embedding (word2vec).



Word Embedding (word2vec)

Fig: http://www.languagejones.com/blog-1/2015/11/1/word-embedding

Why do we embed words?

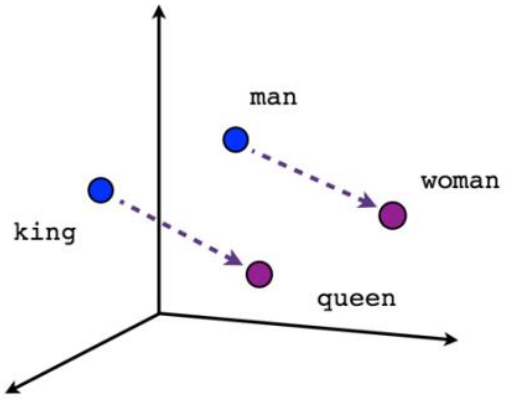
- 1-of-n encoding is not suitable to learn from
  - It is sparse
  - Similar words have different representations
  - Compare this with the pixel-based representation of images: Similar images/objects have similar pixels
- Embedding words in a map allows
  - Encoding them with fixed-length vectors
  - “Similar” words having similar representations
  - Allows complex reasoning between words:
    - king - man + woman = queen

EXPRESSION	NEAREST TOKEN
Paris - France + Italy	Rome
bigger - big + cold	colder
sushi - Japan + Germany	bratwurst
Cu - copper + gold	Au
Windows - Microsoft + Google	Android
Montral Canadiens - Montreal + Toronto	Toronto Maple Leafs

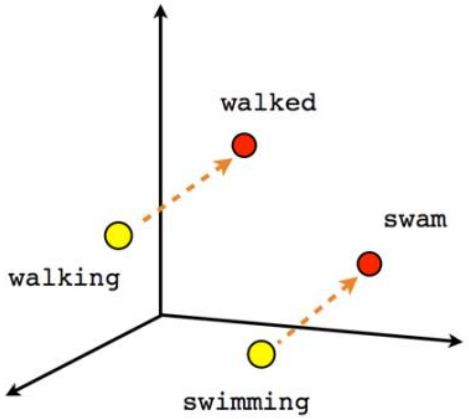
Table 1: Mikolov et al. [3] showcase simple additive properties of their word embeddings.

Table: https://devblogs.nvidia.com/parallelforall/understanding-natural-language-deep-neural-networks-using-torch/

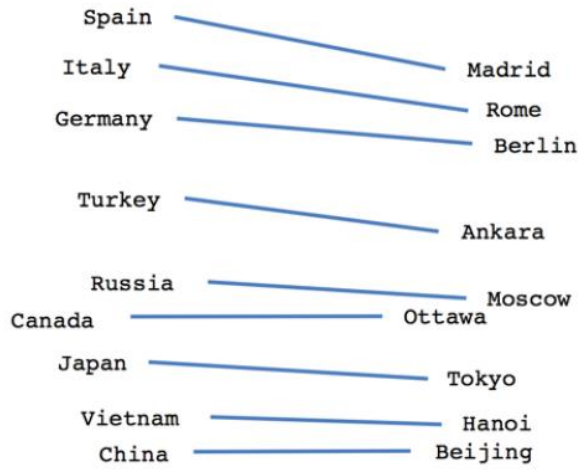
More examples



Male-Female



Verb tense



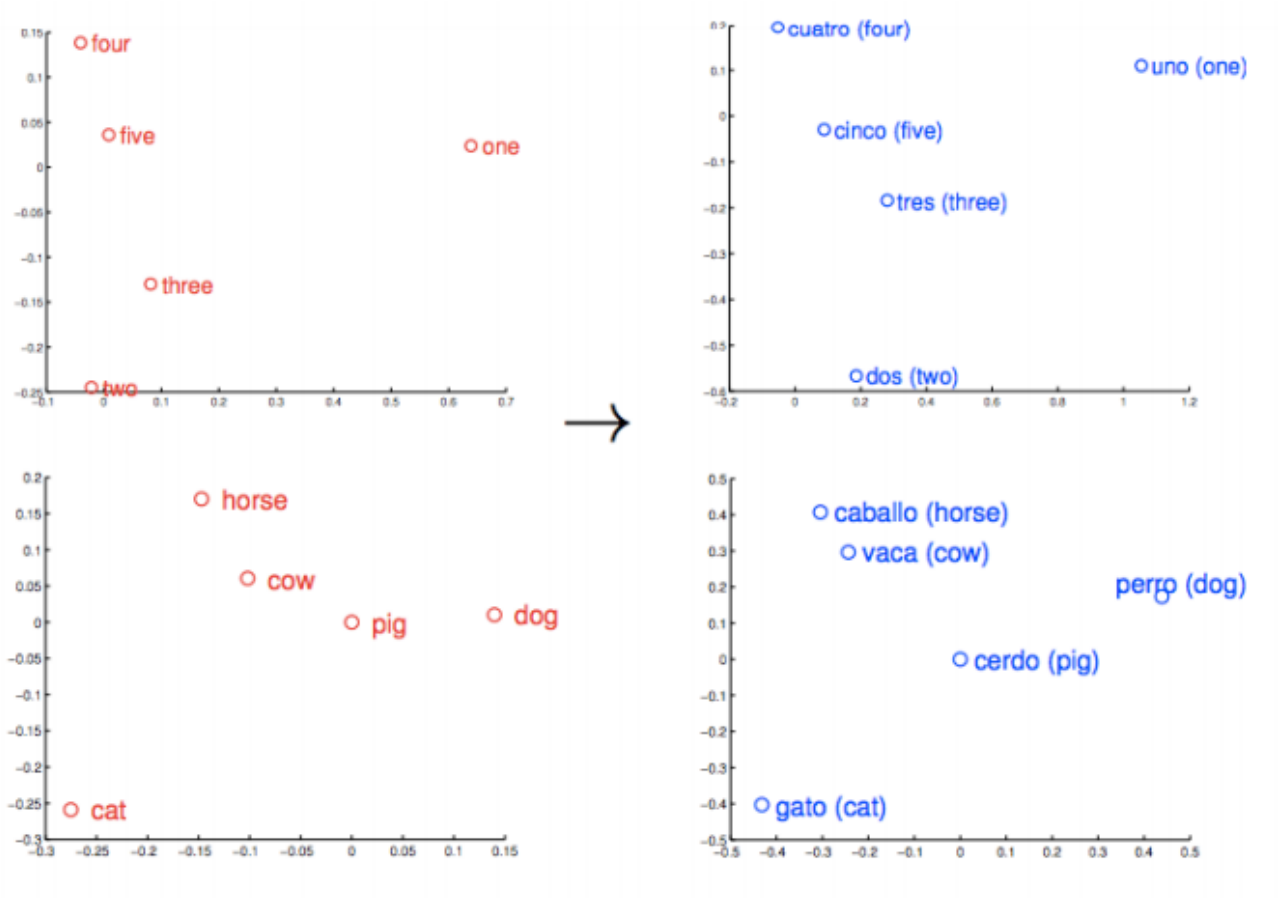
Country-Capital

More examples

- Geopolitics: *Iraq - Violence = Jordan*
- Distinction: *Human - Animal = Ethics*
- *President - Power = Prime Minister*
- *Library - Books = Hall*

http://deeplearning4j.org/word2vec

More examples



http://deeplearning4j.org/word2vec

word2vec

- “Similarity” to Sweden (cosine distance between their vector representations)

Word	Cosine distance
norway	0.760124
denmark	0.715460
finland	0.620022
switzerland	0.588132
belgium	0.585835
netherlands	0.574631
iceland	0.562368
estonia	0.547621
slovenia	0.531408

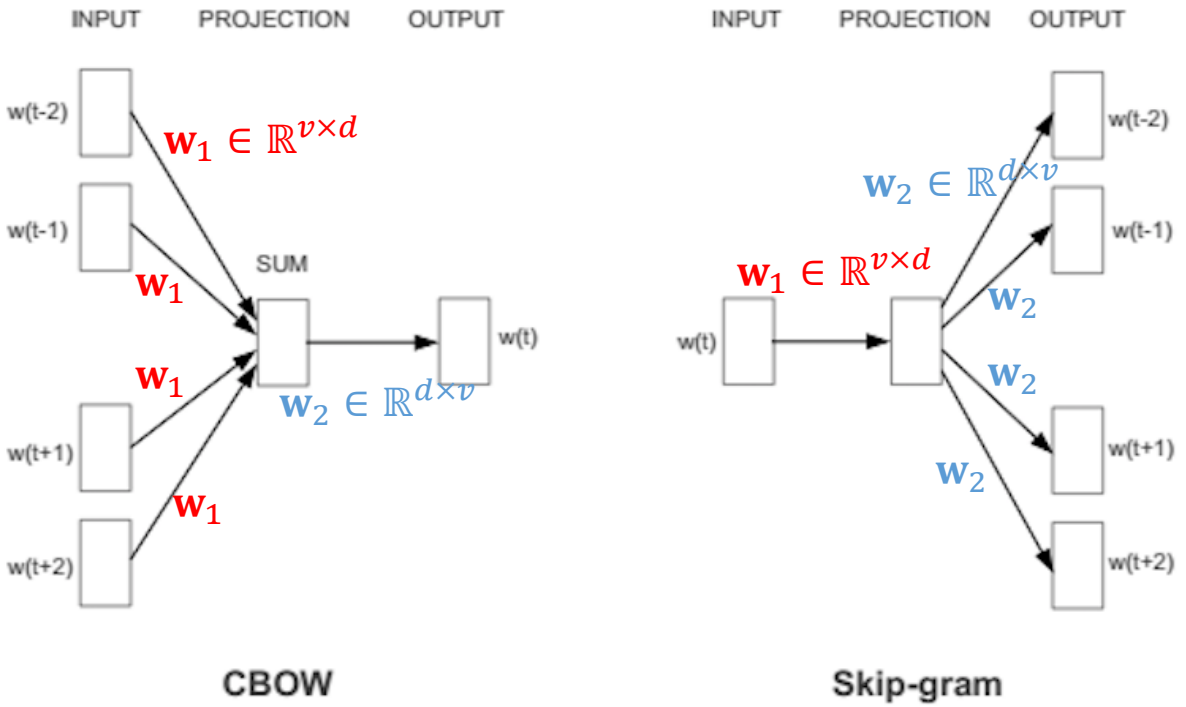
http://deeplearning4j.org/word2vec

Two different ways to train

1. Using context to predict a target word (~ continuous bag-of-words)
2. Using word to predict a target context (skip-gram)

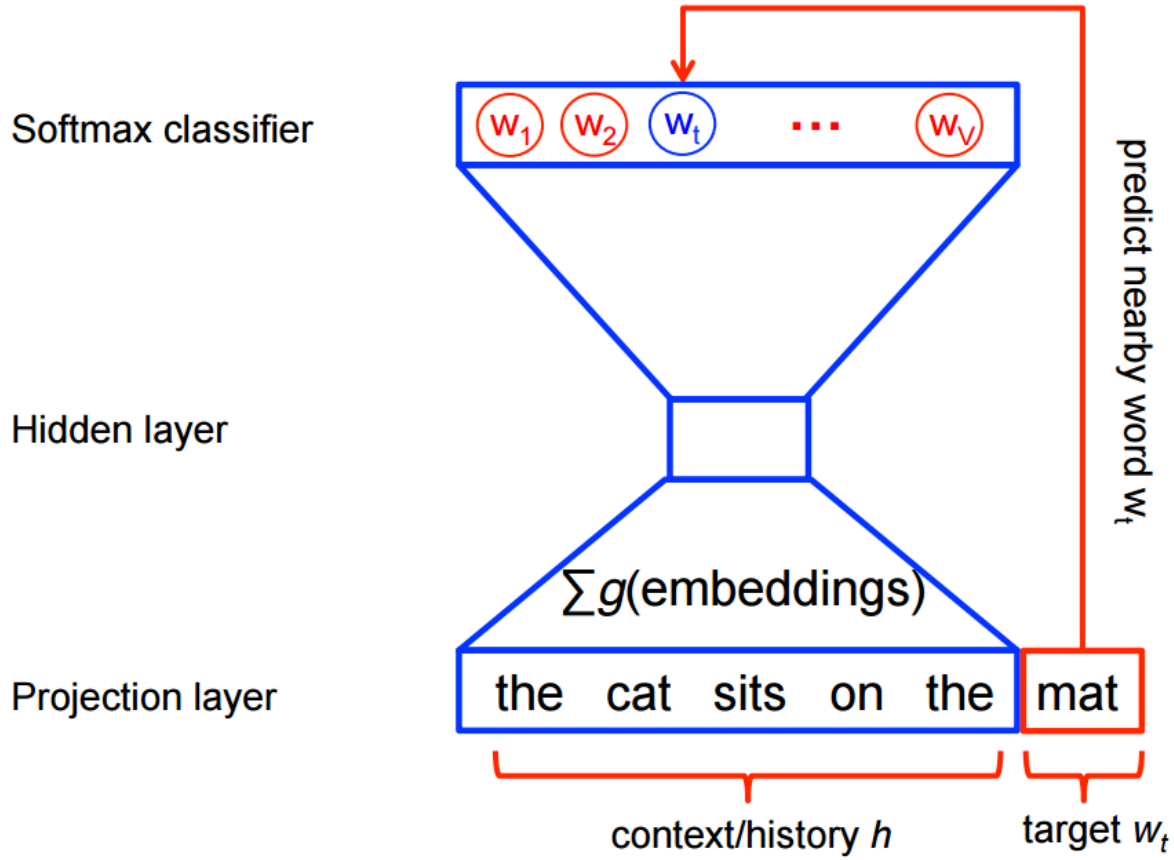
- If the vector for a word cannot predict its context, learning adjusts the mapping to the vector space
- Since similar words should predict the same or similar contexts, their vector representations should end up being similar

v: vocabulary size
d: hidden representation size



Two different ways to train

1.Using context to predict a target word (~ continuous bag-of-words)



https://www.tensorflow.org/versions/r0.7/tutorials/word2vec/index.html

Two different ways to train

2.Using word to predict a target context (skip-gram)

- Given a sentence:

  the quick brown fox jumped over the lazy dog

- For each word, take context to be

  (N-words to the left, N-words to the right)

- If N = 1 (context, word):

  ([the, brown], quick), ([quick, fox], brown), ([brown, jumped], fox), ...

Two different ways to train

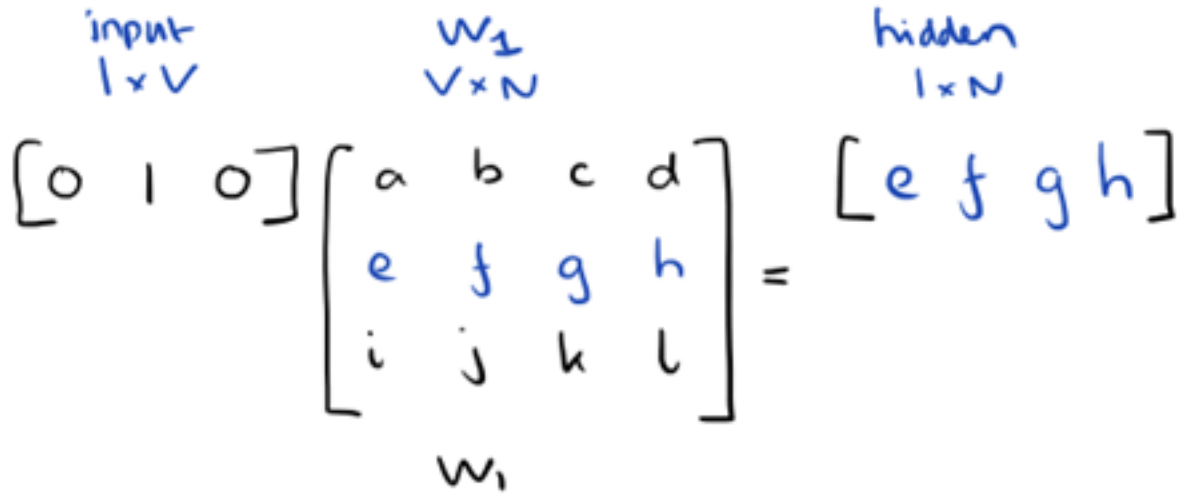
2. Using word to predict a target context (skip-gram)

Window Size	Text	Skip-grams
2	[The **wide** road shimmered] in the hot sun.	wide, the wide, road wide, shimmered
	The [wide road **shimmered** in the] hot sun.	shimmered, wide shimmered, road shimmered, in shimmered, the
	The wide road shimmered in [the hot **sun**].	sun, the sun, hot
3	[The **wide** road shimmered in] the hot sun.	wide, the wide, road wide, shimmered wide, in
	[The wide road **shimmered** in the hot] sun.	shimmered, the shimmered, wide shimmered, road shimmered, in shimmered, the shimmered, hot
	The wide road shimmered [in the hot **sun**].	sun, in sun, the sun, hot

https://www.tensorflow.org/tutorials/text/word2vec

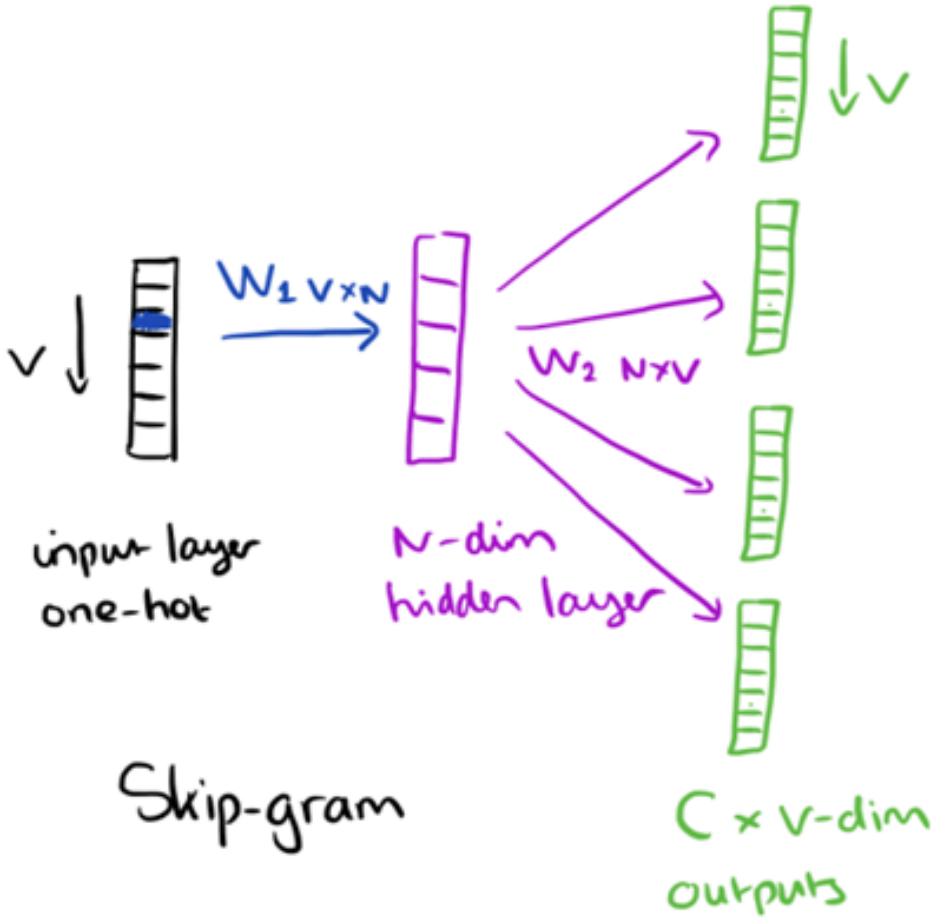
Note that the weight matrix is a look-up table

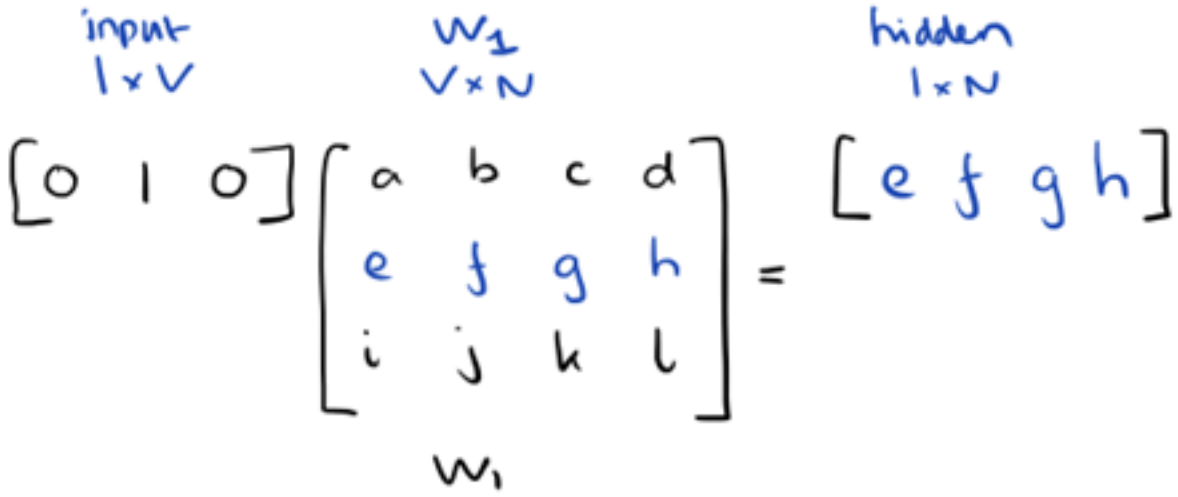
- In both approaches, the weight matrix is used as follows:



https://medium.com/@zafaralibagh6/a-simple-word2vec-tutorial-61e64e38a6a1

Note that the weight matrix is a look-up table





https://medium.com/@zafaralibagh6/a-simple-word2vec-tutorial-61e64e38a6a1