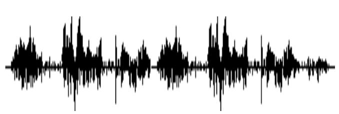
# Language Modeling

Hung-yi Lee
李宏毅

# Language modeling

- Language model: Estimated the probability of word sequence
  - Word sequence: $w_1, w_2, w_3, \ldots., w_n$
  - $P(w_1, w_2, w_3, \ldots., w_n)$
- Application: speech recognition
  - Different word sequence can have the same pronunciation



recognize speech
or
wreck a nice beach

If P(recognize speech)
>P(wreck a nice beach)

Output =
“recognize speech”

- Application: sentence generation

# N-gram

P("wreck a nice beach")
=P(wreck|START)P(a|wreck)
P(nice|a)P(beach|nice)

- How to estimate $P(w_1, w_2, w_3, \ldots., w_n)$
- Collect a large amount of text data as training data
  - However, the word sequence $w_1, w_2, \ldots., w_n$ may not appear in the training data
- *N-gram language model*: $P(w_1, w_2, w_3, \ldots., w_n) = P(w_1|START)P(w_2|w_1) \ldots\ldots P(w_n|w_{n-1})$ ← 2-gram
  - E.g. Estimate P(beach|nice) from training data

$$P(\text{beach}|\text{nice}) = \frac{C(nice\ beach)}{C(nice)}$$

$C(nice\ beach)$ ← Count of "nice beach"

$C(nice)$ ← Count of "nice"

- It is easy to generalize to 3-gram, 4-gram ……

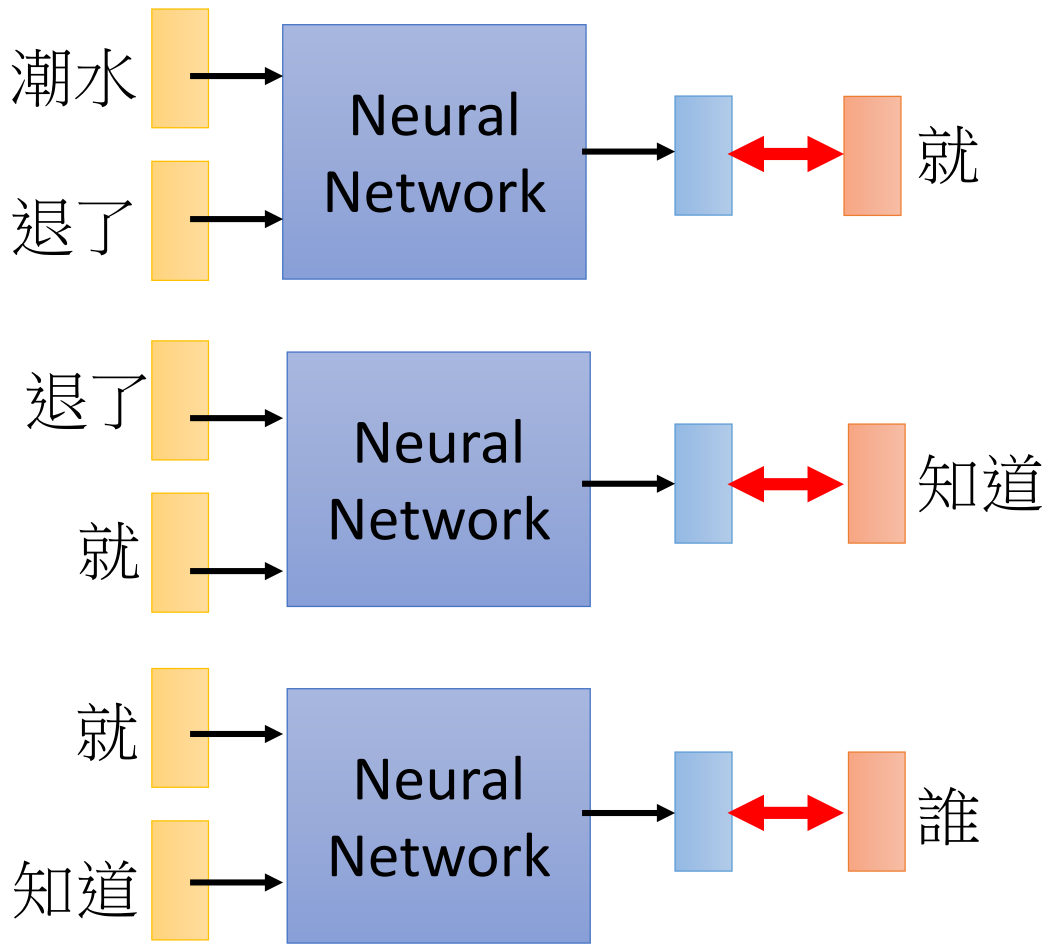
# NN-based LM

- Training:

Collect data:

| |
|---|
| 潮水 退了 就 知道 誰 …<br>不爽 不要 買 …<br>公道價 八萬 一 …<br>……… |

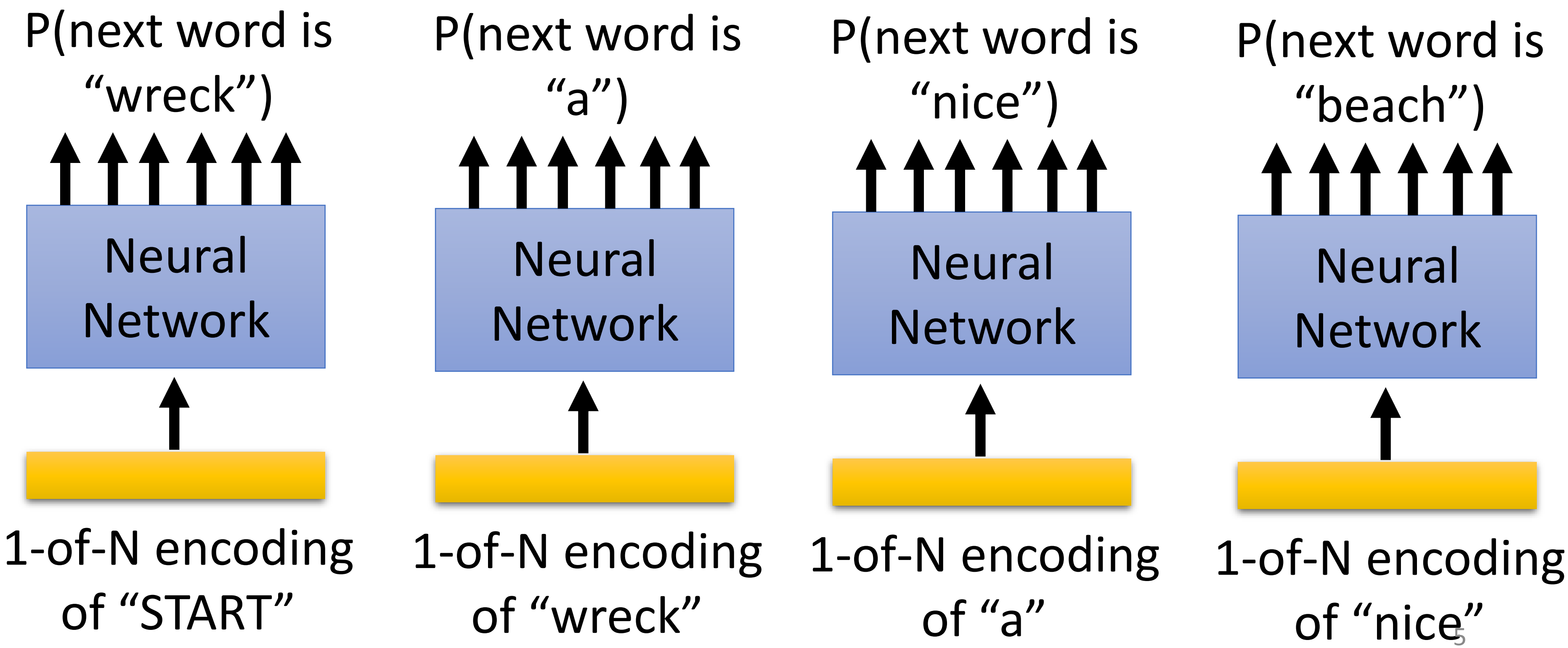
**Minimizing cross entropy**



# NN-based LM

P("wreck a nice beach")
=P(wreck|START)P(a|wreck)P(nice|a)P(beach|nice)

P(b|a): the probability of NN predicting the next word.

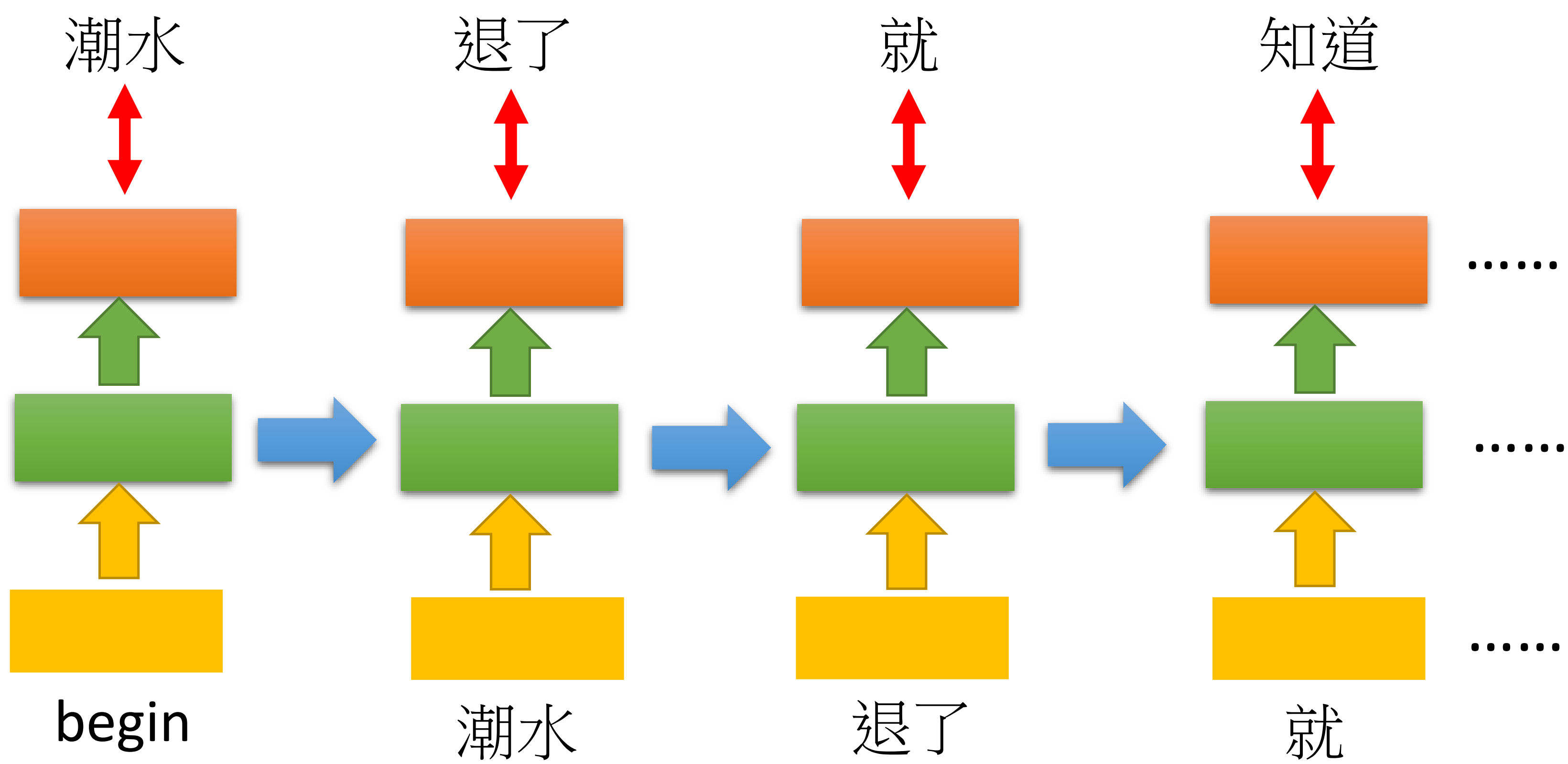


# RNN-based LM

- Training

Collect data:

| |
|---|
| 潮水 退了 就 知道 誰 … |
| 不爽 不要 買 … |
| 公道價 八萬 一 … |
| ……… |

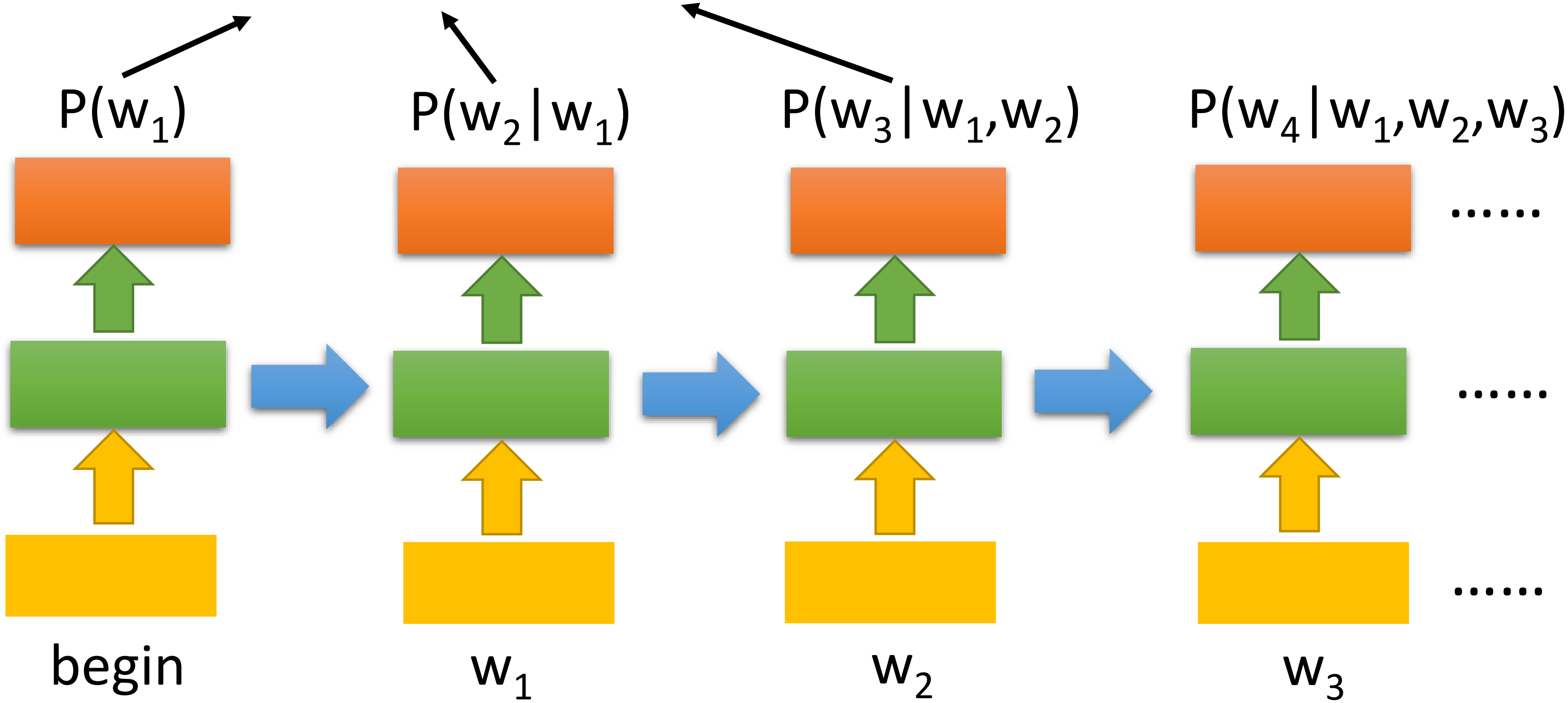


# RNN-based LM

- Modeling long-term information
- People also use Deep RNN or LSTM

- To compute $P(w_1, w_2, w_3, \ldots., w_n)$ by RNN

$$P(w_1, w_2, w_3, \ldots., w_n)$$
$$= P(w_1)P(w_1|w_2)P(w_3|w_1,w_2) \ldots\ldots P(w_n|w_1,w_2 \ldots w_{n-1})$$



# Challenge of N-gram

- The estimated probability is not accurate.
  - Especially when we consider n-gram with large n
  - Because of data sparsity
    - Large model, not sufficient data

Training Data:

The dog ran ……
The cat jumped ……

P( jumped | the, dog ) = ~~0~~ 0.0001

P( ran | the, cat ) = ~~0~~ 0.0001

Give some small probability

This is called **language model smoothing**.

# Matrix Factorization

Recommendation System:
History as customer,
vocabulary as product ……

history

Vocabulary

| | dog $h^1$ | cat $h^2$ | …… | child |
|---|---|---|---|---|
| ran $v^1$ | 0.2 $n_{11}$ | 0.3 | | 0.1 |
| jumped $v^2$ | 0 | 0.2 $n_{22}$ | | 0.1 |
| cried $v^3$ | 0 | 0 | | 0.3 |
| laughed $v^4$ | 0 | 0 | | 0.3 |
| …… | | | | |

Not observed

P( jumped | cat )

$v^i$, $h^j$ are vectors to be learned

$$n_{12} = v^1 \cdot h^2$$
$$n_{21} = v^2 \cdot h^1 \ \ldots$$

Minimizing

$$L = \sum_{(i,j)} \left(v^i \cdot h^j - n_{ij}\right)^2$$

$v^i$, $h^j$ found by gradient descent

# Matrix Factorization

Recommendation System:
History as customer,
vocabulary as product ……

history

Vocabulary

| | dog $h^1$ | cat $h^2$ | …… | child |
|---|---|---|---|---|
| ran $v^1$ | 0.2 $n_{11}$ | 0.3 | | 0.1 |
| jumped $v^2$ | 0 | 0.2 $n_{22}$ | | 0.1 |
| cried $v^3$ | 0 | 0 | | 0.3 |
| laughed $v^4$ | 0 | 0 | | 0.3 |
| …… | | | | |

Not observed

P( jumped | cat )

History "dog" and "cat" can have similar vector $h^{dog}$ and $h^{cat}$

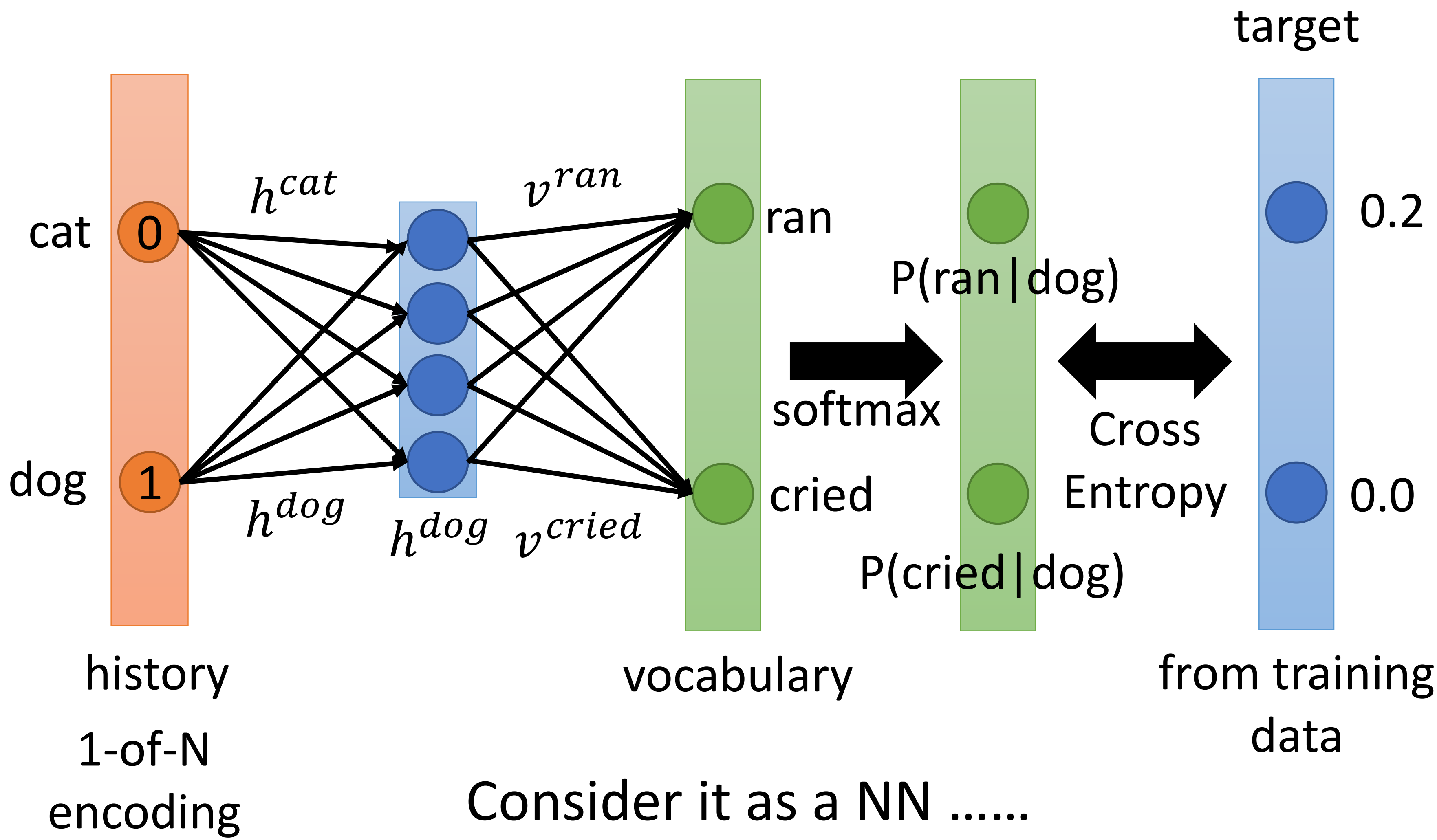
If $v^{jumped} \cdot h^{cat}$ is large, $v^{jumped} \cdot h^{dog}$ would be large accordingly.

Even if we have never seen "dog jumped …"

Smoothing is automatically done.

# Matrix Factorization

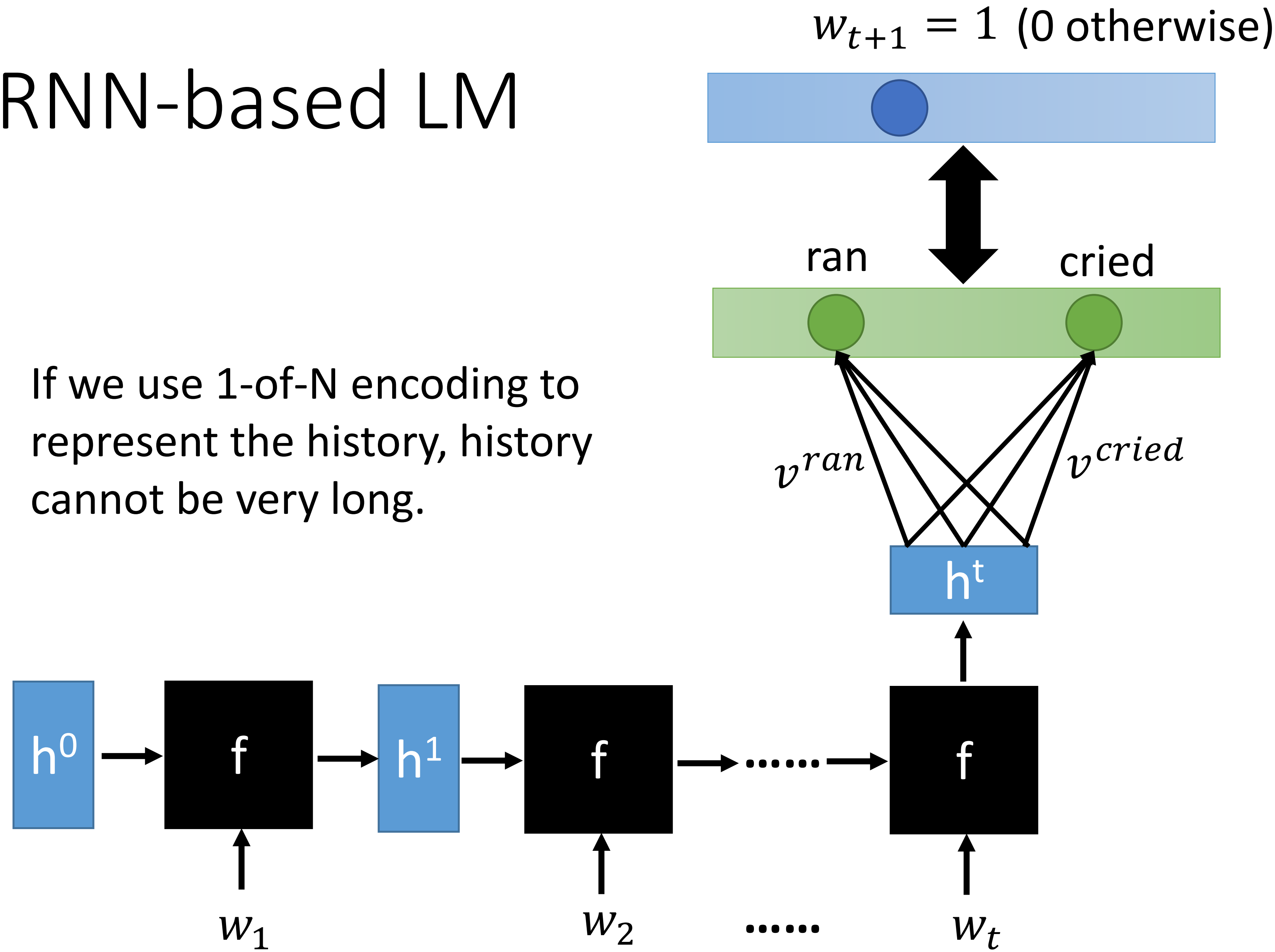
$$L = \sum_{(i,j)} \left(v^i \cdot h^j - n_{ij}\right)^2$$



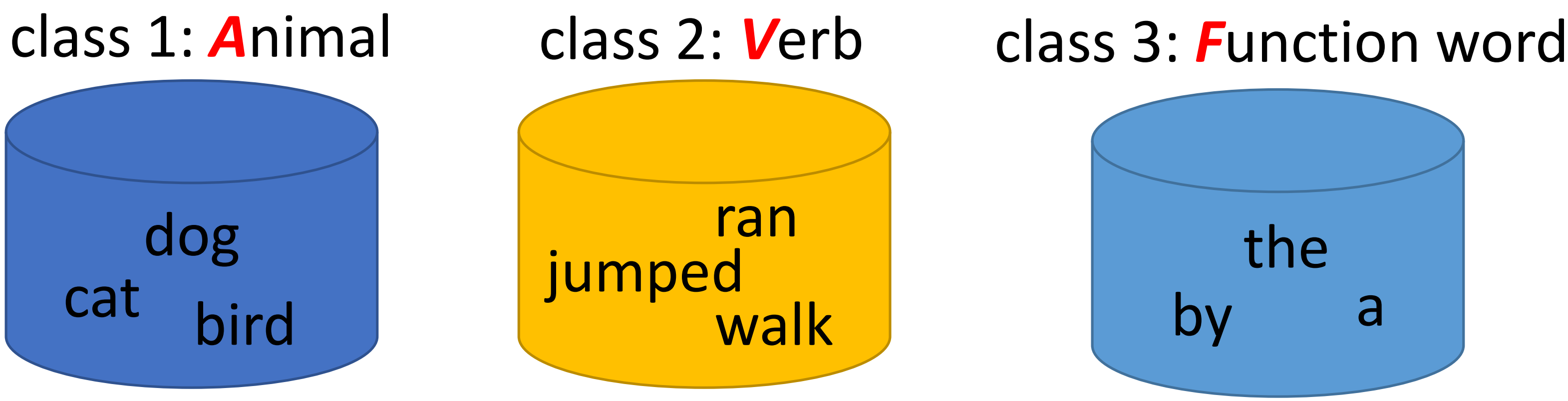

Consider it as a NN ……

# RNN-based LM

If we use 1-of-N encoding to represent the history, history cannot be very long.



# Class-based Language Modeling



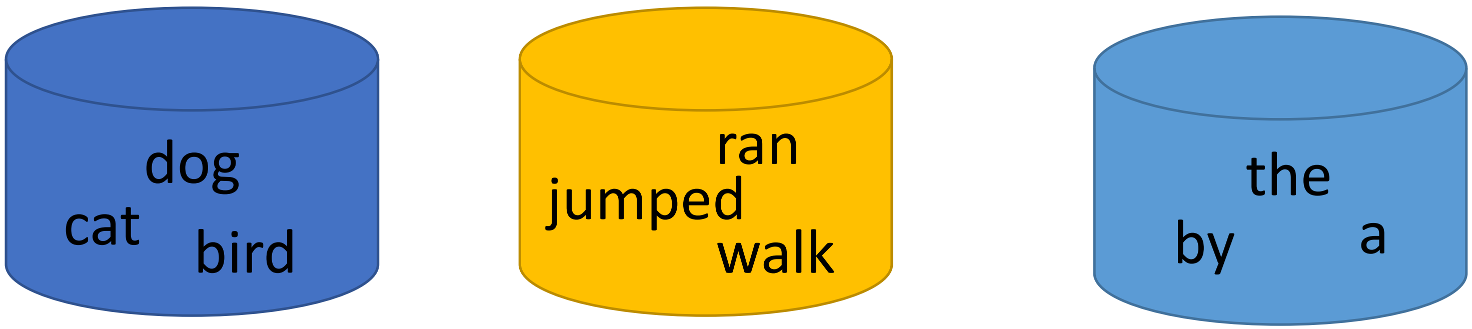

$W$ = "$w_1$ $w_2$ $w_3$"   $C(w_i)$: class of word $w_i$

$$\cancel{P(W) = P(w_1|START)\, P(w_2|w_1)\, P(w_3|w_2)}$$

$$P(W) = P(C(w_1)|START)\, P(C(w_2)|C(w_1))\, P(C(w_3)|C(w_2))$$
$$\times\, P(w_1|C(w_1))\, P(w_2|C(w_2))\, P(w_3|C(w_3))$$

# Class-based Language Modeling

class 1: ***A***nimal class 2: ***V***erb class 3: ***F***unction word




W = "the dog ran"

*F* *A* *V*

$$P(W) = P(F|START)\,P(A|F)\,P(V|A)$$

$$\times\ P(the|F)\,P(dog|A)\,P(ran|V)$$

P( class i | class j ) and P(word w| class i ) are estimated from training data.

# Class-based Language Modeling

P( class i | class j ) and P(word w| class i ) are estimated from training data.

***Training data***

| the | dog | ran |
|---|---|---|
| ***F*** | ***A*** | ***V*** |

| the | cat | jumped |
|---|---|---|
| ***F*** | ***A*** | V |

W = "the cat ran"

***F*** ***A*** ***V***

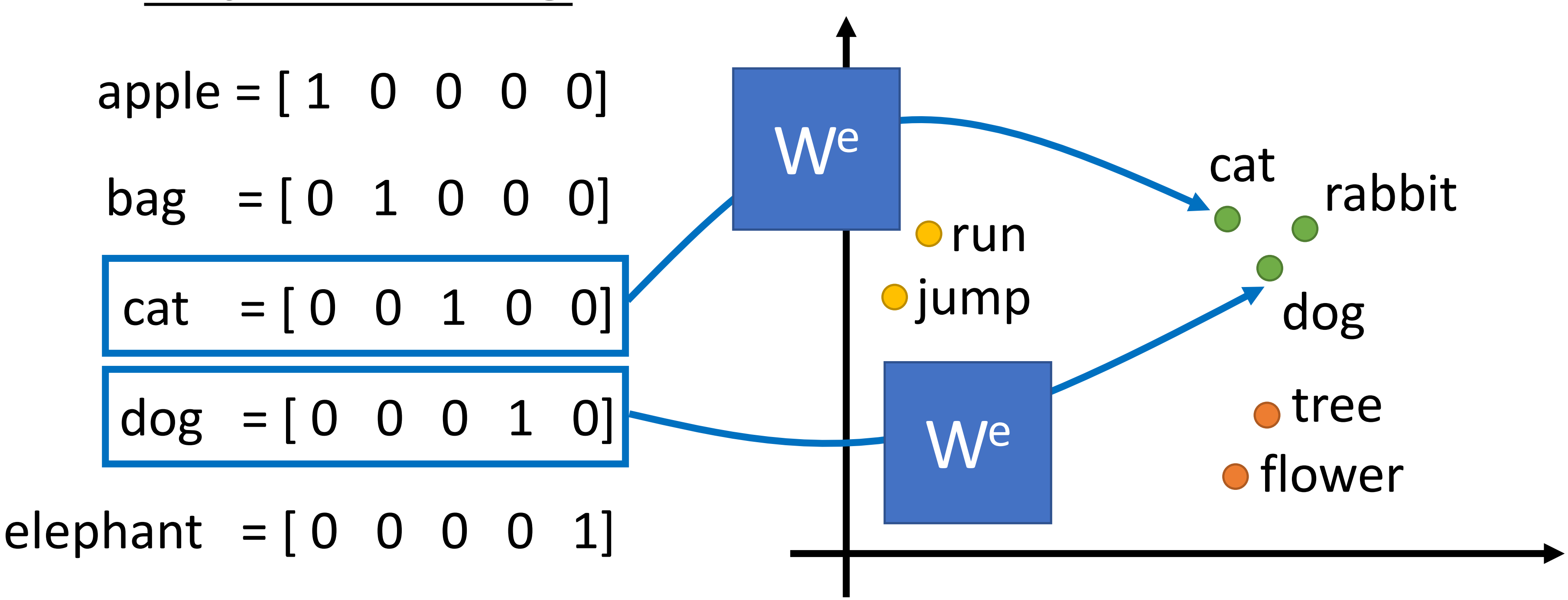
P(ran|cat) is zero given the training data

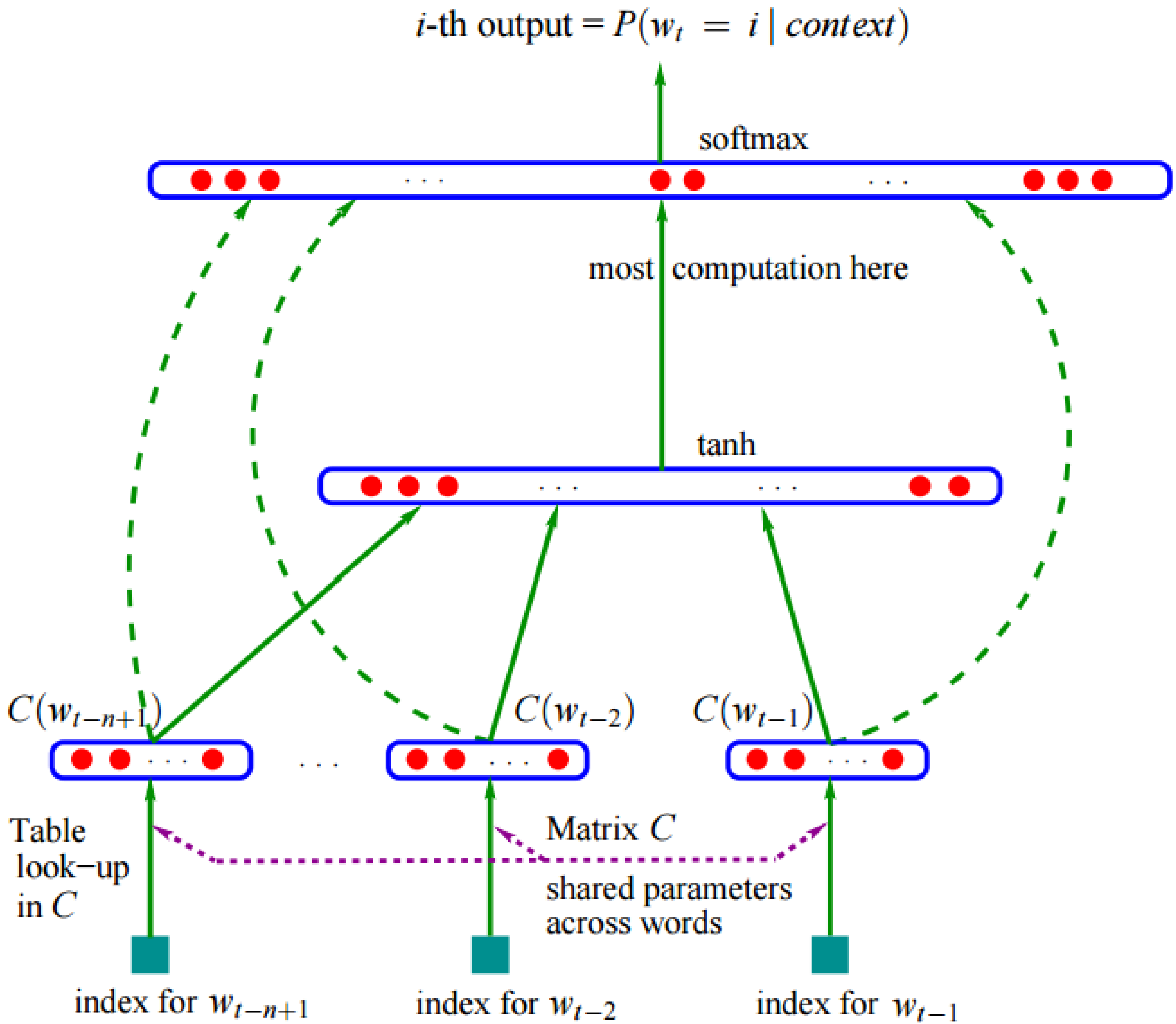
However, P( ***V***erb | ***A***nimal ) is not zero

# Soft Word Class

How to determine the classes of the words?

***1-of-N Encoding*** ***Word Embedding***

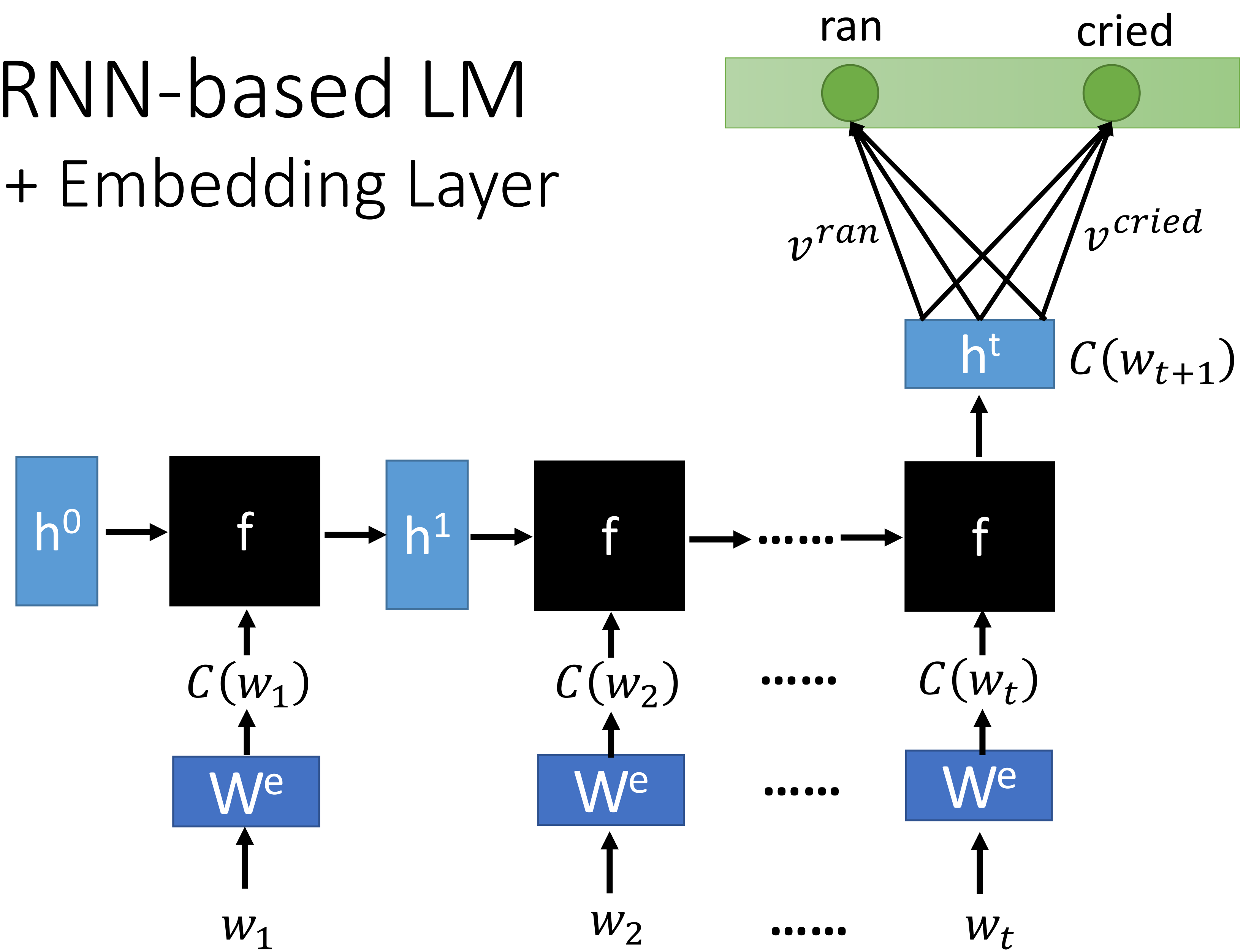




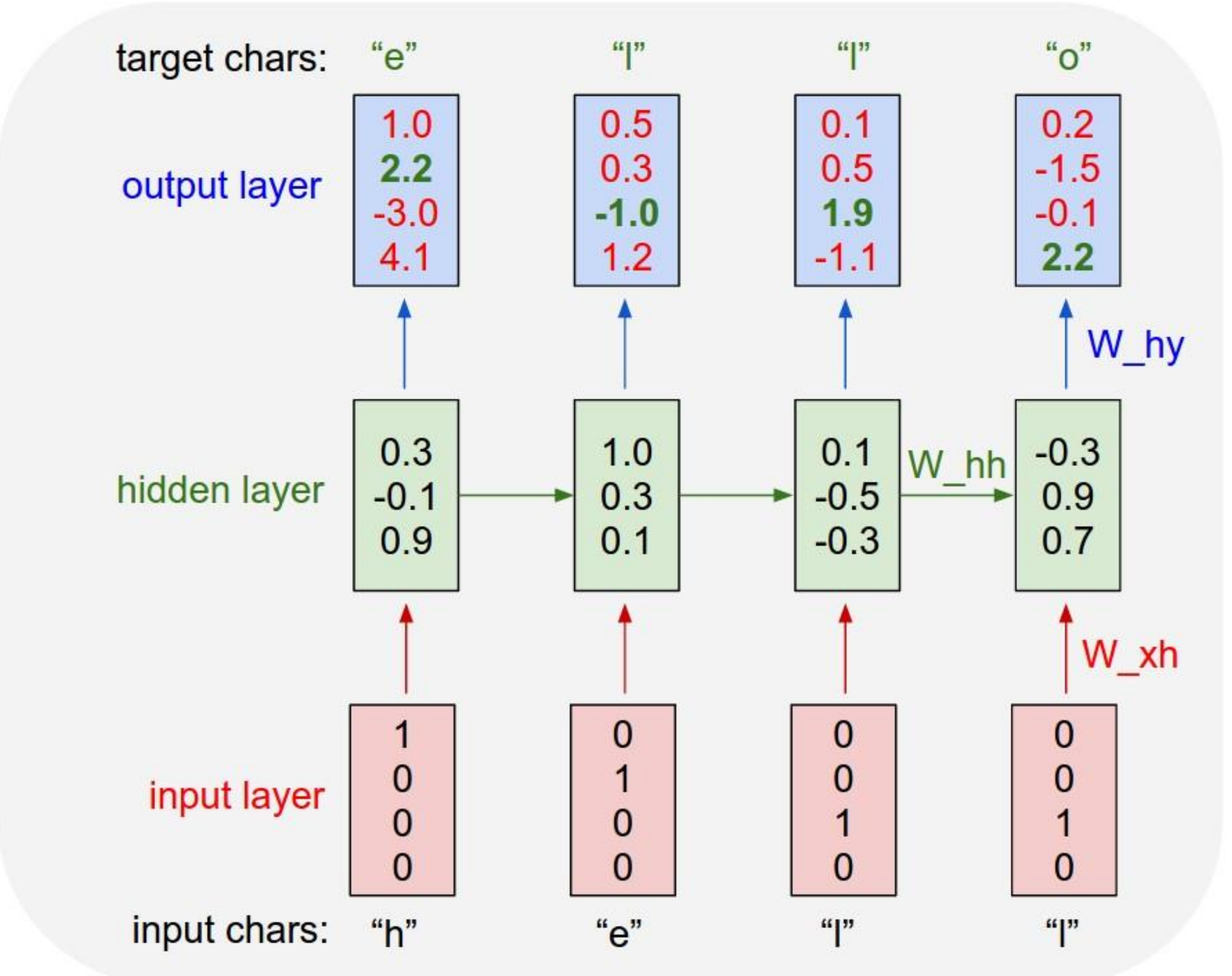

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, *3*(Feb), 1137-1155.

# RNN-based LM

+ Embedding Layer

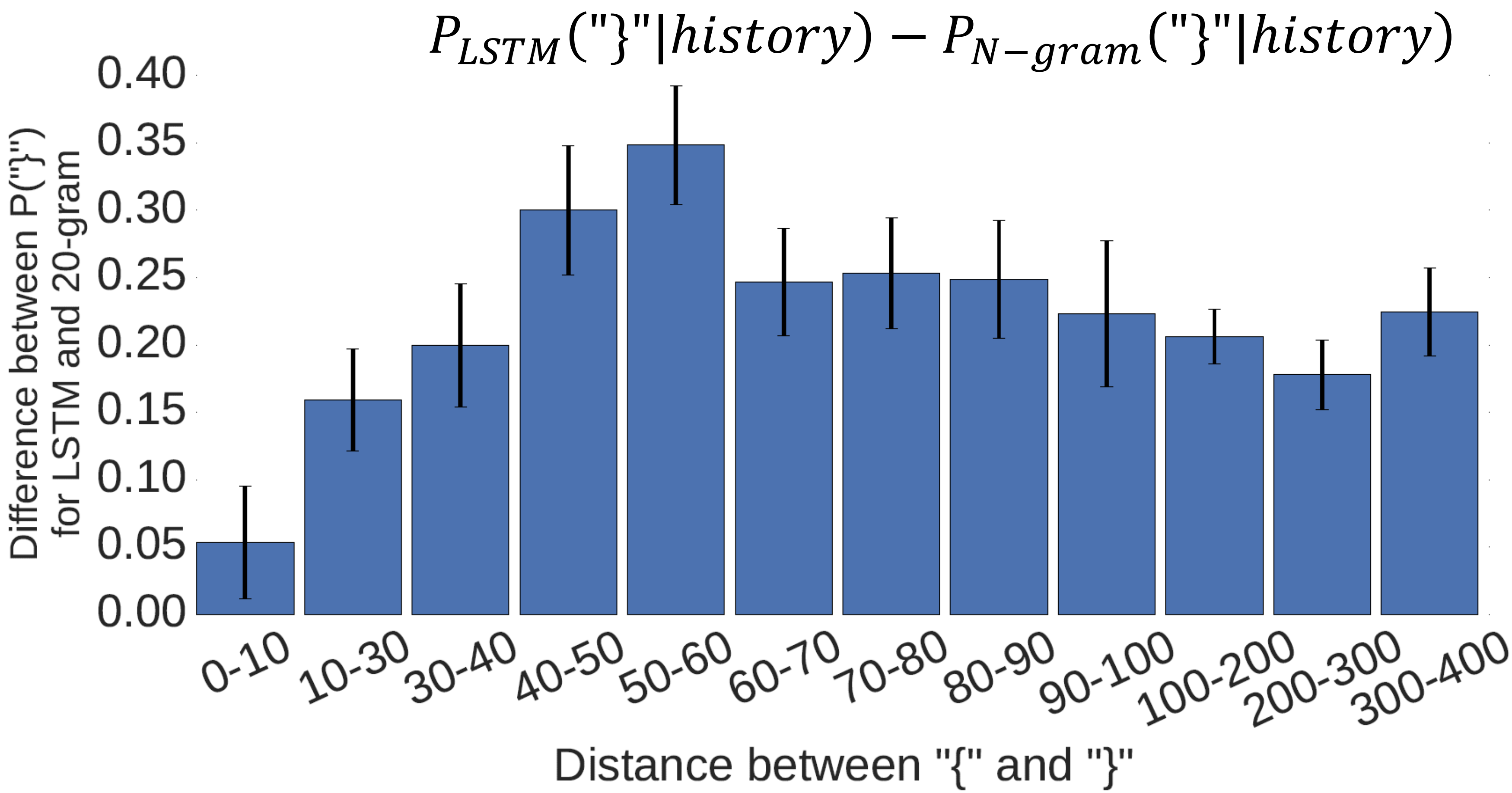


# Character-based LM

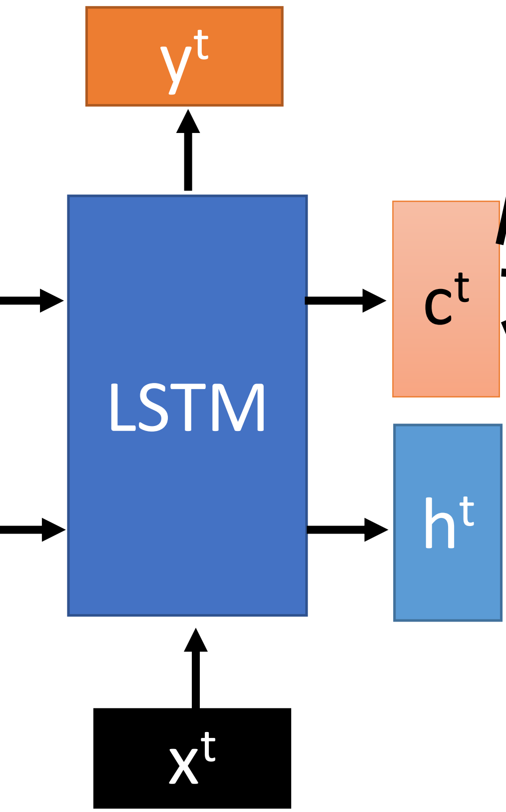



Source of image: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

# Long-term Information




Andrej Karpathy, Justin Johnson, Li Fei-Fei, Visualizing and Understanding Recurrent Networks, https://arxiv.org/abs/1506.02078




-1 is red
+1 is blue

Cell that turns on inside comments and quotes:

```
/* Duplicate LSM field information.  The lsm_rule is opaque, so
 * re-initialized. */
static inline int audit_dupe_lsm_field(struct audit_field *df,
           struct audit_field *sf)
{
 int ret = 0;
 char *lsm_str;
 /* our own copy of lsm_str */
 lsm_str = kstrdup(sf->lsm_str, GFP_KERNEL);
 if (unlikely(!lsm_str))
  return -ENOMEM;
 df->lsm_str = lsm_str;
 /* our own (refreshed) copy of lsm_rule */
 ret = security_audit_rule_init(df->type, df->op, df->lsm_str,
        (void **)&df->lsm_rule);
 /* Keep currently invalid fields around in case they
  * become valid after a policy reload. */
 if (ret == -EINVAL) {
  pr_warn("audit rule for LSM \'%s\' is invalid\n",
   df->lsm_str);
  ret = 0;
 }
 return ret;
}
```

Cell that is sensitive to the depth of an expression:

```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
 int i;
 if (classes[class]) {
  for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
   if (mask[i] & classes[class][i])
    return 0;
 }
 return 1;
}
```

Cell that might be helpful in predicting a new line. Note that it only turns on for some ")":

```
char *audit_unpack_string(void **bufp, size_t *remain, si
{
 char *str;
 if (!*bufp || (len == 0) || (len > *remain))
  return ERR_PTR(-EINVAL);
 /* Of the currently implemented string fields, PATH_MAX
  * defines the longest valid length.
  */
 if (len > PATH_MAX)
  return ERR_PTR(-ENAMETOOLONG);
 str = kmalloc(len + 1, GFP_KERNEL);
 if (unlikely(!str))
  return ERR_PTR(-ENOMEM);
 memcpy(str, *bufp, len);
 str[len] = 0;
 *bufp += len;
 *remain -= len;
 return str;
}
```

Cell sensitive to position in line:

The sole importance of the crossing of the Berezina lies in the fact that it plainly and indubitably proved the fallacy of all the plans for cutting off the enemy's retreat and the soundness of the only possible line of action--the one Kutuzov and the general mass of the army demanded--namely, simply to follow the enemy up. The French crowd fled at a continually increasing speed and all its energy was directed to reaching its goal. It fled like a wounded animal and it was impossible to block its path. This was shown not so much by the arrangements it made for crossing as by what took place at the bridges. When the bridges broke down, unarmed soldiers, people from Moscow and women with children who were with the French transport, all--carried on by vis inertiae--pressed forward into boats and into the ice-covered water and did not, surrender.

Cell that turns on inside quotes:

"You mean to imply that I have nothing to eat out of.... On the contrary, I can supply you with everything even if you want to give dinner parties," warmly replied Chichagov, who tried by every word he spoke to prove his own rectitude and therefore imagined Kutuzov to be animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

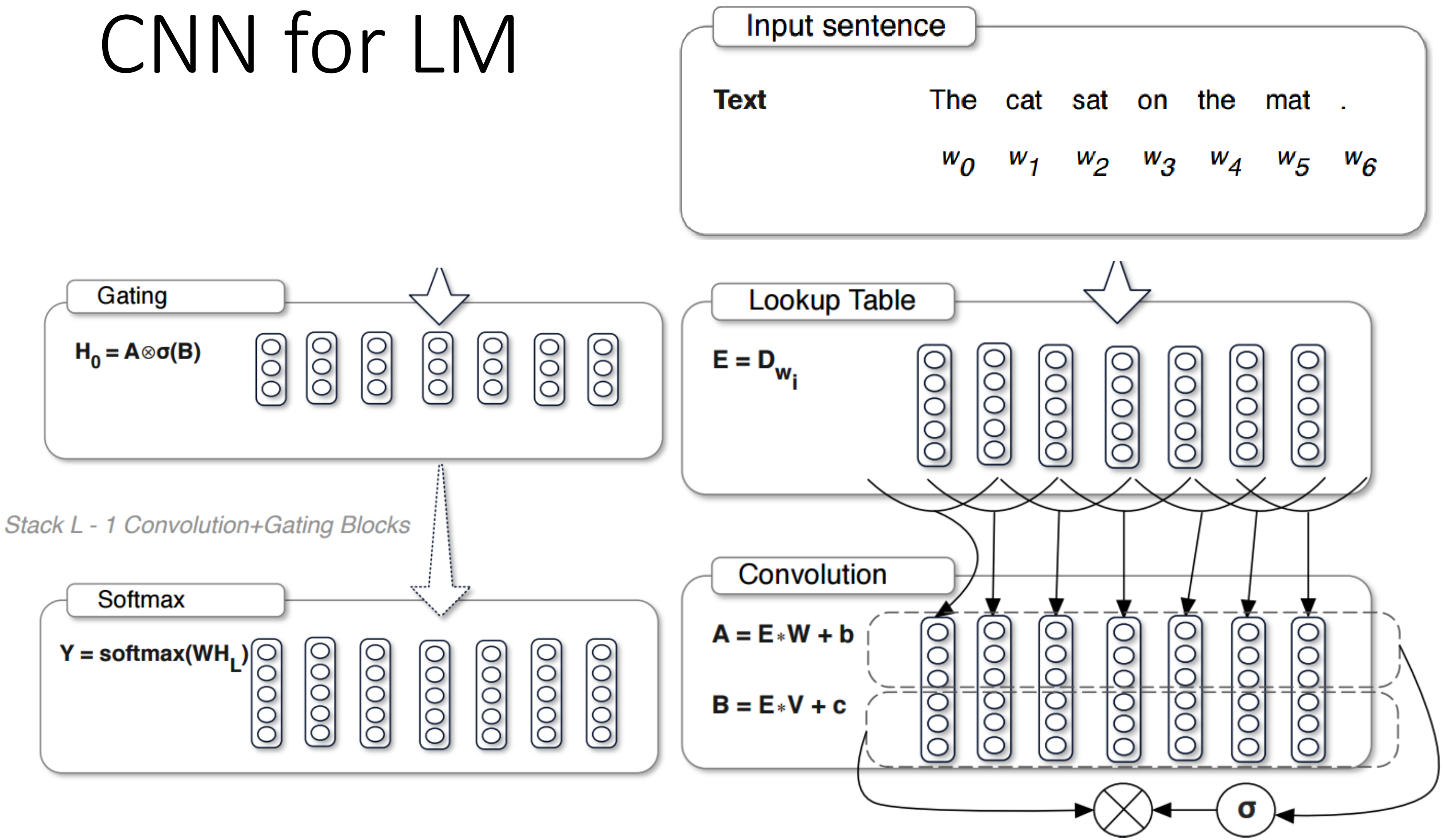
Cell that robustly activates inside if statements:

```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
   siginfo_t *info)
{
 int sig = next_signal(pending, mask);
 if (sig) {
  if (current->notifier) {
   if (sigismember(current->notifier_mask, sig)) {
    if (!(current->notifier)(current->notifier_data)) {
     clear_thread_flag(TIF_SIGPENDING);
     return 0;
    }
   }
  }
  collect_signal(sig, pending, info);
 }
 return sig;
}
```

A large portion of cells are not easily interpretable. Here is a typical example:

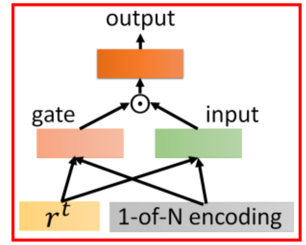
```
/* Unpack a filter field's string representation from user-space
 * buffer. */
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)
{
 char *str;
 if (!*bufp || (len == 0) || (len > *remain))
  return ERR_PTR(-EINVAL);
 /* Of the currently implemented string fields, PATH_MAX
  * defines the longest valid length.
  */
```

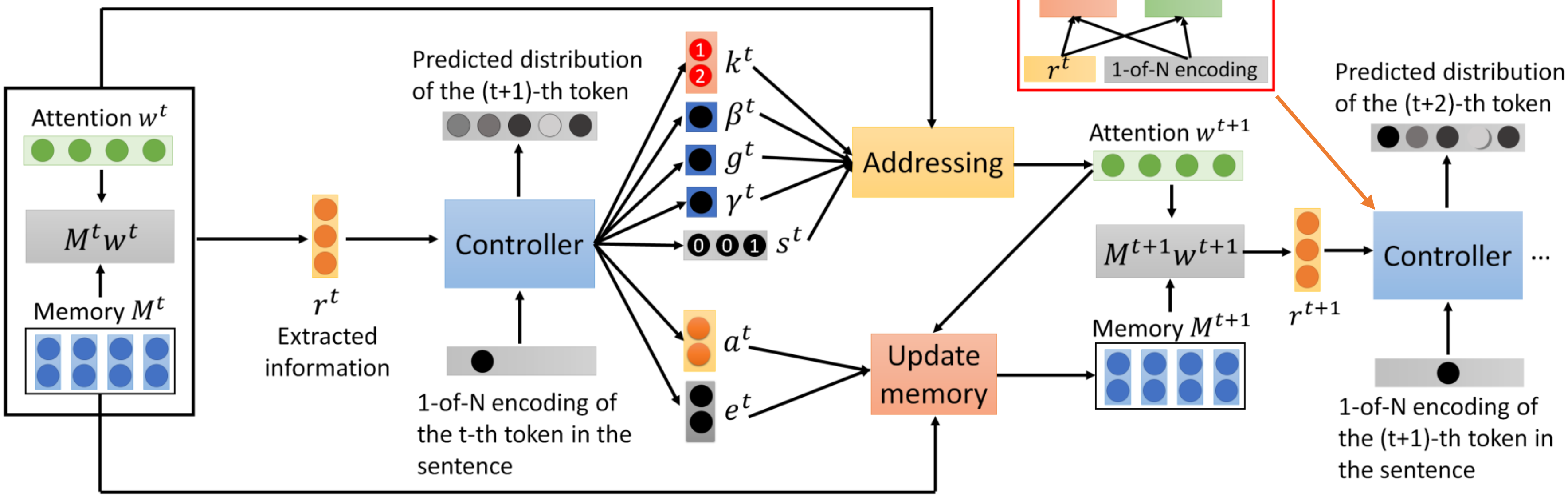
# CNN for LM

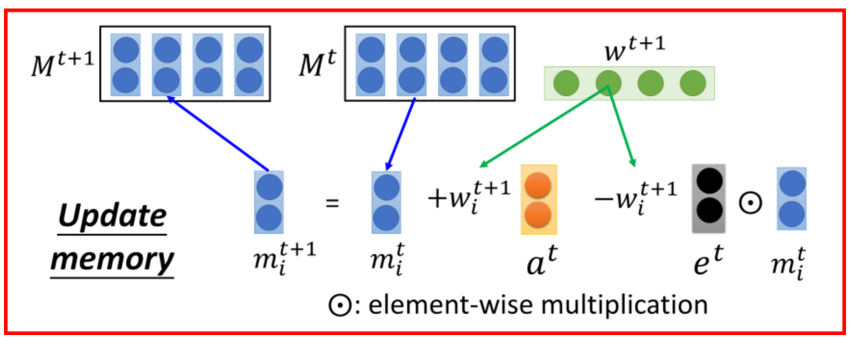


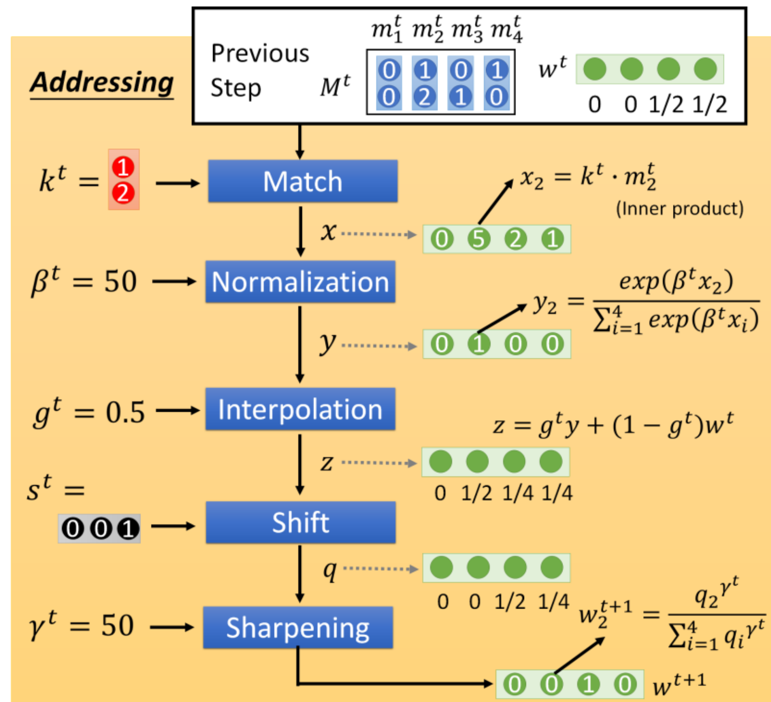

Yann N. Dauphin, Angela Fan, Michael Auli, David Grangier, Language Modeling with Gated Convolutional Networks, https://arxiv.org/abs/1612.08083

# Neural Turing Machine for LM




Wei-Jen Ko, Bo-Hsiang Tseng, Hung-yi Lee, “Recurrent Neural Network based Language Modeling with Controllable External Memory”, ICASSP, 2017

# For Large Output Layer

- Factorization of the Output Layer
  - Mikolov Tomáš: Statistical Language Models based on Neural Networks. PhD thesis, Brno University of Technology, 2012. (chapter 3.4.2)
  - http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015/NN%20Lecture/RNNLM.ecm.mp4/index.html
- Noise Contrastive Estimation (NCE)
  - X. Chen, X. Liu, M. J. F. Gales and P. C. Woodland, "Recurrent neural network language model training with noise contrastive estimation for speech recognition,“ ICASSP, 2015
  - B. Zoph , A. Vaswani, J. May, and K. Knight, “Simple, Fast Noise-Contrastive Estimation for Large RNN Vocabularies” , NAACL, 2016
- Hierarachical Softmax
  - F Morin, Y Bengio, “Hierarchical Probabilistic Neural Network Language Model”, Aistats, 2005
- Blog posts:
  - http://sebastianruder.com/word-embeddings-softmax/index.html
  - http://cpmarkchang.logdown.com/posts/276263--hierarchical-probabilistic-neural-networks-neural-network-language-model

# To learn more ……


- M. Sundermeyer, H. Ney and R. Schlüter, From Feedforward to Recurrent LSTM Neural Networks for Language Modeling, in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 517-529, 2015.
- Kazuki Irie, Zoltan Tuske, Tamer Alkhouli, Ralf Schluter, Hermann Ney, “LSTM, GRU, Highway and a Bit of Attention: An Empirical Overview for Language Modeling in Speech Recognition”, Interspeech, 2016
- Ke Tran, Arianna Bisazza, Christof Monz, Recurrent Memory Networks for Language Modeling, NAACL, 2016
- Jianpeng Cheng, Li Dong and Mirella Lapata, Long Short-Term Memory-Networks for Machine Reading, arXiv preprint, 2016

# Acknowledgement

- 感謝 傅彥禎、楊喻涵 同學發現投影片上的打字錯誤