DATA MINING

/CISC 873 - Steven Ding

Sequential+

NEXT

- Language Model
- Neural Network for Language Model
- RNN for Language Modeling
- Transformer

Language Modelling

- Probabilities associated with language

  all of a sudden I notice three guys standing on the sidewalk

  on guys all I of notice sidewalk three a sudden standing the

- Predicting the next word

  Please turn your homework…

Speech and Language Processing. Daniel Jurafsky & James H. Martin.

Language Modelling

- Probability distribution over text words
  - Assign probability to possible next words
  - State transition

- Speech recognition
- Spelling correction
- Grammatical error correction
- Machine translation

Speech and Language Processing. Daniel Jurafsky & James H. Martin.

Language Modelling

- In machine translation:

他	向	记者	介绍了	主要	内容
He	to	reporters	introduced	main	content

he introduced reporters to the main contents of the statement
he briefed to reporters the main contents of the statement
he briefed reporters on the main contents of the statement

Speech and Language Processing. Daniel Jurafsky & James H. Martin.

N-gram model

- N-gram modeling
- Bigram:

$$P(w_t|w_{t-1})$$

- 3-gram

$$P(w_t|w_{t-1}, w_{t-2})$$

- 4-gram

$$P(w_t|w_{t-1}, w_{t-2}, w_{t-3})$$

- N-gram

$$P(w_t|w_{t-1}, \ldots, w_{t-(n-1)})$$

Speech and Language Processing. Daniel Jurafsky & James H. Martin.

N-gram model

1-gram: To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have

2-gram: What means, sir. I confess she? then all sorts, he is trim, captain

- 4-gram: King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in

- The ***longer*** the context, the more coherent

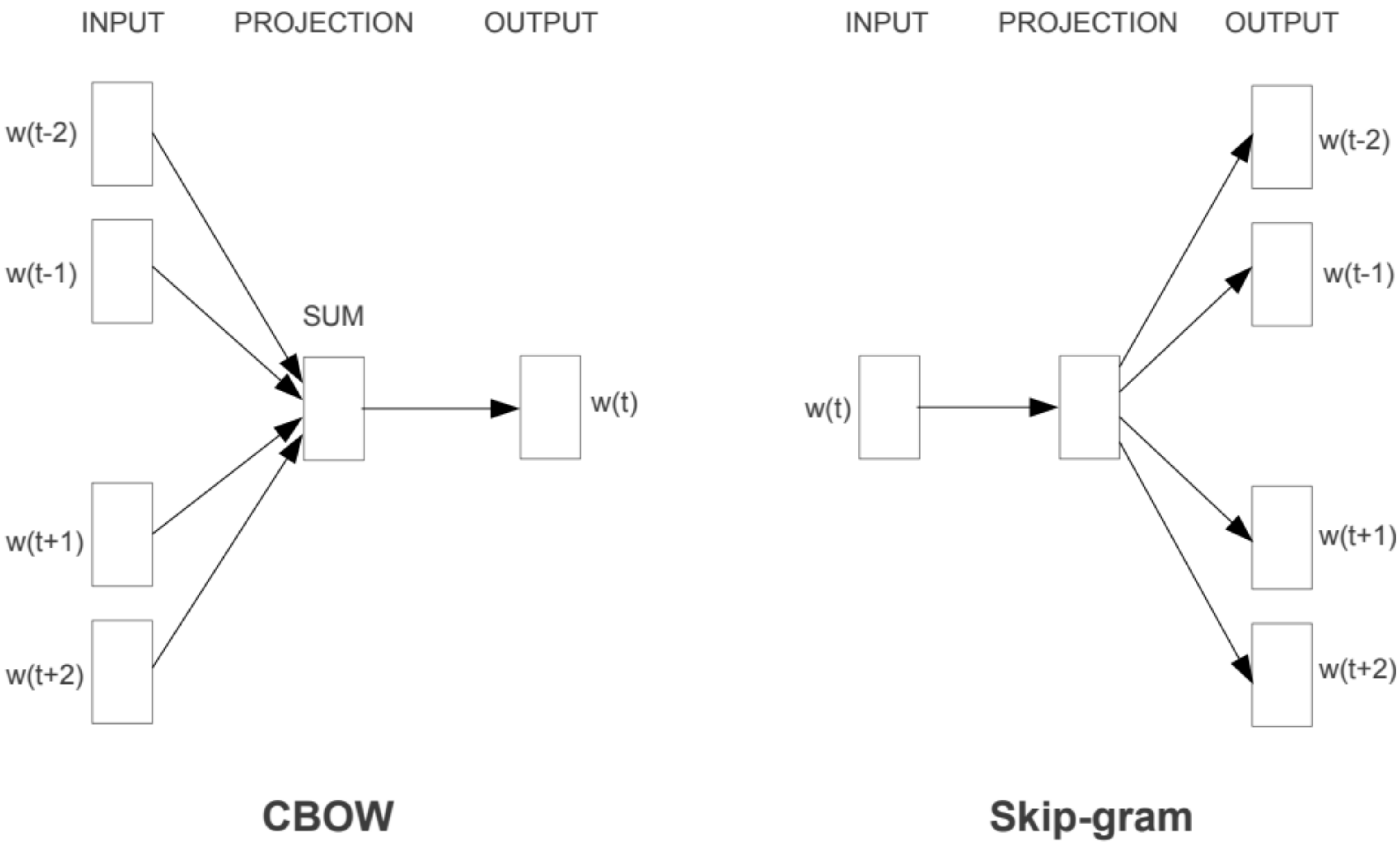
Speech and Language Processing. Daniel Jurafsky & James H. Martin.

N-gram model

- Issues:
  - Storage limitation
  - Shakespeare
    - 844,000,000 bigram
    - 7 x 10^17 4-grams…
- Characteristics:
  - Sparseness

Speech and Language Processing. Daniel Jurafsky & James H. Martin.

Neural Language Model

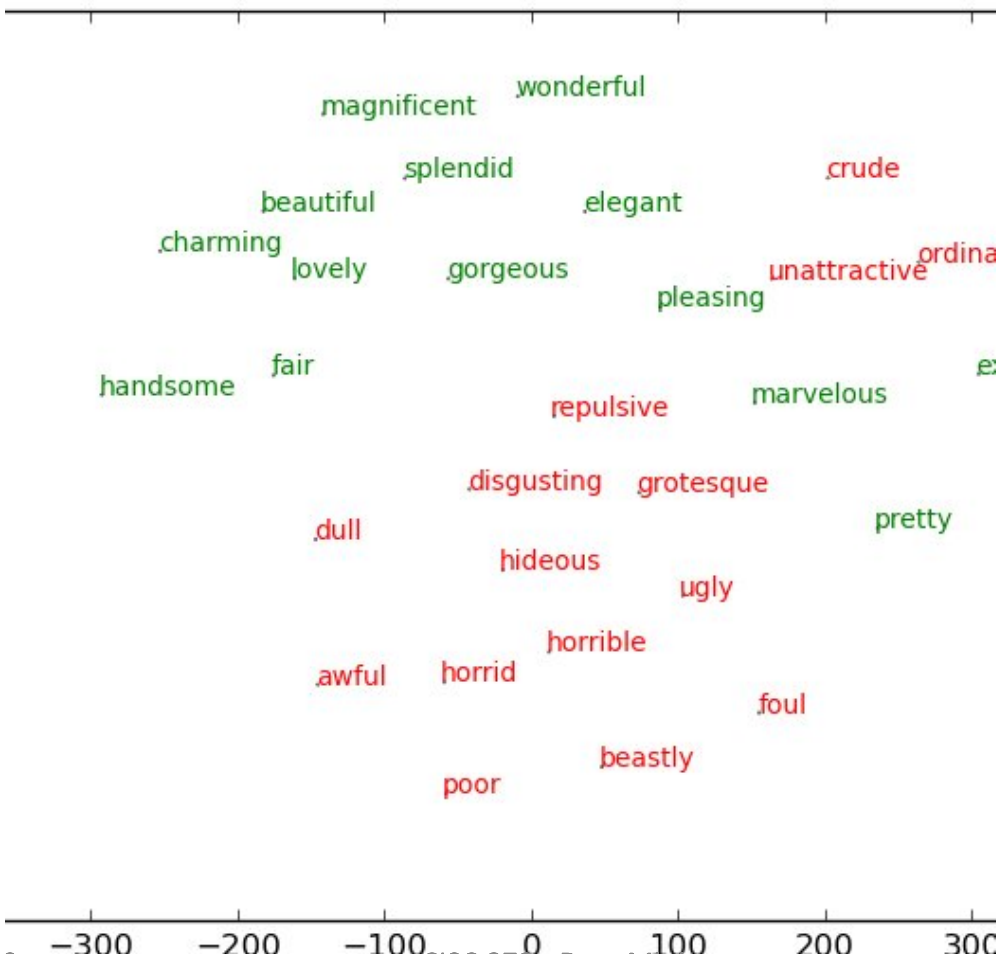


Example: steven is drunk again

Distributed Representations of Words and Phrases and their Compositionality

Neural Language Model

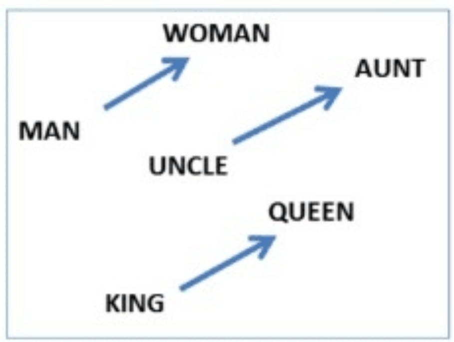
- Embedding matrix visualization (approximated nearest neighbor clustering)

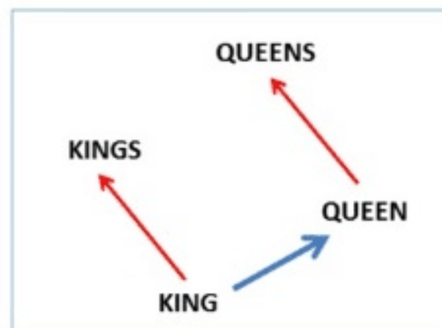


Speech2Vec: A Sequence-to-Sequence Framework for Learning Word Embeddings from Speech

Neural Language Model (vector composition)

- Learned embedding matrix is able to capture relationship between words
- Examples:
- vec(king) – vec(man) + vec(woman) = vec(queen)





Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zolty	Viet Nam	flag carrier Lufthansa	upriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cecile De

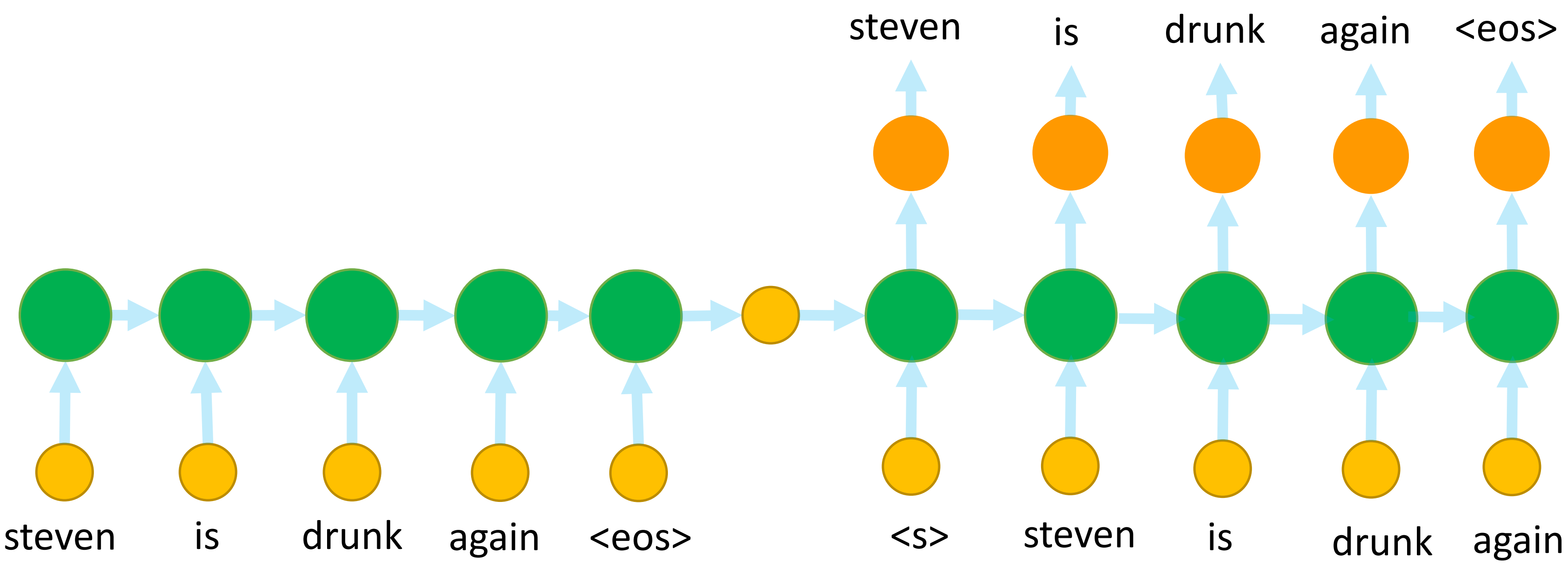
Speech2Vec: A Sequence-to-Sequence Framework for Learning Word Embeddings from Speech

Neural Language Model

- Recurrent Neural Language Model
- Many-to-Many structure
- AKA encoder-decoder structure

RNN Encoder – Decoder

Example: steven is drunk again



Encoder – Decoder

- What does it learn?? (on natural language)

- Early training epochs:
  - we counter. He stutn co des. His stanted out one ofler that concossions and was to gearang reay Jotrets and with fre colt otf paitt thin wall. Which das stimn

- Later:
  - "Why do what that day," replied Natasha, and wishing to himself the fact the princess, Princess Mary was easier, fed in had oftened him. Pierre asking his soul came to the packs and drove up his father-in-law women.

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

Encoder – Decoder

- What does it learn?? (on source code)

```
static int indicate_policy(void)
{
  int error;
  if (fd == MARN_EPT) {
    /*
     * The kernel blank will coeld it to userspace.
     */
    if (ss->segment < mem_total)
      unblock_graph_and_set_blocked();
    else
      ret = 1;
    goto bail;
  }
  segaddr = in_SB(in.addr);
  selector = seg / 16;
  setup_works = true;
  for (i = 0; i < blocks; i++) {
    seq = buf[i++];
    bpf = bd->bd.next + i * search;
    if (fd) {
      current = blocked;
    }
  }
  rw->name = "Getjbbregs";
  bprm_self_clearl(&iv->version);
  regs->new = blocks[(BPF_STATS << info->historidac)] | PFMR_CLOBATHINC_SECONDS << 12;
  return segtable;
}
```


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

Encoder – Decoder

- What does it learn?? (on source code)

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;
}
```

Cell that is sensitive to the depth of an expression:

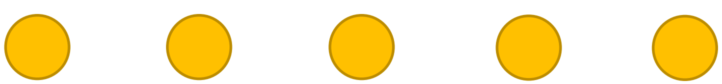
```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
 int i;
 if (classes[class]) {
  for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
   if (mask[i] & classes[class][i])
    return 0;
 }
 return 1;
}
```


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

Issue

- Long-term dependencies
  - Different cell implementation
  - Attention
- Computational complexity
  - The time stamp t's calculation
    - depends on time stamp t-1
  - For encoder/decoder:
    - Required *2 times t* passes over the memory cell to train a batch/sample

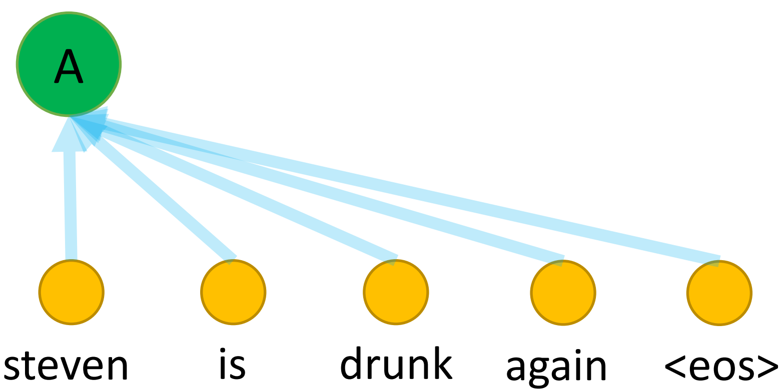
The Transformer – Attention is all you need



steven is drunk again <eos>

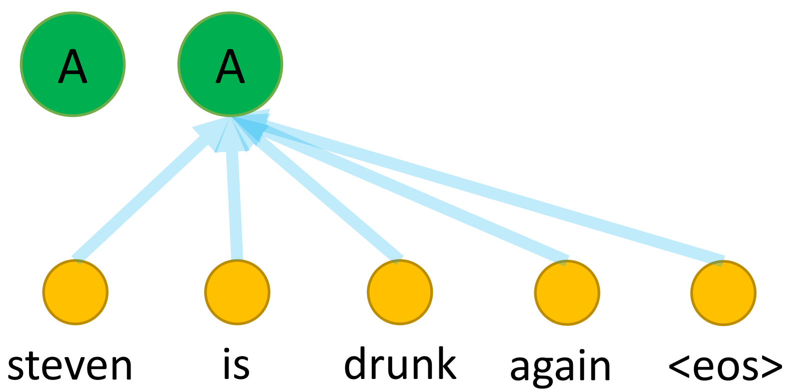
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

The Transformer – Attention is all you need



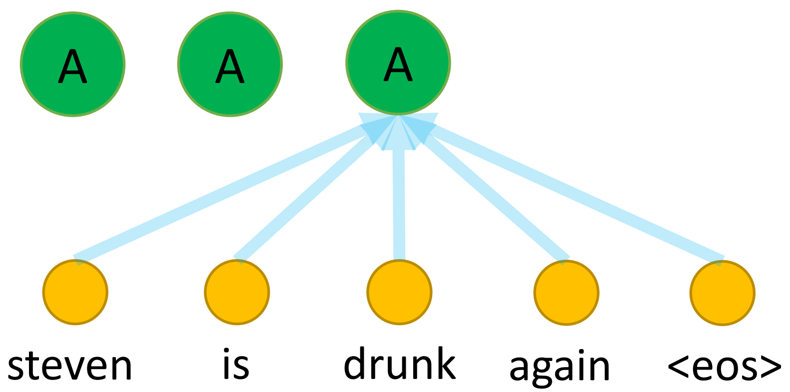
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

The Transformer – Attention is all you need



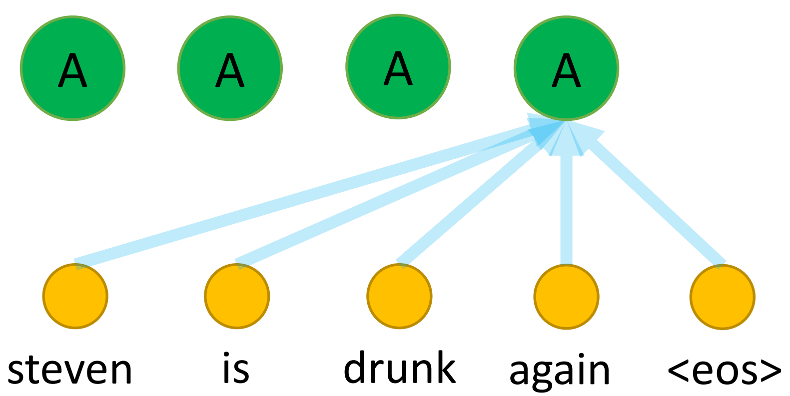
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

The Transformer – Attention is all you need



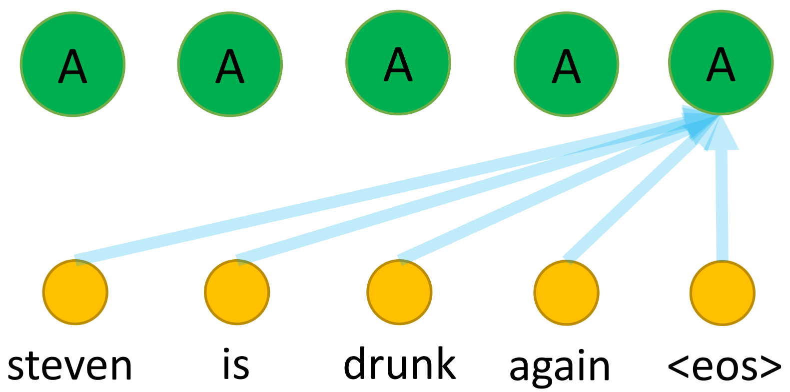
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

The Transformer – Attention is all you need



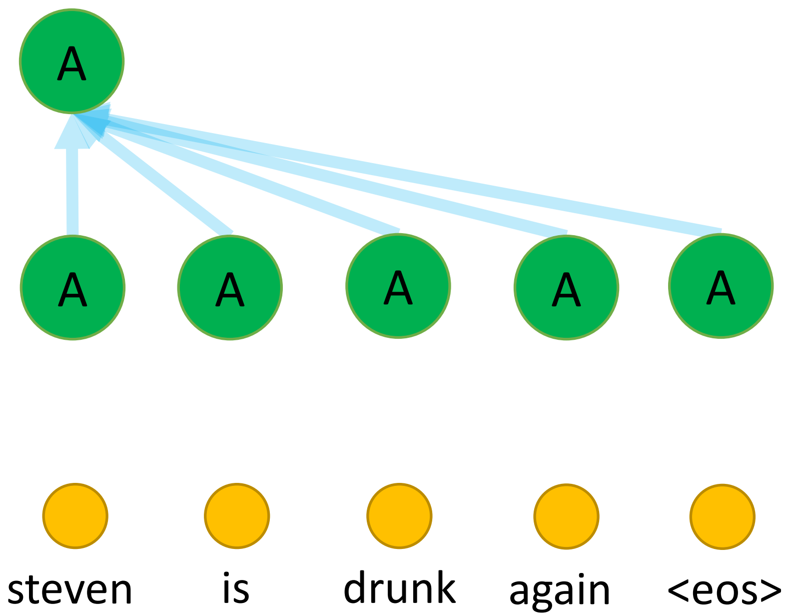
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

The Transformer – Attention is all you need

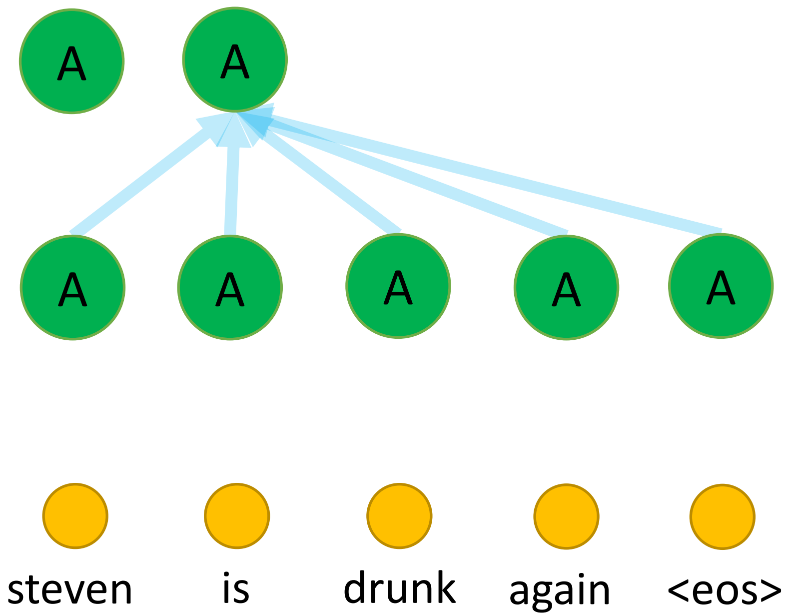


http://karpathy.github.io/2015/05/21/rnn-effectiveness/

The Transformer – Attention is all you need

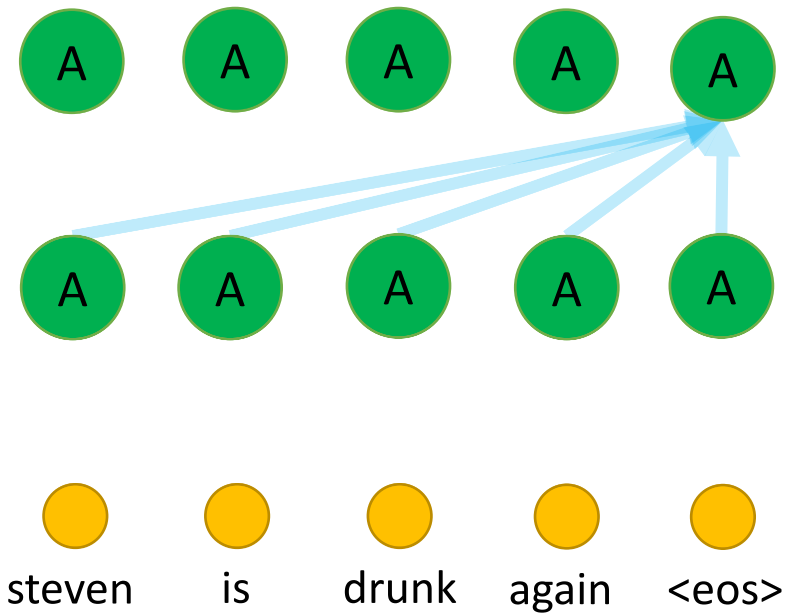


The Transformer – Attention is all you need



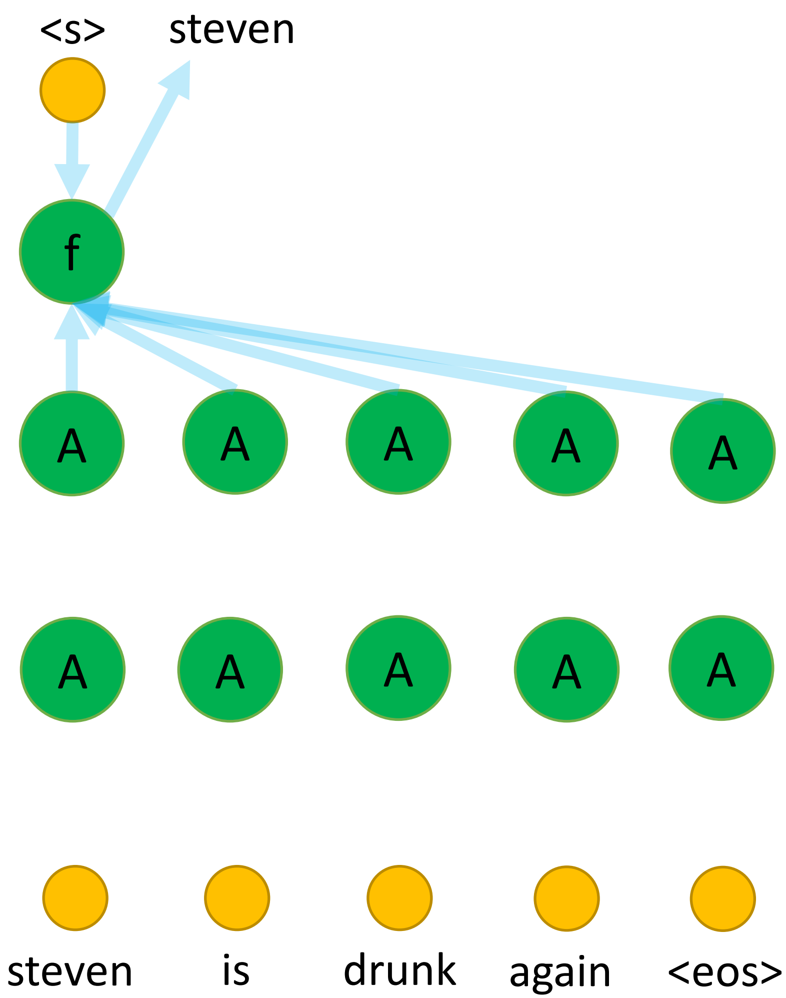
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

The Transformer – Attention is all you need

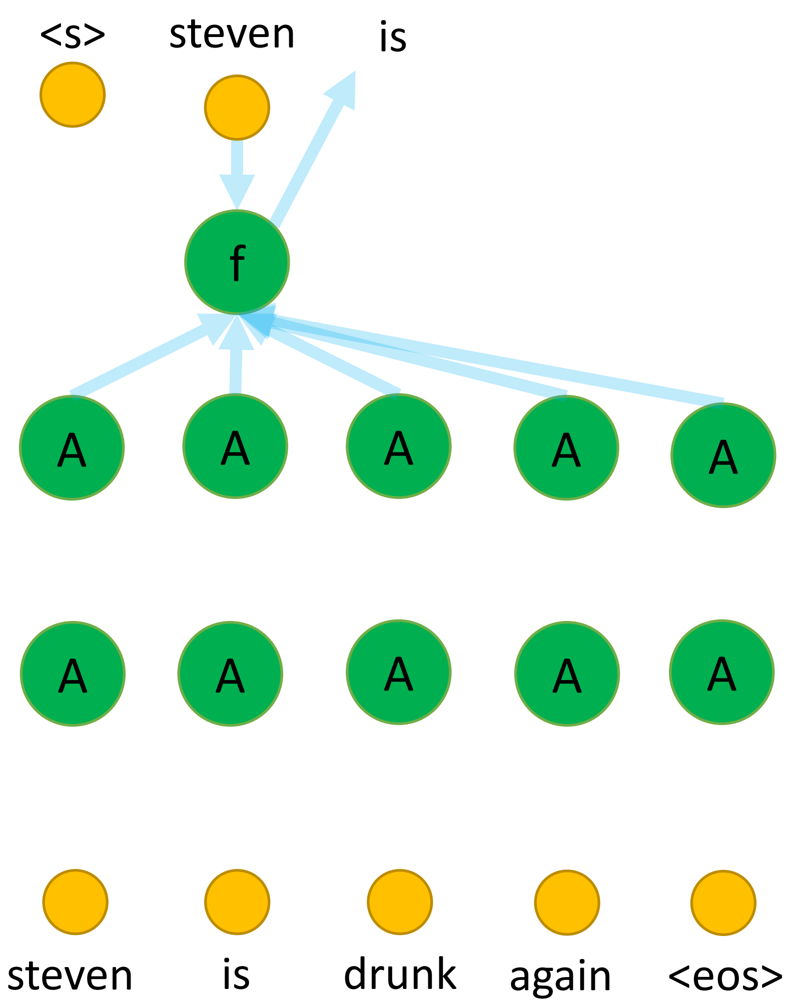


http://karpathy.github.io/2015/05/21/rnn-effectiveness/

The Transformer – Attention is all you need

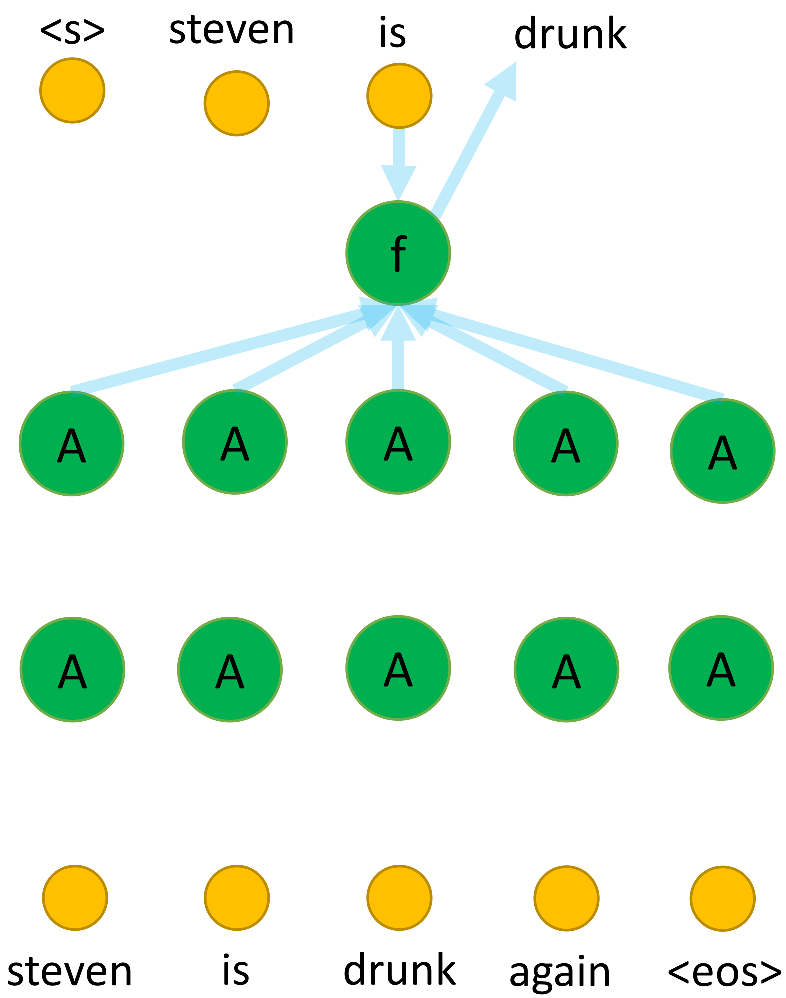


The Transformer – Attention is all you need



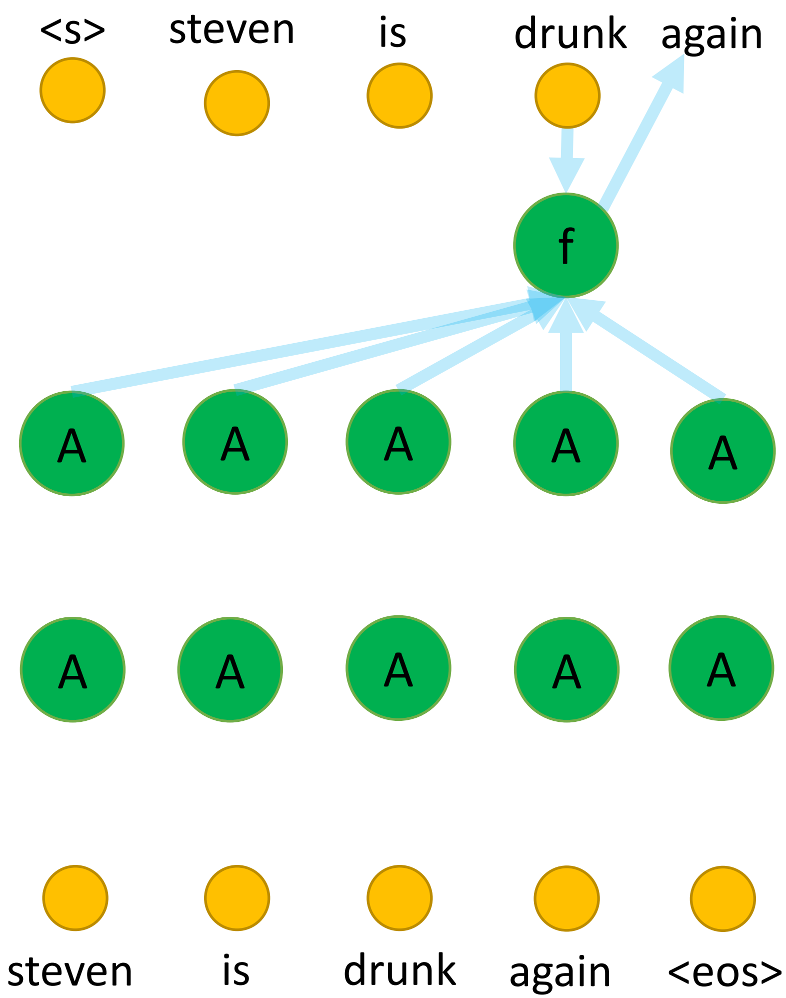
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

The Transformer – Attention is all you need



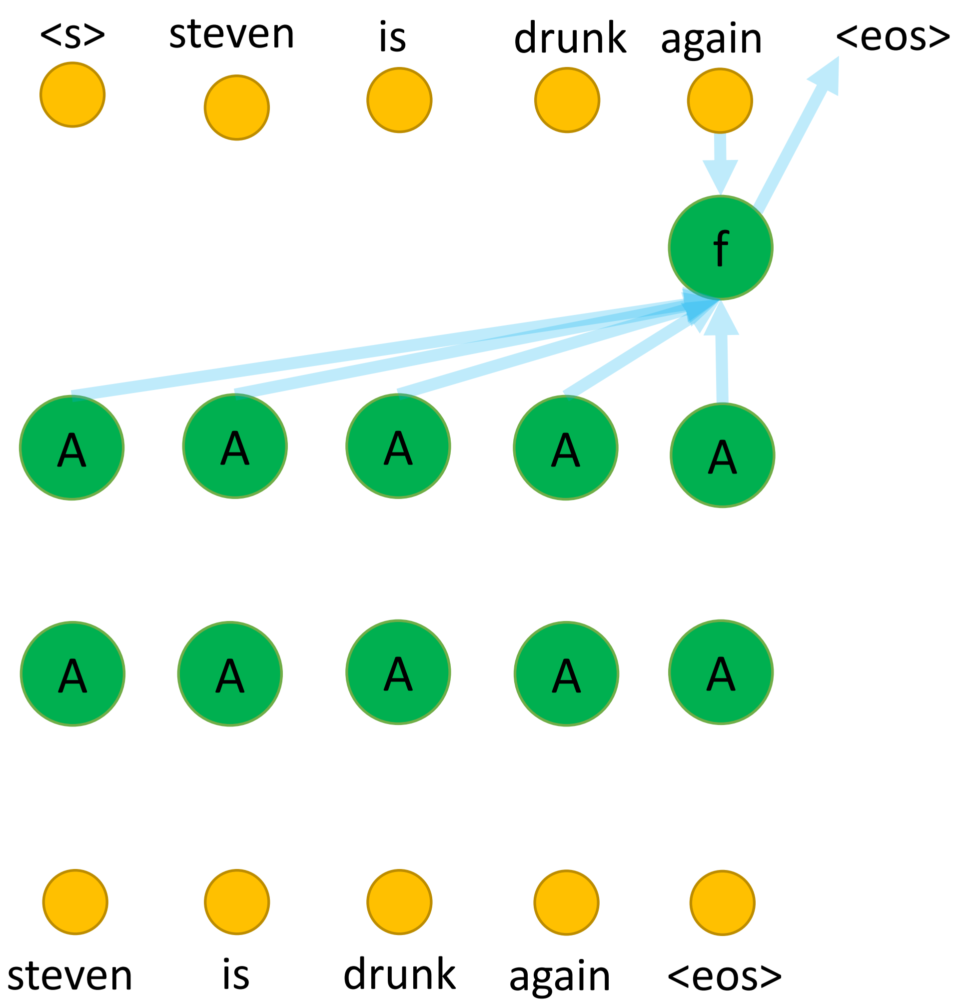
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

The Transformer – Attention is all you need



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

The Transformer – Attention is all you need



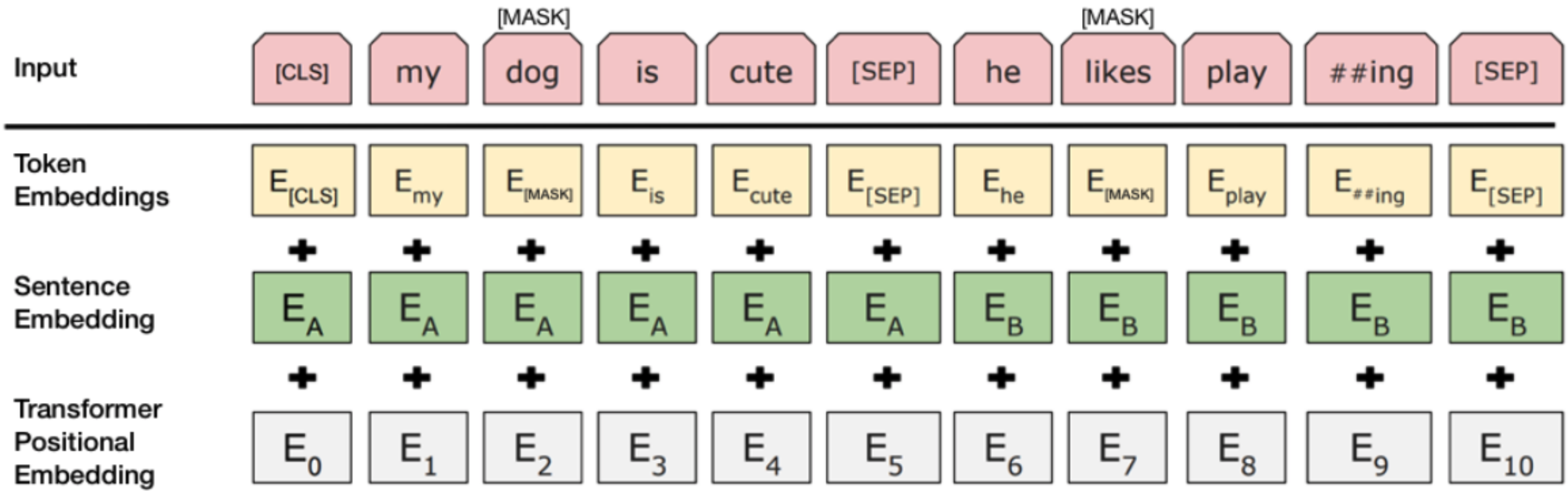
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Performance on down-stream tasks:

Rank	Name	Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m	MNLI-mm	QNLI	RTE	WNLI	AX
1	T5 Team - Google	T5		90.3	71.6	97.5	92.8/90.4	93.1/92.8	75.1/90.6	92.2	91.9	96.9	92.8	94.5	53.1
2	ERNIE Team - Baidu	ERNIE		90.1	72.8	97.5	93.2/91.0	92.9/92.5	75.2/90.8	91.2	90.8	96.1	90.9	94.5	49.4
3	Microsoft D365 AI & MSR AI & GATECH	MT-DNN-SMART		89.9	69.5	97.5	93.7/91.6	92.9/92.5	73.9/90.2	91.0	90.8	99.2	89.7	94.5	50.2
+ 4	Alibaba DAMO NLP	ALICE v2 large ensemble (Alibaba DAMO NLP)		89.7	73.2	97.1	93.9/91.9	93.0/92.5	74.8/91.0	90.8	90.6	95.9	87.4	94.5	48.7
+ 5	Microsoft D365 AI & UMD	FreeLB-RoBERTa (ensemble)		88.4	68.0	96.8	93.1/90.8	92.3/92.1	74.8/90.3	91.1	90.7	95.6	88.7	89.0	50.1
6	Junjie Yang	HIRE-RoBERTa		88.3	68.6	97.1	93.0/90.7	92.4/92.0	74.3/90.2	90.7	90.4	95.5	87.9	89.0	49.3
7	Facebook AI	RoBERTa		88.1	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2	90.8	90.2	95.4	88.2	89.0	48.7
+ 8	Microsoft D365 AI & MSR AI	MT-DNN-ensemble		87.6	68.4	96.5	92.7/90.3	91.1/90.7	73.7/89.9	87.9	87.4	96.0	86.3	89.0	42.8
9	GLUE Human Baselines	GLUE Human Baselines		87.1	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4	92.0	92.8	91.2	93.6	95.9	-

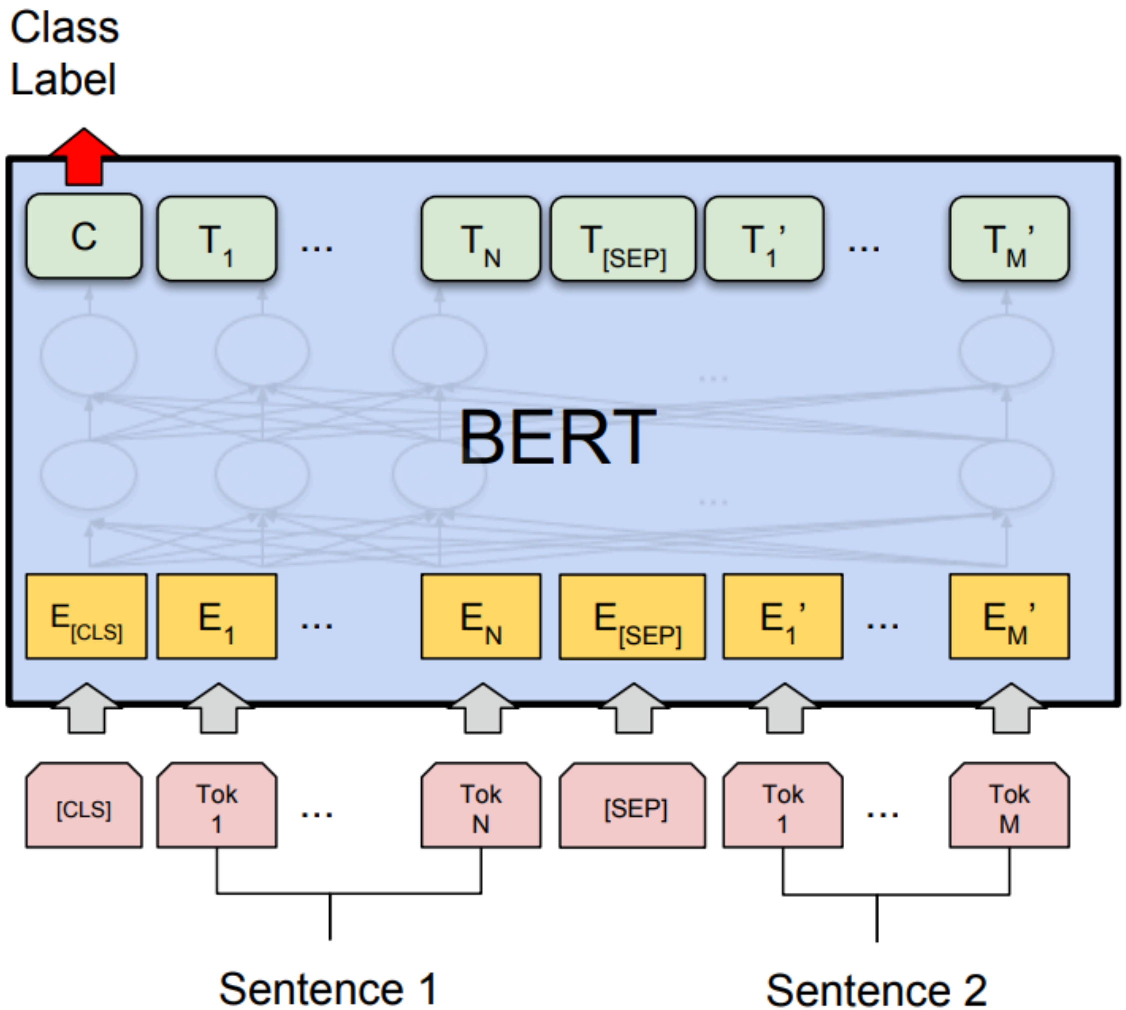
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

BERT

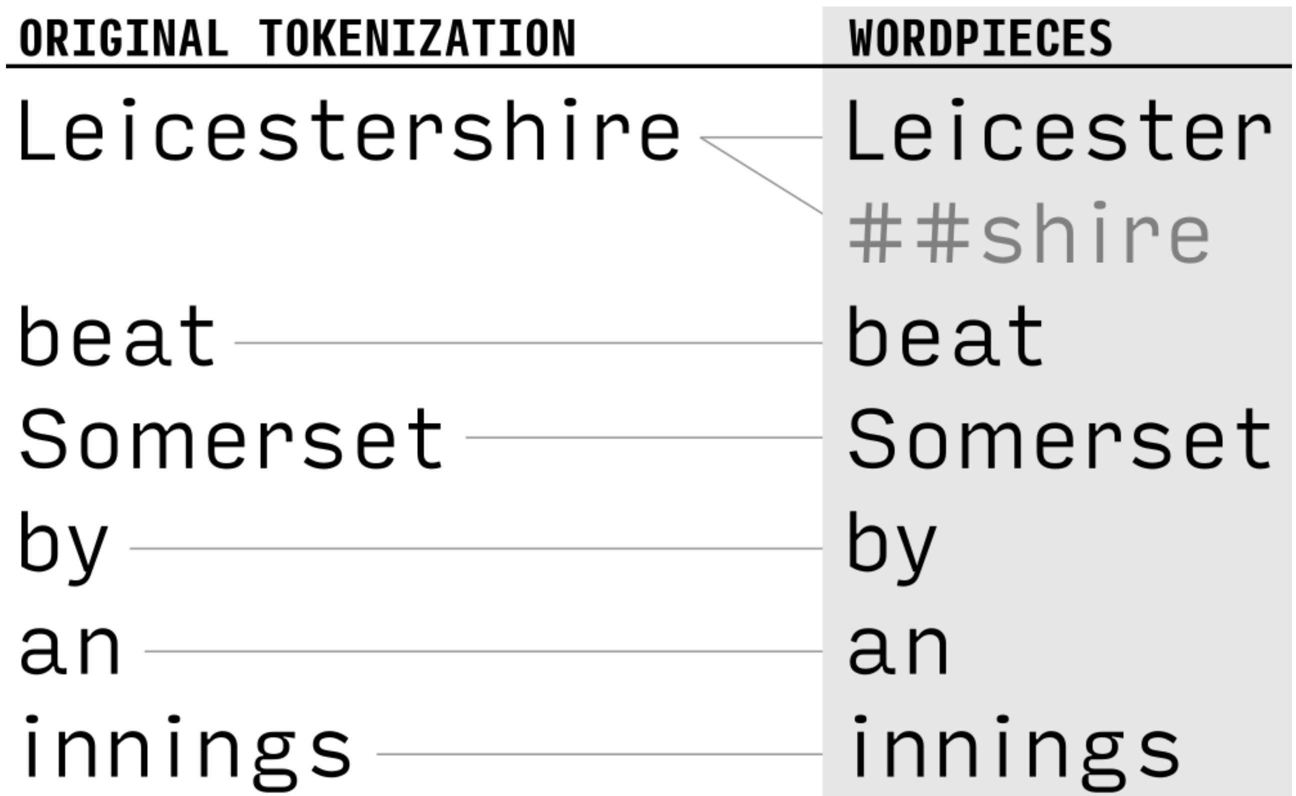


[MASK]
[MASK]
Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$
E_{my}
$E_{[MASK]}$
E_{is}
E_{cute}
$E_{[SEP]}$
E_{he}
$E_{[MASK]}$
E_{play}
$E_{\#\#ing}$
$E_{[SEP]}$
Sentence Embedding
E_A
E_A
E_A
E_A
E_A
E_A
E_B
E_B
E_B
E_B
E_B
Transformer Positional Embedding
E_0
E_1
E_2
E_3
E_4
E_5
E_6
E_7
E_8
E_9
E_{10}

BERT


Class Label
C
T_1
...
T_N
$T_{[SEP]}$
T_1'
...
T_M'
BERT
$E_{[CLS]}$
E_1
...
E_N
$E_{[SEP]}$
E_1'
...
E_M'
[CLS]
Tok 1
...
Tok N
[SEP]
Tok 1
...
Tok M
Sentence 1
Sentence 2

Tokenization (word pieces)

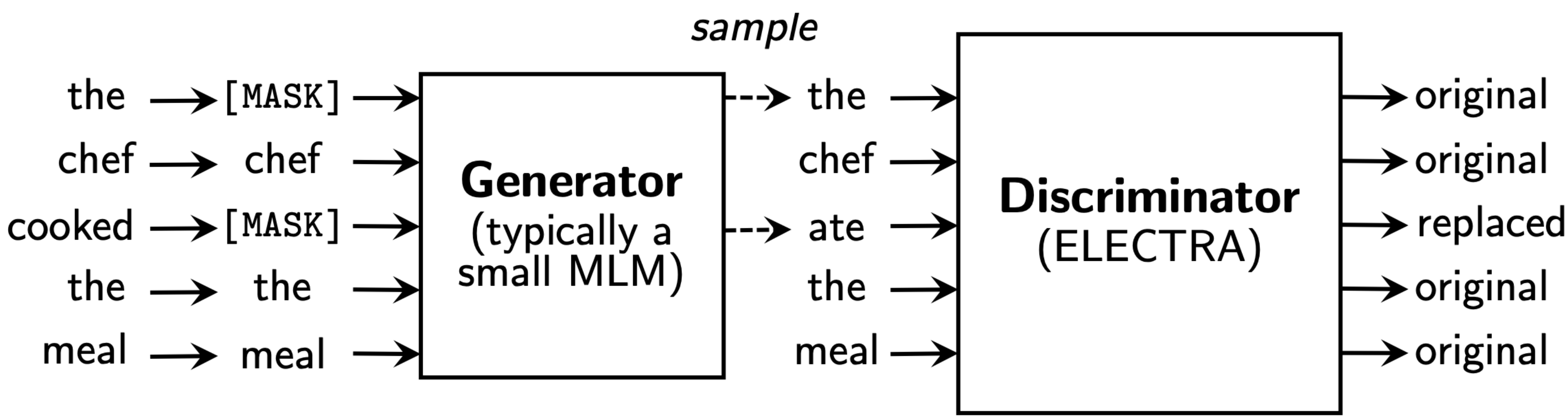


https://github.com/google/sentencepiece/tree/master/python

API

```
>>> import sentencepiece as spm
>>> sp = spm.SentencePieceProcessor(model_file='test/test_model.model')
>>> sp.encode('This is a test')
[284, 47, 11, 4, 15, 400]
>>> sp.encode(['This is a test', 'Hello world'], out_type=int)
[[284, 47, 11, 4, 15, 400], [151, 88, 21, 887]]
>>> sp.encode('This is a test', out_type=str)
['▁This', '▁is', '▁a', '▁', 't', 'est']
>>> sp.encode(['This is a test', 'Hello world'], out_type=str)
[['▁This', '▁is', '▁a', '▁', 't', 'est'], ['▁He', 'll', 'o', '▁world']]
```

ELECTRA



https://openreview.net/pdf?id=r1xMH1BtvB

DATA MINING

/CISC 873 - Steven Ding
/Metric/Loss

Performance Metric

- Not all errors are equally important.
- It depends on what you NEED. (Need-driven)

- Classification:
  - TP, FP, TN, FN, Accuracy, Recall, Precision, Sensitivity, Specificity, AUROC, F1, …
- Prediction (Regression)
  - MSE, MAE, MSLE, R2, …

Confusion Matrix

Real class\Predicted class	C_1	C_2
C_1	True positive (TP)	False negative (FN)
C_2	False positive (FP)	True negative (TN)

Fact\Answer	Yes	no	total
yes	6 (TP)	3 (FN)	9 (P)
no	2 (FP)	3 (TN)	5 (N)
total	8	6	14

- Accuracy: percentage of test set tuples that are correctly classified:
  - $Accuracy = \frac{TP+TN}{P+N} = \frac{6+3}{14}$
  - $Recall = \frac{TP}{P} = \frac{6}{9}$ (percentage of positives correctly answered)
  - $Precision = \frac{TP}{TP+FP} = \frac{6}{6+2}$ (chance that a positive answer is correct)

Confusion Matrix – F1

Real class\Predicted class	C_1	C_2
C_1	True positive (TP)	False negative (FN)
C_2	False positive (FP)	True negative (TN)

Fact\Answer	Yes	no	total
yes	6 (TP)	3 (FN)	9 (P)
no	2 (FP)	3 (TN)	5 (N)
total	8	6	14

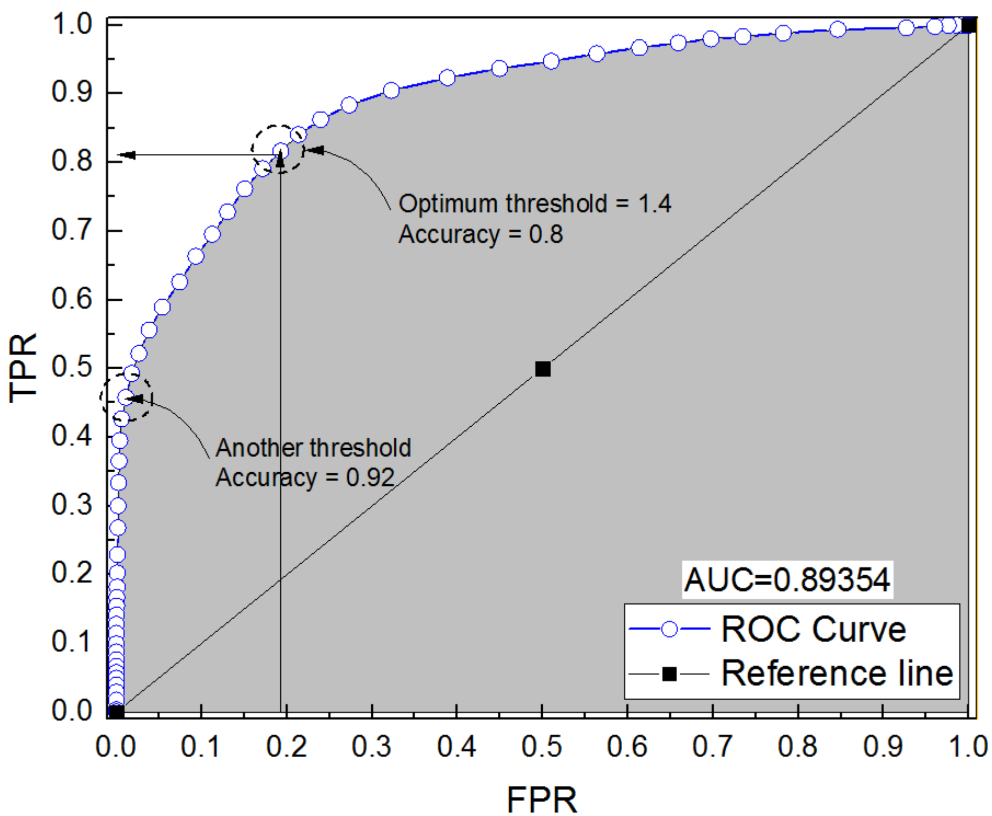
- F measure: the harmonic mean of precision and recall:
  - $F = \frac{2 * Precision * Recall}{Precision + Recall}$

Multi-class Measures

- Micro-average
  - sum up the individual true positives, false positives, and false negatives for different classes and then apply them to get the statistics.
- Macro-average
  - take the average of the chosen metric (e.g. precision, recall etc.) of the system on different classes

AUROC

- Area under receiver operating characteristic curve



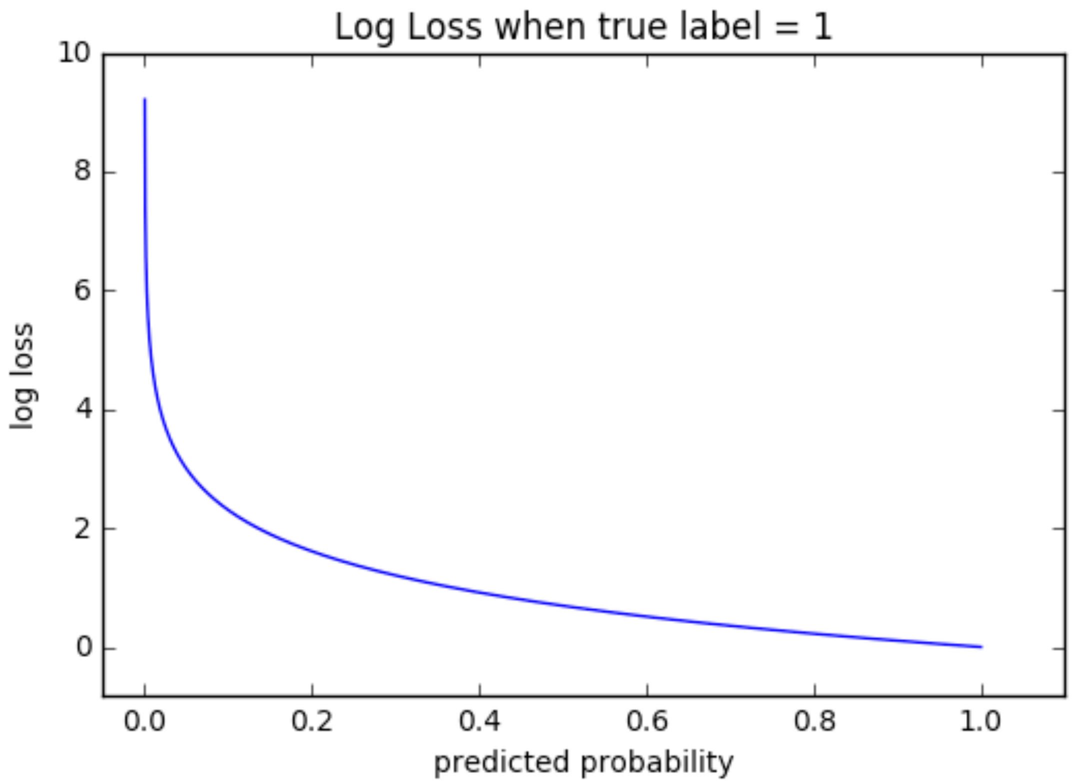
https://stats.stackexchange.com/questions/225210/accuracy-vs-area-under-the-roc-curve

Typical Training Loss

- Classification loss
  - Logistic/Binary Cross Entropy (Binary)
  - Hinge Loss (Binary)
  - KL divergence (Binary)
  - Cross Entropy (Categorical)
  - Sparse Categorical Cross entropy (Categorical)

Typical Training Loss

- Logistic/Binary Cross Entropy (Binary)



$$-(y\log(p) + (1-y)\log(1-p))$$

https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html#cross-entropy

Typical Training Loss

- Logistic/Binary Cross Entropy (Binary)

```
y_true = [[0., 1.], [0., 0.]]
y_pred = [[0.6, 0.4], [0.4, 0.6]]

bce = tf.keras.losses.BinaryCrossentropy()
bce(y_true, y_pred).numpy()
```

https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html#cross-entropy