

Text Generation

Shusen Wang

How Does It Work?

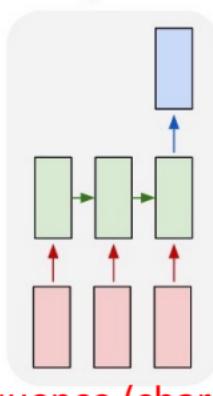
RNN for Text Prediction

Example

- Input text: “a cat sat on the ma”
- Question: what is the next char?

RNN for Text Prediction

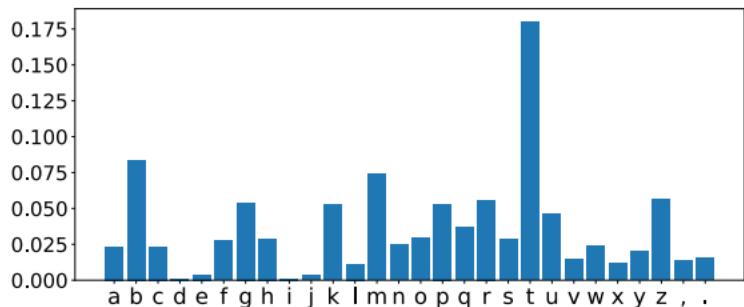
predict the next char



input text

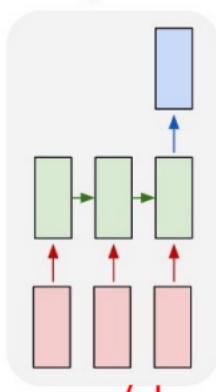
Example

- Input text: “a cat sat on the ma”
- Question: what is the next char?
- RNN outputs a distribution over the chars.



RNN for Text Prediction

predict the next char



sequence (char-level)



input text

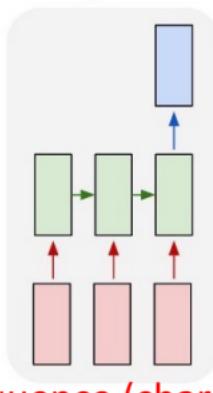
Example

- Input text: “a cat sat on the ma”
- Question: what is the next char?
- RNN outputs a distribution over the chars.
- Sample a char from it; we may get ‘t’.
- Take “a cat sat on the mat” as input.
- Maybe the next char is period ‘.’.

RNN for Text Prediction

How do we train such an RNN?

predict the next char



sequence (char-level)



input text

- Cut text to segments (with overlap).
 - E.g., `seg_len=40` and `stride=3`.

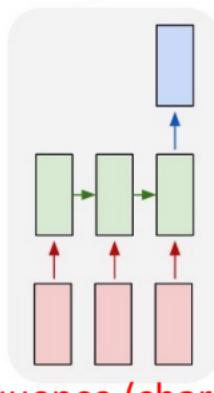
Machine learning is a subset of artificial intelligence in the field of computer science that often uses statistical techniques to give computers the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed. The name machine learning was coined in 1959 by Arthur Samuel.

...

RNN for Text Prediction

How do we train such an RNN?

predict the next char



- Cut text to segments (with overlap).
- Each segment is embedded to a feature vector/matrix.
- Its next char is the label.
- It is a multi-class classification problem.
 - #class = #unique chars.

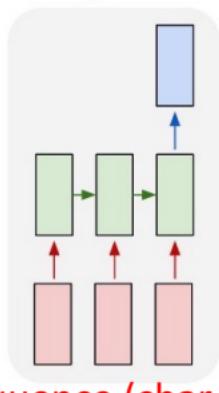
Machine learning is a subset of artificial intelligence in the field of computer science that often uses statistical techniques to give computers the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed. The name machine learning was coined in 1959 by Arthur Samuel.

...
...

RNN for Text Prediction

How do we train such an RNN?

predict the next char



sequence (char-level)

input text

- Cut text to segments (with overlap).
- **Each segment** is embedded to a feature vector/matrix.
- **Its next char** is the label.
- It is a multi-class classification problem.
 - $\# \text{class} = \# \text{unique chars.}$

If the RNN is trained on Shakespeare's books, then the generated text is Shakespeare's style.

Fun with RNNs

Generate academic articles (latex source files).

For $\bigoplus_{i=1,\dots,m} \mathcal{L}_{i,i} = 0$, 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 $U \rightarrow T$ is a separated algebraic space.

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

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

and the comparicoli in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \rightarrow V$. Consider the maps M along the set of points Sch_{fppf} and $U \rightarrow U$ 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 $\text{Sh}(G)$ such that $\text{Spec}(R') \rightarrow S$ 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 $x, x', x'' \in S'$ such that $\mathcal{O}_{X,x'} \rightarrow \mathcal{O}_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\text{GL}_{S'}(x'/S'')$ and we win. \square

To prove study we see that $\mathcal{F}|_U$ is a covering of X' , and 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

$$\tilde{M}^\bullet = I^\bullet \otimes_{\text{Spec}(k)} \mathcal{O}_{S,A} - i_A^{-1} \mathcal{F}$$

is a unique morphism of algebraic stacks. Note that

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

and

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

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. \square

The result for prove any open covering follows from the less of Example ???. It may replace S by $X_{\text{spaces},\text{etale}}$ 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 $X = \lim X_i$ (by the formal open covering X and a single map $\text{Proj}_X(\mathcal{A}) = \text{Spec}(B)$ over U compatible with the complex

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

When in this case of to show that $\mathcal{Q} \rightarrow \mathcal{C}_{Z/X}$ 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 $S = \text{Spec}(R)$ and $Y = \text{Spec}(R)$.

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

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

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 $i_0 \leq p$ is a subset of $\mathcal{J}_{n,0} \circ \tilde{A}_2$ works.

Lemma 0.3. In Situation ???. Hence we may assume $q' = 0$.

Proof. We will use the property we see that \mathfrak{p} is the next 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 . \square

Fun with RNNs

Generate baby names (trained on 8000 baby names).

Rudi Levette Berice Lussa Hany Mareanne Chrestina Carissy Marylen Hammine Janye Marlise Jacacie Hendred Romand Charienna Nenotto Ette Dorane Wallen Marly Darine Salina Elvyn Ersia Maralena Minoria Ellia Charmin Antley Nerille Chelon Walmor Evena Jeryly Stachon Charisa Allisa Anatha Cathanie Geetra Alexie Jerin Cassen Herbett Cossie Velen Daurenge Robester Shermond Terisa Licia Roselen Ferine Jayn Lusine Charyanne Sales Sanny Resa Wallon Martine Merus Jelen Candica Wallin Tel Rachene Tarine Ozila Ketia Shanne Arnande Karella Roselina Alessia Chasty Deland Berther Geamar Jackein Mellisand Sagdy Nenc Lessie Rasemy Guen Gavi Milea Anneda Margoris Janin Rodelin Zeanna Elyne Janah Ferzina Susta Pey Castina

Fun with RNNs

Generate C code (trained on Linux source code).

```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
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(siv->version);
    regs->new = blocks[({BPF_STATS << info->historidac}) | PFMR_CLOBATHINC_SECONDS << 12];
    return segtable;
}
```

Implementation Using Keras

1. Prepare the Training Data

```
print('Text length:', len(text))
```

```
Text length: 600893
```

```
text[0:1000]
```

```
'preface\n\n\nsupposing that truth is a woman--what then? is there  
not ground\nfor suspecting that all philosophers, in so far as they  
have been\nDogmatists, have failed to understand women--that the te  
rrible\nseriousness and clumsy importunity with which they have usu  
ally paid\ntheir addresses to truth, have been unskilled and unseem  
ly methods for\nwinning a woman? certainly she has never allowed he  
rself to be won; and\nat present every kind of dogma stands with sa  
d and discouraged mien--if,\nindeed, it stands at all! for there ar  
e scoffers who maintain that it\nhas fallen, that all dogma lies on  
the ground--nay more, that it is at\nits last gasp. but to speak se  
riously, there are good grounds for hoping\nthat all dogmatizing in  
philosophy, whatever solemn, whatever conclusive\nand decided airs  
it has assumed, may have been only a noble puerilism\nand tyronism;  
and probably the time is at hand when it will be once\nand again un
```

1. Prepare the Training Data

text:

preface\n\n\nsupposing that truth is a woman--what then? is there
not ground\nfor suspecting that all philosophers, in so far as they
have been\nDogmatists, have failed to understand women--that the te
rrible\nseriousness and clumsy importunity with which they have usu
ally paid\ntheir addresses to truth, have been unskilled and unseem

segments[0]: preface\n\n\nsupposing that truth **next_chars[0]:** i

1. Prepare the Training Data

text:

```
'preface\n\n\nsupposing that truth is a woman--what then? is there  
not ground\nfor suspecting that all philosophers, in so far as they  
have been\nDogmatists, have failed to understand women--that the te  
rrible\nseriousness and clumsy importunity with which they have usu  
ally paid\ntheir addresses to truth, have been unskilled and unseem
```

segments[0]: preface\n\n\nsupposing that truth

next_chars[0]: i

segments[1]: face\n\n\nsupposing that truth is

next_chars[1]: a

1. Prepare the Training Data

text:

```
'preface\n\n\nsupposing that truth is a woman--what then? is there  
not ground\nfor suspecting that all philosophers, in so far as they  
have been\nDogmatists, have failed to understand women--that the te  
rrible\nseriousness and clumsy importunity with which they have usu  
ally paid\ntheir addresses to truth, have been unskilled and unseem
```

segments[0]: preface\n\n\nsupposing that truth

next_chars[0]: i

segments[1]: face\n\n\nsupposing that truth is

next_chars[1]: a

segments[2]: e\n\n\nsupposing that truth is a w

next_chars[2]: o

1. Prepare the Training Data

text:

```
'preface\n\n\nsupposing that truth is a woman--what then? is there  
not ground\nfor suspecting that all philosophers, in so far as they  
have been\nDogmatists, have failed to understand women--that the te  
rrible\nseriousness and clumsy importunity with which they have usu  
ally paid\ntheir addresses to truth, have been unskilled and unseem
```

segments[0]: preface\n\n\nsupposing that truth

next_chars[0]: i

segments[1]: face\n\n\nsupposing that truth is

next_chars[1]: a

segments[2]: e\n\n\nsupposing that truth is a w

next_chars[2]: o

segments[3]: \n\nsupposing that truth is a woma

next_chars[3]: n

⋮

1. Prepare the Training Data

```
seg_len = 60
stride = 3
segments = []
next_chars = []

for i in range(0, len(text) - seg_len, stride):
    segments.append(text[i: i + seg_len])
    next_chars.append(text[i + seg_len])

print('Number of segments:', len(segments))
```

Number of segments: 200278

2. Tokenize the Data

```
from keras.preprocessing.text import Tokenizer

tokenizer = Tokenizer(char_level=True, filters='#$%&*+@/=>^_')
tokenizer.fit_on_texts(text)
```

2. Tokenize the Data

```
from keras.preprocessing.text import Tokenizer

tokenizer = Tokenizer(char_level=True, filters='#$%&*+@/=>^_')
tokenizer.fit_on_texts(text)

from keras.preprocessing.sequence import pad_sequences
list[list[int]]
    sequences_raw = tokenizer.texts_to_sequences(segments) list[string]
    sequences = pad_sequences(sequences_raw, maxlen=seg_len)
integer matrix
print(sequences.shape)

(200278, 60)

print(sequences[0, :])

[18  9  2 15  5 13  2 17 17 17  8 14 18 18  6  8  4  7 19  1  3 10  5  3
 1  3  9 14  3 10  1  4  8  1  5  1 21  6 16  5  7 25 25 21 10  5  3  1
 3 10  2  7 35  1  4  8  1  3 10  2]
```

2. Tokenize the Data

```
char_index = tokenizer.word_index
print(char_index)

{' ': 1, 'e': 2, 't': 3, 'i': 4, 'a': 5, 'o': 6, 'n': 7, 's': 8,
'r': 9, 'h': 10, 'l': 11, 'd': 12, 'c': 13, 'u': 14, 'f': 15, 'm':
16, '\n': 17, 'p': 18, 'g': 19, 'y': 20, 'w': 21, ',': 22, 'b': 23,
've': 24, '-': 25, '.': 26, 'k': 27, "'": 28, 'x': 29, ':': 30, ';':
31, 'j': 32, 'q': 33, '!': 34, '?': 35, '(': 36, ')': 37, '1': 38,
'=': 39, '"': 40, 'z': 41, '2': 42, '_': 43, '3': 44, '4': 45, '5':
46, '8': 47, '6': 48, '7': 49, '[': 50, ']': 51, '9': 52, '0': 53,
'ä': 54, 'æ': 55, 'é': 56, 'ë': 57}
```

2. Tokenize the Data

```
char_index = tokenizer.word_index
print(char_index)

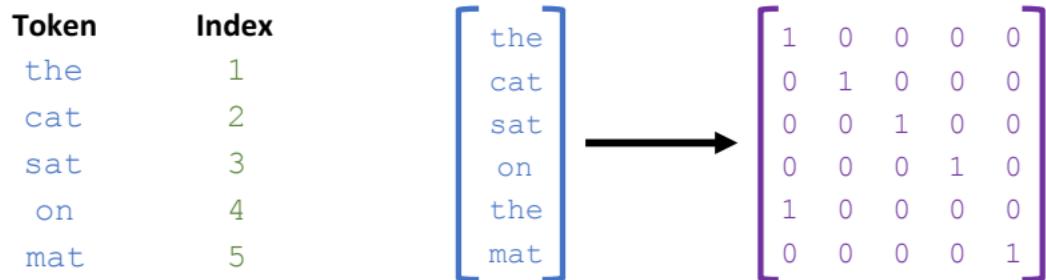
{' ': 1, 'e': 2, 't': 3, 'i': 4, 'a': 5, 'o': 6, 'n': 7, 's': 8,
'r': 9, 'h': 10, 'l': 11, 'd': 12, 'c': 13, 'u': 14, 'f': 15, 'm':
16, '\n': 17, 'p': 18, 'g': 19, 'y': 20, 'w': 21, ',': 22, 'b': 23,
've': 24, '-': 25, '.': 26, 'k': 27, "'": 28, 'x': 29, ':': 30, ';':
31, 'j': 32, 'q': 33, '!': 34, '?': 35, '(': 36, ')': 37, '1': 38,
'=': 39, '"': 40, 'z': 41, '2': 42, '_': 43, '3': 44, '4': 45, '5':
46, '8': 47, '6': 48, '7': 49, '[': 50, ']': 51, '9': 52, '0': 53,
'ä': 54, 'æ': 55, 'é': 56, 'ë': 57}

inv_char_index = {v: k for k, v in char_index.items()}
print(inv_char_index)

{1: ' ', 2: 'e', 3: 't', 4: 'i', 5: 'a', 6: 'o', 7: 'n', 8: 's', 9:
'r', 10: 'h', 11: 'l', 12: 'd', 13: 'c', 14: 'u', 15: 'f', 16: 'm',
17: '\n', 18: 'p', 19: 'g', 20: 'y', 21: 'w', 22: ',', 23: 'b', 24:
've', 25: '-', 26: '.', 27: 'k', 28: "'", 29: 'x', 30: ':', 31: ';',
32: 'j', 33: 'q', 34: '!', 35: '?', 36: '(', 37: ')', 38: '1', 39:
'=', 40: '"', 41: 'z', 42: '2', 43: '_', 44: '3', 45: '4', 46: '5',
47: '8', 48: '6', 49: '7', 50: '[', 51: ']', 52: '9', 53: '0', 54:
'ä', 55: 'æ', 56: 'é', 57: 'ë'}
```

2. Tokenize the Data

One-hot encode

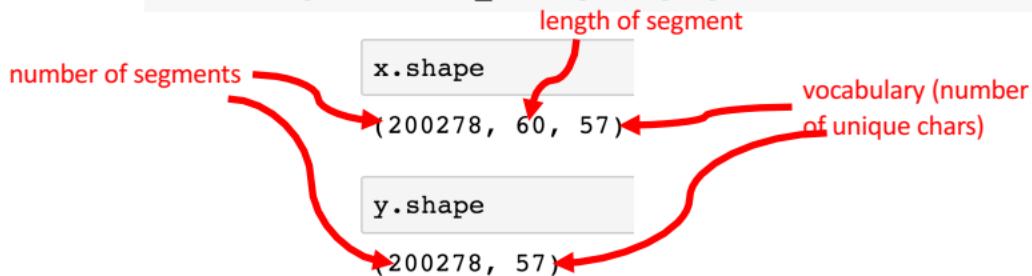


2. Tokenize the Data

```
vocabulary = len(char_index)
seg_num = len(segments)

# one-hot encode
x = np.zeros((seg_num, seg_len, vocabulary), dtype=np.bool)
y = np.zeros((seg_num, vocabulary), dtype=np.bool)

for i, seg in enumerate(segments):
    y[i, char_index[next_chars[i]]-1] = 1
    for t, char in enumerate(seg):
        x[i, t, char_index[char]-1] = 1
```



3. Build an LSTM Network

```
from keras import layers

model = keras.models.Sequential()
model.add(layers.LSTM(128, input_shape=(seg_len, vocabulary)))
model.add(layers.Dense(vocabulary, activation='softmax'))
model.summary()
```

Layer (type)	Output Shape	Param #
lstm_1 (LSTM)	(None, 128)	95232
dense_1 (Dense)	(None, 57)	7353
<hr/>		
Total params: 102,585		
Trainable params: 102,585		
Non-trainable params: 0		

3. Build an LSTM Network

Train the LSTM network

- Input: a segment of 60 chars (represented by a 60×57 matrix).
- Label: the next char (represented by a 57-dim vector).

```
optimizer = keras.optimizers.RMSprop(lr=0.01)
model.compile(loss='categorical_crossentropy', optimizer=optimizer)
```

```
model.fit(x, y, batch_size=128, epochs=1)
```

```
Epoch 1/1
200278/200278 [=====] - 117s 586us/step -
loss: 1.6746
```

4. Generate Text Using the Trained LSTM

Predict the next char

```
pred = model.predict(x_input, verbose=0)[0]
```



- A segment of 60 chars (represented by a 60×57 matrix).
- The “seed” for text generation.

The probability distribution over the 57 unique chars.

4. Generate Text Using the Trained LSTM

Predict the next char

```
pred = model.predict(x_input, verbose=0)[0]
```

Question: Given the distribution, what will be the next char?

- Option 1: greedy selection.
 - `next_index = np.argmax(pred)`
 - It is deterministic.
 - Empirically not good.

4. Generate Text Using the Trained LSTM

Predict the next char

```
pred = model.predict(x_input, verbose=0)[0]
```

Question: Given the distribution, what will be the next char?

- Option 1: greedy selection.
- Option 2: sample from the multinomial distribution.
 - `next_onehot = np.random.multinomial(1, pred, 1)`
 - `next_index = np.argmax(next_onehot)`
 - Maybe too random.

4. Generate Text Using the Trained LSTM

Predict the next char

```
pred = model.predict(x_input, verbose=0)[0]
```

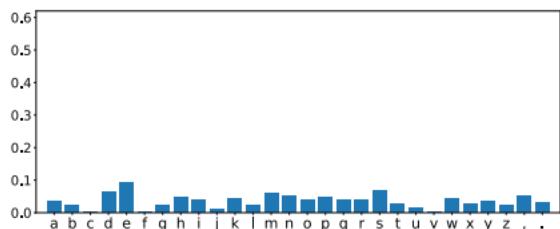
Question: Given the distribution, what will be the next char?

- Option 1: greedy selection.
- Option 2: sample from the multinomial distribution.
- Option 3: adjust the multinomial distribution.
 - `pred = pred ** (1 / temperature)`
 - `pred = pred / np.sum(pred)`
 - Sample according to pred.
 - Between greedy and multinomial (controlled `temperature`).

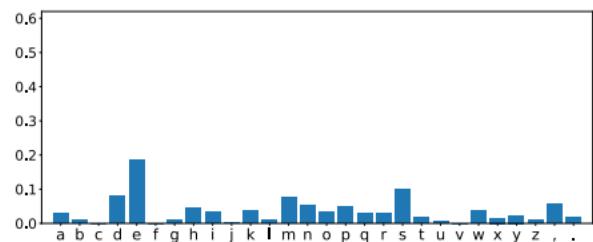
4. Generate Text Using the Trained LSTM

Option 3: adjust the multinomial distribution.

- $\text{pred} = \text{pred} ^\star (1 / \text{temperature})$
- $\text{pred} = \text{pred} / \text{np.sum(pred)}$
- Between greedy and multinomial (controlled **temperature**).



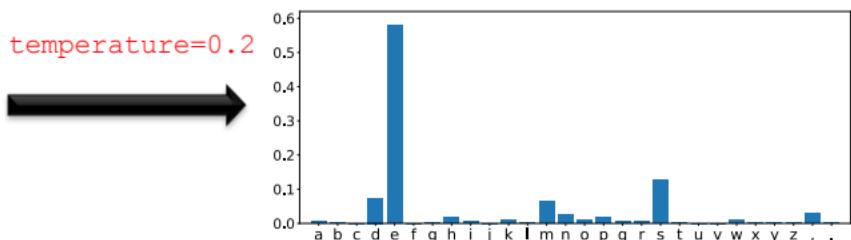
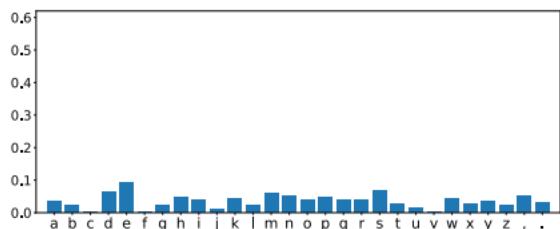
temperature=0.5



4. Generate Text Using the Trained LSTM

Option 3: adjust the multinomial distribution.

- $\text{pred} = \text{pred} ^\star (1 / \text{temperature})$
- $\text{pred} = \text{pred} / \text{np.sum(pred)}$
- Between greedy and multinomial (controlled **temperature**).



4. Generate Text Using the Trained LSTM

Example: seg_len=18

initial_input (seed) : "the cat sat on the"

next_char: '_' (blank)

4. Generate Text Using the Trained LSTM

Example: seg_len=18

initial_input (seed): “**t**he cat sat on the”

next_char: ‘ ’ (blank)

input: “he cat sat on the_”

next_char: ‘m’

4. Generate Text Using the Trained LSTM

Example: seg_len=18

initial_input (seed) : "the cat sat on the"

next_char: '_' (blank)

input: "he cat sat on the "

next_char: 'm'

input: "e cat sat on the m"

next_char: 'a'

4. Generate Text Using the Trained LSTM

Example: seg_len=18

initial_input (seed): "the cat sat on the"

next_char: '_' (blank)

input: "he cat sat on the "

next_char: 'm'

input: "e cat sat on the m"

next_char: 'a'

input: " cat sat on the ma"

next_char: 't'

4. Generate Text Using the Trained LSTM

Initial input (seed)

"immediately disclose what it really is--namely,
a will to the"

Generated text after 1 epoch

"immediately disclose what it really is--namely,
a will to the the believest constive the art of
the persision the says of a gan himself need not
religion a consting be naturing and suld the
pretendents as the constion. the are the good
teach that the most and find of endent and the
self it of instinct a mean constition of can the
constive in this sour from the bad and all the
philosophy to condividuation men and the goence
the condines the all as most strat"

4. Generate Text Using the Trained LSTM

Initial input (seed)

"immediately disclose what it really is--namely,
a will to the"

Generated text after 1 epoch (another sample)

"immediately disclose what it really is--namely,
a will to thes and in the world the mist a dest
reality and lading the been in the most as and
fanition as the reaption of the stenles a prease
and be and disting and regard the goven the most
and distentive to the from stinct forms and
instinct the believes the need itself the feess
and virtue this says of the gone consting to man
instinctian the become and discative of the
present the free the consine of pas"

4. Generate Text Using the Trained LSTM

Initial input (seed)

"immediately disclose what it really is--namely,
a will to the"

Generated text after 5 epoch

"immediately disclose what it really is--namely,
a will to the all and power and that the **complises**
the fundamental and emotions of the **development**
of the **propest** and sympathy of the far the long
standard, and the most present--under the
personally man hard and every stands have been
hat the soul and conscience, the intellectation of
germans of should for the religious profoundly in
this included and naturally as we **progres** and
"will of the most same same and"

4. Generate Text Using the Trained LSTM

Initial input (seed)

"immediately disclose what it really is--namely,
a will to the"

Generated text after 5 epoch (another sample)

"immediately disclose what it really is--namely,
a will to theexample for should and wholling been
are the our at and instinction that is the
consideration of himself, and specially the
present praction of which the course, of the
proportangs, in the really in this fortain of the
valued and even all things the same and
assimplishing in good and instincts in which a
god of the farseed to the same perhaps who has
the same the sentempul and soul individual the"

4. Generate Text Using the Trained LSTM

Initial input (seed)

"immediately disclose what it really is--namely,
a will to the"

Generated text after 20 epoch

"immediately disclose what it really is--namely,
a will to the self-religious superstitial self-
partial religion himself of the superstition of
the contrast in the account in the person of the
assertion, at the valuations and consequences and
things has no longer and how only an man of the
soul and manifest of the disciplines and an whom
all to the constance of its own power, in the
constraining in the truth of the strength of life
of the see to remain in the"

4. Generate Text Using the Trained LSTM

Initial input (seed)

"immediately disclose what it really is--namely,
a will to the"

Generated text after 20 epoch (another sample)

"immediately disclose what it really is--namely,
a will to the science of the original he who has
not in the truth of the point, has not words from
the devilished and sensible and discerence to the
far an morals: the desire to a such as to the
sexual of the morality and their world and with
an ought of the morally and self-possession of
the presumption, of philosophy the sense of the
conduct of the and oppression that a man of a
delicate and can be a man of h"