Continues Bag-of-words Model

* 姓名: 管仁阳　学号:519021911058　邮箱: guanrenyang@sjtu.edu.cn

1 CBOW 模型

本部分简要介绍连续词袋（CBOW: Continues Bag-of-words）模型的提出的直觉，并推导 CBOW 模型正向传播与参数更新公式。1.1节从传统的单热（one-hot）词向量出发介绍了 CBOW 模型的优势；1.2节推导了 CBOW 模型正向计算公式；1.3节运用反向传播算法推导了 CBOW 模型的参数更新公式。其中公式推导主要参考文献 [1]。

1.1 词向量

首先从最原始的词向量——单热词向量（one-hot）出发。假设词 ω_i 的单热词向量为 $\boldsymbol{x}$，它的长度为词表长度 V，其中 $x_i = 1$ 当且仅当 ω_i 在词表中的索引为 i。但是单热词向量有几个主要问题：

1. 单热向量未考虑不同词之间的联系。
2. 当词表大小 V 过大时会导致输入向量维度爆炸。

CBOW 模型的目的时通过机器学习方法学习出词 ω_I 的词向量，其中词向量的维度 N 为一个可调的超参数值。CBOW 模型假设词 ω_i 出现的概率依赖于其上下文中的 $2k$ 个词，即

$$p^{\omega_i} = \hat{P}(\omega_i|\omega_{i-k}, \cdots, \omega_{i-1}, \omega_{i+1}, \cdots, \omega_{i+k})$$

CBOW 模型的任务就是根据上下文中的 C 个词预测当前词 ω_i 出现的概率。

1.2 正向计算

首先考虑 CBOW 模型的最简情况——上下文中只包含一个词，换言之，词 x 出现的条件概率只与上下文中的一个词 y 相关，即

$$p(x) = p(x|y) \tag{1}$$

假设词表长度为 V，隐藏层大小为 N，相邻层之间是全连接网络。网络的输入为单热向量，即对于长度为词表长度 V 的输入向量 $[x_1, \cdots, x_V]$ 来说，只有一个元素是 1，其他元素都是 0。

输入层和隐藏层之间的权重矩阵为 $V \times N$ 维矩阵 W。W 的每一行为一个 N 维向量，可以将其看作输入词的表征向量。如果输入词为 ω_I，其单热词向量满足 $x_k = 1$ 且对于 $k' \neq k$ 有 $x_{k'} = 0$，那么

$$\boldsymbol{h} = \boldsymbol{W}^T\boldsymbol{x} = \boldsymbol{W}^T_{(k,\cdot)} := \boldsymbol{v}^T_{\omega_I} \tag{2}$$

本质上就是将权重矩阵 $\boldsymbol{W}$ 的第 k 行赋值给隐层向量 $\boldsymbol{h}$，$\boldsymbol{v}^T_{\omega_I}$ 是词 ω_I 的表征。

对于上下文中含有多个词的 CBOW 模型来说，假设 $\{x_1, x_2, \cdots, x_C\}$ 是与当前词相关的 C 个词的单热词向量，为了减小输入向量的维度，CBOW 模型的输入是这 C 个单热向量的均值，这本质上是对上下文中 C 个词的表征向量求平均，即

$$\boldsymbol{h} = \frac{1}{C}\boldsymbol{W}^T(\boldsymbol{x}_1 + \boldsymbol{x}_2 + \cdots + \boldsymbol{x}_C) \tag{3}$$

$$= \frac{1}{C}(\boldsymbol{v}_{\omega_1} + \boldsymbol{v}_{\omega_2} + \cdots + \boldsymbol{v}_{\omega_C})^T \tag{4}$$

$$= \boldsymbol{W}^T\bar{\boldsymbol{x}} \tag{5}$$

其中 $\bar{\boldsymbol{x}} = \frac{1}{C}(\boldsymbol{x}_1 + \boldsymbol{x}_2 + \cdots + \boldsymbol{x}_C)$，是多个输入单热向量的均值。隐藏层与输出层之间的权重矩阵 $W' = \{\omega'_{ij}\}$ 是一个与 W 不同的 $N \times V$ 维矩阵，由此可以得出输出层向量为

$$\boldsymbol{u} = \boldsymbol{W}'^T \boldsymbol{h} \tag{6}$$

$\boldsymbol{u}$ 是一个长度为 V 的向量，其中任一个元素 u_j 表示词表中的词 ω_j 的打分，可以通过下式计算 u_j:

$$u_j = \boldsymbol{v}'^T_{\omega_j} \boldsymbol{h} \tag{7}$$

其中 $\boldsymbol{v}'_{\omega_j}$ 为矩阵 $\boldsymbol{W}'$ 的第 j 列。将 $\boldsymbol{u}$ 输入一个对数线性分类器 $softmax$，得到给定输出词 ω_I 的后验概率分布

$$p(\omega_j|\omega_I) = y_i = \frac{\exp(u_j)}{\sum_{j'=1}^{V} \exp(u_{j'})} \tag{8}$$

其中 y_j 是 V 维输出层向量 $\boldsymbol{y}$ 的第 j 个元素。最后我们使用矩阵形式总结正向计算过程

1. 输入层 → 隐藏层

$$\boldsymbol{h} = \frac{1}{C}\boldsymbol{W}^T(\boldsymbol{x}_1 + \boldsymbol{x}_2 + \cdots + \boldsymbol{x}_C) \tag{9}$$

2. 隐藏层 → 输出层

$$y = \text{softmax}(\boldsymbol{u}) = \text{softmax}(\boldsymbol{W}'^T \boldsymbol{h}) \tag{10}$$

1.3 参数更新

在上下文只包含一个词的最简 CBOW 模型中，对于一个训练样本来说，训练目标是最大化后验概率 $p(\omega_j|\omega_I)$，即给定输入上下文 ω_I 的条件下观察到真实输出词 ω_O 的条件概率（ω_O 在词表中的索引为 j^*）。

$$\begin{aligned} \max p(\omega_O|\omega_I) &= \max y_{j^*} \\ &= \max \log y_{j^*} \\ &= u_{j^*} - \log \sum_{j'=1}^{V} \exp(u_{j'}) := -E \end{aligned} \tag{11}$$

其中 $E = -\log p(\omega_O|\omega_I)$ 是损失函数，训练的目标就是最小化 E；j^* 是输出层的真实输出词的索引。

对于上下文中含有多个词的一般 CBOW 模型来说，假设上下文为 $\{\omega_{I,1}, \omega_{I,2}, \cdots, \omega_{I,C}\}$，则损失函数变为

$$\begin{aligned} E &= -\log p(\omega_O|\omega_{I,1}, \cdots, \omega_{I,C}) \\ &= -u_{j^*} + \log \sum_{j^*=1}^{V} \exp(u_{j'}) \\ &= -\boldsymbol{v}'^T_{\omega_O} \cdot \boldsymbol{h} + \log \sum_{j'=1}^{V} \exp(\boldsymbol{v}'_{\omega_j} \cdot \boldsymbol{h}) \end{aligned} \tag{12}$$

一般 CBOW 模型的损失函数与公式 11一致，区别只在于 $\boldsymbol{h}$ 的计算方法不同。在具体推导前先给出 CBOW 模型的参数更新公式的矩阵形式

1. 输入层 → 隐藏层

$$\begin{aligned}\boldsymbol{W}^{(new)} &= \boldsymbol{W}^{(old)} - \frac{\eta}{C}(\boldsymbol{x}_1 + \boldsymbol{x}_2 + \cdots + \boldsymbol{x}_C)\mathrm{EH}^T \\ &= \boldsymbol{W}^{(old)} - \eta\ \bar{\boldsymbol{x}}\ \mathrm{EH}^T\end{aligned} \tag{13}$$

2. 隐藏层 → 输出层

$$\boldsymbol{W}'^{\ (new)} = \boldsymbol{W}'^{\ (old)} - \eta\ \boldsymbol{h}\boldsymbol{e}^T \tag{14}$$

1.3.1 隐藏层 → 输出层

在定义了损失函数以后，就可以利用反向传播算法推导出参数更新公式。首先计算 E 对输出层第 j 的元素 u_j 的偏导数

$$\frac{\partial E}{\partial u_j} = y_j - t_j := e_j \tag{15}$$

其中 $\boldsymbol{t}$ 也是一个长度为词表长度 V 的向量，满足 $t_j = \mathbf{1}(j = j^*)$，即 $t_j = 1$ 当且仅当词表中第 j 个词是要预测的真实标签，其余位置都有 $t_{j'}|_{j'\neq j} = 0$；$e_j = y_j - t_j$ 是输出层的预测误差。

接下来计算 E 对 ω'_{ij} 的偏导数，就可以得到隐藏层到输出层的权重的梯度

$$\frac{\partial E}{\partial \omega'_{ij}} = \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial \omega'_{ij}} = e_j \cdot h_i \tag{16}$$

使用梯度下降法（SGD）可以得到隐藏层到输出层之间的参数更新公式

$$\omega'^{\ (new)}_{i,j} = \omega'^{\ (old)}_{ij} - \eta \cdot e_j \cdot h_i. \tag{17}$$

其中 η 是学习率，满足 $\eta > 0$；$e_j = y_j - t_j$；h_i 是隐藏层向量 $\boldsymbol{h}$ 的第 i 个元素。

1.3.2 输入层 → 隐藏层

再得到权重矩阵 W' 的更新公式以后就要可以推导 W 的更新方程。首先求 E 对隐层向量 $\boldsymbol{h}$ 的偏导数

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{V} \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial h_i} = \sum_{j=1}^{V} e_j \cdot \omega'_{ij} := \mathrm{EH}_i \tag{18}$$

其中 h_i 为隐藏层的第 i 个元素；u_j 是输出层的第 j 个元素；$e_j = y_j - t_j$ 是输出层第 j 个词的预测误差；N 维向量 EH 是词表中所有词的输出向量对预测误差的加权和。

下面计算 E 对输入层到隐藏层的参数 $\boldsymbol{W}$。首先将公式 3展开为

$$h_i = \sum_{k=1}^{V} \bar{x}_k \cdot \omega_{ki} \tag{19}$$

那么 E 对 $\boldsymbol{W}$ 的偏导数为

$$\frac{\partial E}{\partial \omega_{ki}} = \frac{\partial E}{\partial h_i} \cdot \frac{\partial h_i}{\partial \omega_{ki}} = \mathrm{EH}_i \cdot \bar{x}_k \tag{20}$$

将上式用矩阵表示即为

$$\frac{\partial E}{\partial \boldsymbol{W}} = \bar{\boldsymbol{x}}\mathrm{EH}^T \tag{21}$$

由此可以得到一个维度为 $V \times N$ 的权重矩阵。由于 $\boldsymbol{x}$ 中只有 C 个元素非零，$\frac{\partial E}{\partial \boldsymbol{W}}$ 中只有 C 行非零，我们也就可以得到更新权重更新公式

$$\boldsymbol{v}^{(new)}_{\omega_I} = \boldsymbol{v}^{(old)}_{\omega_I} - \eta\mathrm{EH}^T \tag{22}$$

或写成矩阵形式

$$\boldsymbol{W}^{(new)} = \boldsymbol{W}^{(old)} - \eta\bar{\boldsymbol{x}}\mathrm{EH}^{\mathrm{T}} \tag{23}$$

2 代码实现

这部分首先给出一次迭代的所有代码，如图 1所示。接下来逐个行解释 train_one_epoch 函数的实现方式

Line

69 根据上下文词列表 `vocab` 为每个词创建对应的单热向量，并求这些单热向量的和值 `x_bar`)。

71 将真实标签转化为单热向量，亦即计算在公式 15中首次提出的向量 $\boldsymbol{t}$。

74 输入层 → 隐藏层正向计算，根据公式 9。

75 隐藏层 → 输出层正向计算，根据公式 10，得到词表中每个词的分数值 $\boldsymbol{u}$，$\boldsymbol{u}$ 在公式 6 中定义。

79 softmax 分类器，根据公式 8。

80 计算损失函数，根据公式 12。

83 计算预测误差 $\boldsymbol{e}$，$\boldsymbol{e}$ 在公式 15中定义。

84 计算输出向量对预测误差的加权和 EH，根据公式 18。

86 隐藏层 → 输出层参数更新，根据公式 14。

87 输入层 → 隐藏层参数更新，根据公式 13。

```
58  def train_one_step(self, context_tokens: List[str], target_token: str, learning_rate: float) -> float:
59      """
60      Predict the probability of the target token given context tokens.
61
62      :param context_tokens:  List of tokens around the target token
63      :param target_token:    Target (center) token
64      :param learning_rate:   Learning rate of each step
65      :return:    loss of the target token
66      """
67      # ==== TODO: Construct one-hot vectors ====
68      # average of one hot vectors of `context_tokens`
69      x_bar = (sum([one_hot(len(self.vocab), self.vocab.token_to_idx(item)) for item in context_tokens])).reshape(-1, 1) # (V,1)
70      # ground true
71      t = one_hot(len(self.vocab), self.vocab.token_to_idx(target_token)).reshape(-1, 1) # (V,1)
72
73      # ==== TODO: Forward step ====
74      h = (self.W1.transpose() @ x_bar)/len(context_tokens) # (N,1)
75      u = self.W2.transpose() @ h # (V,1)
76
77      # ==== TODO: Calculate loss ====
78      # prediction
79      y = softmax(u) # (V,1)
80      loss = -np.log(softmax(u))[t==1].squeeze() #(V,1)
81
82      # ==== TODO: Update parameters ====
83      e = y - t
84      eh = self.W2 @ e # (N,1)
85      #update
86      self.W2 = self.W2 - learning_rate * (h @ e.transpose())
87      self.W1 = self.W1 - learning_rate / len(context_tokens) * (x_bar @ eh.transpose())
88
89      return loss
```

图 1: train_one_epoch 函数

3 输出结果

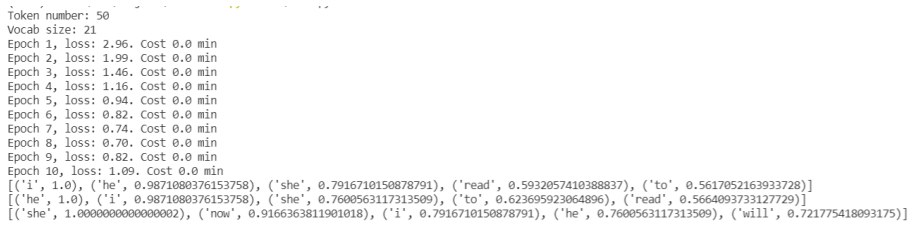

图 2: 全输出-2

图 3: 全输出-2

图 4: 全输出-2

如上代码可以通过全部三个测试点，如图 2、图 3和图 4所示。程序所有的输出截图见图 5和 6。

参考文献

[1] Rong X. word2vec parameter learning explained[J]. arXiv preprint arXiv:1411.2738, 2014.

```
Token number: 50
Vocab size: 21
Epoch 1, loss: 2.96. Cost 0.0 min
Epoch 2, loss: 1.99. Cost 0.0 min
Epoch 3, loss: 1.46. Cost 0.0 min
Epoch 4, loss: 1.16. Cost 0.0 min
Epoch 5, loss: 0.94. Cost 0.0 min
Epoch 6, loss: 0.82. Cost 0.0 min
Epoch 7, loss: 0.74. Cost 0.0 min
Epoch 8, loss: 0.70. Cost 0.0 min
Epoch 9, loss: 0.82. Cost 0.0 min
Epoch 10, loss: 1.09. Cost 0.0 min
[('i', 1.0), ('he', 0.9871080376153758), ('she', 0.7916710150878791), ('read', 0.5932057410388837), ('to', 0.5617052163933
728)]
[('he', 1.0), ('i', 0.9871080376153758), ('she', 0.7600563117313509), ('to', 0.623695923064896), ('read', 0.56640937331277
29)]
[('she', 1.0000000000000002), ('now', 0.9166363811901018), ('i', 0.7916710150878791), ('he', 0.7600563117313509), ('will',
 0.7217754180093175)]

Test-1 pass :)

Token number: 205068
Vocab size: 17832
Step: 10000. Avg. loss:  9.85
Step: 20000. Avg. loss:  9.80
Step: 30000. Avg. loss:  9.69
Step: 40000. Avg. loss:  9.53
Step: 50000. Avg. loss:  9.42
Step: 60000. Avg. loss:  9.30
Step: 70000. Avg. loss:  9.20
Step: 80000. Avg. loss:  9.12
Step: 90000. Avg. loss:  9.04
Step: 100000. Avg. loss:  8.97
Step: 110000. Avg. loss:  8.91
Step: 120000. Avg. loss:  8.86
Step: 130000. Avg. loss:  8.81
Step: 140000. Avg. loss:  8.77
Step: 150000. Avg. loss:  8.73
Step: 160000. Avg. loss:  8.69
Step: 170000. Avg. loss:  8.65
Step: 180000. Avg. loss:  8.61
Step: 190000. Avg. loss:  8.58
Step: 200000. Avg. loss:  8.55
Epoch 1, loss: 8.54. Cost 24.7 min
Save model to ckpt
Step: 10000. Avg. loss:  7.83
Step: 20000. Avg. loss:  7.87
Step: 30000. Avg. loss:  7.85
Step: 40000. Avg. loss:  7.83
Step: 50000. Avg. loss:  7.84
Step: 60000. Avg. loss:  7.82
Step: 70000. Avg. loss:  7.80
Step: 80000. Avg. loss:  7.80
Step: 90000. Avg. loss:  7.79
Step: 100000. Avg. loss:  7.78
Step: 110000. Avg. loss:  7.77
Step: 120000. Avg. loss:  7.76
Step: 130000. Avg. loss:  7.75
Step: 140000. Avg. loss:  7.75
Step: 150000. Avg. loss:  7.74
Step: 160000. Avg. loss:  7.73
Step: 170000. Avg. loss:  7.73
Step: 180000. Avg. loss:  7.72
Step: 190000. Avg. loss:  7.71
Step: 200000. Avg. loss:  7.71
Epoch 2, loss: 7.71. Cost 22.7 min
Save model to ckpt
Step: 10000. Avg. loss:  7.49
Step: 20000. Avg. loss:  7.55
Step: 30000. Avg. loss:  7.53
Step: 40000. Avg. loss:  7.51
Step: 50000. Avg. loss:  7.53
Step: 60000. Avg. loss:  7.52
Step: 70000. Avg. loss:  7.51
Step: 80000. Avg. loss:  7.51
Step: 90000. Avg. loss:  7.50
Step: 100000. Avg. loss:  7.49
Step: 110000. Avg. loss:  7.49
Step: 120000. Avg. loss:  7.48
Step: 130000. Avg. loss:  7.48
Step: 140000. Avg. loss:  7.48
Step: 150000. Avg. loss:  7.48
Step: 160000. Avg. loss:  7.48
Step: 170000. Avg. loss:  7.47
Step: 180000. Avg. loss:  7.47
Step: 190000. Avg. loss:  7.47
Step: 200000. Avg. loss:  7.46
Epoch 3, loss: 7.47. Cost 22.3 min
Save model to ckpt
Step: 10000. Avg. loss:  7.32
Step: 20000. Avg. loss:  7.38
Step: 30000. Avg. loss:  7.36
Step: 40000. Avg. loss:  7.35
Step: 50000. Avg. loss:  7.36
Step: 60000. Avg. loss:  7.35
Step: 70000. Avg. loss:  7.34
Step: 80000. Avg. loss:  7.34
Step: 90000. Avg. loss:  7.34
Step: 100000. Avg. loss:  7.34
Step: 110000. Avg. loss:  7.33
Step: 120000. Avg. loss:  7.33
Step: 130000. Avg. loss:  7.33
Step: 140000. Avg. loss:  7.33
Step: 150000. Avg. loss:  7.33
Step: 160000. Avg. loss:  7.33
Step: 170000. Avg. loss:  7.33
Step: 180000. Avg. loss:  7.32
Step: 190000. Avg. loss:  7.32
Step: 200000. Avg. loss:  7.32
Epoch 4, loss: 7.32. Cost 19.7 min
Save model to ckpt
Step: 10000. Avg. loss:  7.20
Step: 20000. Avg. loss:  7.26
Step: 30000. Avg. loss:  7.25
Step: 40000. Avg. loss:  7.23
Step: 50000. Avg. loss:  7.24
Step: 60000. Avg. loss:  7.24
Step: 70000. Avg. loss:  7.23
Step: 80000. Avg. loss:  7.23
```

图 5: 全输出-1

```
Epoch 4, loss: 7.32. Cost 19.7 min
Save model to ckpt
Step: 10000. Avg. loss:  7.20
Step: 20000. Avg. loss:  7.26
Step: 30000. Avg. loss:  7.25
Step: 40000. Avg. loss:  7.23
Step: 50000. Avg. loss:  7.24
Step: 60000. Avg. loss:  7.24
Step: 70000. Avg. loss:  7.23
Step: 80000. Avg. loss:  7.23
Step: 90000. Avg. loss:  7.23
Step: 100000. Avg. loss:  7.22
Step: 110000. Avg. loss:  7.22
Step: 120000. Avg. loss:  7.22
Step: 130000. Avg. loss:  7.22
Step: 140000. Avg. loss:  7.22
Step: 150000. Avg. loss:  7.22
Step: 160000. Avg. loss:  7.22
Step: 170000. Avg. loss:  7.22
Step: 180000. Avg. loss:  7.22
Step: 190000. Avg. loss:  7.22
Step: 200000. Avg. loss:  7.22
Epoch 5, loss: 7.22. Cost 19.3 min
Save model to ckpt
Step: 10000. Avg. loss:  7.11
Step: 20000. Avg. loss:  7.18
Step: 30000. Avg. loss:  7.16
Step: 40000. Avg. loss:  7.14
Step: 50000. Avg. loss:  7.16
Step: 60000. Avg. loss:  7.15
Step: 70000. Avg. loss:  7.14
Step: 80000. Avg. loss:  7.14
Step: 90000. Avg. loss:  7.14
Step: 100000. Avg. loss:  7.14
Step: 110000. Avg. loss:  7.14
Step: 120000. Avg. loss:  7.14
Step: 130000. Avg. loss:  7.14
Step: 140000. Avg. loss:  7.14
Step: 150000. Avg. loss:  7.14
Step: 160000. Avg. loss:  7.14
Step: 170000. Avg. loss:  7.14
Step: 180000. Avg. loss:  7.14
Step: 190000. Avg. loss:  7.14
Step: 200000. Avg. loss:  7.14
Epoch 6, loss: 7.14. Cost 18.5 min
Save model to ckpt
Step: 10000. Avg. loss:  7.04
Step: 20000. Avg. loss:  7.11
Step: 30000. Avg. loss:  7.09
Step: 40000. Avg. loss:  7.07
Step: 50000. Avg. loss:  7.08
Step: 60000. Avg. loss:  7.08
Step: 70000. Avg. loss:  7.07
Step: 80000. Avg. loss:  7.07
Step: 90000. Avg. loss:  7.07
Step: 100000. Avg. loss:  7.07
Step: 110000. Avg. loss:  7.07
Step: 120000. Avg. loss:  7.07
Step: 130000. Avg. loss:  7.07
Step: 140000. Avg. loss:  7.07
Step: 150000. Avg. loss:  7.07
Step: 160000. Avg. loss:  7.07
Step: 170000. Avg. loss:  7.07
Step: 180000. Avg. loss:  7.07
Step: 190000. Avg. loss:  7.07
Step: 200000. Avg. loss:  7.07
Epoch 7, loss: 7.07. Cost 19.2 min
Save model to ckpt
Step: 10000. Avg. loss:  6.98
Step: 20000. Avg. loss:  7.05
Step: 30000. Avg. loss:  7.03
Step: 40000. Avg. loss:  7.01
Step: 50000. Avg. loss:  7.02
Step: 60000. Avg. loss:  7.02
Step: 70000. Avg. loss:  7.01
Step: 80000. Avg. loss:  7.02
Step: 90000. Avg. loss:  7.01
Step: 100000. Avg. loss:  7.01
Step: 110000. Avg. loss:  7.01
Step: 120000. Avg. loss:  7.01
Step: 130000. Avg. loss:  7.01
Step: 140000. Avg. loss:  7.01
Step: 150000. Avg. loss:  7.02
Step: 160000. Avg. loss:  7.02
Step: 170000. Avg. loss:  7.02
Step: 180000. Avg. loss:  7.02
Step: 190000. Avg. loss:  7.01
Step: 200000. Avg. loss:  7.02
Epoch 8, loss: 7.02. Cost 19.4 min
Save model to ckpt
Step: 10000. Avg. loss:  6.93
Step: 20000. Avg. loss:  6.99
Step: 30000. Avg. loss:  6.97
Step: 40000. Avg. loss:  6.96
Step: 50000. Avg. loss:  6.97
Step: 60000. Avg. loss:  6.97
Step: 70000. Avg. loss:  6.96
Step: 80000. Avg. loss:  6.96
Step: 90000. Avg. loss:  6.96
Step: 100000. Avg. loss:  6.96
Step: 110000. Avg. loss:  6.96
Step: 120000. Avg. loss:  6.96
Step: 130000. Avg. loss:  6.96
Epoch 10, loss: 6.92. Cost 19.7 min
Save model to ckpt
Epoch 10, loss: 6.92. Cost 19.7 min
Save model to ckpt
Cost 204.6 min
[('i', 1.0000000000000004), ('we', 0.912416994475628), ('levy', 0.9051181652637502), ('baboon', 0.8826970358235133), ('adr
ift', 0.8742717194144745), ('you', 0.8730728436218514), ('jim', 0.8700385565478238), ('lounge', 0.8690574718409816), ('the
y', 0.8339341158975253), ('discord', 0.833066550406883)]

Test-2 pass :)

Load model from ckpt

Test-3 pass :)
```

图 6: 全输出-2