COMS 4771 HW1

TA Solutions

1 Analyzing Bayes Classifier

Let p_e and P_e refer to the probability density function and the cumulative density function of the exponential distribution with mean parameter $\lambda = 1$, respectively.
Let $S_2 = X_1 + X_2$ where X_1 and X_2 are two i.i.d variables drawn from an exponential distribution with mean parameter $\lambda = 1$. Let $p_{S_2}(z)$ denote the probability density function of S_2 and observe that

$$p_{S_2}(z) = \int_0^z p_e(a)p_e(z-a)da = \int_0^z e^{-a}e^{-(z-a)}da = \int_0^z e^{-z}da = ze^{-z}$$

1. a. $\mathbf{P[Y = 1|A = a, B = b] = 1 - e^{a+b-7}}$

$$\begin{aligned} P[Y = 1|A = a, B = b] &= P[A + B + C < 7|A = a, B = b] \\ &= P[C < 7 - A - B|A = a, B = b] \\ &= P_e(7 - a - b) \\ &= 1 - e^{a+b-7} \end{aligned}$$

Or, more accurately,

$$P[Y = 1|A = a, B = b] = \begin{cases} 1 - e^{a+b-7} & \text{if} \quad a + b < 7 \\ 0 & \text{otherwise} \end{cases}$$

Bayes Classifier:

$$\texttt{Bayes}(a, b) = \begin{cases} 1 & \text{if} \quad a + b + \ln 2 < 7 \\ 0 & \text{otherwise} \end{cases}$$

Bayes Error:

$$\begin{aligned}
P[\hat{Y} \neq y] &= \int_0^7 p_{s_2}(s_2) \sum_{y \in Y} \mathbb{1}[\hat{Y} = y] P[Y \neq y | S_2 = s_2] ds_2 \\
&= \int_0^7 p_{s_2}(s_2)(\mathbb{1}[\hat{Y} = 1] P[Y = 0 | S_2 = s_2] + \mathbb{1}[\hat{Y} = 0] P[Y = 1 | S_2 = s_2]) ds_2 \\
&= \int_0^{7-\ln 2} p_{s_2}(s_2) P[Y = 0 | S_2 = s_2] ds_2 + \int_{7-\ln 2}^7 p_{s_2}(s_2) P[Y = 1 | S_2 = s_2] ds_2 \\
&= \int_0^{7-\ln 2} s_2 e^{-s_2} e^{s_2 - 7} ds_2 + \int_{7-\ln 2}^7 s_2 e^{-s_2} (1 - e^{s_2 - 7}) ds_2 \\
&= \int_0^{7-\ln 2} s_2 e^{-7} ds_2 + \int_{7-\ln 2}^7 (s_2 e^{-s_2} - s_2 e^{-7}) ds_2 \\
&\approx 0.01996
\end{aligned}$$

b.

$$\begin{aligned}
P[Y = 1 | A = a] &= P[A + B + C < 7 | A = a] \\
&= P[S_2 < 7 - A | A = a] \\
&= \int_0^{7-a} z e^{-z} dz \\
&= -e^{-z}(z + 1) \Big|_0^{7-a} \\
&= 1 - e^{a-7}(8 - a)
\end{aligned}$$

Or, more accurately,

$$P[Y = 1 | A = a] = \begin{cases} 1 - e^{a-7}(8 - a) & \text{if} \quad a < 7 \\ 0 & \text{otherwise} \end{cases}$$

$$\texttt{Bayes}(a) = \begin{cases} 1 & \text{if} \quad 1 - e^{a-7}(8 - a) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

The threshold for the Bayes classifier is at $t \approx 5.32165$.

$$\begin{aligned}
P[\hat{Y} \neq y] &= \int_0^7 p_e(a)(\mathbb{1}[\hat{Y} = 1] P[Y = 0 | A = a] + \mathbb{1}[\hat{Y} = 0] P[Y = 1 | A = a]) da \\
&= \int_0^t p_e(a)(\mathbb{1}[\hat{Y} = 1] P[Y = 0 | A = a]) da + \int_t^7 p_e(a) \mathbb{1}[\hat{Y} = 0] P[Y = 1 | A = a]) da \\
&= \int_0^t e^{-a}(e^{a-7}(8 - a)) da + \int_t^7 e^{-a}(1 - e^{a-7}(8 - a)) da \\
&\approx 0.02707
\end{aligned}$$

c. Let $S_3 = A + B + C$. Observe that

$$p_{S_3}(z) = \int_0^z p_{S_2}(a) p_e(z - a) da = \int_0^z a e^{-a} e^{-(z-a)} da = \int_0^z a e^{-z} da = \frac{z^2}{2} e^{-z}$$

.
Note that $P[Y=1] = \int_0^7 p_{s_3}(z)dz = \int_0^7 \frac{z^2}{2}e^{-z}dz \approx 0.97$
As $P[Y=1] > 0.5$ the Bayes classifer will be

$$\texttt{Bayes}(\cdot) = \Big\{1 \quad \text{always}$$

The Bayes Error will therefore be $P[Y=0] = P[A+B+C \geq 7] = 1 - \int_0^7 p_{S_3}(z)dz = 1 - \int_0^7 \frac{1}{2}z^2e^z dz \approx 0.03$

2. Let C follow a uniform distribution over the two element set $\{-\Delta, \Delta\}$. Observe that the median value of the distribution C is 0 and hence the Bayes classifier will be

$$\texttt{Bayes(a,b)} = \mathbb{1}[a+b<7]$$

Therefore,

$$P[e^*] = \int_{-\infty}^{\infty} p_{s_2}(s)(\mathbb{1}[s<7]P[S+C \geq 7|S=s] + \mathbb{1}[s \geq 7]P[S+C<7|S=s])ds$$

where $S = A + B$.
Now observe that for any fixed distributions A and B,

$$\lim_{\Delta \to \infty} P[e^*] = \frac{1}{2}$$

as $\forall s \in S$, $\lim_{\Delta\to\infty} p_{s_2}(s)(\mathbb{1}[S<7]P[S+C \geq 7|S=s] + \mathbb{1}[S \geq 7]P[S+C<7|S=s]) = p_{s_2}(s)((\mathbb{1}[S<7]\frac{1}{2} + \mathbb{1}[S \geq 7]\frac{1}{2}) = \frac{1}{2}p_{s_2}(s)$.

2 Classification with Asymmetric Costs

For a binary classification task with possible labels $\mathcal{Y} = \{0, 1\}$ and a classifier $f : \mathcal{X} \to \{-1, 0, 1\}$, let the classification loss for a particular point x be

$$\ell(f(x), y) = \begin{cases} 0 & \text{if } f(x) = y \\ p & \text{if } f(x) = 0 \text{ and } y = 1 \\ q & \text{if } f(x) = 1 \text{ and } y = 0 \\ r & \text{if } f(x) = -1 \end{cases}$$

(a) This model of classification might be more appropriate, for example, in the case of an initial cancer screening test. The cost of a false negative, p, should be much more than the cost of a false positive, q, as the potential harm of an untreated case is worse than preparing for a case that turns out to be false. In addition, having a test that can specify when it is uncertain would be useful if it is an initial screening that can refer to a possibly more expensive or time-consuming, but more accurate, test for the cases when it is uncertain; a small cost r would prioritize giving correct results in the initial test, but allow for using the more accurate test when necessary.

(b) For a generic classifier $g : \mathcal{X} \to \{-1, 0, 1\}$, we have:

$$\ell(g(x), y) = p \cdot \mathbb{1}[y = 1] \cdot \mathbb{1}[g(x) = 0] + q \cdot \mathbb{1}[y = 0] \cdot \mathbb{1}[g(x) = 1] + r \cdot \mathbb{1}[g(x) = -1]$$

Hence, the expected loss for a particular x is:

$$\begin{aligned}
&\mathbb{E}_y[\ell(g(x), y) \mid X = x] \\
&= \mathbb{E}_y \left[p \cdot \mathbb{1}[y = 1] \cdot \mathbb{1}[g(x) = 0] + q \cdot \mathbb{1}[y = 0] \cdot \mathbb{1}[g(x) = 1] + r \cdot \mathbb{1}[g(x) = -1] \mid X = x\right] \\
&= p \cdot \mathbb{E}_y \left[\mathbb{1}[y = 1] \mid X = x\right] \cdot \mathbb{1}[g(x) = 0] + q \cdot \mathbb{E}_y \left[\mathbb{1}[y = 0] \mid X = x\right] \cdot \mathbb{1}[g(x) = 1] + r \cdot \mathbb{1}[g(x) = -1] \\
&= p \cdot \eta(x) \cdot \mathbb{1}[g(x) = 0] + q \cdot (1 - \eta(x)) \cdot \mathbb{1}[g(x) = 1] + r \cdot \mathbb{1}[g(x) = -1]
\end{aligned}$$

Since there are three loss terms each with a mutually-exclusive indicator, for a particular x exactly one term is nonzero. Therefore, the loss can be rearranged as:

$$\mathbb{E}_y[\ell(g(x), y) \mid X = x] = \begin{cases} p \cdot \eta(x) & \text{if } g(x) = 0 \\ q \cdot (1 - \eta(x)) & \text{if } g(x) = 1 \\ r & \text{if } g(x) = -1 \end{cases}$$

As seen in class, the Bayes classifier must be the one that predicts the label with the minimum associated loss term for each x. In other words letting f^* denote the Bayes classifier, we saw in class[1] that:

$$f^*(x) = \underset{z \in \{-1,0,1\}}{\arg\min}\ \mathbb{E}_y[\ell(z, y) \mid X = x].$$

Hence, all that is left to do is to solve for when (wrt $\eta(x)$) each loss term is smallest. First we solve for when the first term is smallest.

$$\begin{aligned}
f^*(x) = 0 &\iff p \cdot \eta(x) \text{ is the smallest loss term} \\
&\iff \begin{cases} p \cdot \eta(x) \le q \cdot (1 - \eta(x)) \\ p \cdot \eta(x) \le r \end{cases} \\
&\iff \begin{cases} p \cdot \eta(x) + q \cdot \eta(x) \le q \\ p \cdot \eta(x) \le r \end{cases} \\
&\iff \begin{cases} \eta(x) \le \frac{q}{p+q} \\ \eta(x) \le \frac{r}{p} \end{cases} \\
&\iff \eta(x) \le \min\left\{\frac{q}{p+q}, \frac{r}{p}\right\} \\
&\iff \eta(x) \le \frac{r}{p}
\end{aligned}$$

(Since we assume in this part that $r < \frac{pq}{p+q}$ and therefore have $\frac{r}{p} < \frac{q}{p+q}$)

[1]This was mentioned during the Bayes optimality proof in lecture 1 and again at the beginning of lecture 2. It also isn't hard to show again. Take any other classifier $g : \mathcal{X} \to \{-1, 0, 1\}$. By definition of f^* we know that $\forall x \in \mathcal{X} : \mathbb{E}_y[\ell(f^*(x), y) \mid X = x] \le \mathbb{E}_y[\ell(g(x), y) \mid X = x]$. By linearity of expectations we then have $\mathbb{E}_x\left[\mathbb{E}_y[\ell(f^*(x), y) \mid X = x]\right] \le \mathbb{E}_x\left[\mathbb{E}_y[\ell(g(x), y) \mid X = x]\right]$. Finally, the law of total expectation gives us that $\mathbb{E}\left[\ell(f^*(x), y)\right] \le \mathbb{E}\left[\ell(g(x), y)\right]$.

Similarly, we can solve for when the second term is smallest:

$$\begin{aligned} f^*(x) = 1 &\iff q \cdot (1 - \eta(x)) \text{ is the smallest loss term} \\ &\iff \begin{cases} q \cdot (1 - \eta(x)) \leq p \cdot \eta(x) \\ q \cdot (1 - \eta(x)) \leq r \end{cases} \\ &\iff \begin{cases} \eta(x) \geq \frac{q}{p+q} \\ \eta(x) \geq 1 - \frac{r}{q} \end{cases} \\ &\iff \eta(x) \geq \max\left\{ \frac{q}{p+q}, 1 - \frac{r}{q} \right\} \\ &\iff \eta(x) \geq 1 - \frac{r}{q} \end{aligned}$$

(Since we assume in this part that $r < \frac{pq}{p+q}$ and therefore have $1 - \frac{r}{q} > 1 - \frac{p}{p+q} = \frac{q}{p+q}$)

Finally the third term must be smallest (and $f^*(x) = -1$) for the rest of the values that $\eta(x)$ can take, i.e. whenever $1 - \frac{r}{q} < \eta(x) < \frac{r}{p}$.

Hence we have shown that the Bayes classifier, f^*, is such that:

$$f^*(x) = \begin{cases} 0 & \text{if } 0 \leq \eta(x) \leq \frac{r}{p} \\ -1 & \text{if } \frac{r}{p} < \eta(x) < 1 - \frac{r}{q} \\ 1 & \text{if } 1 - \frac{r}{q} \leq \eta(x) \leq 1 \end{cases}$$

which is the desired classifier from the question.

(c) In the case that $r \geq \frac{pq}{p+q}$, we still have that

$$f^*(x) = \operatorname*{arg\,min}_{z \in \{-1,0,1\}} \mathbb{E}_y[\ell(z, y) \mid X = x].$$

But now we have:

$$\begin{aligned} f^*(x) = 0 &\iff p \cdot \eta(x) \text{ is the smallest loss term} \\ &\iff \eta(x) \leq \min\left\{ \frac{q}{p+q}, \frac{r}{p} \right\} \qquad \text{(Same calculation as in (b))} \\ &\iff \eta(x) \leq \frac{q}{p+q} \end{aligned}$$

(Since we assume in this part that $r \geq \frac{pq}{p+q}$ and therefore have $\frac{r}{p} \geq \frac{q}{p+q}$)

Similarly:

$$\begin{aligned} f^*(x) = 1 &\iff q \cdot (1 - \eta(x)) \text{ is the smallest loss term} \\ &\iff \eta(x) \geq \max\left\{ \frac{q}{p+q}, 1 - \frac{r}{q} \right\} \qquad \text{(Same calculation as in (b))} \\ &\iff \eta(x) \geq \frac{q}{p+q} \end{aligned}$$

(Since we assume in this part that $r \geq \frac{pq}{p+q}$ and therefore have $1 - \frac{r}{q} \leq 1 - \frac{p}{p+q} = \frac{q}{p+q}$)

Since this covers all values of $\eta(x)$, f^* will never output -1.

Hence we have shown that the Bayes classifier, f^*, is such that:

$$f^*(x) = \begin{cases} 0 & \text{if } 0 \leq \eta(x) \leq \frac{q}{p+q} \\ 1 & \text{if } \frac{q}{p+q} < \eta(x) \leq 1 \end{cases}$$

which is the desired classifier from the question.

(d) In the case that $p = q$ and $r \geq \frac{p}{2} = \frac{pq}{p+q}$, from (c) the Bayes classifier is given by

$$f^*(x) = \begin{cases} 0 & \text{if } 0 \leq \eta(x) \leq \frac{1}{2} \\ 1 & \text{if } \frac{1}{2} \leq \eta(x) \leq 1 \end{cases}$$

as $\frac{q}{p+q} = \frac{1}{2}$. This is exactly the Bayes classifier from class.

This intuitively makes sense for two reasons. First, if the cost of both types of incorrect prediction are identical, then the optimal classifier should predict the class that is most likely, which is 1 if $\eta(x) = P[Y = 1 \mid X = x] \geq \frac{1}{2}$ and 0 otherwise. Secondly, as there is always at least one class with a 1ess than a half chance of being incorrect, and the cost of giving an uncertain prediction is more than half the cost of an incorrect prediction, there is always a prediction with better expected loss than remaining uncertain.

In other words, when the cost of predicting "unsure" is too high it never makes sense to output "unsure" (-1). In this case, and since $p = q$, the loss in this question becomes equivalent to $p \cdot \ell_{0-1}$ (i.e. p times the 0-1 loss used in class). It is then clear that the minimizers of $p \cdot \ell_{0-1}$ (this Bayes classifier) and of ℓ_{0-1} (the Bayes classifier seen in class) should be the same.

3 Finding (local) minima of generic functions

(i) We first show that f has bounded second derivative. By our assumption that f' has Lipschitz constant L^2, we have that for all $z \in \mathbb{R}$ and $\epsilon > 0$,

$$|f'(z+\epsilon) - f'(z)| \leq L^2|(z+\epsilon) - z| = L^2\epsilon.$$

Dividing by ϵ and allowing ϵ to tend to 0 gives

$$\lim_{\epsilon \to 0^+} \left| \frac{f'(z+\epsilon) - f'(z)}{\epsilon} \right| \leq L^2.$$

Since $|\cdot|$ is continuous, we can move the limit inside the absolute value so that

$$|f''(z)| = \left| \lim_{\epsilon \to 0^+} \frac{f'(z+\epsilon) - f'(z)}{\epsilon} \right| \leq L^2.$$

Next, by Taylor's remainder theorem, there is a $z \in \mathbb{R}$ between x and $\bar{x}$ such that

$$f(\bar{x}) = f(x) + f'(x)(\bar{x} - x) + \frac{1}{2}f''(z)(\bar{x} - x)^2.$$

Since $f''(z) \leq L^2$ and $(\bar{x} - x)^2 \geq 0$, this implies

$$f(\bar{x}) \leq f(x) + f'(x)(\bar{x} - x) + \frac{1}{2}L^2(\bar{x} - x)^2.$$

Now, take $0 < \eta < 2/L^2$; substituting $\bar{x} := x - \eta f'(x)$ and rearranging, we see that

$$\begin{aligned} f(x) - f(\bar{x}) &\geq -f'(x)(-\eta f'(x)) - \frac{1}{2}L^2(-\eta f'(x))^2 \\ &= \eta f'(x)^2 - \frac{L^2}{2}\eta^2 f'(x)^2 \\ &= \left(1 - \frac{L^2}{2}\eta\right)\eta f'(x)^2 \\ &\geq 0, \end{aligned}$$

with equality exactly when $f'(x) = 0$.

(ii) The following iterative algorithm finds a local minimum x of f, starting from an initial value x_0, with step size η and tolerance ϵ.

Require: $f' : \mathbb{R} \to \mathbb{R}$, $x_0 \in \mathbb{R}$, $\eta > 0$, $\epsilon > 0$
Ensure: x is a local minimum of f
  $x \leftarrow x_0$
  repeat
    $\bar{x} \leftarrow x$
    $x \leftarrow x - \eta f'(x)$
  until $|\bar{x} - x| \leq \epsilon$

(iii) The derivative of f is given by

$$f'(x) = 2x + 8e^{2x} - 6.$$

Using the following Julia code, we find that the minimum of f occurs at $\boxed{x = -0.12364}$.

```
function gradDesc(x_0, eta, epsilon)
    x = x_0
    xbar = Inf
    while abs(x - xbar) > epsilon
        xbar = x
        x = x - eta * (2 * x + 8 * exp(2x) - 6)
    end
    return x
end

print(gradDesc(0, 0.01, 1e-10))

-0.12364662519726605
```

(iv) If the function is non-convex, then there is no guarantee that a minimum point found by gradient descent is the global minimum of the function.

We list a number of example answers for possible improvements. Note that many possibilities exist.

Random Restarts allow you to start along different locations in the loss landscape. Picking the lowest among several trials where, each trial starts at a different location gives a higher probability of picking the global minimum than simply running the algorithm once.

Higher order derivatives allow you to see if the point is actually a minimum instead of just a critical point, allowing you to avoid being stuck at saddle points.

Changing the step size may improve the algorithm as if the step size is too large we can encounter numerical stability issues and if the step size is too small we may erroneously stop due to aforementioned numerical stability issues.

4 Exploring the Limits Current Language Models

(i) The trigram Markov assumption is

$$P(w_{1:n}) = \prod_{i=2}^{n} P(w_i|w_{i-1}w_{i-2}).$$

Let $C(w_{i-2}w_{i-1}w_i)$ denote the number of times the trigram $(w_{i-2}w_{i-1}w_i)$ appears in the training corpus. Then, we can approximate the conditional probability as

$$P(w_i|w_{i-1}w_{i-2}) \underset{\text{approx.}}{=} \frac{C(w_{i-2}w_{i-1}w_i)}{C(w_{i-2}w_{i-1})}.$$

With Laplacian smoothing, the conditional probability becomes

$$P(w_i|w_{i-1}w_{i-2}) \underset{\text{approx.}}{=} \frac{C(w_{i-2}w_{i-1}w_i)+1}{C(w_{i-2}w_{i-1})+|V|}.$$

Thus, the class probability is

$$P(y|w_{1:n}) = \frac{P(y)P(w_{1:n}|y)}{\sum_{y'} P(y')P(w_{1:n}|y')} = \frac{P(y)\prod_{i=2}^{n} P(w_i|w_{i-1}w_{i-2},y)}{\sum_{y'} P(y')\prod_{i=2}^{n} P(w_i|w_{i-1}w_{i-2},y')},$$

where the conditional probability is given by the expression calculated above.

(ii) (a) See code on the following page for solutions. Note that since we will code a trigram model, we add *two* <START> tokens and *one* <END> token to every sentence.

(b) The bigram model has a 9.7% OOV rate and the trigram model has a 35.1% OOV rate.

(c) The bigram and trigram models have 96% and 95% classification accuracy respectively (this is pretty good for a naive model!). While trigram models *generally* perform better than bigram models, this is not always the case. In this problem, the bigram model performs better because it has a significantly lower OOV rate. Consequently, there are fewer approximations from Laplacian smoothing when calculating the bigram conditional probability.

(iii) (a) Determining which model and text corpus generates the best sentences is somewhat subjective, but *generally*, the trigram model trained on the ChatGPT text corpus does the

best. Decreasing the temperature causes the sentences to be more random and nonsensical. Increasing the temperature causes the sentences to be more deterministic and repetitive. There is a "sweet spot" temperature for the model to generate good sentences which are not repetitive but also somewhat grammatically correct.

(b) N-gram models can not capture long-range context because of the Markov independence assumption (i.e. the next token only depends on the $N-1$ previous tokens). Thus, N-grams do not use the entire "history" of a sentence, which causes it to lose context as it produces more tokens.

In [1]:
```
import re, math
import numpy as np
```

Load the data

In [2]:
```
hum_file = open('humvgpt/hum.txt', 'r', encoding="utf8")
hum = hum_file.readlines()

gpt_file = open('humvgpt/gpt.txt', 'r', encoding="utf8")
gpt = gpt_file.readlines()

def clean_string(s):
    s = re.sub(r"[^\w\d'\s,.?!-]+", '', s).lower()
    s = re.sub(r" - ", ' ', s)
    s = s.strip('\n').strip()
    s = re.sub(' +', ' ', s)
    return s

for k,s in enumerate(hum):
    hum[k] = f'<START> <START> {clean_string(s)} <END>'
for k,s in enumerate(gpt):
    gpt[k] = f'<START> <START> {clean_string(s)} <END>'
```

In [3]:
```
vocab = []
for sentence in hum:
    vocab += sentence.split(' ')
for sentence in gpt:
    vocab += sentence.split(' ')
vocab = set(vocab)
vocab_size = len(vocab)
print(f'Vocab size: {vocab_size}')
```

```
Vocab size: 75646
```

In [4]:
```
hum_train, hum_test = hum[:int(len(hum)*0.9)], hum[int(len(hum)*0.9):]
gpt_train, gpt_test = gpt[:int(len(gpt)*0.9)], gpt[int(len(gpt)*0.9):]
print(f'{len(hum)} human responses')
print(f'{len(gpt)} chatgpt responses')
```

```
39713 human responses
20398 chatgpt responses
```

Train the bi/trigram model

In [5]:
```
# This functions trains the model using the human/gpt text corpus
# and calculates the class prior (document frequency)
def train(hum, gpt):
    doc_freq = {'hum': len(hum) / (len(hum) + len(gpt)), 'gpt': len(gpt) / (len(hum)
    hum_trigrams = get_trigrams(hum)
    gpt_trigrams = get_trigrams(gpt)
    return (doc_freq, hum_trigrams, gpt_trigrams)

# This function gets the uni/bi/trigrams for a training text corpus
# trigrams[w1][w2][w3]      retrieves the number of times (w1,w2,w3) appears
# trigrams[w1][w2]['TOTAL'] retrieves the number of times (w1,w2) appears
# trigrams[w1]['TOTAL']     retrieves the number of times (w1) appears
def get_trigrams(data):
    trigrams = {'TOTAL': 0, '<START>': {'TOTAL': 0, '<START>': {'TOTAL': 0}}}
    for resp in data:
        words = resp.split(' ')
        assert words[0] == '<START>' and words[1] == '<START>'
        trigrams['<START>']['<START>']['TOTAL'] += 1
```

```
        trigrams['<START>']['TOTAL'] += 1
        for k in range(2, len(words)):
            w1, w2, w3 = words[k-2], words[k-1], words[k]
            if w1 not in trigrams.keys():
                trigrams[w1] = {}
            if w2 not in trigrams[w1].keys():
                trigrams[w1][w2] = {}
            if w3 not in trigrams[w1][w2].keys():
                trigrams[w1][w2][w3] = 0
            trigrams[w1][w2][w3] += 1

            if w2 not in trigrams.keys():
                trigrams[w2] = {}
            if w3 not in trigrams[w2].keys():
                trigrams[w2][w3] = {}
            if 'TOTAL' not in trigrams[w2][w3].keys():
                trigrams[w2][w3]['TOTAL'] = 0
            trigrams[w2][w3]['TOTAL'] += 1

            if w3 not in trigrams.keys():
                trigrams[w3] = {}
            if 'TOTAL' not in trigrams[w3].keys():
                trigrams[w3]['TOTAL'] = 0
            trigrams[w3]['TOTAL'] += 1
            trigrams['TOTAL'] += 1

    return trigrams
```

In [6]:

```
doc_freq, hum_trigrams, gpt_trigrams = train(hum_train, gpt_train)
print('Finished training!')
```

```
Finished training!
```

OOV rate

In [7]:

```
unk_bigrams, tot_bigrams = 0, 0
unk_trigrams, tot_trigrams = 0, 0

for data in [hum_test, gpt_test]:
    for resp in data:
        words = resp.split(' ')
        for k in range(2, len(words)):
            w1, w2 = words[k-1], words[k]
            tot_bigrams += 1
            try:
                bigrams_count = hum_trigrams[w1][w2]
            except:
                try:
                    bigrams_count = gpt_trigrams[w1][w2]
                except:
                    unk_bigrams += 1
            w1, w2, w3 = words[k-2], words[k-1], words[k]
            tot_trigrams += 1
            try:
                trigrams_count = hum_trigrams[w1][w2][w3]
            except:
                try:
                    trigrams_count = gpt_trigrams[w1][w2][w3]
                except:
                    unk_trigrams += 1
```

In [8]:

```
print('Bigrams OOV rate: {:.2f}%'.format(unk_bigrams / tot_bigrams * 100))
print('Trigrams OOV rate: {:.2f}%'.format(unk_trigrams / tot_trigrams * 100))
```

```
Bigrams OOV rate: 9.68%
```

```
Trigrams OOV rate: 35.10%
```

Evaluation

In [9]:

```python
# This function returns the log-probability of the given class using a bigram model
def prob_bigrams(words, doc_prob, trigrams, vocab_size):
    log_prob = math.log(doc_prob)
    for k in range(2, len(words)):
        w1, w2 = words[k-1], words[k]
        try:
            bigram_count = trigrams[w1][w2]['TOTAL']
        except:
            bigram_count = 0
        try:
            unigram_count = trigrams[w1]['TOTAL']
        except:
            unigram_count = 0
        log_prob += math.log((bigram_count + 1) / (unigram_count + vocab_size))
    return log_prob

# This function returns the log-probability of the given class using a trigram model
def prob_trigrams(words, doc_prob, trigrams, vocab_size):
    log_prob = math.log(doc_prob)
    for k in range(2, len(words)):
        w1, w2, w3 = words[k-2], words[k-1], words[k]
        try:
            trigram_count = trigrams[w1][w2][w3]
        except:
            trigram_count = 0
        try:
            bigram_count = trigrams[w1][w2]['TOTAL']
        except:
            bigram_count = 0
        log_prob += math.log((trigram_count + 1) / (bigram_count + vocab_size))
    return log_prob
```

In [10]:

```python
# This function calculates the class log-probability for y = human and y = ChatGPT
# and then selects the class with the higher probability
def inference(resp, doc_freq, hum_trigrams, gpt_trigrams, vocab_size, mode='bigrams'):
    assert mode in ['bigrams', 'trigrams']
    if mode == 'bigrams':
        prob_func = prob_bigrams
    elif mode == 'trigrams':
        prob_func = prob_trigrams
    words = resp.split(' ')
    hum_prob = prob_func(words, doc_freq['hum'], hum_trigrams, vocab_size)
    gpt_prob = prob_func(words, doc_freq['gpt'], gpt_trigrams, vocab_size)
    if hum_prob > gpt_prob:
        return 0
    return 1

# This function evalutes the given bi/trigram model
def test_accuracy(hum, gpt, doc_freq, hum_trigrams, gpt_trigrams, vocab_size, mode='k
    assert mode in ['bigrams', 'trigrams']
    total = 0
    correct = 0
    for c, data in enumerate([hum, gpt]):
        for resp in data:
            class_ = inference(resp, doc_freq, hum_trigrams, gpt_trigrams, vocab_size
            if class_ == c:
                correct += 1
            total += 1
    return correct / total
```

In [11]:

```python
acc = test_accuracy(hum_test, gpt_test, doc_freq, hum_trigrams, gpt_trigrams, vocab_s
print(f'Bigrams accuracy: {round(acc, 4)*100}%')
```

```
acc = test_accuracy(hum_test, gpt_test, doc_freq, hum_trigrams, gpt_trigrams, vocab_s
print(f'Trigrams accuracy: {round(acc, 4)*100}%')
```

```
Bigrams accuracy: 96.07%
Trigrams accuracy: 94.96%
```

Text generation

In [12]:

```
# This function is used to calculate the probability of the next word given
# the bi/trigram counts scaled by temperature (i.e. the formula before 3a)
def softmax(z):
    return np.exp(z) / np.sum(np.exp(z), axis=0)

toks, T, temps = 20, 50, [10, 25, 50, 100, 200]
```

In [13]:

```
def bigram_sentences(text_trigrams, num_sentences, toks, T):
    for idx in range(num_sentences):
        sentence = ""
        w1 = "<START>"
        for _ in range(toks):
            trigrams = text_trigrams[w1]
            tmp = {k:v for k,v in trigrams.items() if k != 'TOTAL' and k != '<START>'
            next_words = sorted(tmp.items(), key=lambda x:x[1]['TOTAL'], reverse=True
            counts = [w[1]['TOTAL'] / T for w in next_words]
            probs = softmax(counts)

            next_words = [w[0] for w in next_words]
            next_word = np.random.choice(next_words, p=probs)
            if next_word == '<END>':
                break
            sentence += f'{next_word} '
            w1 = next_word
        print(f'{idx+1}: {sentence}')

def trigram_sentences(text_trigrams, num_sentences, toks, T):
    for idx in range(num_sentences):
        sentence = ""
        w1, w2 = "<START>", "<START>"
        for _ in range(toks):
            trigrams = text_trigrams[w1][w2]
            next_words = sorted(trigrams.items(), key=lambda x:x[1], reverse=True)[1:
            counts = [w[1] / T for w in next_words]
            probs = softmax(counts)

            next_words = [w[0] for w in next_words]
            next_word = np.random.choice(next_words, p=probs)
            if next_word == '<END>':
                break
            sentence += f'{next_word} '
            w1, w2 = w2, next_word
        print(f'{idx+1}: {sentence}')
```

In [18]:

```
print('Human bigrams')
print('='*125)
bigram_sentences(hum_trigrams, 5, toks, T)
print('\nChatGPT bigrams')
print('='*125)
bigram_sentences(gpt_trigrams, 5, toks, T)
print('\nHuman trigrams')
print('='*125)
trigram_sentences(hum_trigrams, 5, toks, T)
print('\nChatGPT trigrams')
print('='*125)
trigram_sentences(gpt_trigrams, 5, toks, T)
```

```
Human bigrams
================================================================================
========================================
1: the same way to the same thing .
2: the same .
3: the same time .
4: the same way to the same way to the same time .
5: the same factor it 's a lot of the same thing .

ChatGPT bigrams
================================================================================
========================================
1: the same way to the same time .
2: the united states , and the same time .
3: the same way to the same bank account , and the same way to the same way to the sa
me
4: the same way to the same time .
5: the same way to the same way to the same nondiscrimination rules related to the sa
me font they are a

Human trigrams
================================================================================
========================================
1: the liver for storage 2 to the cabinet could be chronic sinus infection , after se
veral highly active anon leaders
2: the spoon ? nope . they are n't idiots . but also causes ringing . over thousands
of trees .
3: the f22 has new technology the us wanted to placed a limit where an individual in
existence , purpouse ,
4: the ticklish feeling evolved as the rivers lakes are constantly coming into their
lives so it forgets that the hyper
5: the completion of course , sexual intercourse . we can compare this to become froz
en at exactly the moment you

ChatGPT trigrams
================================================================================
========================================
1: the observer . thunder is a type of unconscious movement that began operating in a
way to imagine other kinds
2: the stepped reckoner in the united states , great job! which aired from september
22 , your body that can
3: the district and the countries involved have agreed upon in the united states , ch
ina was taken up a liability
4: the temporal dependencies between variables precisely , or household budget . your
surgeon can help to build cash value are
5: the guitarists on kiss's creatures of habit , without sending them false signals t
hat might have updates that are built
```

In [21]:

```
for temp in temps:
    print(f'TEMPERATURE = {temp}')
    print('='*125)
    trigram_sentences(gpt_trigrams, 5, toks, temp)
    print()
```

```
TEMPERATURE = 10
================================================================================
========================================
1: the united states , the value of the united states , the government . just like ho
w a computer .
2: the united states , the water . one reason is that it is important to note that th
e government .
3: the united states , the water . when you are experiencing following your prescribe
d medications as needed . this is
4: the united states , the government . for example , if you have any other questions
.
5: the doomsday clock should be able to provide a steadier return over a state of the
united states , and

TEMPERATURE = 25
```

==========
1: the unexpected and does not have the same way that deaf people do bad things . sci
ssors are good at
2: the fibonacci sequence because it is important to note that the us supports israel
. many healthy foods can't taste
3: the sp 500 etf voo this etf tracks the 500 loss on a typical breakfast food and ca
use significant damage
4: the oklahoma city bombing , there are a few reasons why people like themselves or
commit crimes , and the
5: the iran-contra affair was a vietnamese communist revolutionary leader who rules o
ver a fixed shape and a physical branch to

TEMPERATURE = 50
==========
1: the dalai lama . the county safe and protect your personal loan with a tax profess
ional or commercial sound ,
2: the battle , disrupting your daily expenses , for most females , and the old way o
f relating different optimization
3: the taliban to try to restart . defibrillators can be a good idea to create a grap
h to show proof
4: the bully vs main character must try to generate , the value of the increase , mak
ing charging interest on
5: the mayan system , they might hire norfolk southern to transport their goods cheap
er for other financial transactions because it

TEMPERATURE = 100
==========
1: the oxygen from earth in order to secure enough , backup generators that can stand
on their canadian-source income .
2: the atanasoffberry computer abc was intended . launch your bank , to a counter tha
t records the record in air-to-air
3: the nelson mandela university is a type of placeholder text , the operators of the
sound signals that coordinate the
4: the disk defragmenter tool is not touching it . ddr4 is faster and do activities l
ike walking , and the
5: the atari video computer system to install cell phone near an airport because it's
used because they add different features

TEMPERATURE = 200
==========
1: the tree didn't have access to legal restrictions . it is important to note that s
tanding orders are filled at
2: the dark . sometimes babies cry for help as soon as you grew in your pelvis , and
the nationalization
3: the osi model , my training , logistics , and the grenadines seychelles sierra leo
ne singapore solomon islands south africa
4: bronies are grown men who are illegally selling or renting would be nothing left o
ver , which can make for
5: the persian wars , economic indicators that are chambered for . these polymers can
be a red herring . the