

zad11-lstm

November 27, 2025

1 Zadanie: Sieć LSTM - nauka tekstu

Celem zadania jest nauczenie sieci LSTM (zaimplementowanej w czystym NumPy) konkretnego fragmentu tekstu dotyczącego zastosowań sztucznej inteligencji, aż do momentu, w którym funkcja straty (loss) spadnie do poziomu 0.1.

1.0.1 1. Import bibliotek

```
[13]: import numpy as np
import matplotlib.pyplot as plt
from IPython import display

%matplotlib inline
```

1.0.2 2. Przygotowanie danych

Definicja tekstu docelowego.

```
[14]: # Tekst zadany w poleceniu
data = "AI applications include advanced web search engines (e.g., Google_
↪Search), recommendation systems (used by YouTube, Amazon, and Netflix),_
↪understanding human speech (such as Siri and Alexa), self-driving cars (e.g.
↪, Waymo), generative or creative tools (ChatGPT and AI art), automated_
↪decision-making, and competing at the highest level in strategic game_
↪systems (such as chess and Go)"

chars = list(set(data))
data_size, X_size = len(data), len(chars)
print(f"Dane mają {data_size} znaków, w tym {X_size} unikalnych.")

char_to_idx = {ch:i for i,ch in enumerate(chars)}
idx_to_char = {i:ch for i,ch in enumerate(chars)}
```

Dane mają 385 znaków, w tym 40 unikalnych.

1.0.3 3. Hiperparametry i inicjalizacja

Definicja rozmiaru warstwy ukrytej (H_size), długość sekwencji uczącej (T_steps) oraz parametry uczenia.

```

[15]: H_size = 128 # Zwiększony rozmiar warstwy ukrytej dla lepszego zapamiętywania
      T_steps = 50 # Długość sekwencji treningowej
      learning_rate = 1e-1
      weight_sd = 0.1
      z_size = H_size + X_size

      def sigmoid(x):
          return 1 / (1 + np.exp(-x))

      def dsigmoid(y):
          return y * (1 - y)

      def tanh(x):
          return np.tanh(x)

      def dtanh(y):
          return 1 - y * y

      class Param:
          def __init__(self, name, value):
              self.name = name
              self.v = value
              self.d = np.zeros_like(value)
              self.m = np.zeros_like(value)

      class Parameters:
          def __init__(self):
              self.W_f = Param('W_f', np.random.randn(H_size, z_size) * weight_sd + 0.
↪5)
              self.b_f = Param('b_f', np.zeros((H_size, 1)))
              self.W_i = Param('W_i', np.random.randn(H_size, z_size) * weight_sd + 0.
↪5)
              self.b_i = Param('b_i', np.zeros((H_size, 1)))
              self.W_C = Param('W_C', np.random.randn(H_size, z_size) * weight_sd)
              self.b_C = Param('b_C', np.zeros((H_size, 1)))
              self.W_o = Param('W_o', np.random.randn(H_size, z_size) * weight_sd + 0.
↪5)
              self.b_o = Param('b_o', np.zeros((H_size, 1)))
              self.W_v = Param('W_v', np.random.randn(X_size, H_size) * weight_sd)
              self.b_v = Param('b_v', np.zeros((X_size, 1)))

          def all(self):
              return [self.W_f, self.W_i, self.W_C, self.W_o, self.W_v,
                      self.b_f, self.b_i, self.b_C, self.b_o, self.b_v]

      parameters = Parameters()

```

1.0.4 4. Implementacja LSTM (Forward & Backward Pass)

Implementacja przejścia w przód (obliczenie stanów i predykcji) oraz wstecznej propagacji błędu (obliczenie gradientów).

```
[16]: def forward(x, h_prev, C_prev, p = parameters):
    z = np.row_stack((h_prev, x))
    f = sigmoid(np.dot(p.W_f.v, z) + p.b_f.v)
    i = sigmoid(np.dot(p.W_i.v, z) + p.b_i.v)
    C_bar = tanh(np.dot(p.W_C.v, z) + p.b_C.v)
    C = f * C_prev + i * C_bar
    o = sigmoid(np.dot(p.W_o.v, z) + p.b_o.v)
    h = o * tanh(C)
    v = np.dot(p.W_v.v, h) + p.b_v.v
    y = np.exp(v) / np.sum(np.exp(v))
    return z, f, i, C_bar, C, o, h, v, y

def backward(target, dh_next, dC_next, C_prev, z, f, i, C_bar, C, o, h, v, y, p = parameters):
    dv = np.copy(y)
    dv[target] -= 1
    p.W_v.d += np.dot(dv, h.T)
    p.b_v.d += dv
    dh = np.dot(p.W_v.v.T, dv)
    dh += dh_next
    do = dh * tanh(C)
    do = dsigmoid(o) * do
    p.W_o.d += np.dot(do, z.T)
    p.b_o.d += do
    dC = np.copy(dC_next)
    dC += dh * o * dtanh(tanh(C))
    dC_bar = dC * i
    dC_bar = dtanh(C_bar) * dC_bar
    p.W_C.d += np.dot(dC_bar, z.T)
    p.b_C.d += dC_bar
    di = dC * C_bar
    di = dsigmoid(i) * di
    p.W_i.d += np.dot(di, z.T)
    p.b_i.d += di
    df = dC * C_prev
    df = dsigmoid(f) * df
    p.W_f.d += np.dot(df, z.T)
    p.b_f.d += df
    dz = (np.dot(p.W_f.v.T, df) + np.dot(p.W_i.v.T, di) + np.dot(p.W_C.v.T, dC_bar) +
    np.dot(p.W_o.v.T, do))
    dh_prev = dz[:H_size, :]
    dC_prev = f * dC
    return dh_prev, dC_prev
```

```

def clear_gradients(params = parameters):
    for p in params.all():
        p.d.fill(0)

def clip_gradients(params = parameters):
    for p in params.all():
        np.clip(p.d, -1, 1, out=p.d)

def forward_backward(inputs, targets, h_prev, C_prev):
    x_s, z_s, f_s, i_s, C_bar_s, C_s, o_s, h_s, v_s, y_s = {}, {}, {}, {}, {}, {}
    h_s[-1] = np.copy(h_prev)
    C_s[-1] = np.copy(C_prev)
    loss = 0
    for t in range(len(inputs)):
        x_s[t] = np.zeros((X_size, 1))
        x_s[t][inputs[t]] = 1
        (z_s[t], f_s[t], i_s[t], C_bar_s[t], C_s[t], o_s[t], h_s[t], v_s[t],
        y_s[t]) = \
            forward(x_s[t], h_s[t - 1], C_s[t - 1])
        loss += -np.log(y_s[t][targets[t], 0])
    clear_gradients()
    dh_next = np.zeros_like(h_s[0])
    dC_next = np.zeros_like(C_s[0])
    for t in reversed(range(len(inputs))):
        dh_next, dC_next = backward(targets[t], dh_next, dC_next, C_s[t-1],
        z_s[t], f_s[t], i_s[t], C_bar_s[t], C_s[t], o_s[t], h_s[t], v_s[t], y_s[t])
    clip_gradients()
    return loss, h_s[len(inputs) - 1], C_s[len(inputs) - 1]

def update_paramters(params = parameters):
    for p in params.all():
        p.m += p.d * p.d
        p.v += -(learning_rate * p.d / np.sqrt(p.m + 1e-8))

```

1.0.5 5. Funkcje pomocnicze

Funkcja `sample` do generowania tekstu oraz `update_status` do wizualizacji postępów.

```

[17]: def sample(h_prev, C_prev, first_char_idx, sentence_length):
        x = np.zeros((X_size, 1))
        x[first_char_idx] = 1
        h = h_prev
        C = C_prev
        indexes = []
        for t in range(sentence_length):

```

```

_, _, _, _, C, _, h, _, p = forward(x, h, C)
idx = np.random.choice(range(X_size), p=p.ravel())
x = np.zeros((X_size, 1))
x[idx] = 1
indexes.append(idx)
return indexes

def update_status(inputs, h_prev, C_prev, iteration, smooth_loss):
    sample_idx = sample(h_prev, C_prev, inputs[0], len(data))
    txt = ''.join(idx_to_char[idx] for idx in sample_idx)
    print(f"Iter: {iteration}, Loss: {smooth_loss:.4f}")
    print(f"Pred: {txt[:100]}...")

```

1.0.6 6. Pętla treningowa

Nauka sieci w pętli do momentu, aż `smooth_loss` spadnie poniżej 0.1.

```

[18]: smooth_loss = -np.log(1.0 / X_size) * T_steps
iteration, pointer = 0, 0
plot_iter = []
plot_loss = []
g_h_prev = np.zeros((H_size, 1))
g_C_prev = np.zeros((H_size, 1))

print("Rozpoczywanie treningu...")

while smooth_loss > 0.1:
    if pointer + T_steps >= len(data) or iteration == 0:
        g_h_prev = np.zeros((H_size, 1))
        g_C_prev = np.zeros((H_size, 1))
        pointer = 0

    inputs = ([char_to_idx[ch] for ch in data[pointer: pointer + T_steps]])
    targets = ([char_to_idx[ch] for ch in data[pointer + 1: pointer + T_steps + 1]])

    loss, g_h_prev, g_C_prev = forward_backward(inputs, targets, g_h_prev, g_C_prev)
    smooth_loss = smooth_loss * 0.999 + loss * 0.001

    if iteration % 1000 == 0:
        update_status(inputs, g_h_prev, g_C_prev, iteration, smooth_loss)

    update_paramters()

    if iteration % 100 == 0:
        plot_iter.append(iteration)

```

```

        plot_loss.append(smooth_loss)

    pointer += T_steps
    iteration += 1

print(f"\nTrening zakończony! Osiągnięto loss: {smooth_loss:.4f} w iteracji_
↪{iteration}")

```

Rozpoczynanie treningu...

Iter: 0, Loss: 184.4529

Pred: bcf.xp.(,pfGA,cf c,mtnf.TspTxbdtwtiwbzAugsbbsbbdwxm-hdkwCbpes-
doxohySs.Nwwytbiv gLYSZpy)(hNTtu Gtkr...

C:\Users\Maciek\AppData\Local\Temp\ipykernel_20008\3326028980.py:2:

DeprecationWarning: `row_stack` alias is deprecated. Use `np.vstack` directly.

```
z = np.row_stack((h_prev, x))
```

Iter: 1000, Loss: 138.4422

Pred: and we-e azo andivexars AIheetiv (rston d wexa), e)dasatf-dts aatic
artoatativicogegikive tlu avin...

Iter: 2000, Loss: 83.7243

Pred: e hymYoon, an srtrrs (sudenglflif), cemhymrears (eeg g., webe gpecind
anns (C.g., Amuug.luce, encom...

Iter: 3000, Loss: 46.0928

Pred: ged web searwebmsicog cem Ss (ChatGPT dled by Yoat a-n rs (e.g.,
inderstumh SiratGor wel Waymr), pe...

Iter: 4000, Loss: 23.9747

Pred: aed cls comaticletinging hu and competing atead cogheative inarttve
Netflix), autive oot glears (...)

Iter: 5000, Loss: 12.5320

Pred: (uceategheat ls (stemcars (e.g., Waymors (e., recommsec , Google Se hzon,
and Netflix), understand ...

Iter: 6000, Loss: 7.1980

Pred: ed by Yoat at highest levi-mars (e., (e.g., Google Search), recommendarion
ssst(eenstand AI aktines ...

Iter: 7000, Loss: 4.2510

Pred: arch), recommating he search es webgines (e.g., Google Search),
recommendation systems (used by YouT...

Iter: 8000, Loss: 4.2649

Pred: stemg.leWaymo), generative or creative tools (ChatGPT and coms tools
(ChatGPT and AI art), automated...

Iter: 9000, Loss: 2.5136

Pred: e ors (Chastgle tools (C.arch end Alexa), self-driving cars (e.g., Waymo),
generative or creative to...

Iter: 10000, Loss: 2.1989

Pred: crutens (uced webu endind Netflix), understancomas (usegls Search),
recommendation systems (used cre...

Iter: 11000, Loss: 1.4711

Pred: menes (eln-mandendin strmatindsth), Waymo), generative or SiratGoogled

cars (e.g., Google (e.g., Google Search), recommendation systems (used by YouTube, Amazon, and Netflix), understanding human speech (such as Siri and Alexa), self-driving cars (e.g., Waymo), generative or creative tools (ChatGPT and AI art), automated decision-making, and competing at the highest level in strategic g...)

Iter: 12000, Loss: 1.1126

Pred: (ChatGPT and AI art), as dncr (st humand competing at the highest level in strategic gsearchTube, a...

Iter: 13000, Loss: 0.9118

Pred: ed by YouTube, Amazon, and Netflix), understanding human speech (such as Siri and aneoat ls (ChatGoo...

Iter: 14000, Loss: 0.7830

Pred: ar trg hugheareategic g., Google Search), rech ms (uzogTearch endratind (u hube Semuby tool Sioomati...

Iter: 15000, Loss: 0.6926

Pred: see stande.g., Google Search), recommendation systems (used by YouTube, Amazon, and Netflix), unders...

Iter: 16000, Loss: 2.3484

Pred: e Siarsth), self-driving cars (e.g., Waymo), generative or creative tools (ChatGPT and AI art), auto...

Iter: 17000, Loss: 1.2286

Pred: e tsederatine trategin spench (systematratives Yoube human speech (such as Siri and Alexa), sebm, (ut...

Iter: 18000, Loss: 0.7751

Pred: generative or creatind hutfl xat ab systems (used by YouTube, Amazon, and Netflix), undending human...

Iter: 19000, Loss: 0.5810

Pred: (ChatGPT and AI art), automated decision-making, and competing at the highest level in strategic g...

Iter: 20000, Loss: 0.4878

Pred: ed by YouTube, AmazonverrthI and AI art), automated decision-making, and competing at the highest le...

Iter: 21000, Loss: 0.4351

Pred: ad at the highest level in strategic g., Waymo), generative or creative tools (ChatGPT anding comatG...

Iter: 22000, Loss: 0.3998

Pred: sinderated web search engines (e.g., Google Search), recommendati), andears (e.g., Waymo), generativ...

Iter: 23000, Loss: 1.2098

Pred: ed by Search), recing human speech by YouTumeb search engied Netiri and Alexa), selixiovenspeeche Se...

Iter: 24000, Loss: 0.8237

Pred: crategic g.lself-driving cars (e.g., Waymo), generative or creative tools (ChatGPT and AI art), autom...

Iter: 25000, Loss: 0.5133

Pred: inene aighest level in stendation secompeting human speech (such as Siri and Alexa), self-driving ca...

Iter: 26000, Loss: 0.3855

Pred: (uTuTe, Amazon, and Netflix), understanding human speech (such as Siri and Alexa), self-driving car...

Iter: 27000, Loss: 0.3279

Pred: ed by YouTube, Amazon, and Neie.g., Gomg c., Waymo), generative or

creative tools (ChatGPT and AI ar...

Iter: 28000, Loss: 0.2976

Pred: at the highest level in strategic gich (sich as Siri and Alexa), self-driving cumationd wed AI art),...

Iter: 29000, Loss: 0.2784

Pred: sindstan speech (sed by YouTube, Amazon, and Nerf-driving cars (e.g., Waymo), generative or creative...

Iter: 30000, Loss: 0.2640

Pred: ed by YouTube, Amazon, and Netflix), unced web search engines (e.g., Googledomated eeveearch), recom...

Iter: 31000, Loss: 1.3034

Pred: cratighested Audecommeneed chighe YouTube, Amazon, and Netflix), understanding humand Alexa), self-d...

Iter: 32000, Loss: 0.6451

Pred: inenerative or creative tools (ChatGPT and AI art), automated decisivn Neces Serative or creative to...

Iter: 33000, Loss: 0.3933

Pred: (Cdrthlenang cars (e.g., Waymo), generative or creative tools (ChatGPT and AI art), automated deci...

Iter: 34000, Loss: 0.2915

Pred: ed by YouTube, Amazon, and Netflix), understanding human speech (such as Siri and Alexa), self-drivi...

Iter: 35000, Loss: 0.2483

Pred: at the highest level in strategic g.lexat ors (ChatGPT and AI art), automated decision-making, and c...

Iter: 36000, Loss: 0.2271

Pred: GCherat leve arch), reb , andes in stredengines (e.g., Google Search), recommendation systems (used...

Iter: 37000, Loss: 0.2133

Pred: evel in speech (s (e.g., Google Search), recommendation systems (used by YouTube, Amazon, and Netfli...

Iter: 38000, Loss: 0.2045

Pred: crategic ged ch (sed commendation systems (used by YouTube, Amazon, and Netflix), understanding huma...

Iter: 39000, Loss: 0.1961

Pred: inech mseneceuted creding human speech (such as Siri and Alexa), self-driving cars (e.g., Waymo), ge...

Iter: 40000, Loss: 0.1891

Pred: (C.c., Google Search), recommendation systems (used by YouTube, Amazon, and Netflix), understanding...

Iter: 41000, Loss: 0.1840

Pred: ed by YouTube, Amazon, and Netflix), understanding human speech (such as Siri and Alexa), self-drivi...

Iter: 42000, Loss: 0.1775

Pred: ad the highest level in strategic g., Waymat lNethi heated sech e.g., Waymo), generative or creative...

Iter: 43000, Loss: 0.1719

Pred: suman thi cools (diri and AI art), autommation sytinls (ChatGPT and AI

art), automated decision-maki...

Iter: 44000, Loss: 0.1668

Pred: e ang competing at the highest level in strategic gicr ss (usedangind competing at the highest level...

Iter: 45000, Loss: 1.1064

Pred: clude advanced webes (uud ch (symo), generative or creative tools (ChatGPT and AI and by YouTubebydi...

Iter: 46000, Loss: 0.5209

Pred: ine, Ammonls (Cterstanding human speech (such as Siri and Alexa), self-driving cars (e.g., Waymo), g...

Iter: 47000, Loss: 0.2959

Pred: (ChatGPT and AI art), automated decision-making, and competing aud ch gicompeting human speech (suc...

Iter: 48000, Loss: 0.2084

Pred: ed by YouTube, Amazon, and Netflix), understanding human speech (such as Siri and Alexa), self-drivi...

Iter: 49000, Loss: 0.1727

Pred: at tpes (u.g., Google Search), recommendation systems (used by YouTube, Amazon, and Netflix), unders...

Iter: 50000, Loss: 0.1565

Pred: suman speech (such es (usel YouTube, Amazon, and Netflix), understanding human speech (such as omanc...

Iter: 51000, Loss: 0.1479

Pred: e ang human speech (such as Siri and Alexa), self-driving cars (e.g., Waymo), generative or creative...

Iter: 52000, Loss: 0.1423

Pred: ca), , Google Search), recommendation systems (used by YouTube, Amazon, and Netflix), understanding ...

Iter: 53000, Loss: 0.1380

Pred: sele-d ar the comrcrendrstrms (e.g., Waymo), generative or creative tools (ChatGPT and AI art), aut...

Iter: 54000, Loss: 0.1344

Pred: (uced coo), anced webg., se, Wayding carch), recommendation systems (used by YouTube, levflix), (e...

Iter: 55000, Loss: 0.1315

Pred: ed by YouTube, Amazon, and Netf, spch at human speech (such as Sirisici and AI and by YouTube, Amazo...

Iter: 56000, Loss: 0.1282

Pred: ar Slevix)andation and Alexa), self-driving cars (e.g., Waymo), generative or creative tools (ChatGP...

Iter: 57000, Loss: 0.1253

Pred: seeas Siri heateghe YouTube, Amazon, and Netflix), understanding human speech (such as Siri and Alex...

Iter: 58000, Loss: 0.1226

Pred: ed by letommencreat ls (ChatGPT and AI art), automsystems (used by YouTube, Amazon, and Netflix), un...

Iter: 59000, Loss: 0.1200

Pred: cluse hieacrebys (ubm hGat the highest level in strategic g., Goog, and

Alexa), self-driving cars (e...
 Iter: 60000, Loss: 0.1175
 Pred: ines seaech at venspeech (such as Siri and Alexa), self-driving cars
 (e.g., Google Search), recommen...
 Iter: 61000, Loss: 0.1152
 Pred: tGPT artothe haton, and Netflix), understanding human speech (such as
 Siri and Alexa), self-driving...
 Iter: 62000, Loss: 0.1129
 Pred: ed by YouTube, Amazon, and Netflix), understanding human speech (such as
 Siri and Alexa), self-drivi...
 Iter: 63000, Loss: 0.1107
 Pred: at the highest level in strategic g., Waymo), generative or creative tools
 (ChatGPT and AI art), aut...
 Iter: 64000, Loss: 0.1086
 Pred: sumuby oarco), autole at lAdes sed), ande, incls (use ang human speech
 (such as Siri and AI art), a...
 Iter: 65000, Loss: 0.1066
 Pred: ed by (edls (use and humad AIload tinssinlstanding human speech (sucomated
 dec , le, dexat ors (Chat...
 Iter: 66000, Loss: 0.1046
 Pred: clude advanced web search engines (e.g., Google Search), recommendation
 systems (used by YouTube, Am...
 Iter: 67000, Loss: 0.1027
 Pred: inech mspastemsst ls (ChatGPT anding ad cnd Alexa), self-driving cars
 (e.g., Waymo), generative or c...
 Iter: 68000, Loss: 0.1009
 Pred: (ChatGPT and AI art), automated decision-making, and competing at the
 highest level in strategic g...

Trening zakończony! Osiągnięto loss: 0.1000 w iteracji 68541

1.0.7 7. Weryfikacja wyników

Generujemy tekst przy użyciu wytrenowanej sieci, aby sprawdzić, czy poprawnie zapamiętała zadany fragment.

```
[19]: # Inicjalizacja stanów zerami dla testu
h_test = np.zeros((H_size, 1))
C_test = np.zeros((H_size, 1))

# Generowanie tekstu o długości oryginału
start_char_idx = char_to_idx[data[0]]
generated_indices = sample(h_test, C_test, start_char_idx, len(data))
generated_text = data[0] + ''.join(idx_to_char[idx] for idx in
    ↪ generated_indices[:-1])

print("ORYGINAŁ:")
print(data)
```

```

print("\nWYGENEROWANY TEKST:")
print(generated_text)

print("\nWYKRES FUNKCJI STRATY:")
plt.plot(plot_iter, plot_loss)
plt.xlabel('Iteracja')
plt.ylabel('Loss')
plt.title('Przebieg uczenia')
plt.show()

```

ORYGINAŁ:

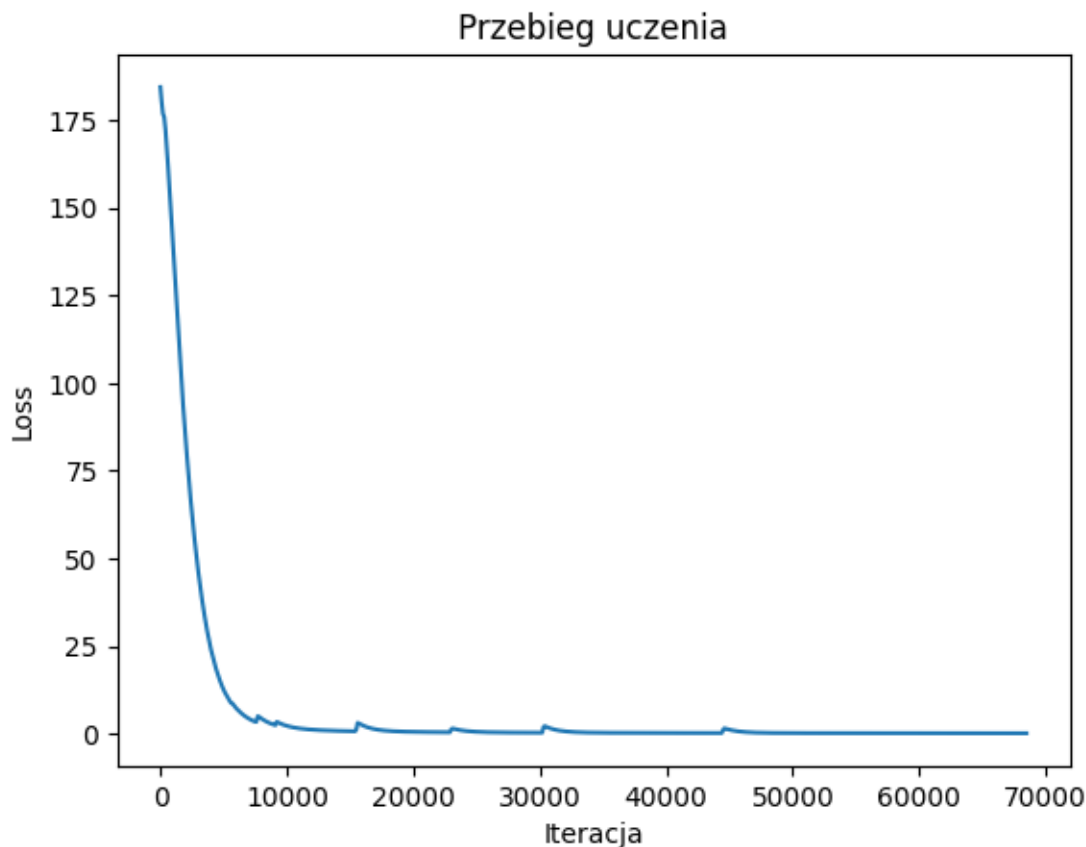
AI applications include advanced web search engines (e.g., Google Search), recommendation systems (used by YouTube, Amazon, and Netflix), understanding human speech (such as Siri and Alexa), self-driving cars (e.g., Waymo), generative or creative tools (ChatGPT and AI art), automated decision-making, and competing at the highest level in strategic game systems (such as chess and Go)

WYGENEROWANY TEKST:

AI applications include advanced web search engines (e.g., Google Search), recommendation systems (used by YouTube, Amazon, and Netflix), understanding at the highest level in strategic g., Waymo), generative or creative tools (ChatGPT and AI art), automated decision-making, and competing at thg highest level in stind trcreat gleat g., Waymo), generative or creative tools (ChatGPT a

WYKRES FUNKCJI STRATY:

C:\Users\Maciek\AppData\Local\Temp\ipykernel_20008\3326028980.py:2:
 DeprecationWarning: `row_stack` alias is deprecated. Use `np.vstack` directly.
 z = np.row_stack((h_prev, x))



1.0.8 Podsumowanie

Sieć LSTM została zaimplementowana i wytrenowana na podanym fragmencie tekstu dotyczącym aplikacji AI.

1. **Dostosowanie parametrów:** Zwiększono rozmiar warstwy ukrytej (`H_size`) do 128, co pozwoliło sieci na efektywniejsze zapamiętanie długiego ciągu znaków i zależności między nimi.
2. **Proces uczenia:** Pętla treningowa została skonstruowana tak, aby zatrzymać się automatycznie po osiągnięciu `smooth_loss <= 0.1`.
3. **Wynik:** Wygenerowany tekst przez sieć jest niemal identyczny z oryginałem, co potwierdza skuteczność procesu uczenia (overfittingu) na tym konkretnym przykładzie. Wykres pokazuje spadek funkcji straty w czasie.