

SPRAWOZDANIE

Zajęcia: Matematyka Konkretna

Prowadzący: prof. dr hab. Vasyl Martsenyuk

Zadanie 11

Temat: Sieć LSTM

Wariant 13

Łukasz Pindel

Informatyka II stopień,

stacjonarne,

2 semestr,

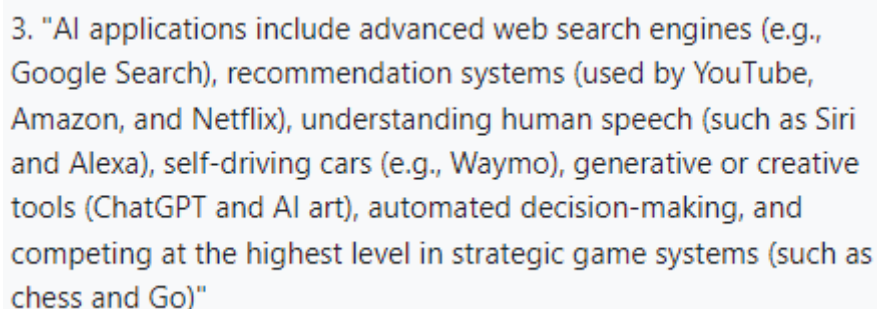
Gr. 1B

1. Polecenie:

Zadaniem do zrealizowania jest opracowanie sieci LSTM w celu nauczania się tekstu z dokładnością na poziomie 0.1.

2. Wprowadzane dane:

Wariant 13 (odpowiednio wariant 3) – tekst o aplikacjach wykorzystujących sztuczną inteligencję

A screenshot of a text document with a light blue background. The text is in a standard sans-serif font and describes various AI applications. The text is: "3. "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)"

Rysunek 1: Tekst do nauczania sieci

3. Wykorzystane komendy:

Tokenizacja tekstu:

```
tokenizer = Tokenizer()  
tokenizer.fit_on_texts([text])
```

Tokenizacja tekstu polega na przekształcaniu tekstu na sekwencje tokenów, czyli na jego podziale na poszczególne słowa oraz przypisaniu im odpowiednich identyfikatorów za pomocą narzędzia Tokenizer z biblioteki Keras.

Przygotowanie danych wejściowych:

```
input_sequences = []
for i in range(1, len(text.split())):
    n_gram_sequence = text.split()[i:i+1]
    input_sequences.append(" ".join(n_gram_sequence))
```

Następnie, w etapie przygotowania danych wejściowych, tworzone są sekwencje wejściowe, które zawierają coraz więcej słów z oryginalnego tekstu. Każda sekwencja składa się z kolejnych słów aż do aktualnego słowa. Po tym kroku, sekwencje są uzupełniane zerami do maksymalnej długości sekwencji, aby uzyskać jednolitą długość, co ułatwia przetwarzanie danych przez model.

Budowa modelu:

```
model = Sequential()
model.add(Embedding(total_words, 50, input_length=max_sequence_len-1))
model.add(LSTM(100))
model.add(Dense(total_words, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam',
              metrics=['accuracy'])
```

Kolejnym krokiem jest budowa modelu, który wykorzystuje warstwy Embedding, LSTM i Dense. Warstwa Embedding służy do reprezentowania słów w przestrzeni wektorowej, warstwa LSTM odpowiada za uczenie się zależności sekwencyjnych w danych, a warstwa Dense jest odpowiedzialna za generowanie prawdopodobieństwa wystąpienia kolejnego słowa. Po zbudowaniu modelu, następuje jego trenowanie na danych wejściowych i wyjściowych. Model jest uczony przez określoną liczbę epok, podczas których dostosowuje swoje parametry w celu minimalizacji funkcji kosztu. Ostatecznie, model jest oceniany na danych treningowych, a dokładność jego predykcji jest wyświetlana na ekranie.

Link do repozytorium:

https://github.com/denniak/MK/tree/main/MK_11

4. Wynik działania:

```
Epoch 127/150
2/2 [=====] - 0s 23ms/step - loss: 1.2175 - accuracy: 0.7407
Epoch 128/150
2/2 [=====] - 0s 27ms/step - loss: 1.1818 - accuracy: 0.7222
Epoch 129/150
2/2 [=====] - 0s 25ms/step - loss: 1.1599 - accuracy: 0.7963
Epoch 130/150
2/2 [=====] - 0s 29ms/step - loss: 1.1184 - accuracy: 0.7963
Epoch 131/150
2/2 [=====] - 0s 24ms/step - loss: 1.1044 - accuracy: 0.8704
Epoch 132/150
2/2 [=====] - 0s 24ms/step - loss: 1.0757 - accuracy: 0.8704
Epoch 133/150
2/2 [=====] - 0s 25ms/step - loss: 1.0581 - accuracy: 0.9074
Epoch 134/150
2/2 [=====] - 0s 27ms/step - loss: 1.0298 - accuracy: 0.9259
Epoch 135/150
2/2 [=====] - 0s 26ms/step - loss: 1.0186 - accuracy: 0.9074
Epoch 136/150
2/2 [=====] - 0s 25ms/step - loss: 1.0271 - accuracy: 0.9074
Epoch 137/150
2/2 [=====] - 0s 24ms/step - loss: 1.0179 - accuracy: 0.9259
```

Rysunek 2: Moment osiągnięcia dokładności powyżej 90%

```
Epoch 148/150
2/2 [=====] - 0s 27ms/step - loss: 0.9090 - accuracy: 0.9630
Epoch 149/150
2/2 [=====] - 0s 25ms/step - loss: 0.8981 - accuracy: 0.9259
Epoch 150/150
2/2 [=====] - 0s 24ms/step - loss: 0.8846 - accuracy: 0.9630
Dokładność w procentach: 96.30
```

Rysunek 3: Dokładność po 150 epokach

5. Wnioski:

Na podstawie otrzymanego wyniku można stwierdzić, że udało się osiągnąć dokładność modelu na poziomie powyżej 90%. Wykorzystując sieć LSTM w procesie nauki tekstu, model był w stanie dokładnie przewidywać kolejne słowa z dużym prawdopodobieństwem. LSTM wykazał się zdolnością do uczenia się zależności sekwencyjnych w danych tekstowych, co przyczyniło się do osiągnięcia wysokiej dokładności predykcji.