Natural Language Processing - Assignment 3

h.j.kraaijveld

May 2023

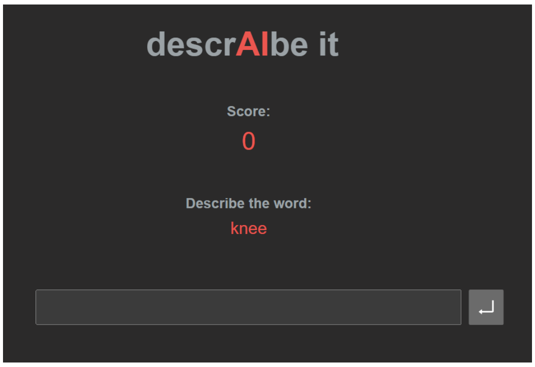
The link to the web app and the Git repository where the code is hosted can be found below:

- Web app name: descrAIbe it
- GitHub repository: Koen-Kraaijveld/nlp-assignment-3
- The assignment was completed individually.

1 Introduction

The web app *descrAIbe it* is based on a game in which players are given a word and are tasked with describing it as best they can without using the word itself. A player can score points if they get it right, otherwise their score is reset back to zero and they must start over. Every round consists of a word being randomly sampled from a pool of 100 unique words. For every word description a player submits, they are also given the class probabilities associated with their answer in descending order.

Figure 1 shows the initial state of the app when running it. It shows the starting score, the first word that the user is tasked with describing, and the input box where they should submit their answer. After entering their answer, the user can either press Enter or click on the button to the right of the input box. Figure 2 shows the results after submitting. If the answer is correct, the score is incremented. An ordered list of probabilities is also generated below the input box showing how well the user describe each of the 100 possible words. In this case, the answer is correct and a new word has already been selected.



descrAIbe it

Score:

0

Describe the word:

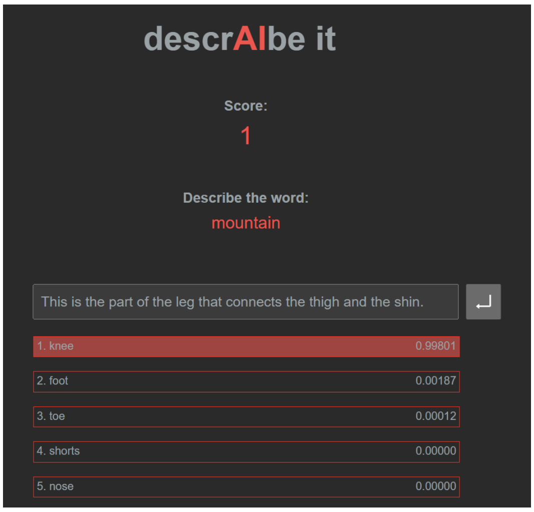
knee

Figure 1: Screenshot showing the initial state of the app.

1.1 Why is *descrAIbe it* linguistically meaningful?

The app is linguistically meaningful because it is a text classification task spread out over 100 different classes (i.e., the possible words that must be described by the player). The model must take into account several linguistic elements in order to determine which class fits best with the player written description. As such, it is able to uncover some of the important patterns underlying how we interpret meanings of words, such as word choice, order, context and literary devices.

As an example, Figure 3 and 4 show the predicted class probabilities for the input *predator* and *prey*. We can see that animals generally considered higher up on the food chain, such as a lion, is associated with the word predator, while more submissive animals lower down are associated with the word prey. Furthermore, the model is also able to linguistically detect some of the more obscure patterns, such as how the word predator may be used in the context of a car through similes and metaphors.



descrAIbe it

Score:

1

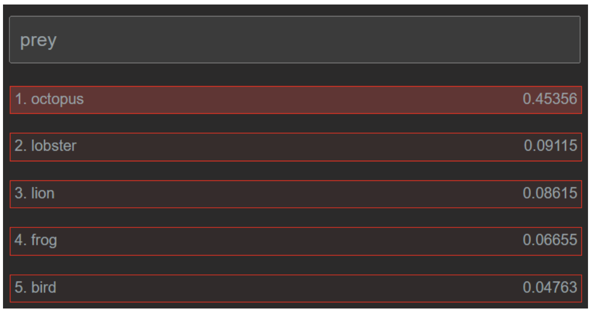
Describe the word:

mountain

This is the part of the leg that connects the thigh and the shin.

1. knee 0.99801
2. foot 0.00187
3. toe 0.00012
4. shorts 0.00000
5. nose 0.00000

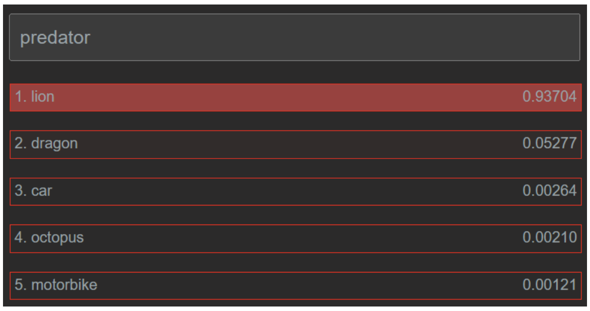
Figure 2: Screenshot showing the state of the app after submitting a description of the given word.



prey

1. octopus 0.45356
2. lobster 0.09115
3. lion 0.08615
4. frog 0.06655
5. bird 0.04763

Figure 3: Predicted class probabilities for the input word *prey*.



predator

1. lion 0.93704
2. dragon 0.05277
3. car 0.00264
4. octopus 0.00210
5. motorbike 0.00121

Figure 4: Predicted class probabilities for the input word *predator*.

2 Dataset

The entire dataset that is used consists of 96,000 descriptions of varying length for 100 words with 960 descriptions each. These were collected by prompting OpenAI's ChatGPT and asking it to give 20 descriptions of each word per prompt. The prompts were parameterized by the following:

- Level of detail, which determines how long each response should. These parameters were *very short*, *regular*, and *very long*.
- Complexity, which determines how advanced the word choice in the response should be. These parameters were *very simple*, *simple*, *complex*, and *very complex*.
- Prefix, which determines how each ChatGPT should start each of its responses. These parameters were *It*, *This*, *A*, and *The*.

Every combinations of these parameters were used to design a prompt, giving a total of 48 prompts per word.

3 Method

3.1 Preprocessing

The preprocessing that was conducted was basic text cleaning (e.g., removing punctuation, hyperlinks, double whitespaces, etc.) and tokenization, which was done in Keras. Furthermore, the labels were encoded as integers (no one-hot encoding). The train-test split ratio was 60:40 with an additional 20% split for the validation data.

3.2 Model

The data was passed to an LSTM model, which consisted of the following layers:

1. Embedding layer. This embedding layer used pretrained GloVe word embeddings, which had a dimensionality of 100. It was trained on 6 billion tokens from Wikipedia 2014 + Gigaword (newswire text data) and has a vocabulary of 400,000 words.

2. LSTM layer (no dropout). This layer consists of 128 units.

3. 1D Global Max Pooling layer. This layer reduces the dimensionality to the maximum vector over the steps dimension.

4. 3 Linear layers. These layers consist of 256 hidden units with a ReLU activation. Each layer also adds Dropout with a probability of 0.7 in order to combat overfitting.

5. 1 Linear output layer. This layer consists of 100 outputs with a Softmax activation.

An Adam optimizer was used with learning rate $\alpha = 0.001$, as well as the default $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Sparse Cross Entropy was used for the loss function, as opposed to regular Cross Entropy because the labels were not one-hot encoded. Early Stopping was also employed with a patience of 15 epochs and which was based on the validation loss. Lastly, the model was fitted with a batch size of 64.

4 Evaluation

For the evaluation, we report the accuracy because the dataset has a class imbalance. The results are as follows:

- Training accuracy: 83.48%
- Validation accuracy: 79.55%
- Test set accuracy: 78.68%