Problem Set 1: Locating Word Referents

Discovery Procedure, Fall 2023
Logan Swanson

Code Structure

word_learning.py: this file loads the training and testing data, and provides three functions for performing learning. The `pursuit` function takes four arguments: the learning rate, the smoothing parameter, the threshold value for learning, and a boolean parameter determining whether sampling should take place. The output of each function is a dictionary matching words to objects.

Results

Reported results for cross-situational, pursuit, and pursuit with sampling are the average scores from 1,000 trials. Hyperparemeter values for the Pursuit algorithm were set to: learning rate of 0.02, smoothing parameter of 0.05, and learning threshold of 0.15.

	Precision	Recall	F1
PbV	0.03	0.47	0.07
X-Situational	0.04	0.53	0.07
Pursuit	0.20	0.26	0.23
Pursuit with Sampling	0.06	0.39	0.10

Input data shuffling: The experiment was repeated with shuffled input data:
trial 1:

	Precision	Recall	F1
PbV	0.04	0.50	0.07
X-Situational	0.03	0.47	0.06
Pursuit	0.09	0.06	0.07
Pursuit with Sampling	0.06	0.38	0.10

trial 2:

	Precision	Recall	F1
PbV	0.03	0.47	0.06

X-Situational	0.04	0.56	0.08
Pursuit	0.18	0.12	0.14
Pursuit with Sampling	0.04	0.26	0.07

Discussion

My results generally corroborate those in Stevens et al 2017, with pursuit being the strongest-performing model. The addition of sampling seems to weaken the Pursuit algorithm, rather than helping it.

Interestingly, while Propose but Verify and Cross-Situational learning appear agnostic to the ordering of the input data, Pursuit seems sensitive to it. This may be because of the structure of typical discourse: the same objects tend to be salient in the discourse for consecutive periods. This may enable the association between the correct word and object to become repeatedly strengthened without taking intervening penalties.