ODD Framework for an Agent Based Language Shift Model

Tyler Pawlaczyk
pawlactb@clarkson.edu

September 26, 2017

1 Purpose

This model seeks to explain the dynamics of Language Shift in a location, specifically Southern Austria. With little modification from the provided example, uses include simulating the spatial spread of language trends. and any other phenomena that can be postulated as a spatial spread.

2 Entities, State Variables and Scales

2.1 Overview

There is only one acting agent in this model, the LanguageAgent. All LanguageAgents are contained within the same LanguageModel. LanguageModels also contain another singular entity to handle neighborhood calculations, the NeighborList.

2.2 LanguageAgent

The LanguageAgent represents a single location during the duration of the simulation. In all typical uses of this model, the typical state variables will be employed by the LanguageAgents. Extraneous state variables are used in special cases. The extraneous state variables mentioned here are primarily for comparing with results with Prochazaka's 2017 findings.

Many LanguageAgents are contained in a LanguageModel. The neighborhood relation of LanguageAgents is determined by the LanguageModel's NeighborList.

2.2.1 Typical State Variables

- `name` (string)
  Each LanguageAgent contains a string field that is the name of the location it represents.
- `pos` (tuple: (float, float))
  The position of the LanguageAgent. In the provided example, this is the latitude/longitude coordinates of the location. `pos` is used by the NeighborList to determine the LanguageAgent's neighborhood.
- `probability, next_probability` (double)
  The `probability` and `next_probability` store information regarding the probability of speaking either language for the current and next timestep. `next_probability` is required due to the design of the scheduler.
- `population` (integer)
  Each region represented by a LanguageAgent has a population for each timestep. The population is updated every timestep.

2.2.2 Extraneous State Variables

- `p_probability`, `p_next_probability` (double)
  Used to store the probabilities calculated as described in Prochazaka's 2017 paper.

2.2.3 Scale

In the provided example, each LanguageAgent represents a region approximately 1 square kilometer in area.

2.3 LanguageModel

The LanguageModel is the "container" in which the simulations happen. The LanguageModel is responsible for reading in initialization data, initializing LanguageAgent objects, and collecting data.

2.3.1 Typical State Variables

- `diffusion` (list: [float])
  Represents the diffusivity of each language. In the provided simulation, the first element of the list is the German diffusivity, and the second is the Slovene diffusivity. Usually size of two.
- `pop_data` (DataFrame: int)
  Maintains a pandas dataframe of populations for every timestep.
- `neighbors` (NeighborList)
  Maintains a list of neighbors to each agent.
- `agents` (list: [LanguageAgent])
  A list of all agents indexed by their `unique_id`.

2.3.2 Scale

The LanguageModel is scaled for the whole simulation in all times and spaces.

2.4 NeighborList

The NeighborList is responsible initially for generating the nearest neighbors of each agent, and then maintaining a list of neighbors for each agent.

2.4.1 Implementation Details

The NeighborList first obtains the position of all LanguageAgents in the LanguageModel. The NeighborList first calculates the distance for each LanguageAgent to all other LanguageAgents. The LanguageAgents are then loaded into a priority minimum queue by their distance. In order to save computing time, the priority queue is only popped for the number of neighbors needed for the simulation (usually 2-10). The neighbors are then stored in a list within a dictionary. To obtain the neighbors of a LanguageAgent, one queries the dictionary of neighbors with the `unique_id` of the LanguageAgent and is returned a list of unique_ids of neighbors, in order of increasing distance. The NeighborList can also be serialized and stored as a Python Pickle, to avoid repeated lengthy calculation when changing other model parameters.

3 Process Overview and Scheduling

When the model is ran under typical execution, the LanguageModel is initialized first. The LanguageModel reads in the location and populations of all LanguageAgents (creating them as needed) and passes this information onto the NeighborList, if no NeighborList serialization is available.

3.1 Data Loading and Neighbor Calculations

If serialization is available via Python Pickle, the pickle file is loaded into the NeighborList to avoid the (potentially lengthy) neightbor calculation. To determine neighbors, first the NeighborList calculates the distance from every LanguageAgent to every other LanguageAgent. Then, for every LanguageAgent in the simulation the other LanguageAgents' `unique_id` is loaded into a minimum-priority queue. The queue is then popped a satisfactory amount to obtain the closest neighbors to each LanguageAgent. The `unique_id` of each neighbor is then stored in a list as the attachment of a dictionary for ease of neighbor lookup.

3.2 Model Runtime

After the neighbors have been calculated and the LanguageAgents are initialized by the LanguageModel, the model is ready to begin simulation.

3.2.1 Scheduling

Each LanguageAgent has two methods, called `step()` and `advance()`. The `step()` method is responsible for calculating the contribution to the current cell from it's neighbors. In order to solve problems with concurrency of LanguageAgents updating before their neighbors (causing temporal inconsistencies) the LanguageAgent's do not update their state variables until the `advance()` method is called.
In the `step()` routine, the LanguageAgent k first obtains its neighbors from the LanguageModel's NeighborList. The contribution from each neighbor r to k is defined as

$$C_{\alpha r} = \frac{n_{\alpha r}}{4\pi D_\alpha \Delta t} \exp\left(-\frac{|\text{dist}|^2}{4D_\alpha \Delta t}\right), \tag{1}$$

with $n_{\alpha r}$ being the number of speakers of language α in neighboring LanguageAgent r at time t, dist being the distance between the LanguageAgents, D_α being the diffusion of language α, and Δt being the timestep size of the model. The contribution from all neighbors for language α, F is defined as

$$F_\alpha = \sum_{\text{neigh.}} C_\alpha. \tag{2}$$

The `next_probability`, P_α, of a LanguageAgent is determined as:

$$P_\alpha = \frac{n_{\alpha k} + F_\alpha}{\sum_{\text{lang.}} n_{\alpha k} + \sum_{\text{lang.}} F_\alpha}\ . \tag{3}$$

After the `step()` for all LanguageAgents, the `advance()` method updates the state variables for the next timestep.

3.2.2 Data Collection

4 Input Data

5 Submodels

6 Extensions/Future Work