Individual Project June 2025

Modeling Jespersen's Cycle: A Closer Look at a Computational Approach to the Evolution of Sentential Negation

Raya Mezeklieva

Master of Logic, ILLC, University of Amsterdam

July 6, 2025

Abstract

Jespersen's Cycle (JC) is a well-established diachronic pattern in the expression of sentential negation in various languages. It involves a shift from preverbal negation to a bipartite or discontinuous stage (where negation is expressed using two markers–one on either side of the verb), to postverbal negation, and possibly, though controversially, back to preverbal negation. While it has predominantly been the object of corpus-based descriptive analyses (e.g.,), Lopopolo and Biró (2011) delineate it using a computational model of language evolution. Their approach extends Optimality Theory (Swart, 2010) with a Simulated Annealing algorithm that models error-prone language production and successfully captures the individual stages of JC in isolation. This model is then embedded within an agent-based, iterated learning framework to simulate language evolution over generations. Although the simulation reproduces the first transition of JC, it fails to account for the later stages.

The current work extends my efforts to address the limitations of this model. The first part of the project consists of a closer examination of my initial reproduction of Lopopolo and Biró (2011) with a diagnosis of the model's shortcomings. In the second part, I propose an alternative, simpler modeling approach based on Bayesian inference.

1 Introduction

I began this project with a critical review of my initial replication of Lopopolo and Biró (2011)[1]. This inspection revealed some inaccuracies and superfluous convolutions in my original implementation and motivated me to create a cleaner, more faithful version that facilitates experimentation. However, upon comparing the results from the original and revised versions, I observed unexpected differences in simulation outcomes. To diagnose the source of this divergence, I systematically investigated the behavior of the individual model components and assessed the influence of core parameters on model dynamics. These investigations ultimately revealed a deeper limitation in the formalization of Simulated Annealing for Optimality Theory (SA-OT) as a model of JC, which I hypothesize constrains the model's ability to generate the entirety of JC within the agent-based simulation.

[1]The code and report can be found here: https://github.com/raya-mez/JC-ABM.

Finally, this project aimed to (at least) prototype a more comprehensive computational account of JC that captures the entire cycle as an emergent outcome of language transmission and learning.

Thus, the work presented in this report is organized around the following goals:

1. To simplify and reimplement the simulation in a more accurate and accessible way;
2. To isolate and investigate the behavior of individual model components;
3. To explore the impact of different parametrizations on the model's fit and internal dynamics;
4. To develop a model that accounts for the full trajectory of JC.

The code and this report can be found on the following GitHub page: https://github.com/rayamez/JC-ABM-2.0.

2 Background

2.1 Jespersen's Cycle

Jespersen's cycle (JC) (Dahl, 1979; Jespersen, 1917) is one of the most influential theories explaining the diachronic development in the expression of sentential negation (SN) across numerous languages. It has been documented in Germanic languages (Breitbarth & Haegeman, 2010), Arabic (Lucas, 2020), and is currently underway in French (Labelle, 2019). JC consists of a recurring pattern in which the original SN marker placed preverbally weakens over time, prompting reinforcement by an additional word–often a minimizer rather than a true negator, placed postverbally–which eventually supplants the original marker. According to the initial formalization of JC, the new SN marker may itself undergo further renewal (Jespersen, 1917), however, whether the historical attestation of a closure of the cycle is contested (Lopopolo & Biró, 2011; Swart, 2010). JC is schematically summarized in Figure 1.

2.2 SA-OT Model of JC

Simulated Annealing for Optimality Theory (SA-OT) (Lopopolo & Biró, 2011) extends classical Optimality Theory (OT)–a model of grammar, which can be understood as linguistic competence–by integrating a stochastic search algorithm that models linguistic performance. As such, it relies on the basic components of standard OT: a candidate set of linguistic forms, and a set of constraints, ranked into hierarchies that represent different grammars (more on this below).

In addition to the components inherited from OT, the SA-OT model introduces two new elements that allow modeling variation in linguistic output given the same grammar competence: a **neighborhood structure** (or topology) that organizes the candidate set by structural similarity, and a **search**

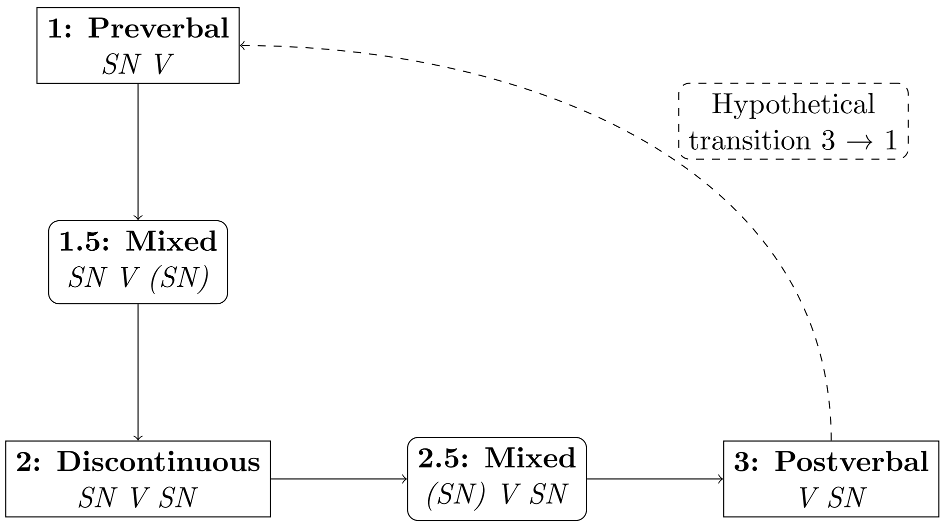


Figure 1: Jespersen's cycle: three core stages and two mixed stages. The return from postverbal to preverbal negation is controversial and lacks strong evidence. Figure adapted from (Lopopolo & Biró, 2011).

algorithm that performs a random walk through this candidate space, guided by a harmony function aligning the search with the grammar. Since the search is stochastic, SA-OT does not always find the globally optimal candidate, i.e., the grammatical form. Instead, it might settle on local optima–suboptimal outputs, e.g., errors arising from fast speech, slips of the tongue, or tolerable irregularities. This is how SA-OT models linguistic performance, rather than solely idealized grammatical competence, and accounts for linguistic variation and change.

2.2.1 Candidates and Topology

A **candidate** is a pair of underlying form and surface form (uf, sf), where uf represents the semantics, i.e., the polarity of the utterance (positive or negative), and sf is a binary syntactic tree, composed of a main verb (V) and zero or more sentential negation markers (SN). Note that in this model, we are only interested in negative uf.

The **neighbors** of a candidate are candidates whose sf's are the results of transforming the original candidate's sf via the following basic steps: (1) Add an SN at the beginning (left side); (2) Add an SN at the end (right side); (3) Remove the outermost SN; (4) Reverse the order of any daughter nodes (e.g., turn [SN, V] into [V, SN]). Note that resulting the neighborhood structure (topology) is thus infinite.

2.2.2 Grammar: Constraints, Hierarchies, and K-values

The **grammar** is defined based on a set of violable constraints, which express preferences or requirements on surface forms. Directly based on Swart (2010) the following four constraints are considered:

1. **Faith[Neg]**: a faithfulness constraint requiring that the polarity (uf) of the candidate matches the presence or absence of SN in the sf. It assigns one violation mark for a mismatch. Note that since the input will only be negative polarity, 1 mark is given if there is no SN in the candidate sf.

2. *Neg: a markedness constraint assigning punishing every occurrence of SN in the sf with one violation mark, that is, it assigns as many violation marks as the number of SN's in the sf.

3. **NegFirst**: a markedness constraint assigning one violation mark if a candidate has no preverbal SN.

4. **NegLast** (called FocusLast in Swart, 2010): a markedness constraint assigning one violation mark to candidates with no postverbal SN.

These constraints are ranked in hierarchies and assigned ranking values (called K-values), such that the higher the position of a constraint in a hierarchy, the higher its ranking value, and thus the more costly its violation. These constraints rakings represent the possible grammars or language variants. Table 1 provides an overview of the different grammars considered in the model, and the stage in JC that they represent. For example, one possible grammar could be [Faith-Neg: 4, *Neg: 3, NegFirst: 2, NegLast: 1], which corresponds to Hierarchy 1, representing preverbal negation.

Hierarchy					**JC stage**
H1	Faith[Neg]	*Neg	NegFirst	NegLast	1: preverbal
H2	Faith[Neg]	NegFirst	*Neg	NegLast	1.5: mixed
H3	Faith[Neg]	NegFirst	NegLast	*Neg	2: discontinuous
H4	Faith[Neg]	NegLast	NegFirst	*Neg	2: discontinuous
H5	Faith[Neg]	NegLast	*Neg	NegFirst	2.5: mixed
H6	Faith[Neg]	*Neg	NegLast	NegFirst	3: postverbal
K-value	4	3	2	1	

Table 1: Six possible constraint rankings, where Faith[Neg] is always ranked highest. The K-values represent the relative importance of each constraint, with higher K-values indicating higher priority. Each hierarchy is associated with a corresponding JC stage as identified by Lopopolo and Biró (2011).

2.2.3 Harmony and SA-OT Algorithm

In OT, the optimal form given a constraint hierarchy is determined by the harmony function, which compares the constraint violations of the candidates in the candidate set. The candidate that best satisfies the constraints is the one that minimizes the number of violations of higher-ranked constraints. Thus, the harmony function in OT always finds the optimal form given the grammar. Note that this requires that the candidate set be finite.

On the other hand, SA-OT models linguistic performance by *searching* for the best candidate in the candidate topology given a certain grammar. Thus, performance emerges from the topology (which may be infinite) and the search algorithm heuristic. The search algorithm used in SA-OT is simulated annealing (SA). It consists of a random walk over the topology, starting from one candidate and exploring its neighbors. The search is guided by the harmony function, i.e., how well each form satisfies the constraint hierarchy[2]. At each step, the algorithm: (1) picks a random neighbor, (2) compares the harmony of the initial candidate and the chosen neighborz: If the neighbor is equally good or better, move to it; if the neighbor is worse, decide whether to move to it based on a transition probability function. The algorithm controls the exploration with a temperature parameter: at the start, the temperature is high, so the algorithm explores widely, even worse options, but over time, the temperature cools, and the algorithm becomes more selective (moves only to better neighbors). The search terminates when no better neighbors are accepted, meaning that at least a local (if not global) optimum has been reached.

2.2.4 Agent-Based Simulation

The SA-OT algorithm is then integrated within an agent-based model using iterated learning (Kirby & Hurford, 2002) to simulate the population-level change of dominant grammar across generations. This is possible since the frequency with which the SA-OT algorithm produces different forms is directly related to the input grammar (see section 5.2 in Lopopolo & Biró, 2011) and the iterated learning framework allows agents to learn from data with specific frequencies (Kirby & Hurford, 2002). Learning is performed using the Gradual Learning Algoritm (Boersma, 1970). When the learner's prediction differs from the observed target, the ranking values of the constraints in the learner's grammar are updated by increasing the ranking of constraints that favor the target form, and decreasing the ranking of constraints that favor the learner's erroneous form. Once agents have seen enough learning data as specified by the learning data parameter, a sample of surface forms produced through SA-OT is recorded. This sample then allows tracking linguistic change over generations.

The SA-OT agent-based model successfully reproduces the first transition of JC–the shift from preverbal to discontinuous negation. However, it stabilizes at this stage and fails to account for the rest of cycle's trajectory.

2.3 Previous Replication of the SA-OT Model

In a previous project, I set out to address the shortcomings of the SA-OT model in accounting for JC (the code and report are available on https://github.com/rayamez/JC-ABM). I replicated the findings of Lopopolo and Biró (2011) in Python using the Mesa framework for agent-based modeling (Hoeven et al., 2025) after reducing the learning data parameter from the 300 set by Lopopolo and Biró to only 25. I then explored a Bayesian variant of the model, replacing the complex SA-OT algorithm with simple Bayesian inference. The first transition of

[2]More precisely, the random walk becomes hill climbing where the horizontal component is the search space of candidates defined by the neighborhood function and the vertical component is the harmony function.

JC emerged under a Maximum A Posteriori (MAP) strategu for surface form generation, where agents always select the grammar with the highest posterior probability based on which they sample an utterance from the candidate space, weighted by the candidate likelihoods given the chosen grammar. However, the transition did not occur under a Sampling strategy, where agents sample a grammar based on their posterior probabilities rather than always selecting the most probable one given their posterior. This was expected in light of the comparison of the effect of the two strategies in iterated learning scenarios in Kirby, Griffiths, and Smith (2014), showing that only the Sampler leads to *convergence to the prior* (Griffiths, Kalish, & Lewandowsky, 2008).

Given that none of the models accounted for the entirety of JC, this project continued the effort towards a full account of this common diocrhonic linguistic pattern. It focused on analysing the shortcomings of the discussed models to enable a more informed approach to their refinement.

3 Improving SA-OT Model Implementation

3.1 Correcting Constraint Rank Values

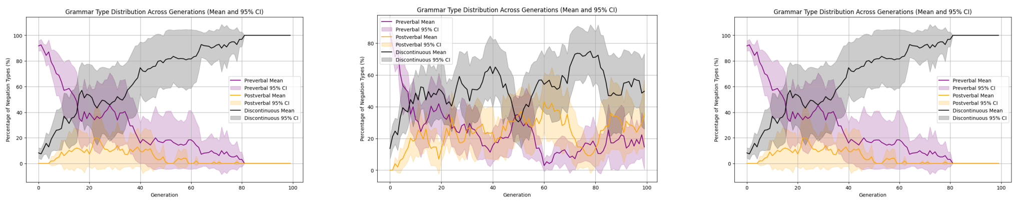
A review of my earlier implementation of the agent-based simulation revealed a discrepancy in the initialization of the constraint rank values or K-values of agents' grammars, relative to the procedure described in Lopopolo and Biró (2011). I had initialized them using the default ranks listed in the Appendix: "The default ranks were: 4 for the highest ranked constraint Faith[Neg], and 3, 2 and 1 for the markedness constraints, in decreasing order following the hierarchy." To the best of my understanding, these default parameters were used in the experiments leading up to the agent-based simulation–namely, those demonstrating that particular hierarchies correspond to distinct stages of Jespersen's Cycle, and those exploring the effect of lowering the K-value of the lowest-ranked constraint (see Section 5 of Lopopolo & Biró, 2011).

However, footnote 4 in Section 6 of Lopopolo and Biró (2011), where the agent-based simulation is introduced, specifies a different initialization procedure: "Constraint Faith[Neg] was assigned rank 4.9, and the markedness constraints were associated with a random floating point value between -0.1 and 4.9. The standard parameters of the SA-OT Algorithm (`K_max` = 5, `K_step` = 1, `t_step` = 1, etc.) were used, as discussed in the Appendix" (Lopopolo & Biró, 2011, p. 14). In light of this, I reran the simulation using the corrected K-value parametrization, keeping all other settings constant. However, the paper does not explicitly state how the K-values of agents in the *initial* generation were initialized–whether using the default values or random floats–therefore, I considered both possibilities. The mean outcomes of 10 repeated simulations under each initialization scenario are shown in Figure 2[3]. Panel A uses default K-values for all generations, Panel B uses random initialization throughout, and

[3]Note that Lopopolo and Biró (2011) and my earlier replication repeated the simulation 20 times, but this number was reduced here due to low added insights for a considerably higher computational cost

Panel C combines default initialization for Generation 0 with randomized values for subsequent generations.

The plots reveal that the simulation outcome is highly sensitive to how K-values are initialized. Panel A confirms that using default K-values throughout generations leads to a robust and stable shift from preverbal to discontinuous grammar, reproducing the intended diachronic trajectory observed by Lopopolo and Biró (2011). However, a full randomization of K-values for every generation leads to unpredictable trajectories, preventing reliable convergence. The model behaves more stochastically and fails to simulate consistent historical change. Interestingly, when only the initial generation uses default K-values and subsequent ones are randomly initialized, the model still converges reliably toward a discontinuous grammar. This suggests that the initial grammar bias is more influential than ongoing K-value randomness, at least under the current parameter setting and highlights that initial conditions are critical in modeling grammatical evolution through iterated learning.



Panel A: Generation 0 + subsequent generations use default K-values

Panel B: Generation 0 + subsequent generations use random K-values

Panel C: Generation 0 uses default; subsequent generations use random K-values

Figure 2: Comparison of agent-based simulation outcomes under different K-value initialization conditions. Lines represent the mean proportion of negation types (preverbal, discontinuous, and postverbal in purple, black, and yellow, respectively) over 10 runs with the shaded region indicating the 95% confidence interval.

Two implementation details concerning the use of randomly initialized K-values require clarification. First, because these values are sampled as floating-point numbers with inconsistent spacing, an additional criterion had to be enforced to ensure that the markedness constraints do not surpass the rank of the faithfulness constraint (Faith[Neg]), which must remain highest with a K-value of 4.9. Without this cap, invalid hierarchies where the faithfulness constraint is not assigned the highest importance, could arise. Second, in some runs, the SA-OT algorithm occasionally produced structurally invalid forms containing more than two negation markers. The exact cause of this behavior remains unclear.

3.2 Moving Away From Mesa

Lopopolo and Biró (2011) have created their computational model using OTKit (Biró, 2010)–a Java-based software package developed by one of the authors, offering tools for Optimality Theory. Despite the availability of this toolkit, I did not utilize it and instead built all of my implementations from scratch in Python. This decision was motivated by my desire to thoroughly understand the model–all of its individual components as well as how they interact–, to have greater control and flexibility over its workings, and to test the reproducibility of the results when not relying on the dedicated platform.

My initial implementation of the agent-based simulation was built using the Mesa framework for Python (Hoeven et al., 2025), a library designed for modeling multi-agent systems. While Mesa offers a range of useful functionalities, many of these were unnecessary for the relatively simple structure of the model under investigation, which could be implemented using basic Python functions and classes. In line with the first goal of this project–enhancing accuracy and accessibility of the code–I reimplemented the model without relying on Mesa.

When comparing the outputs of my original and simplified implementations, I occasionally observed divergent patterns, despite all individual components behaving identically when tested in isolation. Given the model's strong reliance on stochasticity, this prompted a closer examination of Mesa's internal randomness-related mechanisms. Notably, Mesa uses its own random number generator, which is automatically seeded with the current time. Although it is possible to manually set this seed during model initialization, the risk of subtle inconsistencies led me to further favor a full removal of Mesa. This decision improves the reproducibility and interpretability of the simulation by allowing explicit control over all sources of randomness, should it be needed.

3.3 Investigating the Role of Model Parameters

In my original replication of the agent-based simulation, the transition from preverbal to discontinuous negation only emerged when I substantially reduced the amount of learning data available to each new agent from the 300 examples used by Lopopolo and Biró (2011) to just 25. When agents received more than 30 examples, the change failed to occur: each new generation fully acquired the constraint hierarchy of its predecessor, resulting in a stable preverbal negation pattern. This underscores that 'imperfect mental computation" (i.e., "production errors") alone is insufficient to drive diachronic change in this model. Instead, "imperfect learning" due to limited exposure to learning data (which may still contain production errors) appears to be a necessary condition for grammar divergence across generations (refer to Lopopolo & Biró, 2011 Sections 5 and 6 for a more detailed discussion on the factors driving linguistic change in their model).

This difference in parametrization persisted in the simplified implementation without Mesa, prompting a closer look at the learning data parameter. In the agent-based model, learning proceeds as follows: a newborn agent uses its current grammar (i.e., ranked constraints) to generate a surface form via the SA-OT algorithm. The output represents an utterance that the learner would produce given its current grammar competence, with a possibility of committing a production error. An adult agent from the previous generation is then randomly sampled to produce an utterance using its own grammar. The adult production will be used by the learner to adjust its linguistic competence–i.e., the K-value parameters of is grammar. If the two outputs differ, the learner updates the ranks of the constraints in its grammar by one step (set to 0.1 following Lopopolo & Biró, 2011) using the Gradual Learning Algorithm (GLA; Boersma, 1970) to more closely match the adult's production. This is done by increasing the K-value of constraints that favor the target form and decrease the K-value of constraints that favor the learner's erroneous form in terms of violation scores assigned to the respective surface forms. This learning process is repeated as many times as specified by the learning data parameter.

Naturally, the distance between the initial grammar of a newborn and that of the agents in the adult generation will affect how well the learner will be able to acquire the target grammar. Having established that imperfect learning due to limited exposure to learning data plays a crucial role in the occurrence of the desired grammatical change, and given the stochastic initialization of newborns' grammar, it was interesting to analyze the impact of an agent's starting grammar on its learning trajectory.

To this end, I designed a separate experiment (see exp2-convergence_test.ipynb in the project's GitHub repository) investigating how quickly and consistently learners with different initial grammars converge to a target grammar and how their production patterns evolve during the learning process. The goal was to better understand the role of the random initialization of newborn agents in the dynamics of the simulated language evolution. Thus, the experiment was set up as a minimal variant of the agent-based simulation, tracking the learning trajectories of five agents, each representing a different starting grammar, while learning from a single adult agent with the target grammar.

3.3.1 Convergence Speed Analysis

The first test aimed to determine how many learning iterations are required for agents born with different initial grammars to acquire a target grammar corresponding to Hierarchy 1. Thus, one teacher agent was initialized with a Hierarchy 1 grammar, which remained unchanged throughout the experiment. This agent produced target surface forms for five learner agents. Each learner was initialized with one of the remaining hierarchies–Hierarchies 2 through 6–and was exposed to learning data from the teacher until its grammar matched Hierarchy 1 or until a maximum of 300 learning iterations–the amount of learning data used in Lopopolo and Biró (2011)–was reached. The number of learning iterations performed was recorded and the procedure was repeated 100 times for each agent, i.e., for each hierarchy, both with the default and the randomly initialized floating point K-values.

As shown in Figure 3, convergence speed–i.e., the amount of learning iterations needed to transition to a Hierarchy 1 grammar–varied considerably depending on the starting hierarchy. Nevertheless, the mean number of learning iterations needed across starting hierarchies and runs was approximately 25, confirming the reduction of the learning data parameter required in my replication.

Hierarchies 4-6 showed very consistent and fast convergence, especially when

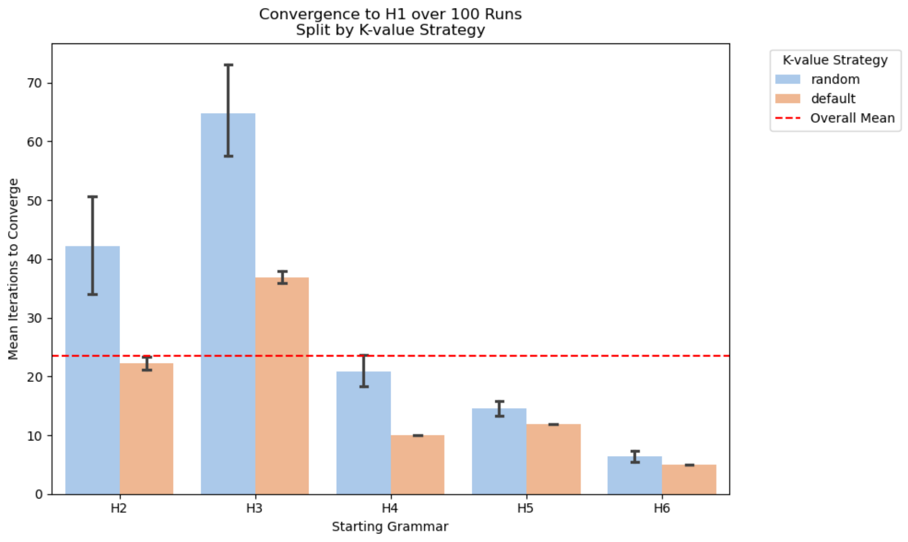


Figure 3: Bar plot of the mean (bar height) and standard deviation (error bars) of the number of learning iterations needed to converge from grammars initialized with Hierarchies 2 through 6 with default K-values (constraint ranks being the integers from 4 down to 1) and random float K-values (constrain ranks being random floats between 4.9 and -0.1 in descending order) over 100 test runs. The horizontal line displays the mean number of learning iterations over all starting hierarchies and runs.

initialized with default K-values. In the case of Hierarchy 4 and Hierarchy 6, convergence time had no variance across runs, suggesting highly stable learning trajectories. In contrast, the learning iterations reached for Hierarchies 2 and 3 were considerably more variable across runs, especially in the case of random float K-values. All in all, Hierarchy 3 was the most disadvantageous as a starting grammar, and Hierarchy 2 converged significantly more slowly than Hierarchies 4-6, despite being structurally closest[4], together with Hierarchy 6, to Hierarchy 1 and having the same global optimum (which Hierarchy 6 does not).

Therefore, it seems that hierarchies that are further away from Hierarchy 1 require fewer learning iterations to converge to this target hierarchy. However, Hierarchy 6 is as distant from Hierarchy 1 as Hierarchy 2 is, indicating that structural dissimilarity is not sufficient to explain the observed results. One distinguishing trait between Hierarchies 2 and 6 is that the former represents a mixed stage of JC, where the SA-OT algorithm produces not only the global optimum [SN V] but also a local optimum [SN V SN], while Hierarchy 6 represents a pure postverbal stage with the SA-OT algorithm producing a single

[4]Structural proximity can be understood here as the number of switches in the order of constraints in a hierarchy that have to be done to obtain the other hierarchy. For instance, Hierarchies 2 and 6 are closest to Hierarchy 1 since they only require one switch (between *Neg and NegFirst for the former and between NegFirst and NegLast for the latter). Another way of visualizing this concept is to count the number of rows that set the two hierarchies apart in the table in section 4.4 on page 30 of Lopopolo and Biró (2011).

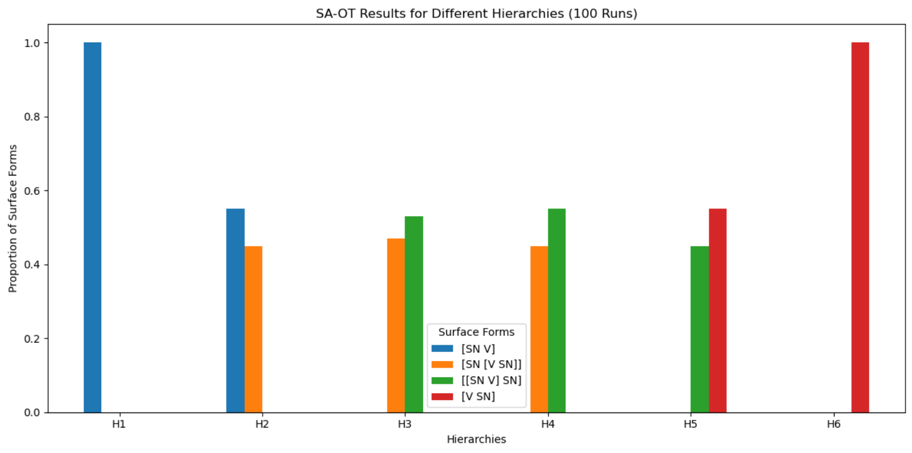


Figure 4: Proportions of surface forms produced by the SA-OT algorithm across six different constraint hierarchies, each run 100 times. The four surface form types–[SN V], [SN [V SN]], [[SN V] SN], and [V SN]–reflect different stages of the JC grammatical development. The plot illustrates Hierarchies 2 and 5–those representing the mixed stages of JC–yield multiple optima, while the others (corresponding to the pure stages of the cycle) consistently converge on a single form.

optimum [V SN] (refer to Table 1 for an overview of the different hierarchies and the stages of JC their represent). Yet, Hierarchy 5 is also a mixed stage with global optimum [V SN] and local optimum [SN V SN], but it converges relatively quickly, while Hierarchy 3 represents the pure discontinuous stage yet it is the slowest to converge. Interestingly, its dual– Hierarchy 4, which also represents the pure discontinuous stage of JC–requires less than average iterations to converge. Hierarchies 3 and 4 only differ in the ordering of the NegFirst and Neg Last constraints resulting in a different preference in terms of SA-OT optima in the branching of the surface form's binary tree such that Hierarchy 3 prefers [[SN V] SN] while Hierarchy 4 prefers [SN [V SN]] as shown in Figure 4.

Taken together, these observations suggest that hierarchies that (1) a more structurally distant from Hierarchy 1, and (2) have SA-OT global optima different from that of the target Hierarchy 1, require fewer learning iterations to converge to this target hierarchy. To test this hypothesis and to better understand the underlying reasons for these observations, a second analysis zoomed into the learning dynamics–specifically, the patterns in the productions of agents during learning which guide their grammar updates.

3.3.2 Analysis of Productions During Learning

The second phase of the learning data investigation focused on analyzing agent productions during the convergence test. Specifically, for each agent, productions that differed from the adult's target form were recorded–i.e., instances in which the learner generated a local (rather than a global) optimum. The results, shown in Figure 5, confirm that only Hierarchies 2 and 5 (x-axis) yield

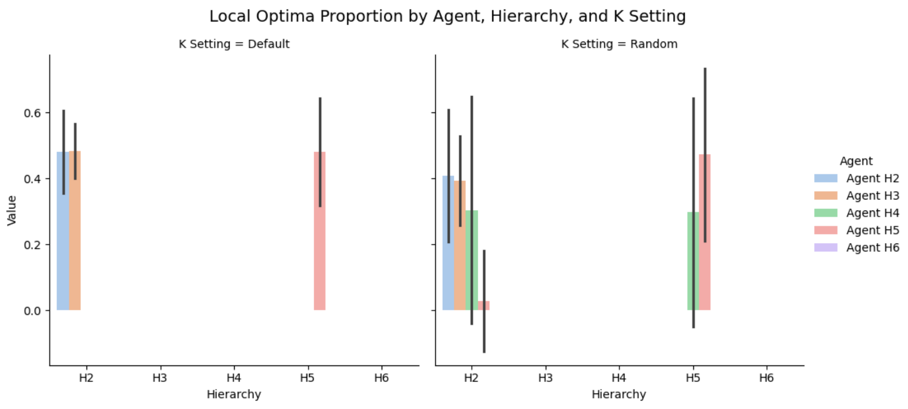


Figure 5: Bar plot of local optima produced by agents during the convergence test.

stable local optima. More interestingly, some agents initialized with pure-stage grammars also produced such production errors during learning–most notably the agent starting from Hierarchy 3. This can be explained by the incremental nature of GLA learning: as the learner adjusts its grammar step by step, it may temporarily pass through intermediate states that yield suboptimal outputs.

This analysis helps clarify the differences in convergence speed across starting hierarchies. Although both Hierarchies 2 and 5 generate local optima, their behavior during learning diverges. Hierarchy 2 converges more slowly, despite being structurally closer to the target (Hierarchy 1), because it often produces the same global optimum ([SN V]), triggering no updates and stalling progress. In contrast, Hierarchy 5 consistently produces divergent forms ([V SN] or [SN V SN]), which elicit corrections at each step, accelerating convergence. A similar dynamic explains why Hierarchy 6, though equally distant from Hierarchy 1 as Hierarchy 2, converges faster: its single global optimum ([V SN]) reliably differs from the target, ensuring regular updates.

Finally, the stark difference between Hierarchies 3 and 4–both representing the discontinuous stage–can be accounted for in similar terms. Hierarchy 3 converges slowly, likely because it often transitions through Hierarchy 2, where learning may stall due to shared optima with the target. Hierarchy 4, in contrast, tends to pass through Hierarchies 5 or 6–or occasionally transitions directly to H1. These pathways produce more consistent learning signals through persistent mismatches. In sum, these findings suggest that convergence speed depends not only on structural proximity to the target grammar, but also on the availability and persistence of corrective feedback during learning.

Finally, since the agent-based simulation tracks production patterns (linguistic performance) at each generation rather than the grammars (linguistic competence) used to produce them, a final analysis was conducted to investigate how the number of learning iterations affects the surface forms produced. This experiment was carried out in the same minimal setup used for the convergence test, with agents initialized using Hierarchies 2–6 and learning from a target

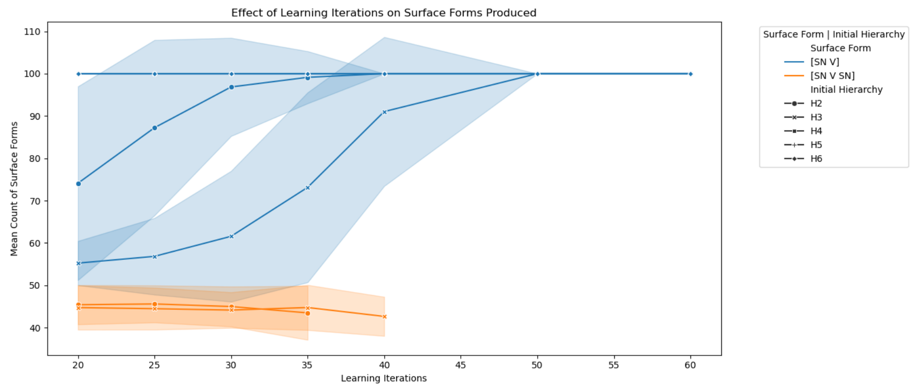


Figure 6: Caption

grammar corresponding to Hierarchy 1. The goal was to examine whether the quantity of learning input (in terms of iteration count) influences the degree to which learners reproduce the target surface form ([SN V]) versus produce locally optimal or residual forms such as [SN V SN]. The results are shown in Figure 6.

The plot reveals a clear pattern: as the number of learning iterations increases, the production of the target form [SN V] increases across all initial hierarchies. The effect is especially pronounced for learners initialized with Hierarchies 3 and 5, which begin with high proportions of discontinuous forms but shift decisively toward the target with sufficient learning iterations. In contrast, Hierarchy 6 starts with a high rate of [SN V] productions and exhibits minimal change–reflecting its structural closeness but output difference to the target.

Overall, the results demonstrate that allowing 50 or more learning iterations leads to exclusive production of preverbal forms, and further justifies the use of around $25 - 30$ iterations to allow some imperfect learning to occur.

3.3.3 Analytical Approach to Learning Data Parameter

The 10-fold discrepancy in the learning data parameter between the model of Lopopolo and Biró (2011) and my replications is quite puzzling. Adopting an analytical approach, it can be easily computed that in the worst-case scenario, where a constraint has to be updated from the lowest to the highest possible rank value, that is, from -0.1 to 4.9, exactly $\frac{4.9-(-0.1)}{0.1} = 50$ update steps are required, as confirmed with the last experiment in the previous subsection. Hence, if 300 learning examples are provided, then at least 251 of them would have to have the production of the learner to match the target if even the minimal form of imperfect learning were to occur in this worst-case scenario in terms of grammar updates needed.

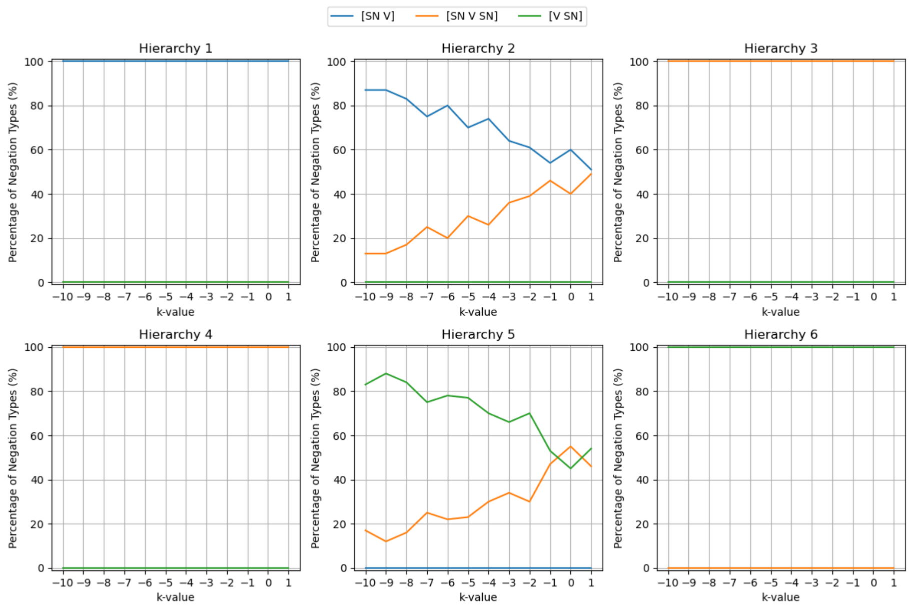


Figure 7: Reducing the rank (K-value) of the lowest constraint leads to a higher frequency of the global optimum in the performance output of the mixed-stage hierarchies, i.e., Hierarchies 2 and 5.

4 New Insights on Model Limitations

4.1 Sensitivity to Parameter Initialization

Having gained a better understanding of the internal dynamics of the model, I sought to improve the model so that it accounts for the entire trajectory of JC. Given the impact of the K-value and the learning data parameters, a promising avenue for exploration seemed to be parameter tuning, as also suggested by Lopopolo and Biró (2011).

One of the analyses conducted in Lopopolo and Biró (2011, Section 5.2) revealed that decreasing the rank value of the lowest constraint in a mixed-stage hierarchy increases the number of global optima produced by the SA-OT algorithm. For instance, decreasing the K-value of NegLast in Hierarchy 2 from 1 down to -10, with all other constraints preserving their default integer values from 4 to 2, increases the frequency of global optima produced from just over 50% up to nearly 100%. This result was confirmed by my replication as shown in Figure 7.

One strategy to try to manipulate the course of linguistic change in the simulation would thus be to test different K-value initializations. K-values play a crucial role in the SA-OT algorithm by determining the constraint strata where "errors" are tolerated, especially early in the annealing process when the system is more "exploratory." The spacing between K-values–that is, the difference in numerical magnitude between successive constraint ranks–directly influences how frequently the algorithm permits performance errors (i.e., suboptimal candidates) during its execution. Smaller intervals allow less temporal room for the algorithm to escape local optima. It quickly becomes more deterministic. Conversely, wider intervals prolong the stochastic search phase, giving the random walker more time to explore the space and escape suboptimal local optima before convergence. As shown in the results of Lopopolo and Biró (2011), expanding the rank gap between two lower constraints (e.g., between *Neg and NegLast in Hierarchy 2) significantly increases the production of the grammatical form by reducing the "stickiness" of a competing local optimum.

Following this reasoning, we would expect that scaling the K-values logarithmically would yield faster linguistic change due to more frequent performance errors, making suboptimal forms more prominent in learning data for the next generations. Conversely, exponentiating the K-values would be expected to have the opposite effect. However, if any of these scaling strategies were to be applied to the K-values in the model as is, the expected results may not be observed due to the way in which the SA-OT algorithm operates. Specifically, the criteria for accepting or rejecting candidate states are based on a direct comparison between the K-value of the fatal constraint (the highest-ranked constraint for which the violation score of the current and candidate surface form differ), and the value of a temperature parameter following a cooling schedule. Hence, modifying the scale of the constraint ranks would need to be harmonized with the decision logic of the SA-OT algorithm. While such an analysis may be fruitful, another fundamental observation discouraged me from pursuing it.

4.2 Asymmetry in SA-OT Representations of the Pure Stages of Jespersen's Cycle

Lopopolo and Biró (2011) mention and my replications confirm that the discontinuous stage acts as a stable attractor: regardless of whether the initial generation is set to the preverbal or postverbal stage, the population eventually converges on discontinuous negation through a transitional mixed stage. In contrast, transitions between the preverbal and postverbal stages, bypassing the discontinuous stage, do not occur. The authors liken this pattern to a pendulum: the outer stages (preverbal and postverbal) are unstable, while the middle stage (discontinuous) exerts a stabilizing force.

In light of the supplementary analyses presented in this report, I hypothesize an explanation of this attractor behavior as a direct consequence of how the SA-OT model formalizes grammatical competence and performance. In this framework, production is modeled as a stochastic search (via simulated annealing) over a space of candidates. While global optima represent grammatical forms under a given constraint hierarchy, local optima–less optimal candidates that are more harmonic than their neighbors–may also be produced due to the probabilistic nature of the algorithm. These local optima are crucial: they enable transitions between grammars over generations by introducing systematic variation into the learner's input.

Discontinuous negation is uniquely favored in this setup. It is the only stage whose characteristic forms ([[SN V] SN] and [SN [V SN]], which can both be simplified to [SN V SN]) arise as local optima under multiple grammars (e.g., Hierarchies 2 and 5). This means that discontinuous forms can appear even when the learner's grammar does not rank them as globally optimal. In contrast, preverbal ([SN V]) and postverbal ([V SN]) forms only emerge as global optima in Hierarchies 1 and 6, respectively, and do not appear as local optima elsewhere in the landscape. As a result, there are no natural stochastic pathways into the preverbal or postverbal stages through production errors. Discontinuous forms, however, can be accessed from various starting points via local optima and thus serve as frequent, learnable targets.

This is complemented by a deeper formal asymmetry in the model. Each of the preverbal and postverbal stages is represented by a single constraint hierarchy, while the discontinuous stage is supported by two (Hierarchies 3 and 4). This duplication in representational space, coupled with the availability of local optima, increases both the likelihood and persistence of discontinuous forms in the population. Consequently, the SA-OT formalization structurally biases the model toward discontinuous negation, regardless of initial conditions. These properties lead me to cast doubt on the model's capacity to simulate the full trajectory of JC.

Given the elaborate Optimality Theory-based formalization behind this model (stretching back to Swart, 2010), it is not evident how this design flaw could be bypassed so that Jespersen's Cycle may be simulated in its entirety. The reliance on local optima to drive change structurally privileges the discontinuous stage, while the rigid mapping between constraint hierarchies and grammatical forms limits the system's flexibility to move across the full cycle. Any attempt to rebalance the landscape would likely require major revisions to the constraint set, the topology, or the learning algorithm itself–undermining the elegance and interpretability of the original SA-OT proposal.

5 Toward a Better Model

Rather than continuing to patch the SA-OT framework, I again propose a shift toward a Bayesian approach. This alternative dispenses with much of the machinery specific to Optimality Theory and instead builds on well-established probabilistic principles that have been widely adopted in recent research on human cognition and language evolution. In particular, I draw inspiration from Bayesian models of language evolution using iterated learning (e.g., Kirby, Dowman, & Griffiths, 2007; Kirby, Tamariz, Cornish, & Smith, 2015; Smith & Kirby, 2008).

In my previous project, I developed an initial Bayesian model aimed at capturing JC. However, this early attempt neither outperformed the SA-OT model nor offered a conceptually grounded account of the phenomenon. The model assumed that learners entertain the six SA-OT constraint hierarchies as discrete hypotheses and select among them using Bayesian inference. This assumption is difficult to justify in the context of Jespersen's Cycle, which is typically described as a gradual process involving the phonological erosion of an original negator and the subsequent introduction of an additional marker to reinforce the sentence's negative meaning. Modeling the cycle as inference over fixed hierarchies thus fails to capture the incremental and morphosyntactic nature of the change.

The Bayesian model I sketch here aims to simulate the diachronic trajectory of Jespersen's Cycle by modeling how learners infer and transmit grammatical preferences over generations. Like the SA-OT agent-based model, it assumes an iterated learning setup in which each generation of agents learns from the productions of the previous one. However, it replaces constraint hierarchies and local optima with probabilistic reasoning about grammatical options–namely, the placement of sentential negation, along with additional constraints related to compositionality (each form contributing new meaning) and noise handling (allowing a level of redundancy to ensure successful communication). While still preliminary, I believe this model offers a promising starting point for simulating language change in a way that is conceptually simple, computationally tractable, and cognitively grounded.

The grammar of an agent is represented by a combination of factors, reminiscent of the different constraints in the (SA-)OT model but assigned probabilities instead of rankings. Unlike my earlier Bayesian model, rather than entertaining the three possible types of negation or the SA-OT hierarchies, agents will consider different structural constraints, such as preverbal negation, postverbal negation, ambiguity minimization, redundancy to ensure communicative success in noisy environments, etc. Each of these is modeled through as a Binomial distribution (where success corresponds to the presence of the factor) and all factors are combined into a Dirichlet-multinomial model representing an agent's grammar. This grammar is then used to generate forms probabilistically. Within an iterated learning framework, the agents in the new generation use Bayesian inference with the productions of the previous generation to update their priors, the probability of each of the factors being present in generated utterance. A preminary implementation of the model can be found in bayesian.ipynb in the project repository.

References

Biró, T. (2010). *Otkit: Tools for optimality theory.* Retrieved 2025-01-17, from https://www.birot.hu/OTKit/ (Accessed: 2025-01-17)

Boersma, P. (1970, February). How We Learn Variation, Optionality, And Probability. *Proceedings of the Institute of Phonetic Sciences Amsterdam*, *21*.

Breitbarth, A., & Haegeman, L. (2010). Continuity is change. *Continuity and Change in Grammar. Benjamins, Amsterdam/Philadelphia*, 61–76.

Dahl, Ö. (1979, January). Typology of sentence negation. *Linguistics*, *17*(1-2), 79–106. Retrieved 2025-01-07, from https://www.degruyter.com/document/doi/10.1515/ling.1979.17.1-2.79/html (Publisher: De Gruyter Mouton Section: Linguistics) doi: 10.1515/ling.1979.17.1-2.79

Griffiths, T. L., Kalish, M. L., & Lewandowsky, S. (2008, September). Theoretical and empirical evidence for the impact of inductive biases on cultural evolution. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *363*(1509), 3503–3514. Retrieved 2025-01-23, from https://royalsocietypublishing.org/doi/full/10.1098/rstb.2008.0146 (Publisher: Royal Society) doi: 10.1098/rstb.2008.0146

Hoeven, E. t., Kwakkel, J., Hess, V., Pike, T., Wang, B., Rht, & Kazil, J. (2025, March). Mesa 3: Agent-based modeling with Python in 2025. *Journal of Open Source Software*, *10*(107), 7668. Retrieved 2025-07-02, from https://joss.theoj.org/papers/10.21105/joss.07668 doi: 10.21105/joss.07668

Jespersen, O. (1917). Negation in English and other languages. *AF Høst*. Retrieved from https://archive.org/details/cu31924026632947

Kirby, S., Dowman, M., & Griffiths, T. L. (2007). Innateness and culture in the evolution of language. *Proceedings of the National Academy of Sciences*, *104*(12), 5241–5245.

Kirby, S., Griffiths, T., & Smith, K. (2014). Iterated learning and the evolution of language. *Current opinion in neurobiology*, *28*, 108–114.

Kirby, S., & Hurford, J. R. (2002). The emergence of linguistic structure: An overview of the iterated learning model. *Simulating the evolution of language*, 121–147.

Kirby, S., Tamariz, M., Cornish, H., & Smith, K. (2015). Compression and communication in the cultural evolution of linguistic structure. *Cognition*, *141*, 87–102.

Labelle, M. (2019). The French Jespersen's Cycle and Negative Concord. In D. L. Arteaga (Ed.), *Contributions of Romance Languages to Current Linguistic Theory* (pp. 155–172). Cham: Springer International Publishing. Retrieved 2025-02-04, from https://doi.org/10.1007/978-3-030-11006-2_8 doi: 10.1007/978-3-030-11006-2_8

Lopopolo, A., & Biró, T. (2011, December). Language Change and SA-OT: The case of sentential negation. *Computational Linguistics in the Netherlands Journal*. Retrieved 2025-01-17, from https://rucore.libraries.rutgers.edu/rutgers-lib/42013/ doi: 10.7282/T3BC3WKH

Lucas, C. (2020, April). Contact and the expression of negation. *Arabic and contact-induced change*, 643–667. Retrieved 2025-01-15, from https://zenodo.org/records/3744559 (ISBN: 9783961102518 Place: Berlin Publisher: Language Science Press)

Smith, K., & Kirby, S. (2008). Cultural evolution: implications for understanding the human language faculty and its evolution. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *363*(1509), 3591–3603.

Swart, H. d. (2010). *Expression and Interpretation of Negation: An OT Typology*. Springer Science & Business Media.