A Modeling Study of the Effects of Surprisal and Entropy in Perceptual Decision Making of an Adaptive Agent

Pyeong Whan Cho
Department of Psychology
University of Michigan
Ann Arbor, MI 48109
pyeongwc@umich.edu

Richard L. Lewis
Department of Psychology
University of Michigan
Ann Arbor, MI 48109
rickl@umich.edu

Abstract

Processing difficulty in online language comprehension has been explained in terms of surprisal and entropy reduction. Although both hypotheses have been supported by experimental data, we do not fully understand their relative contributions on processing difficulty. To develop a better understanding, we propose a mechanistic model of perceptual decision making that interacts with a simulated task environment with temporal dynamics. The proposed model collects noisy bottom-up evidence over multiple timesteps, integrates it with its top-down expectation, and makes perceptual decisions, producing processing time data directly without relying on any linking hypothesis. Temporal dynamics in the task environment was determined by a simple finite-state grammar, which was designed to create the situations where the surprisal and entropy reduction hypotheses predict different patterns. After the model was trained to maximize rewards, the model developed an adaptive policy and both surprisal and entropy effects were observed especially in a measure reflecting earlier processing.

1 Introduction

Over the past decades, computational models of sentence comprehension have improved our understanding of processing difficulty arising in online language comprehension. It has been discovered that information-theoretic complexity metrics can predict processing difficulty (for review, see Hale, 2016).

The surprisal hypothesis (Hale, 2001; Levy, 2008) proposes processing difficulty of a word w_k in a context $w_{1:k-1}$ is proportional to its surprisal, $-\log p(w_k|w_{1:k-1})$. Levy (2008) proved that surprisal is equivalent to Kullback-Leibler divergence between the probability distributions over parse trees T before and after observing the word w_k, $D_{\mathrm{KL}}(P(T|w_{1:k})\|P(T|w_{1:k-1}))$.

On the other hand, the entropy reduction hypothesis (Hale, 2003) claims that processing difficulty is proportional to a non-negative amount of entropy reduced after observing a word w_k: $\max(H(S|w_{1:k-1}) - H(S|w_{1:k}), 0)$ where S is a random variable of sentences. It is not clear why the language processing system works insensitive to negative entropy changes.

Both hypotheses have been supported by experimental data (for surprisal, see Demberg and Keller, 2008; Smith and Levy, 2013; for entropy reduction, see Frank, 2013; Linzen and Jaeger, 2016). Some behavioral studies reported both effects of surprisal and entropy reduction (Linzen and Jaeger, 2016; Lowder et al., 2018) and in such cases, the surprisal effect was much stronger than the entropy reduction effect.

However, we do not have comprehensive understanding of their relative contribution to processing load. Empirically, the estimation of surprisal and entropy values requires a language model, the quality of which depends on many factors (e.g., the corpus size, the model type) (c.f., Goodkind and Bicknell, 2018 argued the effect of surprisal was robust when the measures were estimated using a wide range of language models with different qualities). Also surprisal and entropy values tend to be highly correlated in natural languages, which makes it difficult to tease apart their relative roles in online language processing.

To avoid these empirical problems, we introduce a simple experimental paradigm, which combines two well-established paradigms: saccade target selection (OReilly et al., 2013) and artificial language paradigm (Harrison et al., 2006), both of which have been used to answer related questions. In the artificial language paradigm, we design a language such that it has some distributional

Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, pages 53–61
Minneapolis, USA, June 7, 2019.

properties of interest. For example, we can design a grammar in which the surprisal and the entropy reduction hypotheses make different predictions. For example, Linzen and Jaeger (2014) used a simple finite-state grammar to create such situation and discussed alternative accounts of processing difficulty. In the present study, we used a variant of their grammar (see Figure 3). Due to the simplicity of the grammar, entropy and entropy reduction measures are perfectly correlated. When we discuss the effect of those measures, we will refer to it as the entropy effect but we are neutral in whether it should be interpreted as the effect of entropy or the effect of entropy reduction; we reserve the question for future work.

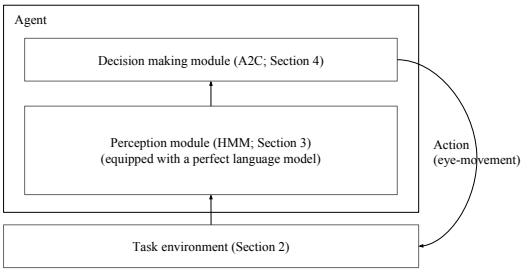


Figure 1: Model architecture. The model consists of two modules: perception module and decision making module. Equipped with a perfect language model, the perception module (implemented as a Hidden Markov Model) integrates noisy inputs from environment with its top-down expectation. The decision making module (implemented as a neural network with the Actor Critic architecture) makes an action based on the output of the perception module.

To develop a better understanding, we propose a mechanistic model of perceptual decision making and investigate its behavior in a simulated task environment with temporal dynamics, focusing on the effects of surprisal and/or entropy. Figure 1 presents the architecture of the model and how it interacts with the task environment. It consists of two components: the perception module at the bottom collects noisy bottom-up evidence from the task environment and updates its state (expressed in [posterior] probability distributions). The decision making module at the top monitors the state of the perception module and makes an action (i.e., decision), which will update the state of the task environment. The design of the perception module was inspired by Bicknell and Levy (2010) that investigated a related research question. Unlike their model, we used reinforcement learning to let the agent develop an optimal policy.

The main contribution of the present study is that we propose a full cognitive architecture that performs perceptual decision making, which we argue shares a core computational problem of uncertainty management with online language comprehension tasks (e.g., self-paced reading) and investigate the optimal behavior by exploring an unrestricted decision policy space.

In the following sections, we will present each component in Figure 1 in detail. In Discussion, we conclude.

2 Task Environment

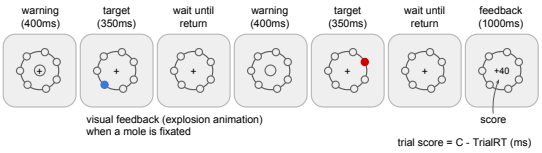


Figure 2: Task environment. The events occurring at two sample trials are shown. The agent is asked to "look at" the target (color dot) as quickly and accurately as possible.

We created a simulated saccade target selection task environment (e.g., OReilly et al., 2013) (see Figure 2). In each trial, a target appears at one of 7 positions and the agent is asked to "look at" the target as quickly and accurately as possible and look back at the center. The returning fixation terminates the trial and initiates the next trial.

In the simulated task environment, each of 7 locations was represented as an angle (in radian) in a circular space $[0, 2\pi)$ and associated with a symbol. The center location was associated with an empty symbol ϵ, representing the absence of a target. A selection of a symbol was treated as the fixation on its associated location.[1] Following OReilly et al. (2013), we measured the number of timesteps that the agent took to select the target (*target arrival*) and the number of timesteps that the agent took to make the first "meaningful" decision, by which we mean the first selection of a non-center location which may or may not be correct (*first saccade onset*).[2]

The locations of the targets changed following a simple finite-state grammar (see Figure 3 so

[1] For modeling convenience, we ignored eye-movement details (e.g., the minimal duration of a saccade).

[2] The proposed model selects a symbol at every timestep. When the model selects the symbol that it previously selected, we treat it as a continuation of the previous fixation.

Sample space	Description
$\mathcal{V} = \{a, b, c, d, e, f, g\}$	the set of input symbols
$\mathcal{U} = \{\epsilon\}$	the set of the empty symbol
$\mathcal{W} = [0, 2\pi)$	a circular space of angles
$\mathcal{S} = (\mathcal{V} \times \mathcal{U}) \cup (\mathcal{U} \times \mathcal{V})$	the set of states
$\mathcal{X} = \mathcal{V} \cup \mathcal{U}$	the set of input symbols
$\mathcal{Y} = \mathcal{W} \cup \mathcal{U}$	the set of observations

Table 1: Sample spaces

were partially predictable. We were interested in whether the onset and arrival measures are dependent on the amount of uncertainty.

3 Agent: Perception Module

For discussion, we introduce some notational conventions. We use the uppercase (e.g., X), lowercase (e.g., x), and calligraphic font (e.g., $\mathcal{X}$) to denote a random variable, a particular sample, and its sample space. We use the superscript to denote the position of a symbol in a sequence of symbols and the subscript to denote a particular element in a sample space.

Let S, X, and Y be a discrete random variable of states, a discrete random variable of input symbols, and a mixed random variable of observations. A probability distribution over states $P(S)$ will be referred to as a "parser state", which should be distinguished from simple states. The sample spaces of those variables are $\mathcal{S}$, $\mathcal{X}$, and $\mathcal{Y}$, respectively (see Table 1).

The perception module was implemented as a Hidden Markov Model (HMM), where the hidden variable $S^{(k)}$ represents states after processing the k-th symbol $x^{*(k)}$ in a sequence of symbols, assuming the agent is equipped with a perfect language model. $X^{(k)}$ representing symbol identities is conditioned on $S^{(k)}$. $Y^{(k)}$ represents the observations of the input symbol (i.e., a particular location in the task environment [see Figure 4]).

3.1 Language Model

Let us begin with the agent's language model. We used a Markov chain (See Figure 3) to implement a language model but other types of models can be used if they can emulate the environmental dynamics. For convenience, unique bigrams were used as states: $s^{(k)} = x^{(k)}x^{(k+1)}$. In this simple language, each state $s^{(k)}$ uniquely specifies the present input symbol $x^{(k)}$; $p(x_j|s_i) = 1$ if $s_i = x_j x_k$ and 0 otherwise. An example sequence of two sentences is $c\,\epsilon\ f\ \epsilon \mid a\ \epsilon\ d\ \epsilon$ where $\mid$ indicates a hidden sentence boundary; the underlying state

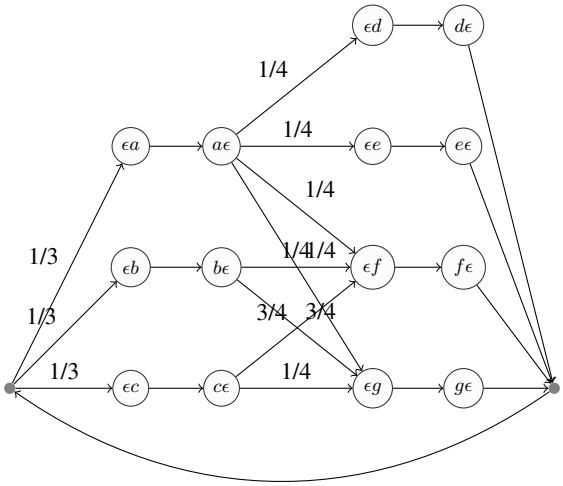


Figure 3: Grammar G specifies probabilistic transitions from a state s_i to another state s_j between symbols. The transition probability $p(s_j|s_i)$ is shown on an edge from node i to node j. The edges with no labels have the transition probability of 1.

change is as follows: $c\epsilon$-ϵf-$f\epsilon$-ϵa-$a\epsilon$-ϵd-$d\epsilon$-ϵb.

Some distributional information of the language is given in Table 2. In the present study, we will focus on three conditions where f is presented in different contexts $a\epsilon$, $b\epsilon$, and $c\epsilon$ at which surprisal and entropy reduction hypotheses predict different patterns.[3] We will refer to the three conditions as HiE/HiS (HighEntropy/HighSurprisal), LoE/HiS, and LoE/LoS. Let RT($\cdot$) be a decision making time (in onset or arrival) at a certain condition. The surprisal hypothesis predicts: RT(LoE/LoS) $<$ RT(LoE/HiS) $=$ RT(HiE/HiS). The entropy reduction hypothesis predicts: RT(LoE/LoS) $=$ RT(LoE/HiS) $<$ RT(HiE/HiS). If both surprisal and entropy (reduction) have unique contributions to processing load, assuming the surprisal effect is stronger than the entropy effect, we expect: RT(LoE/LoS) $<$ RT(LoE/HiS) $<$ RT(HiE/HiS).

3.2 Mapping between Symbols and Observations

Figure 4 presents the mapping of symbols to the locations on the task environment. The empty symbol, representing the absence of target, is mapped to the center location. The other symbols

[3]The target g in the same three contexts was designed to be a mirror case of f and introduced (1) for counterbalancing and (2) to increase the number of data points in the planned human experiment. However, due to the difference in their closest neighbors, processing f and g in the same three contexts can be different.

context	target	p_{targ}	surprisal	entropy
$a\epsilon$	d, e	0.083	1.386	1.386
$a\epsilon$	f, g	0.417	1.386	1.386
$b\epsilon$	f	0.417	1.386	0.562
$b\epsilon$	g	0.417	0.288	0.562
$c\epsilon$	f	0.417	0.288	0.562
$c\epsilon$	g	0.417	1.386	0.562

Table 2: Distributional information for unique context-target combinations. p_{targ} represents the unigram target probability. Entropy measures the amount of uncertainty after processing context but before receiving target: $H(P(X|\text{context})) = -\sum_x p(x|\text{context}) \log p(x|\text{context})$. Surprisal and entropy were calculated with e as base.

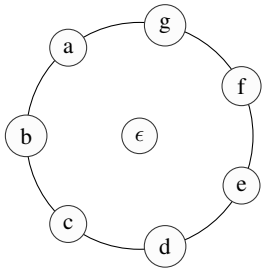


Figure 4: Screen configuration. Each of 7 symbols (a - g) is mapped to a unique location on the ring. At the center, there is a fixation cross, corresponding to a null symbol ϵ.

in $\mathcal{V}$ are mapped to 7 equally-distributed real numbers in $\mathcal{W}$ via a bijective function A; the image of $\mathcal{V}$ is $\{(2\pi)(i/7) + j | i \in \{0, \cdots, 6\}\}$ where j can take any arbitrary number in $[0, 2\pi)$. In the following example, we consider a simple mapping: $A(b) = 0$, $A(c) = (2\pi)(1/7)$, $A(d) = (2\pi)(2/7)$, $A(e) = (2\pi)(3/7)$, $A(f) = (2\pi)(4/7)$, $A(g) = (2\pi)(5/7)$, $A(a) = (2\pi)(6/7)$.

3.3 Noisy Input Channel

Let y^* be the noise-free observation of the target symbol x^*. Note that x^* is chosen by the task environment dynamics. We assume that the perception module samples an observation y via a noisy input channel at every timestep. The conditional probability of y given x, $p(y|x)$, is presented in Table 3. Observations y's over multiple timesteps are assumed to be independent from each other given target x^*.[4] We will use the same conditional probability distribution when the module updates the posterior probability of symbol x given noisy observation y. Parameters α and β are false positive and false negative rates, respectively. In the false positive case, we assume every value in the circular space $\mathcal{W}$ is equally likely. In the true positive case (i.e., $x^* \in \mathcal{V}$, we assume $p(y|x^*)$ is higher as y is closer to $y^* = A(x^*)$. This intuition is implemented by introducing a von Mises distribution (with parameters μ and κ), which is a Gaussian-like distribution applied to a circular space.

	sample a location on the ring $y \in \mathcal{W}$	sample the center $y \in \mathcal{U}$
target present ($x \in \mathcal{V}$)	$(1-\beta)F(y; \mu = A(x), \kappa)$	β
target absent ($x \in \mathcal{U}$)	$\alpha/(2\pi)$	$1-\alpha$

Table 3: $p(y|x)$, a conditional probability of a noisy sample y given a symbol x where α and β are the rates of false positive and false negative, F is the probability density function of the von Mises distribution with location and scale parameters μ and κ. As κ increases, larger probability mass is placed near the mean μ

3.4 Noisy Memory

In addition to noisy input, we consider noise in memory. More specifically, we assume the memory of the parser state is noisy such that a state s can be replaced with another state s'. Noisy memory is implemented by applying the confusion matrix to the parser state $P(S)$: $P(S') = P(S) \cdot \mathbf{P}'_{S \to S'}$ where $P(S)$ is a row vector of probabilities over possible states and $\mathbf{P}'_{S \to S'}$ is the transition probability matrix in which the (i, j)-th component represents $p(s'_j|s_i)$.

We consider three types of confusion: (1: rand) purely random noise which allows every transition $s_i \to s_j$ for all pairs of i and j, (2) similarity-based interference allows transitions between two states similar to each other. Two types of similarities were considered. (2a: sim1) symbol-type similarity; e.g., a, b, c are similar because they occur at the same position in a sequence (i.e., as the first word of a two-word sentence) so $a\epsilon$ can be recalled as $a\epsilon$, $b\epsilon$, or $c\epsilon$. (2b: sim2) transposition

[4] The likelihood of observation y is also conditioned on the present fixation location, which is modeled as the symbol chosen by the decision making module at the previous timestep (see Section 4). The likelihood function in the target present condition (see Table 3) assumes that the present fixation is on the center, which is true at the beginning of each target-present trial; the measure of *first saccade onset* is accurate. However, the likelihood function would not be ideal for modeling the belief update from noisy observations after the first saccade to a non-target location. Although not accurate, the measure of *target arrival* can still be informative because it contains information about whether the target was chosen at the first try or not.

+ symbol-type similarity; for example, $a\epsilon$ can be confused as ϵa, ϵb, and ϵc.

More specifically, we consider $p(s_j|s_i) = (1 - \eta_{noise})\delta_{ij} + \eta_{noise}\{\eta_{rand}\, p_{rand}(s_j|s_i) + (1 - \eta_{rand})((1 - \eta_{trans})\, p_{sim1}(s_j|s_i) + \eta_{trans} p_{sim2}(s_j|s_i))\}$; $p_{type}(s_j|s_i)$ (where $type \in$ {rand, sim1, sim2}) was set to the reciprocal of the number of transitions corresponding to the type of confusion if $s_i \rightarrow s_j$ is allowed and 0 otherwise. We aggregate the conditional probabilities into a transition probability matrix $\mathbf{P}'_{S\rightarrow S'}$ such that $p_{i,j} = p(s_j|s_i)$. In the present study, $\eta_{\text{noise}} = 0.001, \eta_{\text{rand}} = 0.1, \eta_{\text{trans}} = 0.1$.

3.5 Belief Update

The module updates posterior probabilities of target locations over multiple timesteps by accumulating bottom-up noisy evidence (likelihood) and integrating it with top-down expectation (prior probabilities) $p(x^{(k)}|s^{(k-1,T_{k-1})})$ where T_k is the last timestep at the previous trial $k-1$. More detailed processes are presented in the below.

Step 1: Each trial begins with the instantaneous update of input symbol from $x^{*(k-1)}$ to $x^{*(k)}$. The model uses the last parser state $P(S^{(k-1,T_{k-1})})$ to set log priors for $X^{(k)}$ and $S^{(k)}$. $\text{LP}_\text{S}(k) = \log\{P(S^{(k-1,T_{k-1})}) \cdot \mathbf{P}'_{S\rightarrow S'} \cdot \mathbf{P}_{S\rightarrow S}\}$ ($\mathbf{P}'_{S\rightarrow S'}$ adds noise to the past parser state and $\mathbf{P}_{S\rightarrow S}$ [from the language model] uses the noisy past parser state to predict the following parser state); $\text{LP}_\text{X}(k) = \log\{P(S_k) \cdot \mathbf{P}_{S\rightarrow X}\}$.

Step 2: At every timestep t, the module collects a noisy observation $y^{(k,t)}$ and updates log-likelihoods of $X^{(k)}$ and $S^{(k)}$: the i-th component of a row vector $\text{LL}_\text{X}(k,t)$ is $\sum_{t'=1}^{t} \log p(y^{(k,t')}|x_i)$; $\text{LL}_\text{S}(k,t) = \mathbf{P}_{S\rightarrow X} \exp\{\text{LL}_\text{X}(k,t)\}^\top$.

Step 3. Posteriors of X_k and S_k given $y_k^{(1:t)}$ are as follows: $P(X^{(k)}|y^{(k,1:t)}) = \sigma(\text{LL}_\text{X}(k,t) + \text{LP}_\text{X}(k))$; $P(S_k|y^{(k,1:t)}) = \sigma(\text{LL}_\text{S}(k,t) + \text{LP}_\text{S}(k))$ where σ is the standard softmax function.

Step 2 and Step 3 are iterated until (1) the decision making module (see the next section) selects the target symbol x^* correctly or (2) the maximum number of timesteps (= 100) has passed.

3.6 The Module Behavior

We created multiple instances of the perception module by setting some module parameters to different values (see Table 4) and investigated how the posterior probabilities changed in the three conditions of our interest. Each of 3 modules processed 200 blocks of 8 different sentence types. In each block, the presentation order of the sentences was randomized. For each sentence, the model processed each symbol over 50 timesteps.

Module	memory noise η_{noise}	perception noise ($1/\kappa$)
M1	0.001	1
M2	0.001	1/3
M3	0.2	1

Table 4: Different module settings. η_{noise} determines the amount of memory noise while $1/\kappa$ determines the amount of input noise. We fixed α (false negative rate) and β (false positive rate) to 0.05 in this study.

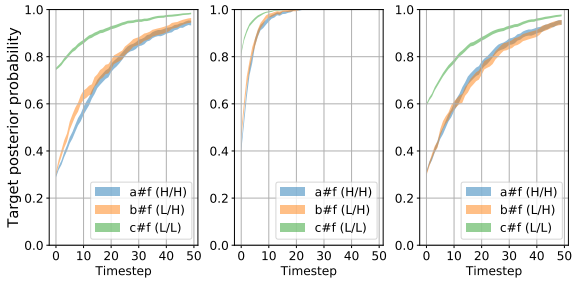


Figure 5: Plots of target posterior probabilities in different conditions. Each ribbon presents mean ± one standard error calculated from 200 trials.

Figure 5 presents the target posterior probability change as the modules processed f in three different contexts $a\epsilon$ (HiE/HiS), $b\epsilon$ (LoE/HiS), and $c\epsilon$ (LoE/LoS). The effect of surprisal is clear in all three modules. This is expected from our belief update process. When a new symbol (i.e., f) is presented, the perception module uses the last parser state to reset log priors, which determine different starting points before evidence integration. When the race begins, the symbol candidate with a low surprisal value is many steps ahead of its competitors with high surprisal values.

On the other hand, the effect of entropy (reduction) was weakly suggested only in Module 1 (see panel A in Figure 5). The target posterior probability increased slightly faster in context $b\epsilon$ (LoE/HiS) than in context $c\epsilon$ (HiE/HiS).

Based on the observed patterns, we chose Module1 as the perception module of the agent.

4 Agent: Decision Making Module

Instead of searching a restricted policy space (e.g., static decision boundary such as $\max_x p(x) > .9$, or as in Bicknell and Levy, 2010), we use reinforcement learning to search a huge policy space

with no restriction to discover a (near-)optimal decision policy in the task environment.

4.1 Advantageous Actor-Critic

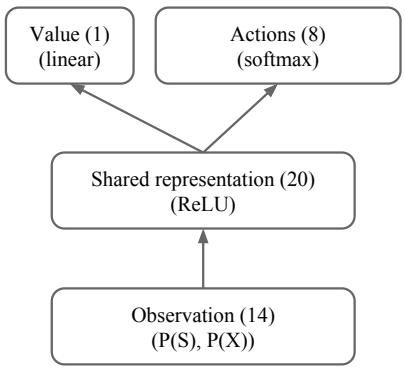


Figure 6: Actor-critic architecture of the decision making module. The number in parentheses indicates the number of units in each layer.

The decision making module has an Advantageous Actor-Critic (A2C) architecture (c.f., for the asynchronous version A3C, see Mnih et al., 2016) (see Figure 6) in which each of 8 actions was mapped to a unique location in the task environment. Let $\mathbf{s}_t$ be the state of the perception module at timestep t. Let $V(\mathbf{s}_t)$ and $\pi(a_i|\mathbf{s}_t)$ be the value output and the probability of choosing an action a_i given input $\mathbf{s}_t$. For the input, an action a_t is sampled from the action probability distribution. The *advantage* of the action is defined as follows:

$$\text{Adv}(\mathbf{s}_t, a_t; \theta) = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V(\mathbf{s}_{t+k}; \theta) - V(\mathbf{s}_t; \theta)$$

where $\gamma (= 0.99)$ is the discount factor for future rewards, r_t is the acquired reward at timestep t by making an action a_t, and θ is the vector of the model parameters. The module makes actions under the current policy over $k (= 5)$ steps and uses the rewards collected over k steps to improve the value estimate.

4.2 Reward in the Task

We constructed 4 instances of the task environment in which the perception module (Module1, see Table 4) was exposed to different sequences of symbols (that were generated by the same grammar). The decision making module interacted with all four environments simultaneously to collect tuples (state, action, reward, next state). This is motivated to collect relatively independent training samples. At every step, the perception module collects a new observation and updates its posterior probabilities over symbols and over states. The decision making module takes both distributions as input and outputs its value and an action sampled from the action probability distribution. When the action chosen at timestep $t (\leq 100)$ corresponds to the target symbol, it terminates the present trial and the new target symbol is presented in the task environment. In this case, the module receives a reward $(100 - t)/100$; faster responses are rewarded more than slower responses. If the module selects a non-target symbol (which is different from its previous selection), the model receives a penalty (= -1). If the model selects the same wrong symbol as in the previous timestep (i.e., $a_t = a_{t-1}$), the model is not penalized; the reward is 0 in this case. For example, let us suppose the decision making module made a sequence of choices $\epsilon, \epsilon, a, a, \epsilon, b, \epsilon, c$ when the target symbol was c, assuming the previous trial ended at the selection of the previous target ϵ. Then, the module would receive a sequence of rewards $0, 0, -1, 0, -1, 0, (100 - 8)/100$. If the model fails to choose the target symbol for 100 timesteps, the task environment is updated to present a new target symbol. Thus, the decision making module has an option not to select any new symbol; technically, the model can keep choosing the previous target symbol over 100 timesteps. This suboptimal policy is better than choosing a non-target symbol; while the maximum reward per trial is 0.99 (if the model chooses the correct target at the first timestep after the task environment update), the model is given -1 for a single wrong selection.

4.3 Training

Over the course of training, the model parameters are updated to minimize the following loss function:

$$\begin{aligned} L(\mathbf{s}_t, a_t; \theta) = &- \log \pi(a_t|\mathbf{s}_t, \theta)\text{Adv}(\mathbf{s}_t, a_t; \theta) \\ &- \lambda_H H(\mathbf{s}_t) \\ &+ \lambda_C \text{Adv}(\mathbf{s}_t, a_t; \theta)^2 \end{aligned}$$

where $H(\mathbf{s}_t)$ is the entropy of action probabilities $\pi(a|\mathbf{s}_t)$. Hyperparameter λ_H determines the strength of entropy regularization, which is intended to encourage the module to explore the policy space without converging to a suboptimal policy too early. In our case, the model developed suboptimal policies when λ_H was fixed at a small

value from the beginning; the model never chose target symbols d and e that have lower unigram frequencies than f and g. When d and e were presented in context $a\epsilon$, the module waited until the trial ended after the deadline (100 timesteps) without choosing any non-center location.

We used the ADAM optimizer (Kingma and Ba, 2014) (learning rate = 0.0003) to update the decision making module's parameters. The coefficient of value prediction cost (λ_C) was fixed at 0.5 but the coefficient of entropy regularization (λ_H) started at 0.01 and reduced to 0.001 after $400,000$ timesteps and 0.0001 after $1,000,000$ timesteps. We stopped training after $1,200,000$ timesteps after observing the performance did not improve. Figure 7 presents the average reward acquired on a randomly generated grammatical sequence of 10 symbols during test.[5]

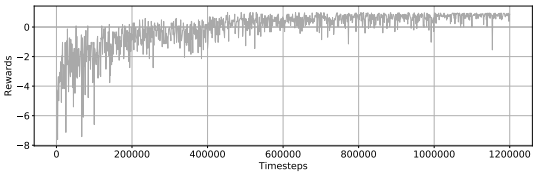


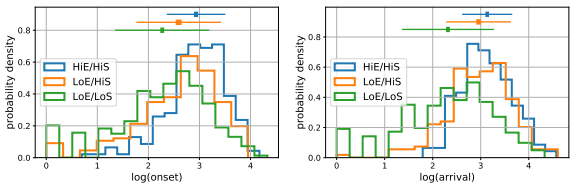
Figure 7: Trajectory of average reward acquired during model evaluation.

4.4 The Model Behaviors

The model consisted of the perception module (Module 1) and the trained decision making module. It was given a long sequence of symbols, a concatenation of 200 blocks of 12 sentences (of 8 sentence types); $b\epsilon g\epsilon$ and $c\epsilon f\epsilon$ were three times more frequent than other sentence types (see Figure 3). We focus on the model's behaviors when f was presented in three different contexts $a\epsilon$ (HiE/HiS), $b\epsilon$ (LoE/HiS), and $c\epsilon$ (LoE/LoS).

Figure 8 presents the distributions of log(onset) and log(arrival) as well as their means, standard errors (thick lines), and standard deviations (thin lines), suggesting the effects of both surprisal and entropy. The entropy effect was more salient in log(onset), which reflects the perception module's states earlier in processing.

[5]We trained three instances of the model with different random seeds. Their behaviors were not identical but similar. In the text, we report the behavior of the best model that achieved the highest reward over 2400 four-symbol sentences because we are interested in the optimal agent's behavior. When the trials with a trivial target ϵ were excluded, the best model achieved average reward of 0.591. Other two models acquired average rewards of 0.566 and 0.485.



(a) log(onset) (b) log(arrival)

Figure 8: Histograms of (a) log onset time and (b) log arrival time in timesteps. The mean ± one standard error (thick line) or one standard deviation (thin line) in each condition was presented at the top.

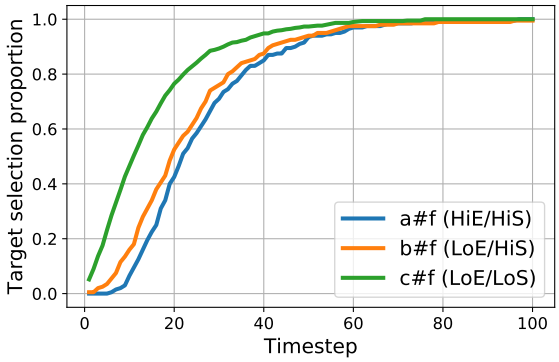
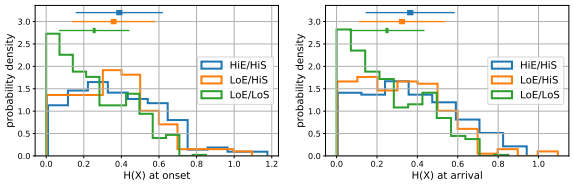


Figure 9: Timecourses of target selection proportions.

Figure 9 presents the proportion of target selection as a function of timesteps and contexts.[6] Both surprisal and entropy effects are clear.



(a) $H(X)$ at onset (b) $H(X)$ at arrival

Figure 10: Histograms of entropy values of posterior probability distributions at (a) onset and (b) arrival. The mean ± one standard error (thick line) or one standard deviation (thin line) is presented for each condition at the top.

Figure 10 presents the distribution of the entropy values of X in the perception module when the decision making module chose the first non-center location (onset) and the target location (ar-

[6]Different trials ended at different timesteps, typically much earlier than the maximum timesteps (= 100). For the purpose of calculating the proportion, we extended the final choice to the maximum timestep; for example, if the last action (i.e., selection of symbol f) was made at timestep 30 in a trial, we treated the module chose the same symbol for the next 70 timesteps.

rival). Distributions in the conditions HiE/HiS and LoE/HiS are largely overlapped but can be distinguished. Note that in all three conditions, the ideal target posterior probability is 1 and the entropy is 0. However, the decision making module made decisions before the perception module developed its belief on the target to the ideal level. This was true especially in HiE/HiS and LoE/HiS conditions. This is because the target posterior probability increased slowly either because the target has many competitors (HiE/HiS) or because the target has a very strong competitor (LoE/HiS). Instead of waiting until the target posterior probability increased enough, the module seemed to take a more risky approach (i.e., making a choice in a more uncertain situation) to obtain more rewards. It makes sense that the model took a safer approach for the target f in the LoS context $c\epsilon$ given that symbol f in the LoS context was three times more frequent than f in each HiS context. Developing a risky policy for such case will be harmful.

5 Discussion

In this study, we introduced a simple task that combines the saccade target selection task (e.g., OReilly et al., 2013) with the artificial language paradigm (e.g., Harrison et al., 2006), both of which have been used to investigate how the human cognitive system deals with uncertainty. Inspired by Linzen and Jaeger (2014), we designed a simple artificial language in which the surprisal hypothesis and the entropy reduction hypothesis predict different patterns. When a perceptual decision making model was trained to maximize rewards in the simulated task environment, both surprisal and entropy effects were observed in the model's behavior; consistent with the literature (Linzen and Jaeger, 2016; Lowder et al., 2018), the surprisal effect was stronger than the entropy effect.

The model developed a flexible decision policy such that it made more risky decisions in the HiE/HiS and LoE/HiS conditions than in the LoE/LoS condition. It was interpreted as the model pursuing a good balance between speed and accuracy because the model could obtain higher rewards from faster responses. The investigation of decision policy reveals the adaptive nature of the system which is not clear from pure rational models.

Our modeling study was intended to explore design-related issues and predict results in human eye-tracking experiments that we plan to run. In human experiments, participants need to learn the grammar hidden in a sequence of symbols. To make learning easier, we chose a simple grammar which made it hard to interpret the effect of entropy; it could be the effect of entropy or the effect of entropy reduction. However, the proposed model is general enough to cover more complex grammars and diverse situations (e.g., self-paced reading). We chose the Hidden Markov Model and the A2C architecture for the perception the decision making modules mainly for modeling convenience. The HMM can be replaced with a more elaborated neural language model when dealing with more complex grammars. The emphasis should be given to our architectual choice. The addition of the decision making module that has the ability to develop a policy on its own provides the system to control the amount of uncertainty flexibly in response to the task situations.

Bicknell and Levy (2010) took the same approach similar to explain reading eye movement patterns, which influenced our work. Our work is different from theirs in that (1) we considered noisy memory more directly and (2) we used reinforcement learning to let the model discover a good decision policy; we believe both additions can lead us to interesting research questions.

References

Klinton Bicknell and Roger Levy. 2010. A rational model of eye movement control in reading. In *Proceedings of the 48th annual meeting of the Association for Computational Linguistics*, pages 1168–1178. Association for Computational Linguistics.

Vera Demberg and Frank Keller. 2008. Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. *Cognition*, 109(2):193–210.

Stefan L. Frank. 2013. Uncertainty reduction as a measure of cognitive load in sentence comprehension. *Topics in Cognitive Science*, 5(3):475–494.

Adam Goodkind and Klinton Bicknell. 2018. Predictive power of word surprisal for reading times is a linear function of language model quality. *Proceedings of the 8th Workshop on Cognitive Modeling and Computational Linguistics (CMCL 2018)*, pages 10–18.

John Hale. 2001. A probabilistic Earley parser as a psycholinguistic model. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics on*

Language Technologies, NAACL '01, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

John Hale. 2003. The information conveyed by words in sentences. *Journal of Psycholinguistic Research*, 32(2):101–123.

John Hale. 2016. Information-theoretical complexity metrics. *Language and Linguistics Compass*, 10(9):397–412.

L. M. Harrison, A. Duggins, and K. J. Friston. 2006. Encoding uncertainty in the hippocampus. *Neural Networks*, 19(5):535–546.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980 [cs]*. ArXiv: 1412.6980.

Roger Levy. 2008. Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–1177.

Tal Linzen and T. Florian Jaeger. 2014. Investigating the role of entropy in sentence processing. In *Proceedings of the Fifth Workshop on Cognitive Modeling and Computational Linguistics*, pages 10–18.

Tal Linzen and T. Florian Jaeger. 2016. Uncertainty and expectation in sentence processing: Evidence from subcategorization distributions. *Cognitive Science*, 40(6):1382–1411.

Matthew W. Lowder, Wonil Choi, Fernanda Ferreira, and John M. Henderson. 2018. Lexical predictability during natural reading: Effects of surprisal and entropy reduction. *Cognitive Science*, 42(S4):1166–1183.

Volodymyr Mnih, Adri Puigdomnech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. *arXiv:1602.01783 [cs]*. ArXiv: 1602.01783.

Jill X. OReilly, Urs Schffelgen, Steven F. Cuell, Timothy E. J. Behrens, Rogier B. Mars, and Matthew F. S. Rushworth. 2013. Dissociable effects of surprise and model update in parietal and anterior cingulate cortex. *Proceedings of the National Academy of Sciences*, 110(38):E3660–E3669.

Nathaniel J. Smith and Roger Levy. 2013. The effect of word predictability on reading time is logarithmic. *Cognition*, 128(3):302–319.