Probabilistic semantic automata in the verification of quantified statements

Jakub Dotlačil (j.dotlacil@gmail.com)
Center for Language and Cognition, University of Groningen

Jakub Szymanik (jakub.szymanik@gmail.com)
Institute for Logic, Language and Computation, University of Amsterdam

Marcin Zajenkowski (zajenkowski@psych.uw.edu.pl)
Faculty of Psychology, University of Warsaw

Abstract

Strategies used by people to verify quantified sentences, like ‘Most cars are white’, have been a popular research topic on the intersection of linguistics, computer science, philosophy, and psychology. A prominent computational model of the task, semantic automata, has been introduced by van Benthem in 1983. In this paper we present a probabilistic extension of the model. We show that the model explains counting errors in the verification process. Furthermore, we observe that the variation in quantifier verification data cannot be explained by Approximate Number Sense, a prominent approach to probabilistic number estimation.

Keywords: quantifier verification; natural language semantics; automata theory; probabilistic computational modeling; Approximate Number Sense

Introduction.

Subjects’ verification strategies used in rejecting/accepting sentences have been a popular research topic in psycholinguistics (see, e.g. Clark and Chase, 1972). Together with the turn to more linguistically-complex phenomena the topic has also received an increased interest in linguistics, semantics, logic, and computer science. Especially the computational and cognitive capacities of recognizing the truth-value of sentences with so-called generalized quantifiers (like ‘some’, ‘an even number of’, ‘more than 7’, ‘less than half’ (Peters and Westerståhl, 2006)) has been intensively studied (see, e.g. Szymanik, 2009; Lidz et al., 2011).

A prominent computational model for verification of quantifiers employs semantic automata (van Benthem, 1986). Intuitively, to check whether sentence (1) is true:

1. Every sentence in this paper is grammatically correct.

it suffices to read the sentences from this article one by one. If we find an incorrect one, then we know that the statement is false. Otherwise, if we read the entire paper without finding any incorrect sentence, then statement (1) is true (see Fig. 1 for a graphical representation). Analogous strategies exist for all other natural language quantifiers.

However, for recognizing some higher-order quantifiers, like “less than half” or “most”, we need computational models making use of internal memory. Intuitively, to check whether sentence (2) is true we must identify the number of correct sentences and store it in working memory to compare with the number of incorrect sentences.

2. Most of the sentences are grammatically correct.

Mathematically speaking, such an algorithm can be realized by a push-down automaton, PDA, see Fig. 2. PDAs can not only read the input and move to the next state, they also have access to the stack memory and depending on the top element of the stack they decide what to do next. Graphically, we represent it by the following labeling of each transition: $s_1,x,y \rightarrow s_2,w$, where $s_1$ is the current state, $x$ is the current input the machine reads (i.e. the element under consideration), $y$ is the top element of the stack, and $s_2$ is the final state and $w$ shows what element is put on the top of the stack next (when the element is added to the previous top element, $w$ is of length 2 and shows both the previous element and the new element) (Hopcroft et al., 2000).

It has been shown that the computational distinction between quantifiers recognized by finite-automata and push-down automata is psychologically relevant, i.e., the more complex the automaton, the longer the reaction time and working memory involvement of subjects asked to solve the verification task (see Szymanik and Zajenkowski, 2010a,b). McMillan et al. (2005), in an fMRI study, have shown that during verification, all sentences recruit the right inferior parietal cortex associated with numerosity, but only proportional quantifiers recruit the prefrontal cortex, which is associated with executive resources, such as working memory. Zajenkowski et al. (2011) have compared the processing of natural language quantifiers in a group of patients with schizophrenia and a healthy control group. In both groups, the difficulty of the quantifiers was consistent with the computational predictions, and patients with schizophrenia took
s_0, \text{correct}, \# \rightarrow s_0, Y
s_0, \text{incorrect}, \# \rightarrow s_0, N
s_0, \text{correct}, N \rightarrow s_0, \varepsilon
s_0, \text{incorrect}, Y \rightarrow s_0, \varepsilon
s_0, \text{correct}, Y \rightarrow s_0, YY
s_0, \text{incorrect}, N \rightarrow s_0, NN

Figure 2: This push-down automaton for statement (2) reads the text sentence by sentence. The automaton needs two states and the stack. It starts in the rejecting state, s_0 with an empty stack marked by #. If it finds a correct sentence it pushes Y on top of the stack and stays in s_0, if it finds an incorrect sentence it pushes N on top of the stack. If it finds a correct (incorrect) sentence and there is already N (Y) on the top of the stack, the automaton pops out the top of the stack (by turning it into the empty string \varepsilon). If it ‘sees’ a correct (incorrect) sentence and there is Y (N) on the top of the stack, then the automaton pushes another Y (N) on the top of the stack. Eventually, when the automaton has analyzed the whole paper (input = \varepsilon) then it looks on the top of the stack. If there is a Y it moves to the accepting state, otherwise it stays in the rejecting state without modifying the stack. For simplification, we omit the situations: s_0, \varepsilon, N \rightarrow s_0, N, s_0, \varepsilon, # \rightarrow s_0, #.

more time to solve the problems. However, they were significantly less accurate only with proportional quantifiers, such as ‘more than half’.

All this evidence speaks in favor of the thesis that the model employing two types of automata can capture some cognitive aspect of the semantics for generalized quantifiers. However, the model, while important, is crude and only qualitative. It distinguishes between quantifier types but it does not offer a cognitive computational story of quantifier processing. For instance, the semantic automata cannot tell us why the verification of the sentence More than half of the cars are blue leads to more problems when there are 8 blue cars and 7 cars of other colors than when there are 9 blue cars and 6 other cars, as we will show below. The model also leaves the relation between semantics of quantifiers and number sense (Dehaene, 1999) completely unspecified. We show that such problems can be alleviated if one considers probabilistic, rather than deterministic automata as a model of verification strategies. As a result we can directly compare semantic automata model with the quantifier verification model relying on Approximate Number System (ANS, Pietroski et al., 2009). We believe that introducing the probabilistic version of the model is a first step towards a computational cognitive model of quantifier processing.

Probabilistic semantic automata.

Probabilistic finite-state automata (PFSA) can be used to model the verification of counting quantifiers (like, ‘more than k’, ‘less than k’). PFSA are tuples \langle S, \Sigma, s_0, F, M, \text{Prob} \rangle, where S = \{s_0, \ldots, s_n \} is the finite set of states, \Sigma the input alphabet, s_0 is the starting state, F is the set of final states, M is a transition relation \( (\Sigma \times \Sigma) \times S \) and \text{Prob} assigns a probability to each element of M, such that for every \( (s_j, a) \in \Sigma \times \Sigma \), \( \sum_{a \in M(s_j, a)} \text{Prob}(s_j, a, s_i) = 1 \) (cf. Rabin, 1963).

As an input PFSA take a string encoding the finite situation (model). They are to decide whether a given quantifier sentence, \( Q(A, B) \), is true in the model.

An example of a PFSA used for the verification of more than one car is blue is shown in Figure 3, where s_0 is the initial state and s_2 is the final accepting state, each transition indicates what happens when a blue car is encountered, and the superscript on a transition indicates where the transition originated. Thus, for instance, \( p_0^0 \) all start at s_0 and, given the conditions on probabilities discussed above, exhaust the events in one probability space (probabilities assigned to them sum up to 1). They are abbreviations. \( p_2^1 \), for example, is an abbreviation of \( (s_0, \text{BLUE CAR}, \text{s}_2) \). The deterministic version discussed by van Benthem (1986) could be expressed by assuming \( \text{Prob}(p_0^0) = \text{Prob}(p_1^1) = 1 \).

Figure 3: Probabilistic finite state automaton for more than one car is blue.

Probabilistic push-down automata (PPDAs) are tuples \( \langle S, \Sigma, \Gamma, M, s_0, F, \text{Prob} \rangle \), where S and \Sigma as previously stand for the set of states and the input alphabet, \Gamma is the stack alphabet (# \in \Gamma marks the bottom of the stack) and the transition relation is \( M \subseteq (\Sigma \times \Sigma \times \Gamma) \times (\Sigma \times \Gamma^*) \), where the length of \( \Gamma^* \) is at most 2 elements. Similarly as with PFSA, \( \sum_{(s_j, a) \in M(s_j, a, s_i)} \text{Prob}(s_j, a, s_i) = 1 \) for every \( (s_j, a, A_j) \in \Sigma \times \Sigma \times \Gamma \). An example of a PPD modeling the verification of more than half of the cars are blue is given in Table 1, where s_1 is the final accepting state. Each non-empty box shows all the transitions in one probability space and each row represents rules with identical effects, so the rules in the same row are expected to have the same probability (for example, in Row 2, reading BLUE CAR leads to either adding one Y to the stack or removing N from the top of the stack; since N keeps track of non-blue cars and Y keeps track of blue cars, the rules in Row 2 lead to the same effect; similarly for the other rows). The non-probabilistic variant could be expressed by stating that the second row in each block has
probability 1, see also Fig. 2 for such a deterministic pushdown automaton.

Testing probabilistic automata.

We tested whether PFSAs and PPDAs could model accuracy in verification of counting and proportional quantifiers. We applied PFSAs and PPDAs to the data collected in two experiments discussed by Szynanik and Zajenkowski (2010b) and Zajenkowski and Szymanik (2013).

In the experiments participants were asked to verify Polish sentences with counting and proportional quantifiers against pictures showing cars in a car park. There was an added difficulty of a working memory task in Exp. 1 (which we will ignore in our models, to keep our study manageable at this point).

The verification tasks consisted of simple propositions in Polish containing a quantifier that probed a color feature of cars on display, e.g., ‘Więcej niż połowa samochodów jest niebieska’ (More than half of the cars are blue). The participants were asked to decide if the proposition accurately described the presented picture. They responded by pressing the button with the letters ‘p’ or ‘f’ if true or false, respectively. (The letters refer to the first letters of Polish words for ‘true’ and ‘false’.) In the first experiment the displayed car park always contained 15 cars, while in the second experiment the accompanying picture had either 15 elements or 17 elements. Each picture contained objects in two colors.

Four different quantifier types were used in the studies. Here, we only consider data from three groups:

1. **PQ**: proportional quantifiers (less than half, more than half); PQs were studied in four different scenarios, which varied according to the number of elements of the probed color vs. another color (9 vs. 6; 8 vs. 7; 10 vs. 7; or 9 vs. 8);

2. **CQ4/5**: counting quantifiers of relatively low rank (less than 5, more than 4); the number of cars in the probed color was maximally close to the criterion for validating or falsifying the proposition;

3. **CQ7/8**: counting quantifiers of relatively high rank (less than 8, more than 7); the number of cars in the probed color was maximally close to the criterion for validating or falsifying the proposition.

PQs were tested in 8 trials for each scenario type. Each CQ type also appeared in 8 trials. 50% of the items were designed to be judged as true, the rest was false.

63 participants took part in Experiment 1, 99 participants took part in Experiment 2.1

The descriptive summary of the data is presented in Figure 4. It shows that the accuracy in responses to CQs decreased with the increase of the number of elements that had to be counted. It also shows that the participants judging PQs became less accurate as the ratio of the elements of the probed color and of another color got closer to 1 (e.g., the scenarios 8 vs. 7 and 9 vs. 8 were more difficult than the scenarios 9 vs. 6 and 10 vs. 7).

We now turn to the modeling of the verification of CQs (CQ4/5, CQ7/8) and PQs. The models implemented PFSAs and PPDAs and captured, among other things, the descriptive observations just mentioned. The PFSAs and PPDAs themselves were embedded in Bayesian hierarchical models, which allowed us to include differences in individual participants’ responses in our analysis. All models were implemented in JAGS (Plummer, 2003).

The model for the verification of CQs is summarized graphically in Figure 5 (see Lee and Wagenmakers 2014 for an introduction into graphical summaries of Bayesian models). The data to be estimated, \(k_{N,j}\) (the number of correct responses given by a subject in each condition) were taken to come from the binomial distribution with \(n = 8\) (the number of items seen by each subject in each condition). The parameter \(\theta_{N,j}\) is modeled as the addition of the fixed effect \(\text{SUM}_N\) and the subject random effect coming from the normal distribution with deviation \(\sigma\). \(\text{SUM}_N\) is the probability calculated using PFSA.

The PFSA we consider has three possibilities at each state when encountering the car of the probed color (BLUE CAR from now on): either it advances to the next state (the ‘correct’ behavior), \(p_{1}^{0}\) and \(p_{1}^{1}\) in Figure 3, or it loops, \(p_{0}^{0}\) and \(p_{0}^{1}\), or it moves by two states, \(p_{2}^{0}\) in Figure 3. When the final state is \(s_{n+1}\), or any higher state, the sentence more than \(n\) cars are blue is accepted and fewer than \(n+1\) cars are blue is rejected, when the final state is lower than \(s_{n+1}\), the situation is reversed. We assumed that the probability of \(p_{1}^{1}\) is the same in every probability space, and that the same is true for \(p_{0}^{0}\) and for \(p_{2}^{0}\). In other words, the PFSA has only three free parameters, \(p_{1}^{0}\) to \(p_{2}^{0}\), irrespective of the value of \(i\).

The prior distribution of the parameters is the Dirichlet distribution, in which all the three parameters are equal. This represents the fact we have no previous information that one transition is more likely than the other ones.

\(\text{SUM}_N\) is the sum of all walks through our probability spaces that lead to the correct response given the number of blue cars and the CQ used. Different SUMs, \(\text{SUM}_{4/5}\) and \(\text{SUM}_{7/8}\), were computed since the number of walks differ in case of CQ4/5 and CQ7/8.

The model simulating the verification of PQs was similar, the only important difference being that \(\text{SUM}_N\) came from the PPDA discussed above (see Table 1). Unlike in case of PFSAs, two types of transitions are estimated here: either \(p_{0}^{0} = p_{2}^{0}\), covering the probability space of transitions when encountering a blue car, or \(q_{0}^{0} = q_{2}^{0}\), covering the probability space of transitions when encountering a non-blue car.2 \(\text{SUM}_N\) dif-

\[1\] The number of participants differed slightly from the original reports as some subjects were excluded from the analysis due to missing values on other cognitive tasks studied therein.

\[2\] The model discussed here ignores \(p_{1}^{1}\) and \(q_{1}^{1}\), so the comparison between PFSAs and PPDAs is more straightforward (i.e., the same number of transitions is assumed for each probability space). We also ran the full model, which included \(p_{1}^{1}\) and \(q_{1}^{1}\). The full model
The posterior distributions of the most relevant parameters are summarized in box-plots of Figure 6. The ‘correct’ transition (p1, q1) is now the most common (in the PFSA) than in the PPDA but the 95% confidence interval of the PPDA posterior includes the mean of p1 of the PFSA and thus, we cannot conclude that the correct transition significantly differs between automaton types. The box-plot of θ (probability of success) shows that we correctly model that the success probability of binomial distribution is lower with CQs of higher ranks, and also did not show any significant divergences from the results discussed here.

The observed mean. (As is common, the p-values were calculated for the simulations are not significantly different from the observed means. (As is common, the p-values were calculated as proportions of simulations which are at least as extreme as the observed mean.)

We conclude that our hierarchical Bayesian models can successfully model the data of two experiments. In particular, the models correctly capture the fact that higher rank CQs decrease accuracy, and the accuracy in verification also decreases when the ratio of sets compared by PPs is smaller. Since the information about number of elements and size of sets enter only in PFSAs and PPDAs in our models, we conclude that PFSAs and PPDAs constitute valid approaches to modeling the verification of CQs and PPs.

**Comparing semantic automata with ANS.**

It is a plausible hypothesis that the observed variation in the data should not be modeled by probabilistic automata but by probabilistic number estimations. One influential model in psychophysics is Approximate Number Sense, used to repre-
sent imprecise cardinalities and the comparison of quantities without counting (see Dehaene, 1999). Recently, ANS has been used to model the verification of the quantifier ‘most’ (Lidz et al., 2011; Pietroski et al., 2009) under 200 and 150 ms time pressure. We used the ANS model of Lidz et al. (2011) for the quantifiers more than half and fewer than half to learn whether our findings could be captured by this model. If this was the case, the strength of our findings would be weakened as one could argue that there is no need for a specific semantic model of verification.

The posterior predictive of the ANS model is summarized in the fourth and fifth rows of Table 2. The predictions do not diverge significantly from the observed values or the values of our PPDA. However, these good predictions come at a cost: one free parameter of the model, the Weber fraction \( w \), is underestimated. Its posterior distribution, shown in Table 2, is at odds with previous findings that French adults’ Weber fraction is .12 (Pica et al., 2004), i.e., on average French adults can discriminate the ratio 9:8 but not finer ratios. In contrast to that, our model would predict that on average participants can discriminate the ratio 15:14.

Unless Polish and French adults differ in their representation of imprecise quantities (which we find extremely unlikely as ANS is know to be language-independent (see Dehaene, 1999)), the estimation of \( w \) argues against ANS as a suitable model for quantifier verification. Therefore, even though ANS explains accuracy data for the verification of ‘most’ under time pressure, it is unlikely that ANS is employed in a similar way in the verification process for related quantifiers without time pressure.

### Discussion and Outlook

We have introduced the probabilistic semantic automata model. It extends a prominent computational approach from logic and linguistics. In the paper, we have modeled two main natural language quantifier classes: counting and proportional quantifiers. Our method of turning semantic automata into probabilistic semantic automata can be straightforwardly applied to obtain a probabilistic verification model for any natural language quantifier. Therefore, the paper provides a new tool for the semantics of natural language.

Our experiments have shown that probabilistic semantic automata can explain judgment accuracy in sentence-picture verification experiments. Moreover, the probabilistic semantic automata explain the distance effect in proportional quantifier verification: a decrease in verification accuracy as the numerical distance between the two sets to be compared decreases. Furthermore, we have critically compared probabilistic semantic automata model with the verification model based on number estimations and we have argued that ANS cannot consistently explain the verification process underlying the semantics of proportional quantifiers. Hence, the probabilistic semantic automata model seems to be a necessary innovation in cognitive science. Additionally, the presented approach illustrates fruitful interaction between computational cognitive modeling and more traditional disci-

---

3The Weber fraction expresses the smallest numerical difference between two quantities that participants can distinguish. The Weber fraction of \( n_1 \) vs \( n_2 \) is calculated as \( (n_1 - n_2) / n_2 \).
plines: linguistics, logic, and formal semantics.

As it always happens introducing a new perspective creates many further questions and research opportunities. In conclusion let us briefly mention a few such themes that we find particularly exciting. The main goal of our modeling is to better understand cognitive resources underlying quantifier processing. As we recalled in the introduction there is an ample psychological evidence in favor of semantic automata. For instance, Zajenkowski and Szymanik (2013) have recently argued that the cognitive mechanism of comparing in memory the cardinalities of two sets is crucial when it comes to the cognitive difficulty of the proportional judgements. However, PPDAs seem to suggest that counting and not comparison is the most important process explaining the accuracy of verification. In fact, in our model it is even hard to distinguish counting errors from memory slips. Therefore, to further explain the role of cognitive resources in quantifier processing we plan to bring probabilistic semantic automata model more extensively to the lab by confronting its predictions with more complex behavioral data, like reaction times. On the other hand, probabilistic semantic automata model poses also many new theoretical challenges: Do probabilistic semantic automata give rise to a new classification of natural language quantifiers (cf. van Benthem, 1986)? Can the model be naturally combined with the modeling of the acquisition of quantifier meanings (see Gierasimczuk, 2007; Clark, 2010; Piantadosi, 2011)? Clearly, such questions are beyond the topic of this paper but offer interesting venues to explore in future applications of probabilistic semantic automata to cognition.

Acknowledgments

JD acknowledges NWO Veni Grant 275.80.005. JS acknowledges NWO Veni Grant 639.021.232. The work of MZ was supported by a grant no. 2011/01/D/HS6/01920 funded by the National Science Centre in Poland.

References


