Role of expectation and memory constraints in Hindi comprehension: An eyetracking corpus analysis

Arpit Agrawal CS1120216@IITD.AC.IN
Sumeet Agarwal SUMEET@IITD.AC.IN
Samar Husain SAMAR@IITD.AC.IN
Indian Institute of Technology, Delhi

Abstract

We have used the Potsdam-Allahabad Hindi Eye-tracking Corpus that contains eye-movement data from 30 participants on 153 sentences to investigate the effect of surprisal and retrieval on reading time, while controlling for word-level predictors (word complexity, syllable length, unigram and bigram frequency) and integration and storage costs. We find that surprisal has a significant coefficient in only in First Pass Reading Time while storage cost shows up only in Total Fixation Time, thus indicating that the two measures of predictability capture different cognitive variables.

1. Introduction

Surprisal, calculated using dependency as well as phrase-structure grammars, and retrieval cost for different extents of parallelism have been found to have a statistically significant effect on reading times using eye-tracking corpora of German sentences. (Boston et al., 2011) Surprisal calculated using an unlexicalised formulation has also been found to have a significant contribution in predicting reading times for English. (Demberg & Keller, 2008) For Hindi, the only large-scale study using eye-tracking data (Husain et al., 2015) so far investigates the effect of low-level predictors (at the word level), integration cost and storage cost on difficulty in comprehension; they leave an open question on whether surprisal-based expectation has a larger effect size than integration- and storage-cost effects which we explore in our study.

Appearing in Proceedings of the 2nd Indian Workshop on Machine Learning, IIT Kanpur, India, 2016.

2. Background

2.1. Surprisal and Retrieval

Surprisal For a given probabilistic grammar G, we define prefix probability at the i^{th} word (α_i) as the sum of probabilities of all partial parses that can explain the first i words (Boston et al., 2008). Surprisal at the i^{th} word then is the logarithm of the ratio of prefix probabilities before and after seeing the word. It is easy to observe that surprisal is always positive and is unbounded. In our computation, we only take the top k parses based on their likelihoods at each word to computed α_i.

$$surprisal(i) = log(\frac{\alpha_{i-1}}{\alpha_i})$$

Retrieval Retrieval cost according to the ACT-R activation-based memory theory (Anderson et al., 2004) is the time taken to retrieve a word from the memory which is a function of decay and interference. A word takes longer to retrieve if it was last seen a long whiile ago or if many words *similar* to the word being retrieved. Here, *similar* words are the ones with the same category POS tag. More formally, retrieval cost T_i at the i^{th} word ($t_{j}{}_{j=1}^{n}$ denote the set of times when the i^{th} word was retrieved) is given as:

$$T_i = Fe^{A_i} \texttt{ where } A_i = ln(\sum_{j=1}^{n} t_j^{-0.5}) + \sum_j W_j S_{ji}$$

$$W_j = 1/j \texttt{ and } S_{ji} = S_{max} - ln(fan_j)$$

where fan_j is the number of words *similar* to the j^{th} cue and S_{max} is set to 1.5.

Finally, productions in ACT-R are assumed to accrue a fixed cost of 50 ms and reading a cost of 1 ms to execute. Formation of a dependency arc accrues the cost of a retrieval along with two productions and a shift

Table 1. Results of linear mixed-effects regression on FPRT

Predictor	Coeff	Std Error	t-stat
Intercept	**5.502**	**0.023**	**237.74**
word_complex	0.003	0.003	0.87
word_freq	−0.0003	0.006	−0.04
word_bifreq	**−0.014**	**0.003**	**−4.00**
syll_len	**0.112**	**0.011**	**9.95**
IC	0.004	0.004	1.00
SC	0.003	0.006	0.50
surprisal (k=10)	**0.013**	**0.004**	**2.88**

operation accrues only one production cost. (Boston et al., 2008)

2.2. Other Predictors

Apart from surprisal and retrieval, we use four word-level predictors and two sentence-level predictors. (Husain et al., 2015) The word-level predictors we use are syllable length (syll_len), word complexity (word_complex), unigram (word_freq) and bigram frequencies (bigram_freq). Here syllable length and word complexity are computed using the Devanagari rendering of the word while the unigram and bigram frequencies were computed from the beta version of the Hindi-Urdu treebank data which contains 400,000 words. (Bhatt et al., 2009)

Integration (IC) and Storage Costs (SC) proposed by Gibson (2000) as part of Dependency Locality Theory (DLT) (Gibson, 2000) are two high-level metrics we control for while testing for the effect of surprisal and retrieval. Integration Cost, intended to capture the retrieval cost of a dependent at its integration site (Lewis & Vasishth, 2005), is computed here as the sum of distances of a word from its left-dependents. Storage Cost, on the other hand, characterizes the processing load incurred as a result of maintaining predictions of upcoming heads.

2.3. Eye-tracking measures

We use eye-tracking data from the Potsdam-Allahabad Hindi Eyetracking Corpus which contains different metrics of reading time for 153 sentences picked from the Hindi-Urdu treebank (HUTB) read by thirty graduate and undergraduate students of the University of Allahabad in the Devanagri script. (Husain et al., 2015) We use three eye-tracking measures in our analysis, namely First-pass reading time, regression path duration and total fixation time calculated using the `em2` package in R.

First Pass Reading Time (FPRT) on a word refers to the sum of the fixation durations on the word after it has been fixated after an incoming saccade from the left, until the word on the right is seen. *Regression Path Duration (RPD)* is the sum of all first-pass fixation durations on the word and all preceding words in the time period between the first fixation on the word and the first fixation on any word right of this word. *Total Fixation Time (TFT)* is the sum of all fixations on a word. This is always greater than (or equal to) the FPRT since this also includes the re-reading time. (Logacev & Vasishth, 2006)

3. Approach

Algorithm 1 Surprisal and Retrieval Calculation

Input: Sentence as a list of words $buffer$
$index \leftarrow 0$
$loglikelihood \leftarrow 0$
$surp, retr \leftarrow []$
$S \leftarrow [empty\_parse(buffer)]$
for $index = 0$ **to** $length(buffer)$ **do**
  while $\exists parse \in S, parse.index <= index$ **do**
    $poss \leftarrow$ set of possible transitions for parse
    $S \leftarrow S \backslash \{parse\}$
    $probs \leftarrow$ probabilities of each $tr \in poss$ using learning algorithm
    for $tr \in poss$ **do**
      $S_{poss} \leftarrow parse.make\_transition(tr, prob[tr])$
      $S \leftarrow S \cup \{S_{poss}\}$
    end for
  end while
  Sort S in decreasing order of likelihoods
  Truncate S to keep top k elements of S
  $ll\_new \leftarrow log(sum(\{parse.prob : parse \in S\}))$
  $surp[index] \leftarrow loglikelihood - ll\_new$
  $retr[index] \leftarrow max(\{parse.retr : parse \in S\})$
  $loglikelihood \leftarrow ll\_new$
end for
Output: $surp, retr, S[0]$

3.1. Surprisal and Retrieval Calculation

We implemented our own probabilistic incremental dependency parser in Python available here. We use the Arc-Eager algorithm which is a transition-based algorithm with four transitions (`Left Arc, Right Arc, Shift, Reduce`) (Nivre, 2008) at the core of our parser to parse sentences and the Maximum Entropy algorithm (Daumé III, 2004) to get probabilities for each transition in order to output surprisal, retrieval and the most likely parse. An outline of the algorithm used has been sketched in Algorithm 1.

Table 2. Results of LMER on RPD (reduced model)

Predictor	Coeff	Std Error	t-stat
(Intercept)	**5.65**	**0.032**	**177.11**
word_complex	0.002	0.003	0.80
word_freq	-0.008	0.005	-1.72
word_bifreq	**-0.025**	**0.003**	**-8.20**
syll len	**0.119**	**0.004**	**26.83**
SC	**-0.016**	**0.004**	**-4.46**
surprisal (k=10)	0.001	0.004	0.35
retrieval (k=10)	**0.008**	**0.004**	**2.15**

3.2. Training the Parser

We have used a morphologically rich incremental feature set that includes the form, lemma, part-of-speech tag, category, tense-aspect-modality and case markers along with the chunking information of the top two elements of the stack, the top element of the buffer and their dependents in the partial parse. We have used the sentences in the Hindi-Urdu treebank (HUTB) to train the parser. The HUTB contains the dependency parse for around 12000 sentences along with morphological information (part-of-speech tag, category, lemma, case marker, chunk information, tense-aspect-modality and type of sentence) about each word in the treebank. (Bhatt et al., 2009) The Unlabeled Attachment score (proportion of words that are correctly attached to their parent) for our parser is close to 88%.

4. Results

We performed a linear mixed effects regression analysis controlling for the random effects due to different subjects and different sentences. Before the regression analysis, all predictors were scaled by centering them around their mean and dividing by their standard deviation. We applied a log-transformation on the eye-tracking metrics to achieve approximate normality of residuals (dropping the data points with $0ms$ for these fixation measures).

Surprisal In the linear mixed-effects regression for log(FPRT), log(RPD) and log(TFT) for different values of k, we do not find a significant effect of surprisal for log (RPD) or log (TFT), but find a statistically significant coefficient for log (FPRT) (Table 1 for $k = 10$) for most values of $k(k > 1)$. Among the coefficients of surprisal in the case of First Pass Reading Time, we note that while the standard deviation of the estimate is nearly constant, the mean estimate first increases with k, reaches a maximum at $k = 10$ and then starts decreasing again. The effect on R^2 is not very pronounced, as it changes from 62.43% to 62.49% on adding surprisal.

Retrieval While testing for the effect of retrieval, we leave out integration cost (IC) from the set of predictors since IC and retrieval have very similar definitions and are thus highly correlated. Even after removing IC though, we do not get a significant coefficient for retrieval in any of the three eye-tracking measures. However in a less conservative model (where we remove the predictors from random effects, thus ignoring the random slopes) fit for Regression Path Duration, we do get a significant positive coefficient for retrieval (Table 2 for k=10).

Storage Cost We find that storage cost has a significant coefficient only in the case of Total Fixation Time which is consistent with previous studies. (Husain et al., 2015)

5. Discussion

5.1. Surprisal, retrieval and position of the word

We notice that both surprisal and retrieval have a significant correlation with the position of the word in the sentence. Words later in the sentence tend to have both a higher surprisal value as well as higher retrieval cost. Surprisal tends to be higher for words later in the sentence probably because more reduce operations are performed while parsing words later in the sentence. Similarly, retrieval tends to be higher later in the sentence as there is more space for long-distance dependencies and more interference. Also independent of the position in the sentence, for words with high retrieval cost due to long-distance dependency, more transitions may be needed to form arc, hence higher surprisal. This could be an due to the parsing algorithm or due to the head-final structure of Hindi sentences.

5.2. Effect of surprisal on FPRT

We find that surprisal has a significant coefficient only in the case of First pass Reading Time while controlling for graphemic complexity, word frequency, bigram frequency, syllable length, integration cost and storage cost. A significant t-value is not seen in RPD (Regression Path Duration) or TFT (Total Fixation Duration). We find significant coefficient for bigram frequency, surprisal and syllable length in FPRT, bigram frequency, integration cost and syllablle length in RPD and in unigram frequncy, bigram frequency, storage cost and syllable length in TFT.

In a simpler model, however, on removing predictors from random effects a significant t-value is found in the case of TFT. In this case, however, there is significant autocorrelation in the residuals and thus the t-value cannot be relied upon.

5.3. Effect of Retrieval on RPD

In order to evaluate the effect of retrieval on the eye-tracking measures, IC is dropped from analysis since it is highly correlated with Integration Cost. Even after dropping IC, we do not see a significant coefficient in any of the three eye-tracking measures we consider. Partially, this was expected since Husain et al (2015) (Husain et al., 2015) too did not find any significant effect of IC (which is defined very similarly to retrieval cost). After removing predictors from the random effects, however, we do find a significant coefficient in the case of Regression Path Duration.

5.4. Exclusiveness of Surprisal and Storage Cost

While surprisal has a significant effect on FPRT, storage cost has a significant coefficient in TFT. Both metrics capture the idea of prediction while parsing and appear significant in different eye-tracking measures (where FPRT is an early measure and TFT is a late measure).

6. Conclusion

We find that surprisal has a significant effect on the First-Pass Reading Time and also some weak evidence of the effect of surprisal on Total Fixation Duration and retrieval cost on Regression Path Duration. This is in contrast to (Boston et al., 2011) where a significant effect was found for both surprisal and retrieval for most eye-tracking measures for German sentences at least for higher k. Also, unlike German, we do not see a consistent increase in the effect size as k increases. We also find that surprisal and storage cost have significant effects on different eye-tracking measures and thus have mutually exclusive effects.

Acknowledgments

We would like to thank Dr Rajkumar Rajakrishnan (Assistant Professor, IIT Delhi) and Dr Ashwini Vaidya (DST-CSRI post-doctoral fellow, IIT Delhi) for their help in the form of useful suggestions and also helping us with the data for this research. We would also like to thank the IIT Delhi HPC facility for the computational resources.

References

Anderson, John R, Bothell, Daniel, Byrne, Michael D, Douglass, Scott, Lebiere, Christian, and Qin, Yulin. An integrated theory of the mind. *Psychological review*, 111(4):1036, 2004.

Bhatt, Rajesh, Narasimhan, Bhuvana, Palmer, Martha, Rambow, Owen, Sharma, Dipti Misra, and Xia, Fei. A multi-representational and multi-layered treebank for hindi/urdu. In *Proceedings of the Third Linguistic Annotation Workshop*, pp. 186–189. Association for Computational Linguistics, 2009.

Boston, Marisa, Hale, John, Kliegl, Reinhold, Patil, Umesh, and Vasishth, Shravan. Parsing costs as predictors of reading difficulty: An evaluation using the potsdam sentence corpus. *The Mind Research Repository (beta)*, (1), 2008.

Boston, Marisa Ferrara, Hale, John T, Vasishth, Shravan, and Kliegl, Reinhold. Parallel processing and sentence comprehension difficulty. *Language and Cognitive Processes*, 26(3):301–349, 2011.

Daumé III, Hal. Notes on CG and LM-BFGS optimization of logistic regression. 2004. Paper available at http://pub.hal3.name#daume04cg-bfgs, implementation avaiable at http://hal3.name/megam/.

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

Gibson, Edward. The dependency locality theory: A distance-based theory of linguistic complexity. *Image, language, brain*, pp. 95–126, 2000.

Husain, Samar, Vasishth, Shravan, and Srinivasan, Narayanan. Integration and prediction difficulty in hindi sentence comprehension: Evidence from an eye-tracking corpus. *JOURNAL OF EYE MOVEMENT RESEARCH*, 8(2), 2015.

Lewis, Richard L and Vasishth, Shravan. An activation-based model of sentence processing as skilled memory retrieval. *Cognitive science*, 29(3): 375–419, 2005.

Logacev, P and Vasishth, S. The em package for computing eyetracking measures. *University of Potsdam, Potsdam, Germany*, 2006.

Nivre, Joakim. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553, 2008.