9.19: Computational Psycholinguistics, Pset 6
due Monday 15 November 2021

1 November 2021

Problem setup

Smith and Levy (2013) explored the shape of the relationship between a word's predictability in context and how long, on average, a comprehender spends on reading that word. In particular they were interested in the hypothesis that a word's average reading time might be linear in the SURPRISAL of the word (Hale, 2001; Levy, 2008):

$$\text{surprisal}(w_i|\text{Context}) = \log \frac{1}{P(w_i|\text{Context})}$$

or in some other function of the word's conditional probability. In this assignment, you will investigate this question yourself, using one of the datasets analyzed by Smith and Levy.

On Canvas, you can access the Smith and Levy (2013) self-paced reading dataset under Files|Datasets|Pset 6: surprisal and RTs. This dataset derives from each subject in the experiment reading a number of several-hundred-word passages selected from the Brown corpus (Kučera & Francis, 1967). This dataset is presented in tabular format, with one reading-time measurement per row and with the following columns:

- The **word** that was read;
- A **code** that uniquely identifies the word/context pair;
- The identifier for the **subject** in the experiment from which the reading-time measurement was taken;
- **text_id**, which is the identifier for the text from the Brown corpus that was being read;
- **text_pos**, which is the word number in the text selection that was being read;
- **word_in_exp** is the word number in the experiment for the particular subject;
- The **time** in milliseconds that the word in question was visible on-screen during the subject's reading. (Remember, in self-paced reading this is the time elapsed between the subject pressing a button/key to reveal the word, and the subject pressing the button/key again to mask that word and reveal the next word.)

Your task

Your task for this assignment is essentially to reproduce the main result of Smith and Levy (2013), and extend their analysis from n-gram models to modern neural network language models.

We provide the data and code that implements most of the heavy lifting, but you will need to keep track of everything as you progress through the assignment. This is all situated within an interactive notebook on the Google Colaboratory ("Colab"), which allows you to write and execute Python in your browser.[1]

The notebook for this pset can be found at https://colab.research.google.com/drive/17Fs4saK3ZLaNkYMII_akOV_9rIYBiLqi?usp=sharing. Please carefully read and follow all instructions in the notebook. You will need to write code or textual responses in every place marked as TODO.

There are a series of non-trivial steps that you will need to take to complete this assignment, which we outline in Figure 1. **Orange boxes** indicate data files that you will download through Stellar or the Colab notebook. **White boxes** indicate files that you will produce by running the provided code in the Colab notebook. **Blue boxes** indicate files, code, etc. that you are expected to produce and submit for this assignment, along with your completed notebook.

Let's walk through everything step-by-step below.

Obtaining n-gram surprisals/RTs

As shown in Figure 1, we will provide the reading time (RT) data in a file called `brown_rts.csv`. We will also provide per-token surprisal data from a pre-trained n-gram model (a Kneser-Ney-smoothed 5-gram model) in a file called `ngram_surprisals.tsv`. Your first step is to combine these two files into a format where you can analyze the relationship between reading time and word surprisal. Please see the Colab notebook for detailed instructions.

Before you can do this, in the process of estimating word probabilities, you will need to attend carefully to issues of **word tokenization**. The way words are presented to humans in the reading experiment does not match the way words are separated and processed by the n-gram model. In the self-paced reading experiment, every button press presented a single contiguous string of non-space ASCII characters, so the dataset reflects tokenization by spaces. In the n-gram training data, by contrast, punctuation characters are tokenized out from the rest of word strings.

You will also need to handle out-of-vocabulary (OOV) items. Since not all words in the self-paced reading dataset appear in the n-gram's vocabulary, some of the surprisal values will correspond to unknown ("`unk`") tokens.

[1] Colab is quite similar to Jupyter notebooks. If you have never used Jupyter or Colab before, we highly recommend this introduction to Colab: https://colab.research.google.com/notebooks/welcome.ipynb).

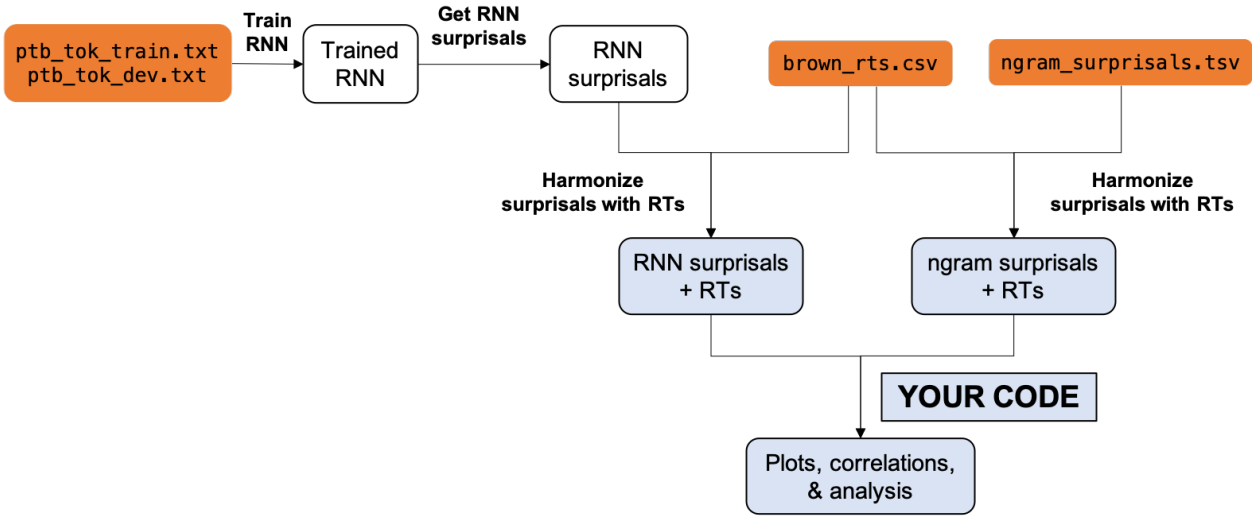


Figure 1: Diagram of problem structure. **Orange boxes** indicate data files that you will download through Stellar or the Colab notebook. **White boxes** indicate files that you will produce by running the provided code in the Colab notebook. **Blue boxes** indicate files, code, etc. that you are expected to produce and submit for this assignment, along with your completed notebook.

Both of these issues prevent you from simply aligning both files on the word-level.To get around this issue, we have provided a script called `harmonize.py`. Run the script in the notebook to produce a harmonized output file. **Please include this harmonized file in your final submission.**

Obtaining RNN surprisals/RTs

Next, we need to perform the same steps using a Recurrent Neural Network (RNN) language model. However, unlike the n-gram model, we will not be providing surprisal values from the RNN. **You will need to train this model yourself from scratch.**

Training. We ask you to train the RNN on a section of the Penn Treebank, a commonly used dataset in natural language processing research.[2] We have provided the files `ptb_tok_train.txt` and `ptb_tok_dev.txt`, which are the training and validation sets, respectively. **Please download the files from Stellar under** `Materials`→`Datasets`→`PTB*`.

[2]Note: If you have previous machine learning experience and want to train your own model, feel free to do so! We have provided these instructions for simplicity and for consistency with the provided n-gram model. If you use a different training or preprocessing method, please describe it in your write-up.

Note that we have already preprocessed these files for you – that is, the files have already been tokenized and unkified.

Once you have downloaded the PTB corpus files from Stellar, pease see the Colab notebook for detailed instructions on how to train your RNN language model. **Note: training may take up to 2 hours, though Colab compute speed is unpredictable and training may finish in as little as 20 minutes.**[3]

Obtaining surprisals. Once you have trained the RNN, you need to obtain surprisals from the model. We have also provided this code for you. Please see the Colab notebook for detailed instructions.

Analysis

Once you have word-conditional probabilities from both models (n-gram and RNN), your job is to interpret your results. For each metric/reading-time investigation, calculate the correlation coefficient, and produce a graphical depiction of the relationship. For the graphical depiction, bin the data by word probability or surprisal, compute the mean RT in each bin, and plot the resulting collection of means. Here is an overview of the analysis we want you to run:

- For each metric in [surprisal, raw probability]:
  - For each model in [n-gram, RNN]:
    1. Obtain metric-RT correlation coefficients;
    2. Draw metric-RT scatterplot with best-fit line, **without** binning RT values; and
    3. Draw metric-RT scatterplot with best-fit line, **with** binning RT values.

We will provide code to run all these analyses for metric = surprisal, but you will need to write the necessary code to reproduce the analysis for metric = raw probability. Please see the Colab notebook for detailed instructions.

Finally: once you've completed the data analysis, provide your interpretation. Does your analysis support the hypothesis of a linear relationship between word surprisal and word reading time? Is that hypothesis better or worse than an alternative hypothesis of a linear relationship between *raw word probability* and word reading time? Are there other alternative hypotheses that might be even more compelling given the data? Justify your reasoning.

If you want to take this further: look at the relationship between word probability and the *next* word's reading time (a "spillover" effect)—is the effect similar as on the current word? Bigger? Smaller? Different shape?

[3] 2 hours is very fast by NLP standards, and it will be running on the cloud, not your local computer. :)

You can also try training an RNN on different or larger datasets, such as Wikitext-2, Wikitext-103, or a still larger dataset (note that training larger models on Colab might require some careful checkpointing work), and see what happens to the relationship with RTs.

What you need to turn in

1. Your full Colab notebook copy, including not only code, cell output, graphs, and quantitative analyses, but also a verbal description of what you did and your interpretation of the results. You can save your notebook as a PDF and submit this file as your write-up. Minimally, you should make sure that your write-up answers all of the questions posed in the *Analysis* section. Further ideas for analysis are given in the Colab notebook.

2. Your harmonized RNN surprisal + reading time file. This is so we can check your work if something goes wrong in the downstream analysis.

3. Your harmonized n-gram surprisal + reading time file. This is so we can check your work if something goes wrong in the downstream analysis.

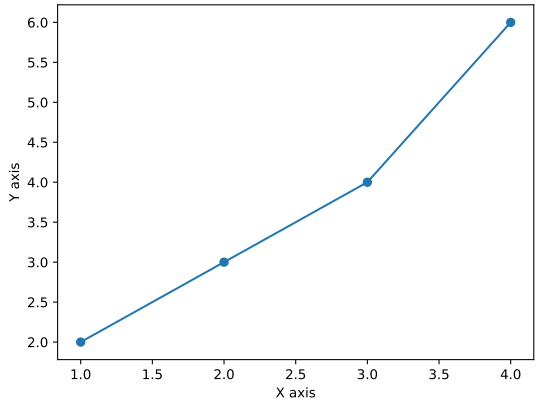
Notes and suggestions

- It is common in self-paced reading to exclude "outlier" reading times that might reflect non-linguistic processing effects, such as the participant taking a break mid-sentence. You can find the outlier-removal policy used in Smith and Levy (2013) in the *Materials and Methods* section of the paper. You can use this policy, an alternative policy, or simply conduct no outlier removal. Be clear about the policy you take and the motivation for it, and take it into account when you provide your interpretation of the results.

- For data visualization, you can use whatever software you are comfortable with. If you have not made data plots before, we recommend using `matplotlib`, a Python visualization library. If you don't have it installed already, you can install it via `pip` with the command `pip install matplotlib`. Here is an example of creating a scatterplot with lines between the points and axis labels:

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4] # x coordinates for the example plot
y = [2, 3, 4, 6] # y coordinates for the example plot
plt.scatter(x, y) # Create a scatterplot with given x and y coordinates
plt.plot(x, y) # Draw lines between the points
plt.xlabel("X axis")
```

```
plt.ylabel("Y axis")
plt.show() # Display it on your screen
plt.savefig("filename.pdf") # Save the result to filename.pdf
```



References

Hale, J. (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*, Pittsburgh, Pennsylvania.

Kučera, H., & Francis, W. N. (1967). *Computational analysis of present-day American English*. Providence, RI: Brown University Press.

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

Smith, N. J., & Levy, R. (2013). The effect of word predictability on reading time is logarithmic. *Cognition*, *128*(3), 302–319.