00960222: Language, Computation and Cognition
Homework Assignment 2
due 4 May 2025

21 April 2025

1 n-gram language models: perplexity (15 points)

Let's assume that we have a vocabulary V which consists of M words ($|V| = M$), and a sentence $S = w_1 w_2 ... w_N$ in which all the words belong to V.
Recall that the perplexity of a language model with respect to S is defined as $\text{PP}(S) = P(w_1, w_2, ..., w_N)^{-\frac{1}{N}}$.

Part (a): Rewrite this expression in terms of bigram probabilities.

Part (b): Now assume that our language model is a *uniform probability* bigram model that assigns $P(w_i|w_{i-1}) = \frac{1}{M}$ for all w_i and w_{i-1}. What is the perplexity of this model on S, as a function of M?
Does the perplexity depend on the particular words that appear in S? Why is this the case?

Part (c): Assume we use MLE to estimate an unsmoothed bigram model from a training corpus which has non zero counts for all the bigram combinations of words from V. If the training and test datasets are coming from the same distribution (e.g., different parts of the same corpus), is it mathematically **possible** for such a model to yield higher test-set perplexity on S compared to the uniform distribution model described in part (b)? If it is possible, is it **likely**? Justify your answers.

2 Forms of perplexity (5 points)

One of the definitions of perplexity of a language model M evaluated on a length-N test corpus $w_{1...N}$ is:

$$\text{PPL}(w_{1...N}) = \sqrt[-N]{\prod_{i=1}^{N} P_M(w_i|w_{1...i-1})}$$

Exercise: prove that this can equivalently be written as:

$$\text{PPL}(w_{1\ldots N}) = \exp\left[\frac{1}{N}\sum_{i=1}^{N}\log\frac{1}{P_M(w_i|w_{1\ldots i-1})}\right]$$

(Note: we could choose a different base & exponent so long as they match; often perplexity is instead written as $2^{\frac{1}{N}\sum_{i=1}^{N}\log_2\frac{1}{P(w_i|w_{1\ldots i-1})}}$, for example.)

3 Tuning an n-gram language model (20 points)

This Colab notebook contains starter code for this problem.

When we covered n-gram models, you learned about MAXIMUM-LIKELIHOOD ESTIMATION and ADDITIVE SMOOTHING for a bigram model. Maximum-likelihood estimation is equivalent to RELATIVE FREQUENCY ESTIMATION:

$$\widehat{P}_{RFE}(w_i|w_{i-1}) = \frac{\text{Count}(w_{i-1}w_i)}{\text{Count}(w_{i-1})}$$

In ADDITIVE SMOOTHING, a pseudo-count of α is added to each n-gram count, so that the bigram probability of w_i given w_{i-1} is estimated as

$$\widehat{P}_{\alpha}(w_i|w_{i-1}) = \frac{\text{Count}(w_{i-1}w_i)+\alpha}{\text{Count}(w_{i-1})+\alpha V}$$

where V is the vocabulary size.

Task: consider the following toy datasets, similar to those appearing in recitation 4:

Training set	**Validation set**	**Test set**
`dogs chase cats`	`the cats meow`	`cats meow`
`dogs bark`	`the dogs bark`	`dogs chase the birds`
`cats meow`	`the dogs chase the cats`	
`dogs chase birds`		
`birds chase cats`		
`dogs chase the cats`		
`the birds chirp`		

1. Implement a bigram language model with additive smoothing, trained on the training set and with the smoothing parameter α optimized to mimimize the held-out perplexity of the validation set (make sure that you include beginning-of-sentence and end-of-sentence tokens in your representation of each sentence, and that you include these tokens appropriately in your perplexity computations). In choosing which possible values of α to consider, you can use any of a variety of methods; the simplest is grid search, but you can use another method if you prefer.

2. Plot the validation-set perplexity against the test-set perplexity, considering a range of α values (think carefully about what range you need to consider in order to fully understand how the validation and test sets test generalization from the training set). What value of α worked the best for the validation set? Was it the same that would have worked best for the test set?

Want to take this farther? (BONUS 5 points) Once you've done the above you've completed the assignment, but for further learning you can try tuning α for a larger-scale dataset, such as Wikitext-2 or Wikitext-103 (training sets of about 2 million words and 103 million words respectively, which are still small by contemporary NLP standards!), This will also give you an opportunity to see how well your implementation scales time- and memory-wise as the dataset gets larger, which is often very important for computational work on language.

Notes and suggestions

Background note: in machine-learning terminology, choosing the value of α based on the held-out perplexity of the validation set is an example of HYPERPARAMETER TUNING. In general, you don't get to look at the test-set performance for all the different choices of the hyperparameters; you have to make your hyperparameter commitments on the basis of exploration against the validation set, and then look at performance on the test set at the end. In the present exercise, We are asking you to look at both together strictly for pedagogical purposes.

Open versus closed vocabulary language modeling: if you try to scale up to a larger dataset like suggested above, you will have to deal with the question of how to define the vocabulary size V. In the general case, there may be words in your validation and/or test sets that do not appear in your training set, so you cannot just set V to the number of distinct words that appear in the training set, or you will wind up with an improper probability distribution (why?). A couple of alternative options include:

- "Peeking" at the validation and test sets and defining the vocabulary as including at least the union of all words that appear in training, validation, and test sets; this is what is known as CLOSED VOCABULARY language modeling.
- Defining a vocabulary that does not necessarily include all words that appear in the validation and/or test sets, together with an "unkification" function that maps words that do not appear in this vocabulary to one or more UNKNOWN WORD categories (when there is only one unknown word category, it's traditionally denoted with `<UNK>`). You then convert your training, validation, and test datasets using this unkification function, and all perplexity evaluations involve these converted dataset. This is OPEN VOCABULARY language modeling.

These issues are covered in some more detail in SLP3 section 3.3.1. Note that you don't have to worry about any of this for the toy dataset for the present problem, because every word appearing in the validation and test sets also appears in the training set.

4 The Relationship between Surprisal and RTs (60 points)

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.

The Smith and Levy (2013) self-paced reading 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 to reproduce the main result of Smith and Levy (2013), using the output of a 5-gram model, trained on part of the Penn Treebank, a commonly used dataset in natural language processing research.

The notebook for this assignment can be found here. 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.

Obtaining n-gram surprisals/RTs

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. In your writeup, describe what preprocessing steps you performed.

Statistical Analyses

Once you have aligned the surprisals and the RTs, your job is to visualize and statistically analyze their relationship. You will examine and compare the relationship of RTs both to surprisals and to raw probabilities.

Specifically, for each metric in [surprisal, raw probability]:

- Fit a linear regression model to predict RTs from the metric. You should report the coefficient for the metric term (slope) and a corresponding t-score and p-value (to determine whether it is significantly different from 0), as well as an R^2-score (the coefficient of determination) of the model.
- Draw metric-RT scatterplot with best-fit line, **without** binning RT values; and
- Draw metric-RT scatterplot with best-fit line, **with** binning RT values.

Interpret the results Does the univariate 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?

Multiple regression analysis: The aim of this analysis is to assess the surprisal-RTs association while controlling *word-length* and *word log-frequency.* First, you will be asked to extract these two variables by yourself (see detailed instructions in the Colab notebook). Then, you should fit a multiple linear regression model to predict RTs from *surprisal + word length + word log frequency*, interpret the model's results, and compare them to the results from analysis 1. The new model should take the form:

$$\text{RT} = \beta_0 + \beta_1 \text{surprisal} + \beta_2 \text{length} + \beta_3 \text{log frequency}.$$

Interpret the results How does the *surprisal* coefficient of this model compare to the surprisal coefficient in the univariate model? Does your conclusion regarding the effect of *suprisal* on RTs from the univariate analysis still hold?

What you need to turn in

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.

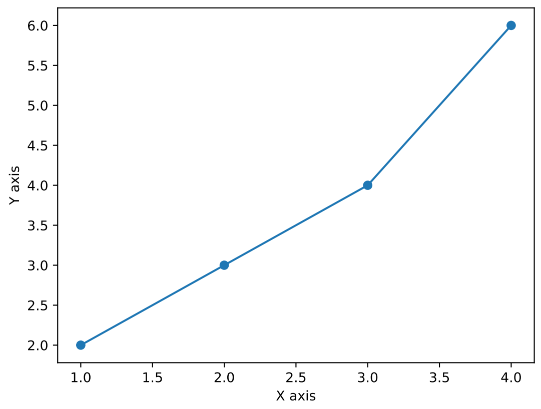
Notes and suggestions

- 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. *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics*, 159–166.

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.