Natural Language Processing 2025
Assignment 2
Due Friday, April 25 (by midnight)

A2: Language Modeling with N-Grams & Word Vectors

Authors *Nathan Schneider, Lucia Donatelli, Alexander Koller, Bas Maat*

Submission One group member should upload a .zip file to Canvas as GROUPX_A2_NLP2025. The file should contain:

1. Your completed files, notebooks, and results (named as instructed below);
2. A PDF with all written responses (separate from your code; see word limits below). Name this document GROUPX_A2_NLP2025. Ensure last names of all group members appear on the document.

Grading The assignment is worth 100 points. Point distributions are listed below.

Notes

- Please use the latest version of Python, preferably with the Anaconda distribution, to complete the assignment.
- Prior to completing the assignment, you should install and familiarize yourself with the spaCy, NLTK, and NumPy libraries.
- The assignment and related skills will be reviewed during Thursday TA sessions. For additional practice notebooks relevant to this assignment, see here.

Overview & Goal

In this assignment, we explore two foundational approaches to modeling language: n-gram models and word vector representations. These techniques capture complementary aspects of how language functions and lay the groundwork for more advanced NLP systems. This assignment combines implementation and analysis to facilitate a deeper understanding of how machines can begin to learn the structure and meaning of language from text.

Please read through the entire assignment before dividing and starting work. Be sure to not miss the bonus!

A2.0 Group Agreement (5 points)

Before beginning, it is important to take time to get to know our group members, define learning goals, and set up a structure that ensures group work will be effective. For this, please complete the following (Thursday TA sessions are a great opportunity for this). All documents are available in the A2 folder:

1. Answer the VU group work Starting Questions on your own. To help you brainstorm goals in different roles, see the VU's website on group work. For question 9, generate a random question from here for the entire group and answer it. (2 points)
2. Share your answers in your first group meeting. Based on your discussion, draft a timeline of the work to be done in A2, how you will allocate the tasks amongst your group, and how you will communicate as a group, including when you will have group meetings. Each group member should sign this agreement document (2 points)
3. Before you hand in your assignment, write a brief summary of how your group worked together and if you adhered to your contract. (1 point)

A2.1 Text Processing & Zipf's Law (10 points)

In this part of the assignment, we will walk through basic text processing steps and empirically verify Zipf's law on the Brown Corpus, a sample of American English texts of varied genres.

Step 1. To access the full Brown corpus, import it from NLTK. Compute a list of unique words sorted by descending frequency for (i) the whole corpus and (ii) two different genres of your choice.
Note: see here for guidance on tokenizing the text and defining your words.

Step 2. Extract the following information for each portion (corpus + each genre). This process and informations should be visible in your code and output files): (i) number of tokens; (ii) number of types; (iii) number of words; (iv) average number of words per sentence; (v) average word length; (vi) number of lemmas.

Step 3. Run a default part-of-speech (POS) tagger on the dataset and identify the ten most frequent POS tags.

Step 4. Use the Python library matplotlib to plot the frequency curves for the corpus and two genres you choose: i.e. x-axis is position in the frequency list, y-axis is frequency. Provide both a plot with linear axes and one with log-log axes.

Provide your source code for all of the above in a file named problem1.py or .ipynb.

Step 5: Answer the following questions in your group document (total word limit 300).

1. How did you define a *word* for your task and why? How was this different from a *lemma*? What difficulties, if any, did you run into when making this decision?

2. What do you notice about the information you extracted from this corpus, specifically pertaining to how properties of the two different genres relate? Highlight key important linguistic observations that relate to class discussions about Zipf's law, linguistic corpora, and bias in data.

A2.2 N-Gram Language Modeling (25 points code, 10 points written)

An n-gram language model is a type of statistical language model that predicts the likelihood of a sequence of words based on the probability of occurrence of smaller word sequences called n-grams.

An n-gram is a contiguous sequence of n items from a given sample of text, where the items can be words, characters, or even phonemes. In the context of language modeling, an n-gram typically refers to a sequence of n words. For example, in the sentence "I love to play soccer," the 2-grams (or bigrams) would be "I love," "love to," "to play," and "play soccer." The 3-grams (or trigrams) would be "I love to," "love to play," and "to play soccer."

The n-gram language model makes assumptions about the probability of a word occurring based on the previous n-1 words. It utilizes a training corpus of text to calculate the frequencies of n-grams and estimates the probability of encountering a particular word given the preceding n-1 words.

For additional reference, see the chapter in SLP on N-Gram Language Models.

Step 1. The coding part of the assignment is contained in the notebook NLP_A2.ipynb. Note: you will be working with a portion of the Brown Corpus, which can be found in brown_100.txt. Results should be saved in a folder and named according to the notebook: namely, unigrams.txt, bigrams.txt, trigrams.txt, bigrams_smoothed.txt, and generated_bigrams.txt.

As part of n-gram modeling, you will use a technique called *smoothing*. This is noted in the code as `alpha value for additive smoothing`. In a language model, smoothing allows the model to generalize more effectively by adjusting the probabilities; specifically, unseen events receive a non-zero probability with smoothing. For the assignment, you should experiment with different alpha values to better understand how this process works.

Step 2. Answer the following questions in your group document. (total word limit 500)

1. Briefly describe and motivate how you implement your n-gram model.

2. Briefly describe and motivate how you implement smoothing based on the results you obtain from your n-gram model.

3. What is the effect of smoothing on your model, and how do different alpha values produce different results?

4. In the SLP chapter, *perplexity* is introduced as a method to evaluate n-gram language models. Define this measure in your own words. How would you implement this to evaluate your n-gram model? (Feel free to implement it instead of answering the second part of this question!)

A2.3 Word Vectors (35 points code; 15 points written)

“Words that occur in similar contexts tend to have similar meanings” and “You shall know a word by the company it keeps.” These are the central claims of the *distributional hypothesis*. A word vector is an attempt to capture this hypothesis mathematically. Word meanings are represented based on looking at the contexts a word can appear in a corpus and modeling similar words, often using an n-gram language model.

More precisely, word vectors represent words as multidimensional continuous floating point numbers where semantically similar words are mapped to proximate points in geometric space. In simpler terms, a word vector is a row of real-valued numbers (as opposed to dummy numbers) where each point captures a dimension of the word's meaning and where semantically similar words have similar vectors. Thus, words such as *wheel* and *engine* should have similar word vectors to the word *car* (because of the similarity of their meanings), whereas the word *banana* should be quite distant.

For additional reference, see the SLP chapter on Vector Semantics and Embeddings.

Step 1. The coding part of the assignment is contained in the notebook NLP_A2.ipynb. You will use your n-gram model for word modeling, so be sure to complete that part first.

In addition to code, the notebook has small questions to check for understanding. Please state your answers both in the notebook and your group document.

Step 2: Answer the following questions in your group document. (total word limit 500)

1. Theoretically and in practice, why is it helpful to represent words as vectors? Based on the results you obtained, can you anticipate cases where word vectors would not be helpful to determining a word’s meaning? List 1-2 specific examples.

2. Describe the two ways of representing words you implement in the notebook. What stands out when comparing these two methods in how they represent words?

3. The code provides simple examples that could be used to evaluate word vectors. Are these effective evaluations for the meaning of a word? Why or why not? Give at least one concrete example.

BONUS (up to 10 extra points)

In statistical NLP, we frequently make independence assumptions about relevant events which are not actually correct in reality. We are asking you to test the independence assumptions of unigram language models. Pointwise mutual information (PMI), is a measure of statistical dependence of the events $X_t = w_1$ and $X_{t+1} = w_2$; $C(w)$ is the absolute frequency and N is the size of the corpus. If the probability of the next word in the corpus being w_2 is unaffected by the probability of the previous word being w_1, then pmi(w_1, w_2) = 0; otherwise the PMI is positive or negative.

Pointwise mutual information,

$$\text{pmi}(w_1, w_2) = \log \frac{P(X_t = w_1, X_{t+1} = w_2)}{P(X_t = w_1) \cdot P(X_{t+1} = w_2)} \approx \log \frac{C(w_1 w_2) \cdot N}{C(w_1) \cdot C(w_2)},$$

Step 1. Calculate the PMI for all successive pairs (w_1, w_2) of words in the Brown corpus. Words (not word pairs!) that occur in the corpus less than 10 times should be ignored. List the 20 word pairs with the highest PMI value and the 20 word pairs with the lowest PMI value. Document and submit your code with observations as comments in pmi.py or .ipynb.

Step 2. Discuss the validity of the independence assumption for unigram models**.** Give 2-3 examples from your results to support your ideas. (word limit 200)

Step 3. Extend step 1 by researching and implementing both PMI and positive pointwise mutual information (PPMI) on the entire Brown corpus and brown100. Document and submit your code with observations as comments in the same file as step 1.

Step 4. How does PPMI differ from PMI, both generally and given specific examples from your results?