Sentence Generation

Myriam Kapon - aid24008 22/04/24

Dataset Choice

The selection process for the ten books aimed to exclude poetic, biblical, or otherwise abnormal texts, such as those containing numbered verses. Moreover, a diverse range of authors was chosen to ensure variations in vocabulary and speech patterns. The size of the vocabulary was 37997 unique words.

Data Structure

The data structure chosen to hold the n-grams is a tree, specifically a trie. (A trie, short for "retrieval tree" or "prefix tree," is a specialized tree data structure designed specifically for storing and efficiently retrieving keys, typically strings, in a manner that facilitates prefix-based searches and word combination generation). Programmatistically, this is the same as a normal tree, but the name gives it context. A trie stores keys by breaking them down into their constituent words or elements, with each node in the trie representing a single word of the key.

In the context of handling n-grams, a tree proves especially advantageous due to its ability to store and retrieve sequences of words efficiently. By organizing the tree such that each level corresponds to a specific word in the n-gram sequence, the structure inherently captures the relationships between successive words. For instance, the first level would consist of unigrams (single words), the second level would represent pairs of words (bigrams), and so forth.

During the construction of the tree, frequencies associated with each n-gram are also computed and stored at the corresponding nodes, effectively making this structure suited for sentence generation.

Visually, it looks like this:

```
                              root
                           /        \
              ("the", 4843)  ("a", 19391)…       (unigrams)
                    /                   /          \
      ("sun", 149)  ("house", 547) ("tree", 663)…    (bigrams) …
```

Tree vs Dictionary

Using a tree data structure for storing n-grams offers several advantages. Firstly, it allows for efficient storage of all n-grams simultaneously, eliminating the need to create separate structures for each n-gram length. This not only saves memory but also simplifies the organization and retrieval of n-grams.

Additionally, the time-consuming process of constructing the trie occurs only once during the initialization phase (Creating a tree of 6-grams takes 30 seconds). Once the tree is built, subsequent operations such as sentence generation become significantly faster. This contrasts with frequency dictionaries, where each operation requires iterating through the entire dataset to find matching n-grams. By front-loading the computational cost to the construction phase, trie-based methods offer faster performance during runtime operations.

Sentence Generation

The program examines the preceding n-1 words of a sentence and identifies all potential words that can follow. If no matches are found, it iterates backward to the previous n-gram length. For instance, if no tetragrams match, it searches for trigrams, followed by bigrams.

Starting n-gram

The fundamental concept behind sentence generation is to commence with a starting n-gram. For instance, if it's a bigram, the start sentence token <s> will be paired with another word; for a trigram, the start token will be combined with two words, and so forth. The program employs the "generate_next_word" function iteratively until the desired length is achieved. If the function fails to find any suitable words after a certain number of attempts, it randomly selects a word from the vocabulary.

Generate next word

The "generate_next_word" function operates on a sentence of length n-1, corresponding to the n-grams used for generating sentences. For instance, if the goal is to generate trigrams, the function takes as input a bigram (i.e., two words). It then traverses the tree structure recursively, utilizing the sentence as a pathway guide. If it encounters children nodes for the final word, it randomly selects one of them, weighted by their frequency.

- In the base case, when the sentence length is zero, indicating that the final word has been reached, the function checks if the word has any children. If so, it randomly selects one; otherwise, it returns None.

- In the general case, the function checks if the current word has the next word as a child. If it does, the function proceeds to the child node and repeats the process; otherwise, it returns None.

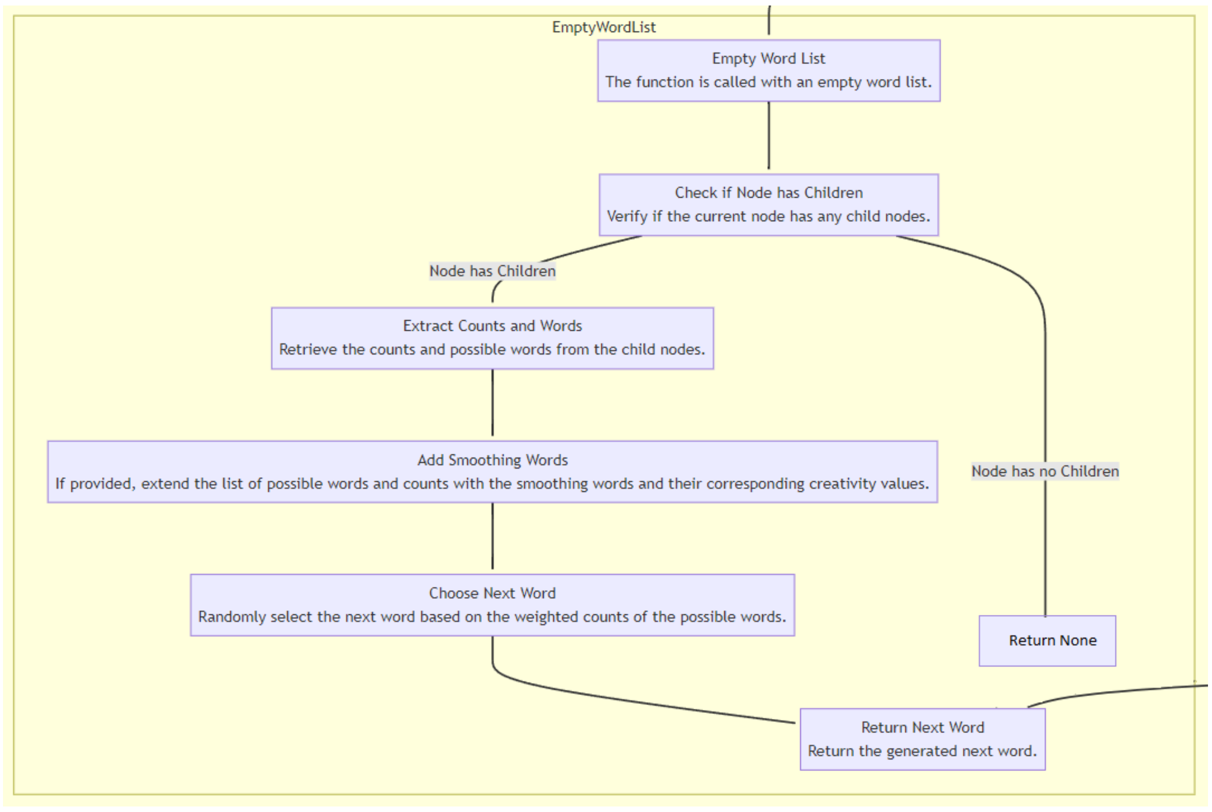
EmptyWordList
Empty Word List
The function is called with an empty word list.
Check if Node has Children
Verify if the current node has any child nodes.
Node has Children
Extract Counts and Words
Retrieve the counts and possible words from the child nodes.
Add Smoothing Words
If provided, extend the list of possible words and counts with the smoothing words and their corresponding creativity values.
Node has no Children
Choose Next Word
Randomly select the next word based on the weighted counts of the possible words.
Return None
Return Next Word
Return the generated next word.

Diagram 1: The Base Case

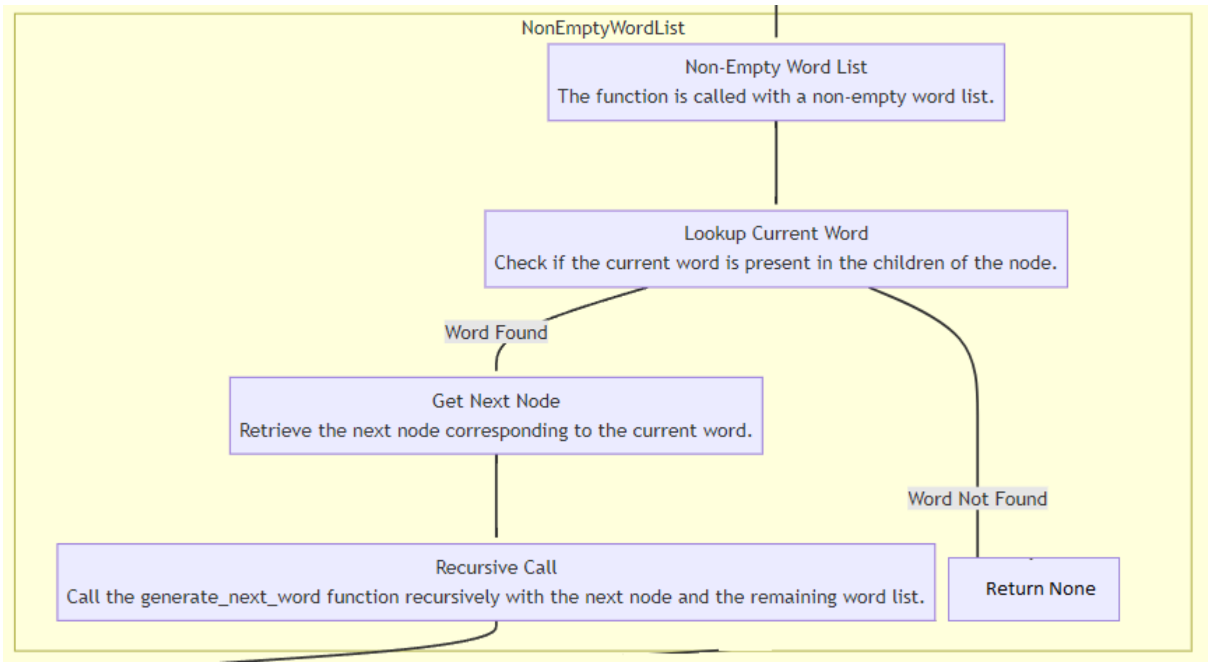
NonEmptyWordList
Non-Empty Word List
The function is called with a non-empty word list.
Lookup Current Word
Check if the current word is present in the children of the node.
Word Found
Get Next Node
Retrieve the next node corresponding to the current word.
Word Not Found
Recursive Call
Call the generate_next_word function recursively with the next node and the remaining word list.
Return None

Diagram 2: The General Case

Generate sentences

This function generates multiple sentences of a specified length. It repeatedly generates starting n-grams, then extends each sentence by using the “generate_next_word” function. It employs the mechanism where if no n-gram is found, the (n-1)-gram is searched. If nothing is found, it resorts to random selection from a given vocabulary or includes a default word ('and') to ensure continuation. Finally, it ensures uniqueness of generated sentences before returning them.

The reason for adding “and” is because it always makes sense in a sentence. However, sometimes it results in this badness:

- oh vestige deprecatory unloitering and hooked and dooks and media and root and tost and dearly and irradiate and

Smoothing

The model incorporates smoothing similar to Laplace smoothing. It builds a vocabulary of unique words and assigns them a frequency of 1 or less. When the "generate_next_word" function creates a list of possible word continuations, it appends this vocabulary. This approach prevents the tree from becoming excessively large, as not all words are made children of every other word. Since the frequency of the smoothed words is minimal, real children are much more likely to be chosen during random selection, as the weights are determined by frequency.

A parameter called "creativity" has been introduced to control the frequency of the smoothed words. Lower values result in a lower likelihood of smoothed words being chosen, while higher values increase the competition among words.

Sentence Examples

Sentences using 2-grams

1. The only way to move him was to depend.
2. His situation is an evil but you must give me a living.
3. The everlasting whip cord I declare.
4. He did not even move a hair and syme could come close enough to a whale before any pitchpoling comes into play.
5. Papa laughed.

Sentences using 3-grams

1. Churchill after being disliked at least years was now spoken of with compassionate allowances.
2. Repeated the doctor with a start but what on earth can he agent he thought mahanaim where he adam at the news with chilling gripe of sorrow stood that.
3. And where do you think he was in rashness leads not on.
4. After the farmer was dead the hosts of light.
5. My dear miss gregory said syme gently there are many kinds of sincerity and insincerity.

<u>Sentences using 4-grams</u>
1. Who will none shall from me withhold thy offered good distance from those she wants to be with but one can not comprehend a young being under such restraint.
2. Doated introductions jonah and the whale.
3. Here comes your beau nancy my cousin said day when she saw him crossing the street to the house.
4. No use sterning all then but as I was groping at innocent he went on eating his dinner in silence.
5. These were the ladies whom emma found herself very frequently able to collect and happy was she for her father sake in the power though as far as she.

<u>Sentences using 5-grams</u>
1. Alice had been looking over his shoulder with some curiosity.
2. Had he anything to tell the prince.
3. Woodhouse saw the letter and he says he never saw such a handsome letter in his life.
4. In places you see it bubblingly up like old wine worked anew.
5. Is there anything in our house can be of service to her.

<u>Sentences using 6-grams</u>
1. Grant when he heard of all this endeavoured to discover what could have offended his neighbour but all explanation was prevented by the obstinate silence of oakly.
2. So almost every hours when the watches of the night were set and the band on deck sentinelled the slumbers of the band below and when if a rope.
3. Ah here comes your champagne.
4. Unjust brittle fishiest grosser and weighs and exhibited that stump to an incredulous world.
5. Then yours was a really good school said the mock turtle in a deep hollow tone down both of you and do speak a word till I finished.

Thoughts
Sentence generation is indeed a fascinating area of study. Achieving realistic and meaningful sentences can be challenging because even a single misplaced word can disrupt the coherence of the entire sentence. Experimenting with larger datasets or different smoothing techniques, such as Kneser-Ney smoothing, could yield interesting results and potentially improve the quality of generated sentences.

Bonus: Text-to-Image Prompts

his sentence generation technique could be used to create prompts from stories that could be fed into a text-to-image model.
But because short stories don't include many words, the vocabulary is small and the n-grams not varied. For this reason, the generated sentences are exact copies of the original ones, since a word can have only 1 or 2 children. However, the sentences are meaningful and could be used to create story illustrations.

An example:

Paper Dragons - Story
“The knight stands opposite of the Dragon. It is a ferocious beast; Fifty times the size of a grown man, its body constantly moving, contorting in strange and unnatural ways. Pure white feathers decorate the creature, sharp like double-edged swords, ready to slice the flesh with the lightest touch. They move with the wind, each one dancing to its own rhythm, making the Dragon feel even more alive. The sight is mesmerizing, but the knight knows better than to look, for this Dragon is special. Its deadliest weapon isn't its needle-like teeth, or its sharp claws, but its black, abysmall eyes. They feel like bottomless pits, like the absense of eyes and everything else, they pierce through shiny armors and feed the emptiness in human souls until there's nothing left. On the ground, the bodies of better knights are scattered like flowers. Some alive, some dead, and some in-between. Lost souls, cursed to wander in places unknown. Their bodies break under the Dragon's legs as it arches its body backwards. It opens its huge mouth, and defeaning silence comes out. It's ready to attack. "I am a knight", the knight thinks. The knight's plain brown horse neighs in disagreement as they move into position. A pointy spear points at the beast's direction. Words have suddenly lost their meaning, and the air is too dense. Still, it seems like the whole battlefield is tingling with anticipation, waiting to see who will make the first mo- The Dragon charges violently forward, tearing everything in its way apart, its claws scratching the ground, leaving deep scars. It's wrong, very wrong, because dragons aren't supposed to move that fast, turn something into nothing this quickly. The world has been replaced by a void, and the void is being replaced by [ ]. The only path the knight can follow lies straight ahead. "I am a knight", the knight says. The two imbalanced bodies collide; for a moment, everything is still. A horse's scream breaks the silence, and a knight's one joins it. But the greatest scream is the one that never gets heard. The Dragon feels the spear leaving its body, just above the shoulder, feels the wound opening, allowing for the inside to become outside. Surprisingly, it's not blood that comes out, but ink, black as a night without stars. It stains the rich white feathers, and it's beautiful in a way, because it paints letters. "I am a knight", the knight writes. Paper dragons dislike words. It's angry now, preparing to attack again. The knight picks up the spear. And so the battle continues, a Dragon and a knight, until one prevails.”

<u>Sentences using 6-grams</u>
1. It ready to attack body and move and up and size and blood and letters and just and horse.
2. A pointy spear points at the beast direction.
3. Surprisingly it not blood that comes out but ink black as a night without stars.
4. Lost souls cursed to wander in places unknown.
5. I am a knight the knight says.