# Part 1

## 1.1 Assumption and Design

**Assumptions and Design Decisions**

Several assumptions and design decisions were made while developing this program:

- The model operates on character-level n-grams, meaning it considers sequences of 'n' characters as contexts for predicting the next character.
- A padding scheme using the '~' character is applied to maintain consistency in n-gram extraction at the beginning of the text.
- The vocabulary is dynamically updated based on input text, ensuring that only observed characters are considered.
- The probability of an unseen character in a given context defaults to a uniform probability over the total character occurrences in that context.
- The random text generation function builds output iteratively, ensuring that contexts evolve as characters are appended.
- The perplexity function follows a standard logarithmic approach to measure the model's predictive uncertainty, with references cited.

**Utility Functions**

- start_pad(c): Generates a padding string of length c to prepend to text before n-gram extraction.
- ngrams(c, text): Extracts n-grams from text as (context, character) pairs, where the context is a sequence of c preceding characters.
- create_ngram_model(model_class, path, c): Reads a text file, updates an n-gram model, and returns the trained model instance.

**N-Gram Model Implementation**

The NgramModel class is designed to support:

- **Initialization (__init__)**: Stores the context length, vocabulary, and a dictionary for n-gram counts.

- **Vocabulary Management (get_vocab)**: Retrieves the vocabulary of observed characters.
- **Model Update (update)**: Processes input text, extracting n-grams and updating frequency counts.
- **Probability Computation (prob)**: Computes the likelihood of a character appearing after a given context.
- **Random Character Sampling (random_char)**: Samples a character probabilistically based on context.
- **Text Generation (random_text)**: Produces synthetic text of a specified length based on learned n-gram probabilities.
- **Perplexity Calculation (perplexity)**: Evaluates how well the model predicts a given sequence, using a logarithmic approach.

## 1.2 Variation of `n`(context length)

With generation word length set at 250

n set at 1:

```
Fout han,
me t;
Thaghe atimad t lepan,
Sighar thid tino leve ve
PENYSeadlar e. ars sorerimys, owhathey bl de nccourp ith,
Ther tfr lly thengers tennghance h h, patono be, wagno?
Wid t isticu.
Falvo'gaturoma stifous m?
widur he.
```

n set at 2:

```
Fids, grou hand han, th,
Youns re ce resojany offend Far.

Let bleadenjoilee,
I we rad,
But trueend my lath lon pain wo eas timosolless ime on after how nothe ill coverever
stait all we hall eat nown loardion.
Whe unt;
How but halithat lon.

Thome va
```

n set at 3:

```
First king inst Lord!

IAGO:
Since:
Yet, and!
News fould stuck no sleep; but lord? I thee discopy dulo:
My here preven is so ress.

DUKE OF WORCESTER:

ANTONY:
Which one,
The Diant compleavel that lord the fight as the eyes raith 'em aloft, and.
Snou
```

n set at 4:

```
First Citizen:
If one to seen,
After for Lord of youth, be place shall too, so; we'll madam.

PANDARUS:
The man, sir,
Where that me we
are of thy magnant fair father
Have part her from humour late,
Make it wan fathere brough thee, and derived; he han
```

n set at 5:

```
First Gentleman:
'Tis our child.
Give up my face for thy father's loved me the would preciously that them.

APEMANTUS:
You haste, an youthful man
should be of
knave; I cannot broke? will shown in the began to the last, as our daughters, now of either
```

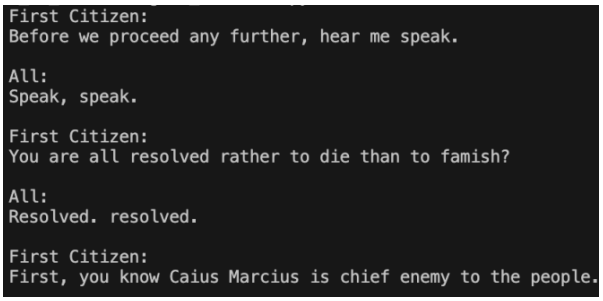
n set at 10:

```
First Citizen:
You must not think his father's life.
Call it not be so long; this
wrestler shall cleave to. What's the matter to mine answer.

PRINCE HENRY:
Very true, sir, the slip; can you loved not at home;
and I pray God his bad
voice bode no mis
```

n set at 20:

```
First Citizen:
Before we proceed any further, hear me speak.

All:
Speak, speak.

First Citizen:
You are all resolved rather to die than to famish?

All:
Resolved. resolved.

First Citizen:
First, you know Caius Marcius is chief enemy to the people.
```

n set at 30:




n = 1: The generated text is highly random and lacks structure, as it selects characters based only on their overall frequency without considering prior context. Words and sentences are mostly nonsensical.

n = 2: Some common letter combinations start appearing, but the text still remains largely unintelligible. The model lacks sufficient context to form real words consistently.

n = 3: Recognizable words begin to emerge, but sentence structure remains weak. The model captures some short patterns in the text but still generates a mix of gibberish and readable words.

n = 4: Words are more structured, and some short meaningful phrases appear. However, sentence fluency is inconsistent.

n = 5: There is a noticeable improvement in readability. Words are mostly well-formed, and sentence fragments start making sense, though the overall structure is still imperfect.

n = 10: The text becomes significantly more readable, with many real words appearing in proper context. Sentence coherence improves, but occasional inconsistencies remain.

n = 20: The generated text closely resembles the training corpus, with well-formed words and mostly understandable sentences. Some grammatical inconsistencies may still occur.

n = 30: The text is almost indistinguishable from the original corpus, with high fluency and well-formed sentences. The model has memorized large portions of the training data, resulting in near-perfect replication of learned sequences.

These results highlight the trade-off between n and generalization. Smaller n values lead to more randomness, while larger n values increase coherence but also risk overfitting to the training data.

## 1.3 Perplexity values for different character-level

| n-gram / Dataset | 3 | 4 | 7 |
|---|---|---|---|
| Nytimes article | 8.41 | 6.84 | 10.91 |
| Shakespeare sonnets | 5.50 | 4.74 | 5.71 |

## 1.4 Discussion

The perplexity values across different test documents indicate how well the n-gram model predicts text sequences in different contexts.

1. **Comparison Across Test Documents**:

   New York Times Article: Higher perplexity suggest that the model struggles more with this dataset. This could be due to more diverse vocabulary, complex sentence structures, or topic variations.

   Shakespeare Sonnets: Lower perplexity values suggest better predictive performance. Shakespearean language might have more repetitive or structured phrasing, which n-grams can capture effectively.

2. **Impact of N-Value on Perplexity**:

   For both datasets, n=4 has the lowest perplexity, indicating it provides a good balance between context size and model generalization. When n=3, the context is shorter, leading to more ambiguity and higher perplexity. When n=7, perplexity increases again, possibly because the model overfits to specific character sequences, making it less generalizable.

3. **Best Performing N-Gram**:

   n=4 performs the best overall, as it provides sufficient context without making the model too sparse or too rigid. The higher perplexity for n=7 suggests that using too long of a context window may lead to poor generalization.

# Part 2

## 2.1 Setting for top-k sampling

With 5 shot testing generate from the beginning

Top-k set at 1:

```
[{'generated_text': 'o the sea of the sea of the sea of the sea, and the sea, and the sea, and the sea, and there......'},
 {'generated_text': 'o the sea of the sea of the sea of the sea, and the sea, and the sea, and the sea, and there......'},
 {'generated_text': 'o the sea of the sea of the sea of the sea, and the sea, and the sea, and the sea, and there......'},
 {'generated_text': 'o the sea of the sea of the sea of the sea, and the sea, and the sea, and the sea, and there......'},
 {'generated_text': 'o the sea of the sea of the sea of the sea, and the sea, and the sea, and the sea, and there......'}]
```

Top-k set at 2:

```
[{'generated_text': 'hat shall see your sight we have some part on me? What is they? I would seen?? you? you? you? you?'},
 {'generated_text': 'or the sense of truth, and so much as I shall never have. I say, and as I a say... I say. I.on..;.'},
 {'generated_text': "or sometimes the sea of the world o' the world, the sense of them, and as the so.e, and, as,.ense,"},
 {'generated_text': 'o set him to the world. I have seen thee, and there is no more. I have. I said, he. he. is. and. h'},
 {'generated_text': 'he world and she was a soldier of the world. I have a servant. I am and. I had. I am.nd, and. I.. '}]
```

Top-k set at 3:

```
[{'generated_text': "ore than he shall be a banking, and thanks, and thanks! he shall not? here. I here. I'll sheear.'s"},
 {'generated_text': "o tale of his sister and he is not answer. Therefore I'll be talk, take. I'll, he wit. you.h....' "},
 {'generated_text': 'et this this to the state that they are asket of a money. If they are. I were. I say, tou, I and. '},
 {'generated_text': 'hen I have been an arms at her father. This is a mighty. I have a heard, at,it.e.,., I have at..,i'},
 {'generated_text': 'he shall hear the sea of this sea. Here, so I should not. I should. I see, it. it. is, I it. I.his'}]
```

Top-k set at 5:

```
[{'generated_text': 'hen thyself were as much for my brother. I will not be so. I will, I were in.wer.,ed. I.o. I that.'},
 {'generated_text': "our sister shall not held to be such a space. I say 'Ay.' What?'s there? I I'll not?'d?'?' there? "},
 {'generated_text': "he poor crys in the stamp of my sword: and I am, at him. Welcome, take; I there:, I take.'s.'t.'.'"},
 {'generated_text': "his water, who will be such a son town, and shepherd: and he's away, and honours,. to me.; and, an"},
 {'generated_text': "he most stick'd was and mighty brother, and there's a speech.' together.,:, and there!'s most!'d?'"}]
```

Top-k set at 10:

```
[{'generated_text': ' humbly and presently. She will suddenly tender him, shere. I pray thee,,; she will.; not, sheee..'},
 {'generated_text': "hy wind our captains. As if you should not have they seen your survey, them. your;folk... I's: if "},
 {'generated_text': "ashions your pregnand foe, but I see thee: you are not. I will.'d you know,.'d.' your hear,'s.'?? "},
 {'generated_text': 'e may, my lord. I cannot have laid him well at this? we may. What say?, I do?no????, I I hear?. I '},
 {'generated_text': 'ere, my most life. What you shall make her? how it thee! lord? If I, it. I, you and, it? I your li'}]
```

# 2.2 Comparison

### GPT-2 Generated Output

The GPT-2 model generates text that, while sometimes repetitive, forms more coherent sentences that resemble natural language.

Examples:

> *"the world and she was a soldier of the world. I have a servant. I am and. I had. I am and, I..."*
>
> *"then I have been an ass at her father. This is a mighty. I have a heard, at, withe, I have at..."*

Despite occasional repetition and awkward phrasing, most words are recognizable, and sentences follow a grammatical structure.

**Character n-gram Generated Output**

The text generated by the character n-gram model appears more random and less structured with a short context length settings, often resulting in incoherent words and phrases, but mitigate when the context length increase

Examples:

*"Fout han, Thage aised t lepan, Sphdar tiad time voe"*

*"Ne discopy dulox; My here preven is so ress."*

These outputs contain more gibberish and meaningless word formations compared to GPT-2, suggesting that the character n-gram model struggles to maintain linguistic rules in a short context length but can achieve a better result with enough context length