Hangman Solver
Using N-gram Approach

Suvrat Pal

Indian Institute of Technology Kanpur

21-01-2024

Contents

1 Introduction

The game of Hangman has long been a captivating challenge, both linguistically and computationally, attracting interest across various domains. Recently, as part of the selection process for a Global Alpha Researcher Internship at TrexQuant, I undertook the task of developing an algorithm that could outperform a base model in the intricate game of Hangman. The primary goal was to design an algorithm capable of achieving an accuracy rate exceeding 50%. This report provides a thorough analysis of the development process, methodology, cross validation and the results obtained in creating an advanced Hangman-solving algorithm.

My approach to this challenge centered around leveraging patterns inherent in English words, employing conditional probabilities, and incorporating statistical techniques to construct a sophisticated model. The key objective was to enhance the algorithm's ability to guess correct letters more accurately. Through the amalgamation of these strategies and a meticulous evaluation process involving cross-validation, I aimed to demonstrate the effectiveness of the developed approach.

The core of my Hangman-solving algorithm lies in the utilization of an n-gram approach, a technique that considers sequences of n consecutive letters in words. This method aims to capture and exploit the inherent linguistic structures present in English words, ultimately enhancing the algorithm's predictive capabilities. The subsequent sections of this report delve into the details of the development process, the rationale behind the chosen approach, and the outcomes obtained through rigorous evaluation.

2 Intuition

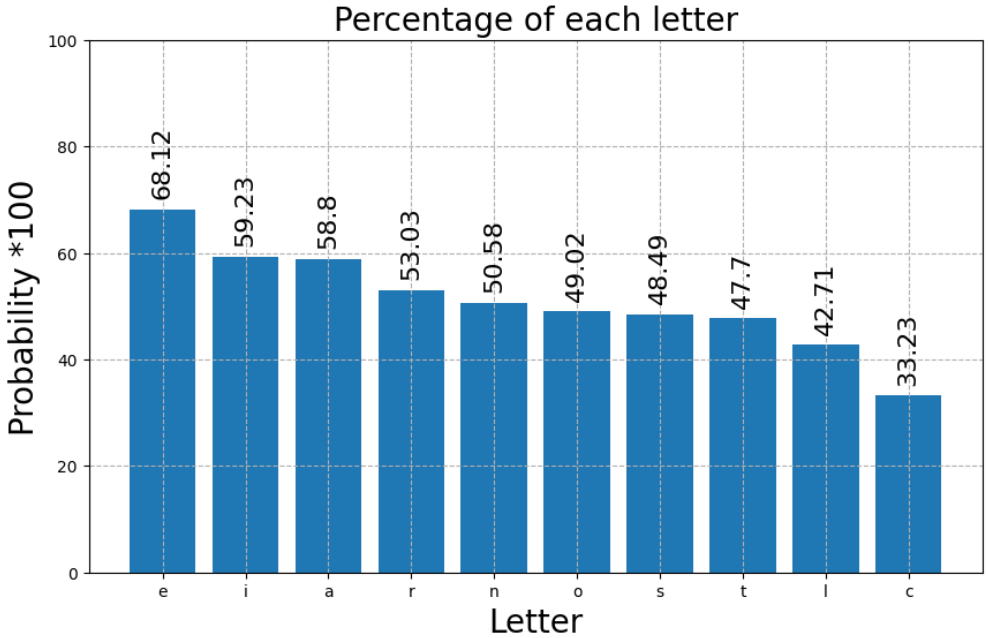
The Hangman-solving algorithm I developed relies on an N-gram approach, drawing inspiration from the inherent linguistic structures present in English words. The primary intuition behind the N-gram model is to consider sequences of N consecutive letters in words, thereby capturing meaningful patterns and correlations between adjacent letters.

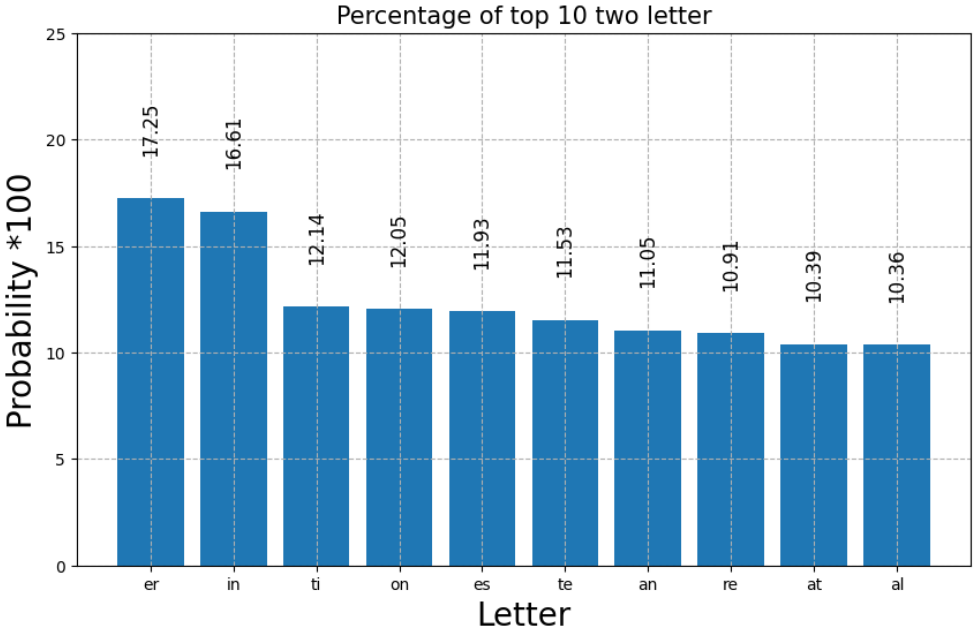
In the context of Hangman, this approach allows the algorithm to exploit the observed correlations and dependencies between letters within a word. By examining not just individual letters but also the contextual relationships between adjacent letters, the algorithm gains a more nuanced understanding of the language patterns embedded in the word to be guessed.

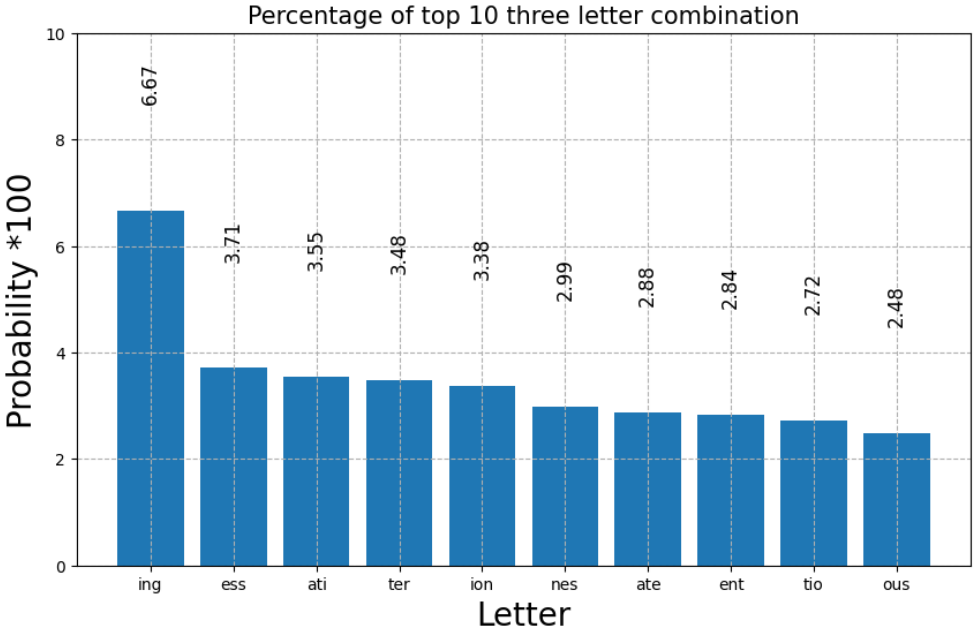
The key assumption driving this approach is that certain letter sequences commonly co-occur in English words, and this knowledge can be harnessed to improve the accuracy of letter predictions in the game. For instance, the combination "QU" is frequently observed, and understanding this pairing can guide more informed guesses.

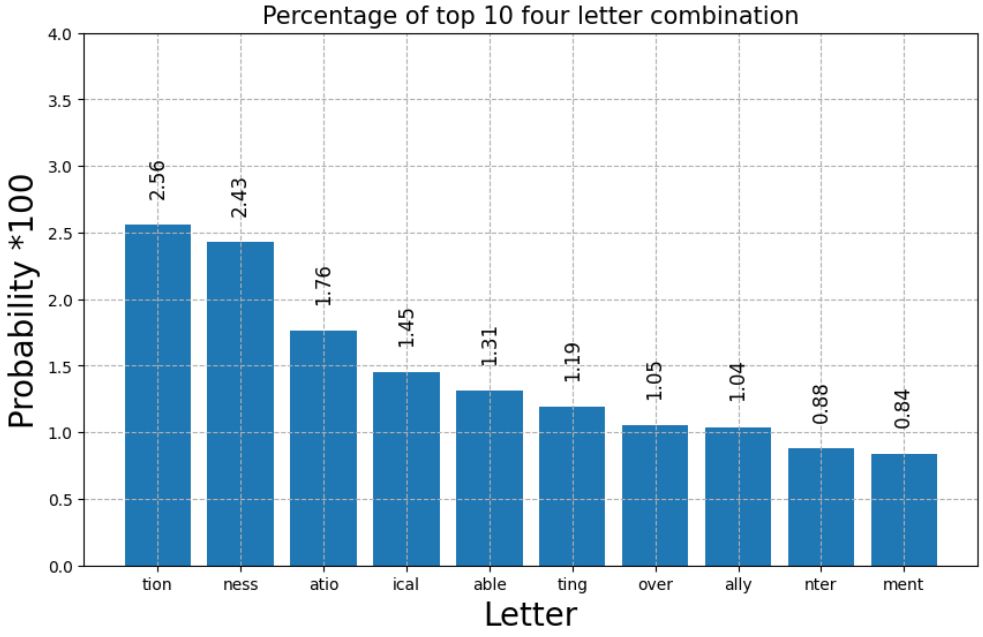
Moreover, the N-gram model inherently captures the idea that letters in a word are not independent of each other. By examining sequences of letters, the algorithm adapts to the linguistic nuances of the English language, recognizing common prefixes, suffixes, and letter combinations that can significantly impact the likelihood of certain letters appearing in a particular position within the word.

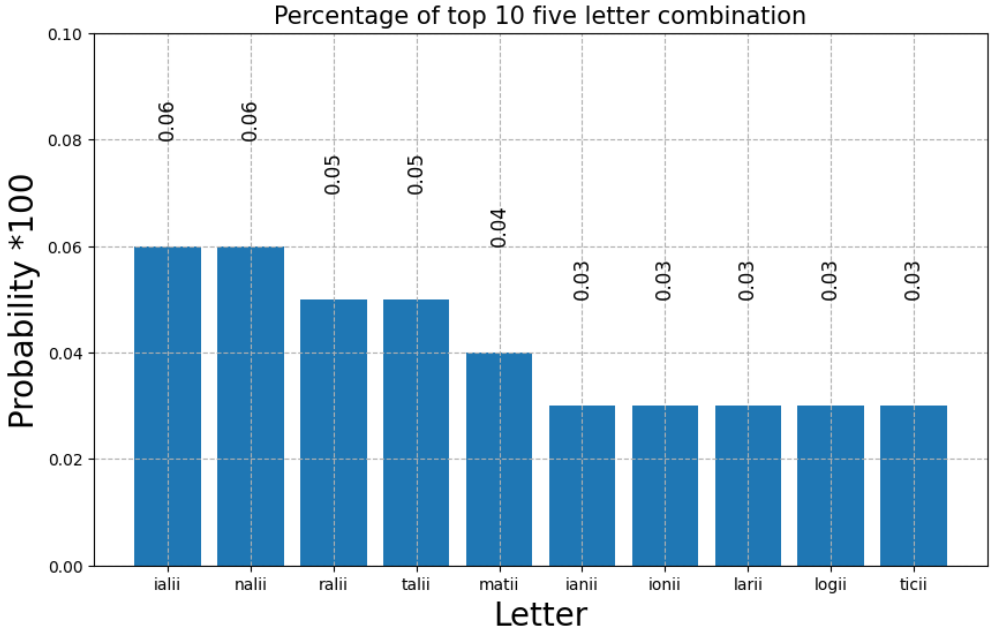
In summary, the N-gram-based Hangman-solving algorithm is rooted in the concept that the arrangement of letters in a word provides valuable information. By considering the sequential relationships between letters, the algorithm aims to unravel the underlying linguistic patterns, enhancing its ability to predict and guess letters more accurately in the game of Hangman.




Percentage of top 10 two letter
Probability *100
Letter
0
5
10
15
20
25
17.25
16.61
12.14
12.05
11.93
11.53
11.05
10.91
10.39
10.36
er
in
ti
on
es
te
an
re
at
al


Percentage of top 10 three letter combination
Probability *100
Letter
0
2
4
6
8
10
6.67
3.71
3.55
3.48
3.38
2.99
2.88
2.84
2.72
2.48
ing
ess
ati
ter
ion
nes
ate
ent
tio
ous


Percentage of top 10 four letter combination
Probability *100
Letter
tion
ness
atio
ical
able
ting
over
ally
nter
ment
2.56
2.43
1.76
1.45
1.31
1.19
1.05
1.04
0.88
0.84


Percentage of top 10 five letter combination
Probability *100
Letter
ialii
nalii
ralii
talii
matii
ianii
ionii
larii
logii
ticii
0.06
0.06
0.05
0.05
0.04
0.03
0.03
0.03
0.03
0.03

3 Methodology

3.1 *Data Preprocessing:*

- **Dictionary Selection:** Utilize the provided challenge dictionary as the primary source of English words for the Hangman-solving algorithm.
- **Dummy Letters Addition:** Append dummy letters (e.g., { and |) to the beginning and end of each word in the dictionary to account for letter combinations occurring at word boundaries.

3.2 *N-gram Model Construction:*

- **N-gram Selection:** Choose an appropriate value for N based on empirical analysis and considerations of word length. A balance must be struck between capturing meaningful patterns and avoiding excessive computational complexity.
- **N-gram Generation:** Created a dataset of N-grams by extracting all possible sequences of N consecutive letters from each word in the preprocessed dictionary.
- **Frequency Calculation:** Calculate the frequency of each N-gram to identify common and relevant letter sequences in the dataset.

3.3 *Pattern Recognition and Letter Prediction:*

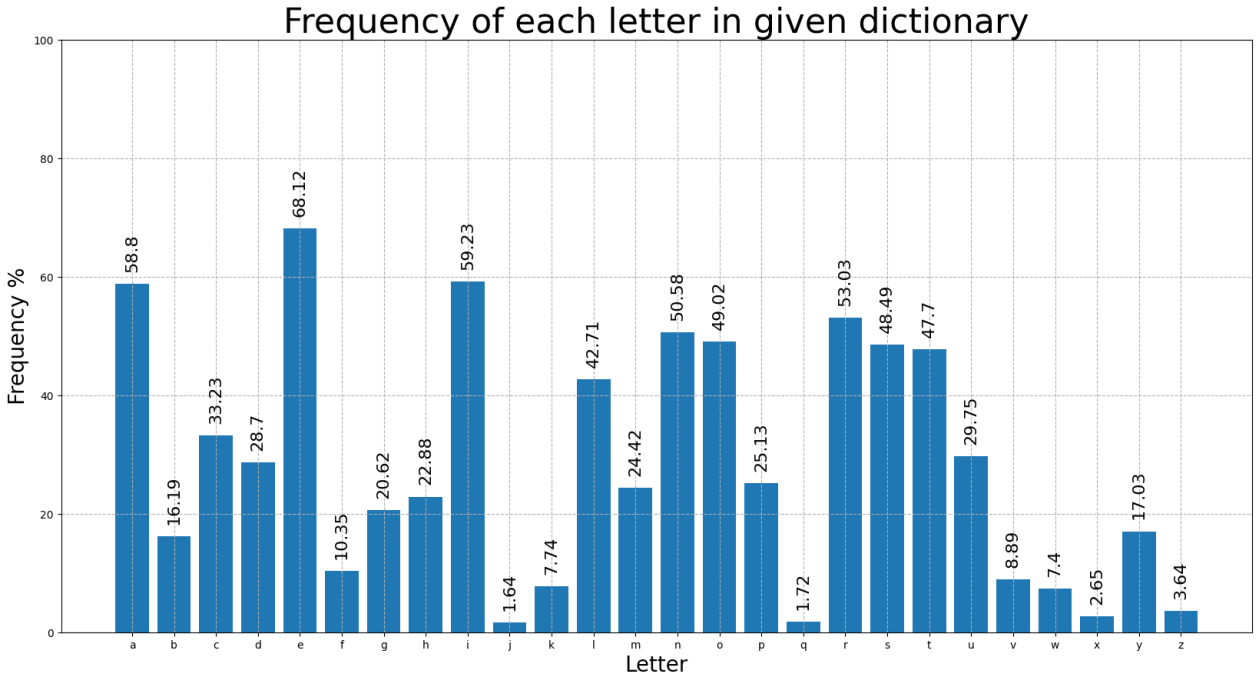
- **Contextual Analysis:** Analyze the context in which specific letters appear by considering their adjacent N-grams. Identify patterns, common prefixes, and suffixes that may guide the guessing algorithm.
  Patterns which are considered in this model:
  - Length 1:
    1. (_)
  - Length 2:
    1. a(_)
    2. (_)a
  - Length 3:
    1. ab(_)
    2. a(_)b
    3. (_)ab
  - Length 4:
    1. abc(_)
    2. ab(_)c
    3. a(_)bc

4. (_)abc

– Length 5:

1. abcd(_)
2. abc(_)d
3. ab(_)cd
4. a(_)bcd
5. (_)abcd

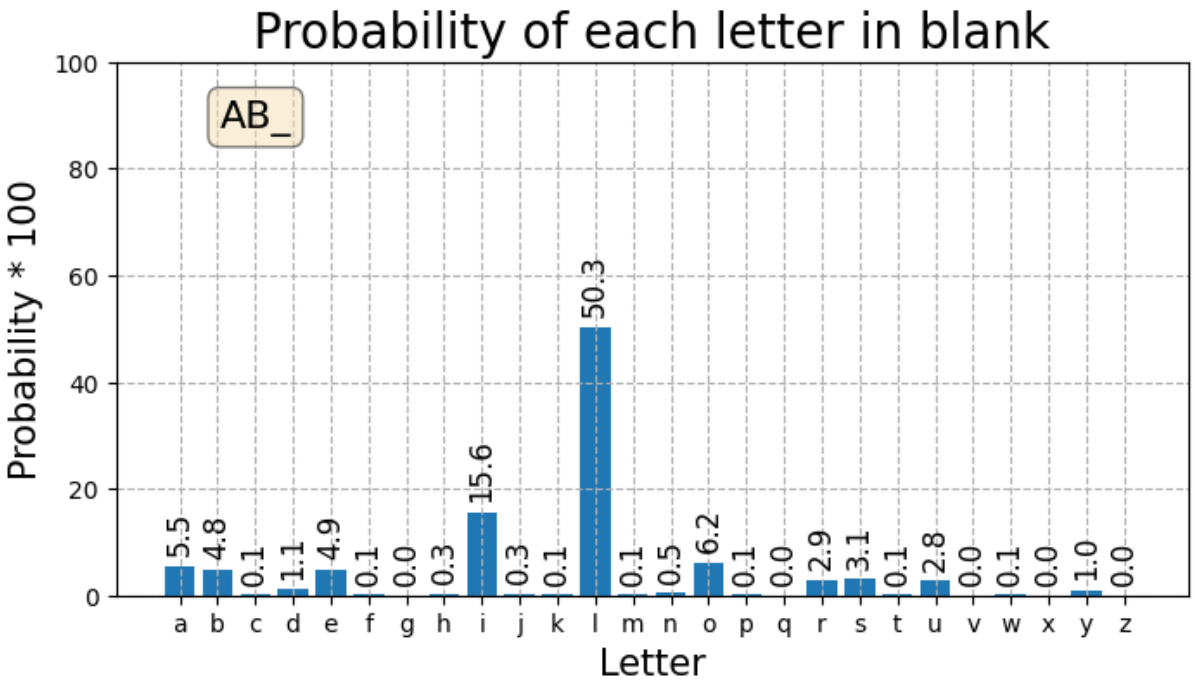
- **Conditional Probabilities:** Calculate conditional probabilities of letters based on the observed N-grams, considering the likelihood of a letter given its preceding N-gram.



3.4 *Prediction and Evaluation:*

- **Letter Prediction:** For each guessing round, predicted the most probable letter based on the learned N-gram patterns and conditional probabilities.
- **Evaluation Metrics:** Assess the algorithm's performance using accuracy. Employed cross-validation to ensure robustness and generalizability.

This methodology outlines a systematic process for developing, implementing, and refining an N-gram-based algorithm for playing Hangman, focusing on leveraging sequential letter patterns for accurate word predictions.


Probability of each letter in blank
AB_
Probability * 100
Letter
100
80
60
40
20
0
a b c d e f g h i j k l m n o p q r s t u v w x y z
5.5
4.8
0.1
1.1
4.9
0.1
0.0
0.3
15.6
0.3
0.1
50.3
0.1
0.5
6.2
0.1
0.0
2.9
3.1
0.1
2.8
0.0
0.1
0.0
1.0
0.0

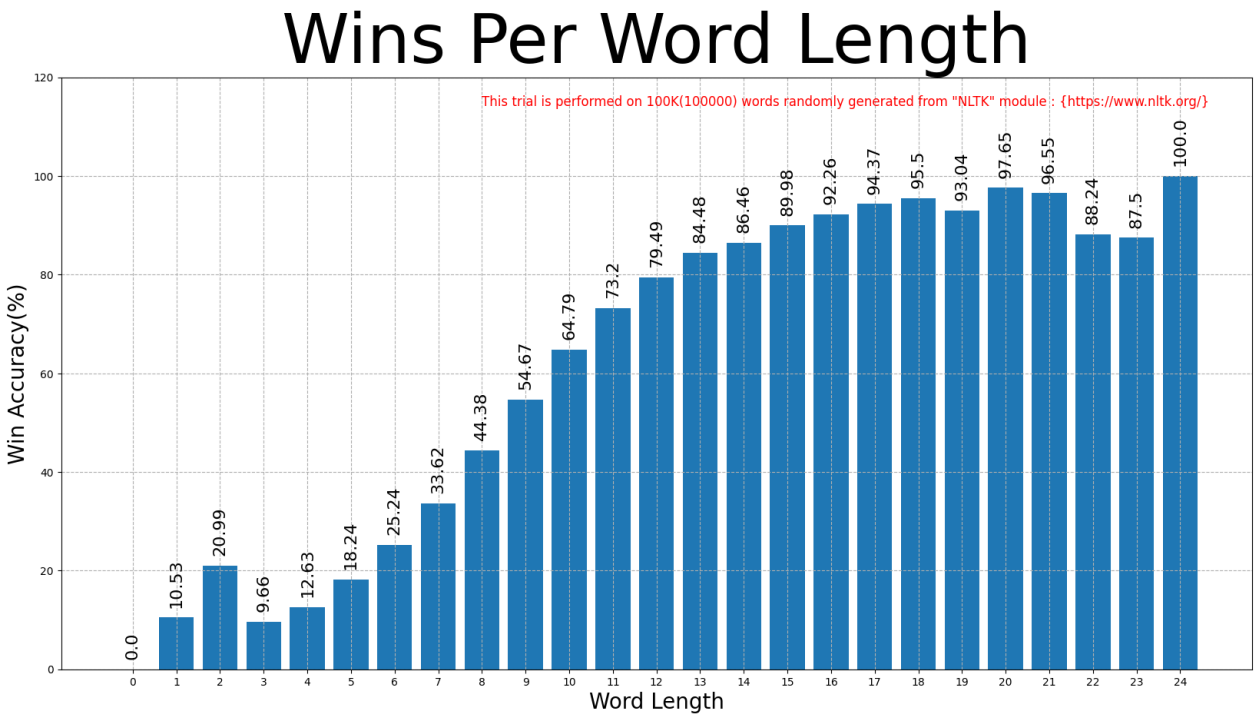
4 Cross Validation

To rigorously assess the performance and generalization capabilities of the Hangman Solver, a comprehensive cross-validation procedure was employed on 100K randomly generated words.

The accuracy of the algorithm, denoted by random variable X, was evaluated through an extensive experiment involving 1000 sets, each set comprising 100 Hangman games. This rigorous process aimed to generate a representative distribution of X through repeated sampling.

4.1 Experimental Design

1. *Random Word Selection :*
    - A subset of 100K words was randomly selected from the NLTK Module.
2. *Sampling Accuracy :*
    - The Hangman algorithm was tested on the set of 100 randomly generated words.
    - This testing process was repeated 1000 times, each time randomly different set of 100 words.
    - This results in a collection of samples denoted as X1, X2,X3,.....X1000

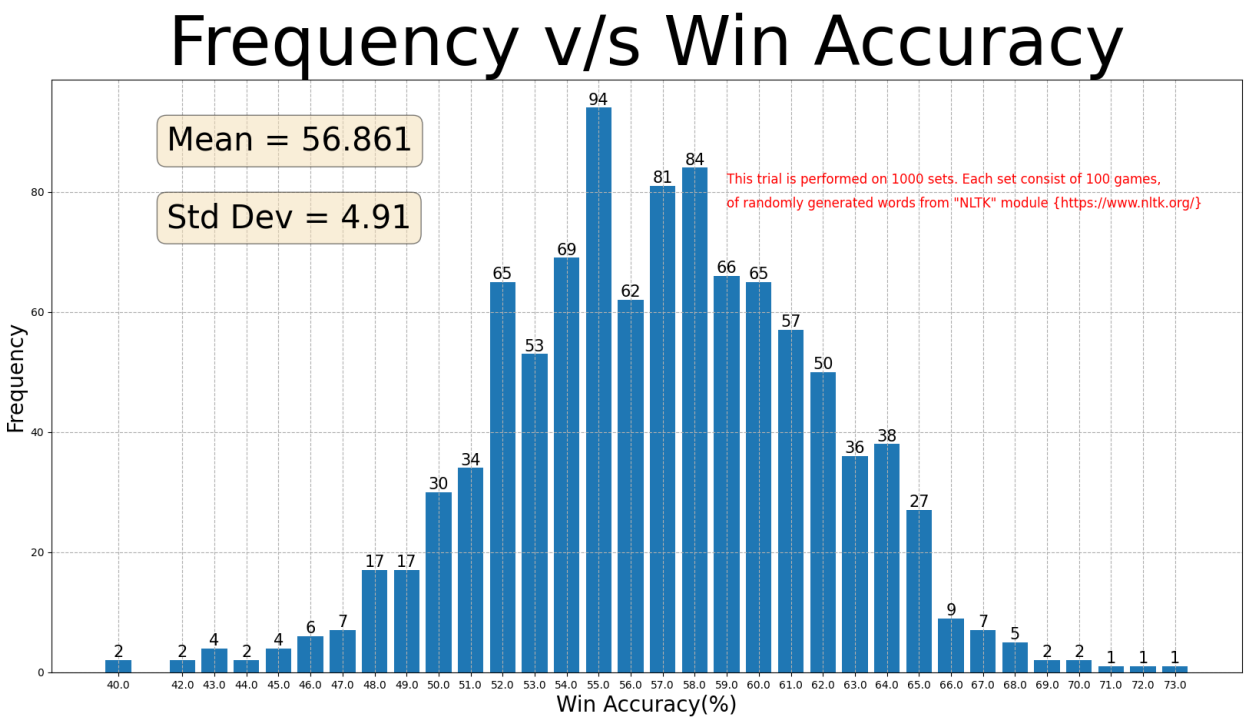


4.2 Statistical Analysis

- Statistical analysis was conducted on the cross-validation results to assess the consistency and variability of the Hangman Solver's performance.
- This analysis provides insights into the solver's robustness and its ability to generalize to unseen words.

4.3 Interpretation :

- **Average accuacy** for 1000 sets is **56.86**1 with a **Standard deviation** of **4.91**
- The variation of Accuracy with words of different length in shown in Figure8.
- The frequency of accuracy on 1000 sets in shown in Figure9.



5 Conclusion

In this Hangman game simulation and analysis, we implemented a strategic letter-guessing algorithm based on n-gram frequencies in a given dictionary of words. The Hangman class was designed to utilize unigrams, bigrams, trigrams, tetragrams, and pentagrams to calculate the conditional probability of each letter in a word subset with blanks. The algorithm successfully demonstrated an ability to make informed guesses, adapting its strategy based on the evolving state of the guessed word.

I conducted extensive simulations on a dataset comprising 100,000 randomly generated words from the NLTK module. The performance was evaluated through a series of 1000 games, each consisting of 100 randomly selected words. The results were analyzed to understand the algorithm's effectiveness in different word-length scenarios.

Key findings from the simulations include:

- The algorithm exhibited varying levels of accuracy, achieving higher success rates on higher word lengths.
- The analysis of win accuracy per word length revealed patterns in the algorithm's performance.
- The frequency distribution of win accuracies provided insights into the overall success rates across multiple game sets.

These observations indicate the algorithm's potential for strategic letter guessing in the Hangman game. Further fine-tuning and optimization could enhance its performance, potentially making it a valuable tool for word-guessing applications.

Overall, the project provides a foundation for exploring and refining probabilistic strategies in word-guessing games, opening avenues for future research and improvements in artificial intelligence-based gameplay.

6 Appendix

1. Source Code for Hangman Computer Solver :
2. Dictionary used to train model
3. Report
4. Analysis Files
5. Cross validation
    - Source Code for Cross Validation
    - My Random Word
    - Random Words Generator Code through NLTK