# Developing a Python 2-gram Language Model Plan of Work Report Introduction

The goal of this project is to improve the language model utilized by a speech recognition program that a client uses to transcribe books. The language model currently being employed is inappropriate, and the existing voice recognition technology is not adequate for this use, according to preliminary research. As a result, we will create a brand-new Python software from scratch, trained on a text dataset supplied by the client, for predicting a 2-gram language model. The software must be capable of predicting the likelihood of the subsequent word given the preceding word and measuring its effectiveness on a test set.

## Criteria for the Program

The program must adhere to the following standards:

- Calculate the M1 2-gram language model.
- Use the customer's book text to train the language model.
- Using the language model, determine the likelihood of the subsequent word given the preceding word.
- Analyze the language model's performance on a test set.
- and assess how well it performs on a test set.

## Design and Preprocessing of Data

The following design and data preprocessing procedures will be followed:

1. Specify the program's prerequisites for estimating the M1 2-gram language model.
2. Choose the most performant data structures, then create a basic code foundation. The design choices will be supported by evidence.
3. Apply data preparation to the sets of data. Think about the type of preprocessing required for the creation of a language model.
4. Stretch objective: FST for text data preparation.
5. Stretch goal: Create a design that accounts for the potential future use of a language model M2 with back-off.

## Stretch Objectives

FST for text data preprocessing

A mathematical model known as a finite state transducer (FST) converts a stream of input symbols into a stream of output symbols. An FST can be used for text preparation tasks including stemming, lemmatization, and entity recognition in the context of natural language processing.

In this project, we'll investigate how to preprocess text data using FST before training a language model. We will research various FST libraries, including OpenFST and PyFST, and test out various FST stemming and lemmatization methods. We will assess how FST-based preprocessing affects the functionality of the language model and contrast it with conventional preprocessing methods.

## Back-off language model M2

The likelihood of higher-order n-grams is estimated using lower-order n-grams in a back-off language model, a form of n-gram language model. In other words, the model "backs off" to the lower-order n-grams to estimate the likelihood of a higher-order n-gram if it has a zero count.

In this project, we'll create the code infrastructure needed to support the eventual implementation of the M2 back-off language paradigm. The M1 language model class's data structures and methods will be changed in order to include the back-off mechanism. To enhance the performance of the back-off model, we will also study several back-off strategies, including Katz, Jelinek-Mercer, and absolute discounting, and test out various smoothing techniques. We will assess the back-off model's effects on the voice recognition system's performance and contrast it with the M1 language model.

## Conclusion

We have described the design and data pretreatment processes for creating a 2-gram language model in Python in this plan of work report. Also, we have specified the stretch goals, chosen the right data structures, and defined the program's requirements. The program will then be put into use, and its performance will be assessed using the test data that the customer has provided.