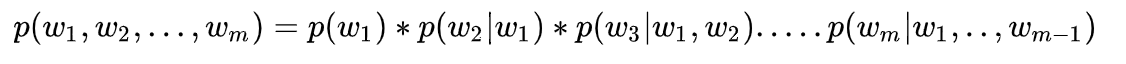
**N-gram Model**

The N-gram model is based on the assumption that the occurrence of the NTH word is only related to the preceding N-1 word and it is not related to any other word. The probability of the whole sentence is the product of the probabilities of the occurrence of each word. These probabilities can be obtained by directly counting the number of simultaneous occurrences of N words in the corpus. Commonly used are binary bi-gram and ternary trigram. The N-gram model is used to evaluate the reasonableness of statements. If we have a sequence of M words (or a sentence) and want to calculate the probability, according to the chain rule: It has the below formula:

(1)

By simplifying the calculation with the Markov chain assumption that the current word is only related to a finite number of previous words and does not have to go back to the beginning of the word, the length of the above calculation can be greatly reduced. we have the below formula:

A close up of a logo

Description automatically generated (2)

After Bayes process, the following formula will be used for mathematical calculations:

 (3)

**N-gram Code Implementation (Python)**

**Corpus preparation**

N-gram analyzes the reasonableness of user input by comparing corpus, so corpus should be prepared before implementing n-gram model. In this paper, three English novels are selected as corpus. At the same time, in order to speed up the analysis, some data preprocessing was carried out on the corpus, including the segmentation of language segments in novels by line and marking with <content> and </content>. Figure 1 shows the statement format of the corpus.

A close up of a newspaper

Description automatically generated

**Figure 1.** The statement format of the corpus

After the corpus is ready, the program first defines a function that reads the corpus as ‘read\_data’

It is shown in Figure 2. Note that regular expressions are used to get each statement. The statements read from the corpus are saved as a list and returned to the main program.

A screenshot of a cell phone

Description automatically generated

**Figure 2.** The corpus read function

**Establishment of n-gram statistical model**

Based on the read function, the program can already obtain data from the corpus and analyze it according to the statement entered by the user and the expected N-gram(Unigram, Bigram,trigram). The algorithm idea is based on formula 3 above. For example, the user enters' I want to go school. ' This function proceeds to edit the list of the corpus. Operations include marking statements with <s> and </s> and separating statements with Spaces. If the user asks for A bigram, the first Bigram is processed by counting all the '(I, want)' in the corpus list as A, and then counting the '(I)' number as B, so the reasonability rate of '(I want)' is P and P = A/B. It should be noted that the corpus used is considered infinite in this paper. Any gram that does not appear in the corpus will be considered a spelling error and will be analysed by editing distance (Mentioned in the edit distance algorithm). Figure 3 shows the statistical model function of n-gram.

A screenshot of a social media post

Description automatically generated

**Figure 3.** N-gram statistical model function

**Precision improvement function**

In the test, it was found that the algorithm in this paper had a flaw in processing ngram model calculation. For example, '(to,want)' did not exist in the corpus list, but '(to,want.)' did. Therefore, a function is designed to improve the accuracy. This function operates with a Python tuple, and when the N-gram statistical model detects that the current gram does not exist, it will call this function named 'tuple\_edit' to modify the current gram. If the modified gram exists in the corpus list, it will be used to replace the original gram for calculation. Figure 4 shows this function.

A screenshot of a cell phone

Description automatically generated

**Figure 4.** N-gram edit function

**N-gram probability function**

It will process the probability of each ngram and store it as a list after N-gram statistical model calculation. Figure 5 shows its function code.

**A screenshot of a cell phone

Description automatically generated**

**Figure 5.** N-gram probability function

**Statement rationality calculation**

Calculate the reasonableness of the sentence according to the probability list of each ngram. Figure 6 is a statistical function of it.

A close up of a logo

Description automatically generated

**Figure 6.** Statement rationality calculation function

**None gram check**

For a ngram that is not in the corpus, its index will be marked so that the edit distance function can be called to replace it, and thus probability statistics will be performed here. Figure 7 shows the index lookup method.

A screenshot of a cell phone

Description automatically generated

**Figure 7.** None gram check function

**Appendix: Complete code**

<https://github.com/YoloForDream/NLP_ngram_model/blob/master/n-gram-model.py>