```
import nltk
from nltk.util importw ngrams
from nltk.metrics.distance import edit_distance
from nltk.corpus import words
from nltk.corpus import reuters
nltk.download('punkt')
nltk.download('words')
nltk.download("reuters")
def train_ngram_model(n):
    words = reuters.words(categories="trade")
    ngrams_model = list(ngrams(words, n))
    return ngrams_model
def next_word_prediction(context, ngrams_model, n, candidates):
    context_tokens = nltk.word_tokenize(context)
    context_ngram = tuple(context_tokens[-n:])

    next_word_candidates = candidates

    predicted_word = max(set(next_word_candidates), key=next_word_candidates.count)

    return predicted_word
unigram_model = train_ngram_model(1)
bigram_model = train_ngram_model(2)
trigram_model = train_ngram_model(3)
input_text = "Mr Patrick is our new"
predicted_word = next_word_prediction(input_text, unigram_model, 2, ['principal', 'principle'])
print(input_text+" "+predicted_word)
input_text = "The company all the terms."
predicted_word = next_word_prediction(input_text, unigram_model, 2, ['expected', 'accepted'])
print(input_text+" "+predicted_word)
input_text = "Please don’t keep your dog on the"
```

```
predicted_word = next_word_prediction(input_text, unigram_model, 2, ['lose', 'loose'])
print(input_text+" "+predicted_word)
```