```
from collections import defaultdict
import numpy as np


def train_translation_prob(sentence_pairs, num_iterations):
    # Step 1: Initialize translation probabilities uniformly
    t = {}
    for (e, f) in sentence_pairs:
        for word_e in e:
            for word_f in f:
                t[(word_e, word_f)] = 1.0 / len(e)

    # Step 2: Training iterations
    for iteration in range(num_iterations):
        print(f"Iteration {iteration + 1}:")

        # Step 4: Initialize count(e|f) and total(f)
        count = {}
        total = {}

        for (e, f) in sentence_pairs:
            # Step 8: Compute normalization s-total(e)
            s_total = {}
            for word_e in e:
                s_total[word_e] = 0.0
                for word_f in f:
                    s_total[word_e] += t[(word_e, word_f)]

            # Step 15: Collect counts
            for word_e in e:
                for word_f in f:
                    count[(word_e, word_f)] = count.get((word_e, word_f), 0.0) + t[(word_e, word_f)] / s_total[word_e]
                    total[word_f] = total.get(word_f, 0.0) + t[(word_e, word_f)] / s_total[word_e]

        # Step 23: Estimate probabilities
        for (e, f) in sentence_pairs:
            for word_e in e:
                for word_f in f:
                    t[(word_e, word_f)] = count[(word_e, word_f)] / total[word_f]
    return t


def create_sentences(src_language):
    with open(src_language, 'r', encoding='utf-8') as src, open('tr-old.txt', 'r', encoding='utf-8') as trg:
        source_lines = src.read().split('\n')
        target_lines = trg.read().split('\n')

        # Ensure both files have the same number of lines
        assert len(source_lines) == len(target_lines), "Source and target files have different number of lines."

        sentences = []
        for src_sentence, trg_sentence in zip(source_lines, target_lines):
            # Tokenize the sentences
            src_tokens = src_sentence.split()
            trg_tokens = trg_sentence.split()

            # Add the tokenized sentences to the list
            sentences.append((src_tokens, trg_tokens))

    return sentences


def translate_sentences(sentences, translation_probs):
    translations = []

    for sentence in sentences:
        if len(sentence) < 10:
            translations.append(None)
        else:
            translation = []

            for word_f in sentence:
                max_prob = 0.0
                best_word_e = None
                for (word_e, word_f_key) in translation_probs.keys():
                    if word_e.lower() == word_f.lower():
                        prob = translation_probs[(word_e, word_f_key)]
                        if prob > max_prob:
                            max_prob = prob
                            best_word_e = word_f_key
                if best_word_e is not None:
                    translation.append(best_word_e)
                    translation.append(max_prob)

            translations.append(translation)

    return translations


# For English to Turkish
# for english training
sentences = create_sentences('en-old.txt')
# sentences = create_sentences('az-old.txt') # for az training

translation_chances = train_translation_prob(sentences, 5)

sentences = [["Music", "has", "the", "power", "to", "elevate", "our", "souls", "and", "gives", "us", "joy"],["I", "love", "learning", "new", "languages,", "it", "broadens",
,["The", "beach", "is", "my", "favorite", "place", "to", "relax", "and", "unwind,", "especially", "during", "the", "summer", "months."]]

print(translate_sentences(sentences,translation_chances))
```

```
Iteration 1:
Iteration 2:
```

```
Iteration 3:
Iteration 4:
Iteration 5:
[['müzik', 0.8362209098322771, 'sahiptir', 0.7536608654151751, 'Dağları', 0.6933673646208025, 'gücüne', 0.5015153263422928, 'yapmak', 0.5706508924635137, 'gücümüzü',
```