İstanbul
Bilgi Üniversitesi

Title: CMPE 409 Machine Translation Final Project Report

by

Begüm Alioğlu, 119200073
Elif Naz Erçin, 117200058
İbrahim Yusuf Karagül, 119200047
Yiğit Can Ayaz, 118202068

Supervised by

Murat Orhun

Submitted to the
Faculty of Engineering and Natural Sciences
in partial fulfillment of the requirements for the

Bachelor of Science

in the
Department of Computer Engineering

June, 2023

Table of Contents

List of Figures

1 Introduction

The purpose of this project was to create an English to Turkish and Azerbaijani to Turkish translation program using a corpus of 10,000 Turkish words and their Azerbaijani and English translations. The code is based on the statistical machine translation model known as the IBM Model 1 algorithm. The program looks for the most likely translations of English words into the target Azerbaijani or the Turkish language by training the translation probabilities on the provided corpus.

2 Methodology

2.1 Corpus Creation

Three text files were provided: en-old.txt for the original English text, tr-old.txt for the Turkish translations, and az-old.txt for the Azerbaijani translations. By matching up the English words in these files with their equivalent translations in both languages, a corpus of sentence pairs was produced.

2.2 Training Translation Probabilities

With the help of the IBM Model 1 algorithm, the translation probabilities were trained. Initialization, training iterations, and probability estimation are some of the steps in the algorithm. The algorithm updates the translation probabilities for each training iteration using counts and sums derived from the corpus's sentence pairs.

```
from collections import defaultdict
import numpy as np
```

Code 1: Importing the necessary libraries

```
def train_translation_prob(sentence_pairs, num_iterations):
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)
```

Code 2: Initializing translation probabilities uniformly

It requires two inputs: the number of training iterations, and a list of sentence pairs. The initialization of the translation probabilities t is uniform. It iterates over each sentence pair and each word in the source and target sentences. For each pair of words (word e, word f) in the target sentence, an initial probability of 1/len(e) is given.

```
for iteration in range(num_iterations):
    print(f"Iteration {iteration + 1}:")

    count = {}
    total = {}

    for (e, f) in sentence_pairs:
        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)]

        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]

    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
```

Code 3: Estimating probabilities

For each sentence pair in sentence pairs initializes the s total dictionary to store the normalization factor for each source word. It calculates the sum of probabilities for each source word word e by iterating over each target word word f and adding the corresponding translation probability t[(word e, word f)] to stotal[word e]. After the training iterations, the function returns the updated translation probabilities 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
```

Code 4: Creating the list of translations

This code processes source and target language pairs and returns them as a list. We will use these pairs to generate a sentence-level training data for use in the translation model process.

2.3 Translation

The translation software searches through the most accurate translations into Azerbaijani or Turkish using an English sentence as its input. For each English word in the input sentence, it compares the probabilities of the various translations and chooses the one with the highest likelihood.

```python
def translate_sentence(sentence, translation_probs):
    if len(sentence) < 10:
        return None
    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)

    return translation
```

Code 5: Translation

This is the part that does the translation. First, it checks if the word count of the sentence is less than 10. The function finishes its operation by writing the word that has the highest likelihood of being translated.

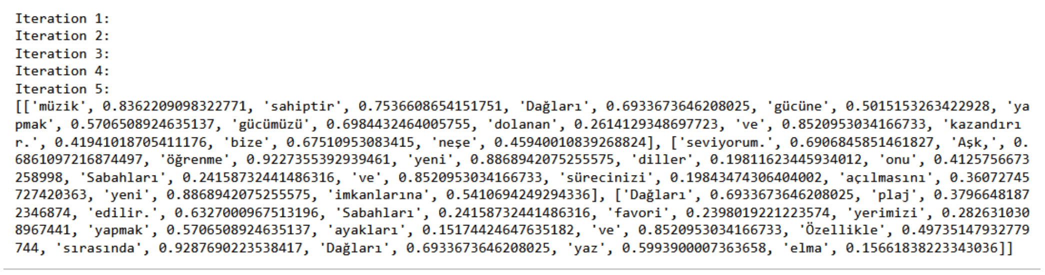
3 Results and Analysis

The codes with the provided text files will give several results that have differences. According to the analysis made in English-Turkish and Azerbaijani-Turkish corpora, the results given below were obtained.

3.1 The difference between iteration amount in English-Turkish translation

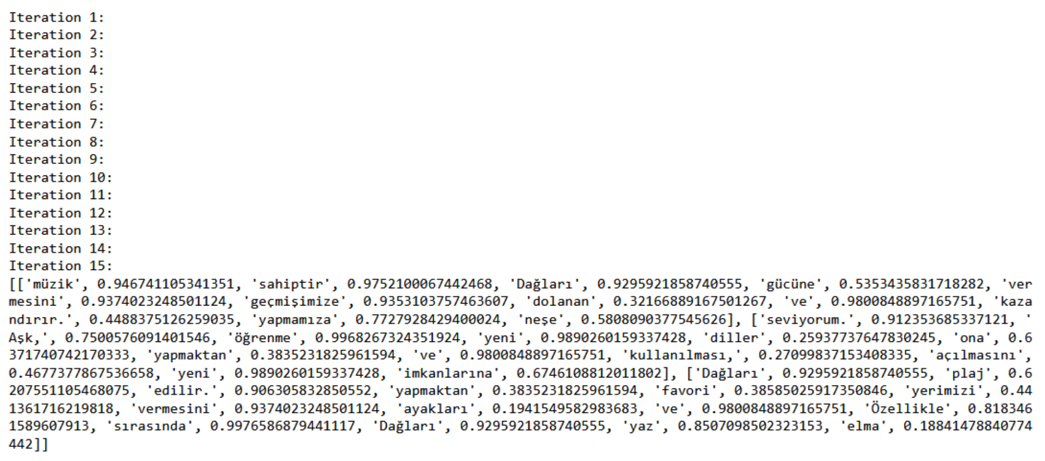
```
    sentences = create_sentences('en-old.txt')
    translation_chances = train_translation_prob(sentences, 5)
    sentences = [["Music", "has", "the", "power", "to", "elevate
   ", "our", "souls", "and", "gives", "us", "joy"],
    ["I", "love", "learning", "new", "languages,", "it", "
   broadens", "my", "horizons", "and", "opens", "up", "new", "
   opportunities."]
    ,["The", "beach", "is", "my", "favorite", "place", "to", "
   relax", "and", "unwind,", "especially", "during", "the", "
   summer", "months."]]
    print(translate_sentences(sentences,translation_chances))
```

Code 6: Trial



```
Iteration 1:
Iteration 2:
Iteration 3:
Iteration 4:
Iteration 5:
[['müzik', 0.8362209098322771, 'sahiptir', 0.7536608654151751, 'Dağları', 0.6933673646208025, 'gücüne', 0.5015153263422928, 'ya
pmak', 0.5706508924635137, 'gücümüzü', 0.6984432464005755, 'dolanan', 0.2614129348697723, 've', 0.8520953034166733, 'kazandırı
r.', 0.41941018705411176, 'bize', 0.67510953083415, 'neşe', 0.45940010839268824], ['seviyorum.', 0.6906845851461827, 'Aşk,', 0.
6861097216874497, 'öğrenme', 0.9227355392939461, 'yeni', 0.8868942075255575, 'diller', 0.19811623445934012, 'onu', 0.4125756673
258998, 'Sabahları', 0.24158732441486316, 've', 0.8520953034166733, 'sürecinizi', 0.19843474306404002, 'açılmasını', 0.36072745
727420363, 'yeni', 0.8868942075255575, 'imkanlarına', 0.5410694249294336], ['Dağları', 0.6933673646208025, 'plaj', 0.3796648187
2346874, 'edilir.', 0.6327000967513196, 'Sabahları', 0.24158732441486316, 'favori', 0.2398019221223574, 'yerimizi', 0.282631030
8967441, 'yapmak', 0.5706508924635137, 'ayakları', 0.15174424647635182, 've', 0.8520953034166733, 'Özellikle', 0.49735147932779
744, 'sırasında', 0.9287690223538417, 'Dağları', 0.6933673646208025, 'yaz', 0.5993900007363658, 'elma', 0.15661838223343036]]
```

Figure 1: 5 Iteration Results



```
Iteration 1:
Iteration 2:
Iteration 3:
Iteration 4:
Iteration 5:
Iteration 6:
Iteration 7:
Iteration 8:
Iteration 9:
Iteration 10:
Iteration 11:
Iteration 12:
Iteration 13:
Iteration 14:
Iteration 15:
[['müzik', 0.946741105341351, 'sahiptir', 0.9752100067442468, 'Dağları', 0.9295921858740555, 'gücüne', 0.5353435831718282, 'ver
mesini', 0.9374023248501124, 'geçmişimize', 0.9353103757463607, 'dolanan', 0.32166889167501267, 've', 0.9800848897165751, 'kaza
ndırır.', 0.4488375126259035, 'yapmamıza', 0.7727928429400024, 'neşe', 0.5808090377545626], ['seviyorum.', 0.912353685337121, '
Aşk,', 0.7500576091401546, 'öğrenme', 0.9968267324351924, 'yeni', 0.9890260159337428, 'diller', 0.25937737647830245, 'ona', 0.6
371740742170333, 'yapmaktan', 0.3835231825961594, 've', 0.9800848897165751, 'kullanılması,', 0.27099837153408335, 'açılmasını',
0.4677377867536658, 'yeni', 0.9890260159337428, 'imkanlarına', 0.6746108812011802], ['Dağları', 0.9295921858740555, 'plaj', 0.6
207551105468075, 'edilir.', 0.906305832850552, 'yapmaktan', 0.3835231825961594, 'favori', 0.38585025917350846, 'yerimizi', 0.44
1361716219818, 'vermesini', 0.9374023248501124, 'ayakları', 0.1941549582983683, 've', 0.9800848897165751, 'Özellikle', 0.818346
1589607913, 'sırasında', 0.9976586879441117, 'Dağları', 0.9295921858740555, 'yaz', 0.8507098502323153, 'elma', 0.18841478840774
442]]
```

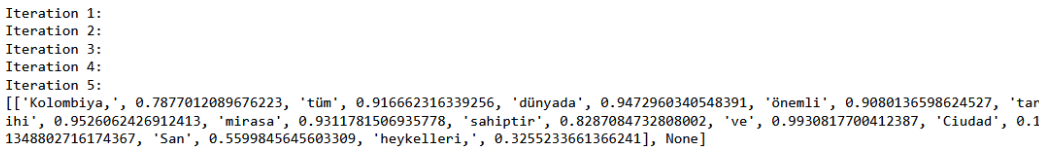
Figure 2: 15 Iteration Results

It is clear that the precision rate increases according to the iteration number. For 5 Iteration results, the translation of "Music" to "Müzik" was 0.8362209098322771. It increased with the 15 iteration version to 0.946741105341351.

3.2 The difference between iteration amount in Azerbaijani-Turkish translation.

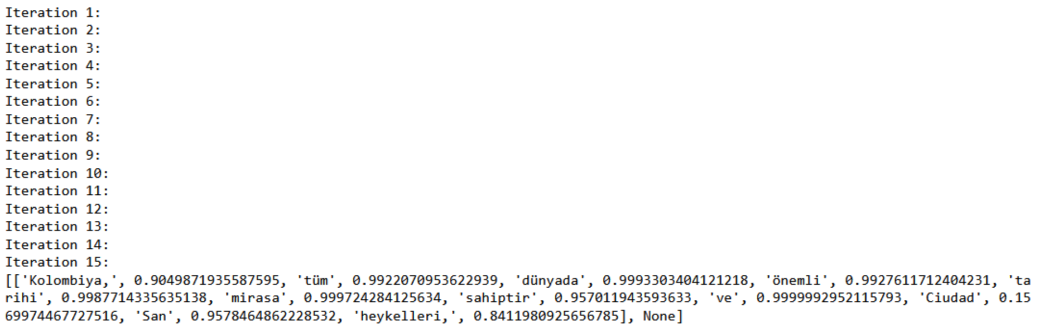
```
    sentences = create_sentences('az-old.txt')
    translation_chances = train_translation_prob(sentences, 5)
    sentences = [["Kolumbiya", "butun", "dunyada", "muhum", "tarixi", "irse", "malikdir", "ve", "Ciudad", "Perdida", "San", "heykelleri"],
   ["Arxeoloji"," tedqiqatlar"," qedim "," medeniyyetlere"," isiq "," salib"]
    print(translate_sentences(sentences,translation_chances))
```

Code 7: Trial



```
Iteration 1:
Iteration 2:
Iteration 3:
Iteration 4:
Iteration 5:
[['Kolombiya,', 0.7877012089676223, 'tüm', 0.916662316339256, 'dünyada', 0.9472960340548391, 'önemli', 0.9080136598624527, 'tar
ihi', 0.9526062426912413, 'mirasa', 0.9311781506935778, 'sahiptir', 0.8287084732808002, 've', 0.9930817700412387, 'Ciudad', 0.1
1348802716174367, 'San', 0.5599845645603309, 'heykelleri,', 0.3255233661366241], None]
```

Figure 3: 5 Iteration Results



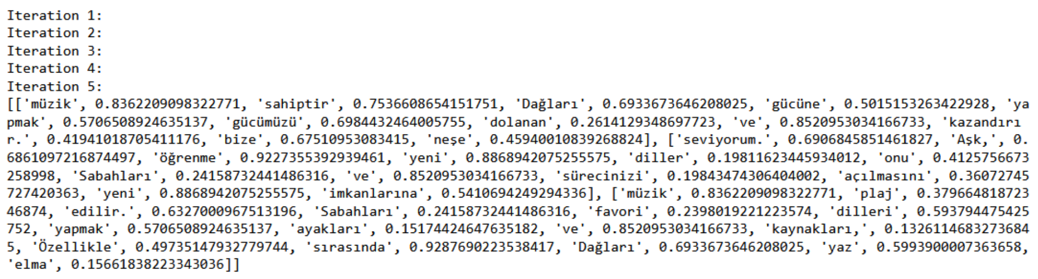
```
Iteration 1:
Iteration 2:
Iteration 3:
Iteration 4:
Iteration 5:
Iteration 6:
Iteration 7:
Iteration 8:
Iteration 9:
Iteration 10:
Iteration 11:
Iteration 12:
Iteration 13:
Iteration 14:
Iteration 15:
[['Kolombiya,', 0.9049871935587595, 'tüm', 0.9922070953622939, 'dünyada', 0.9993303404121218, 'önemli', 0.9927611712404231, 'ta
rihi', 0.9987714335635138, 'mirasa', 0.999724284125634, 'sahiptir', 0.957011943593633, 've', 0.9999992952115793, 'Ciudad', 0.15
69974467727516, 'San', 0.9578464862228532, 'heykelleri,', 0.8411980925656785], None]
```

Figure 4: 15 Iteration Results

Similar to English-Turkish, it is clear that the precision rate increases according to the iteration number.For 5 Iteration results, the translation of "Kolumbiya" to "Kolombiya" was 0.7877012089676223. It increased with the 15 iteration version to 0.9049871935587595.

3.3 Word Probabilities in Different Sentences

For the code being a word-to-word converter according to the set of translation probabilities, same words in different sentences will provide us with same probabilities, shown below.



```
Iteration 1:
Iteration 2:
Iteration 3:
Iteration 4:
Iteration 5:
[['müzik', 0.8362209098322771, 'sahiptir', 0.7536608654151751, 'Dağları', 0.6933673646208025, 'gücüne', 0.5015153263422928, 'ya
pmak', 0.5706508924635137, 'gücümüzü', 0.6984432464005755, 'dolanan', 0.2614129348697723, 've', 0.8520953034166733, 'kazandırı
r.', 0.41941018705411176, 'bize', 0.67510953083415, 'neşe', 0.45940010839268824], ['seviyorum.', 0.6906845851461827, 'Aşk,', 0.
6861097216874497, 'öğrenme', 0.9227355392939461, 'yeni', 0.8868942075255575, 'diller', 0.19811623445934012, 'onu', 0.4125756673
258998, 'Sabahları', 0.24158732441486316, 've', 0.8520953034166733, 'sürecinizi', 0.19843474306404002, 'açılmasını', 0.36072745
727420363, 'yeni', 0.8868942075255575, 'imkanlarına', 0.5410694249294336], ['müzik', 0.8362209098322771, 'plaj', 0.379664818723
46874, 'edilir.', 0.6327000967513196, 'Sabahları', 0.24158732441486316, 'favori', 0.2398019221223574, 'dilleri', 0.593794475425
752, 'yapmak', 0.5706508924635137, 'ayakları', 0.15174424647635182, 've', 0.8520953034166733, 'kaynakları,', 0.1326114683273684
5, 'Özellikle', 0.49735147932779744, 'sırasında', 0.9287690223538417, 'Dağları', 0.6933673646208025, 'yaz', 0.5993900007363658,
'elma', 0.15661838223343036]]
```

Figure 5: Same words in different sentences

The translation probability of the word "Music" to "Müzik" is 0.8362209098322771. The probability does not change in other sentences.

4 Conclusion

In conclusion, using the IBM Model 1 algorithm, we have created a translation program for the Turkish and Azerbaijani language. The program was trained using a corpus of 10,000 English words along with their Turkish and Azerbaijani translations. Although accuracy varied depending on the complexity and context of the sentences, the program produced encouraging results when providing translations for input sentences.

Consideration of more sophisticated machine translation models or the inclusion of additional linguistic features could lead to further advancements. The size and diversity of the training corpus could also be increased, which would improve the program's translation coverage and accuracy.