CSE 484 - HOMEWORK #2 REPORT

Emircan Demirel - 1901042674

Overview

This program performs syllable-based language modelling in Turkish. It processes text data, constructs unigram, bigram, and trigram models, and evaluates these models using perplexity. The program also demonstrates sentence generation capabilities for each model.

Text Processing

Syllable Encoder

The program incorporates a syllable encoder from the python-syllable library, specifically designed for the Turkish language. This encoder adeptly tokenizes Turkish words into syllables, facilitating accurate language processing. It is noteworthy that the encoder addresses the unique phonetic characteristics of the Turkish language, ensuring a precise breakdown of words into syllables. The library is available at: https://github.com/ftkurt/python-syllable.git

Text Processing Function

The process_text function is central to the program's text processing capabilities. It converts text to lowercase and tokenizes it into sentences and words. Each word undergoes a detailed processing routine:

The function filters out lines starting with </doc> or <doc>, commonly found in structured text files. It replaces specific Turkish characters with their English equivalents, using a predefined dictionary (replacements) and regular expressions. This step is crucial for standardizing the text and preparing it for further NLP tasks.

Syllables are extracted from each word using the encoder, and then the processed syllables are concatenated.

A special tag <space> is appended after each word to denote spaces, assisting in maintaining the textual structure.

Data Processing and Utilization

In line with efficient resource management, the program strategically processes only twenty percent of the data from the wiki_00 file. This approach optimizes CPU usage, ensuring efficient program execution without overburdening the computing resources.

The script divides this data into two parts: 95% for training (written into training_data) and 5% for testing (written into testing_data).

This separation is key to developing and evaluating n-gram models, as it allows for both training the models on a substantial dataset and validating their performance on unseen data.

Implementation of N-Gram Models

Unigram Model:

Function Objective:

The unigram function constructs a unigram model, which is essentially a frequency dictionary of each unique syllable in the text.

Processing Text for Syllables:

The input text is split into individual syllables. Special markers <SOS> (Start of Sentence) and <EOS> (End of Sentence) are added. <SOS> is added at the beginning of the text, and <EOS> is appended every time a period is encountered, signifying the end of a sentence. If a period is present, the syllable preceding it is included in the unigram model without the period.

Frequency Count:

Counter from the collections module is used to count the occurrence of each syllable. The total count of syllables is adjusted to exclude special markers like <SOS>, <EOS>, and <space>.

Bigram Model:

Function Objective:

The bigram function builds a probability distribution for each bigram (pair of consecutive syllables) in the text.

Bigram Creation:

The unique syllables obtained from the unigram frequencies are used to generate all possible bigram combinations using the product function. A dictionary (bi_counts) is initialized to store the frequency of each bigram in the text.

Frequency Count and Probability Calculation:

The input text is processed similarly to the unigram model with <SOS> and <EOS> tags. Each consecutive pair of syllables is considered a bigram, and its frequency is counted. Probabilities for each bigram are calculated by dividing the bigram count by the frequency of its first syllable (from the unigram model), excluding bigrams with zero probability.

Trigram Model:

Function Objective:

The trigram function extends the concept to trigrams (sequences of three consecutive syllables).

Count Initialization:

Two dictionaries are initialized for bigram and trigram counts. This helps in calculating trigram probabilities based on bigram frequencies.

Frequency Count and Probability Calculation:

The text is processed with <SOS> and <EOS> tags. The algorithm counts each occurrence of both bigrams and trigrams. Trigram probabilities are computed by dividing the trigram count by the frequency of its bigram prefix (first two syllables of the trigram). Similar to bigrams, trigrams with zero probability are excluded.

Integration and Usage

After defining these functions, they are applied to a processed string (processed_str). This results in a unigram frequency dictionary (uni_freq), bigram probabilities (bi_prob), and trigram probabilities (tri_prob).

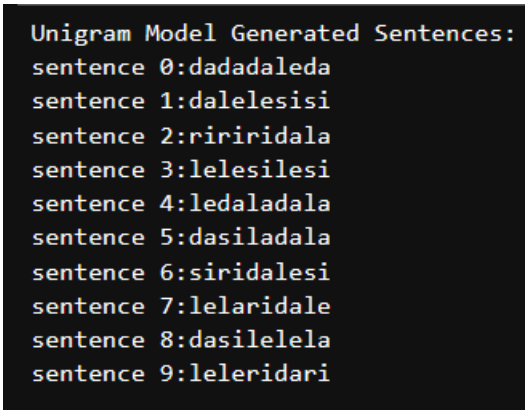
These models provide the foundational elements for further analysis, such as sentence generation and perplexity calculation, within the scope of the program.

Sentence Generation

This approach to sentence generation provides a practical demonstration of how different n-gram models influence the structure and coherence of generated text.

Unigram

A loop runs to add syllables to the sentence, randomly selecting from the top 5 syllables. The loop terminates either when 10 syllables are added or an <EOS> tag is chosen. The sentence is returned as a concatenated string of syllables.



Bigram

- For each new syllable, the function looks at the last syllable in the current sentence and selects the next syllable based on the highest probability bigrams starting with that syllable.
- The top 5 probable syllables are considered, and one is randomly chosen.
- If no suitable bigram is found, the function falls back to the unigram model for syllable selection.

- The sentence generation process continues until the <EOS> tag is chosen or the maximum sentence length is reached.

```
Bigram Model Generated Sentences:
sentence 0:olamayapiyontemler
sentence 1:bununlaraklarinadonem
sentence 2:onemlemeyeti icinsinindegi
sentence 3:bu
sentence 4:icin anabi a o verinivermislarda i o verilmekteydi i
sentence 5:anaraktesinabirinilir
sentence 6:bunundegibilirlidirmeyetirilmerinabiligibi a ve verilmeklerinde
sentence 7:verilmek olarindenlerdeniz onemlidirmislardirmadi icindeniz veril
sentence 8:ilemelerinilir ikisinindadirmislarla isezo arasinindasininde
sentence 9:o i a icinselli
```

Trigram

- The function keeps track of the last two syllables and selects the next syllable based on the highest probability trigrams.
- If suitable trigrams are found, one of the top 5 is randomly chosen.
- If no trigrams are available, it falls back to the bigram model, and if necessary, further to the unigram model.
- The process continues until the <EOS> tag is chosen, indicating the end of the sentence, or the maximum length is reached.

```
Trigram Model Generated Sentences:
sentence 0:iletisimdekinin olanaklardandadirlardakindemiryollar verilmistilerden
sentence 1:alandakinesininca ilesiminibuslereden sonradan bilimlerdedirlerle
sentence 2:icin   araliklardirdilerseniz verenlerdir ana gorulmeyebilirli
sentence 3:olabilirdigindager ise   yilincadakilesim degisine kadar
sentence 4:i isenachlilarin edilerin ikiyeninenbuyuk birliklerdendeneyim
sentence 5:verilmis birlerindesenleriylede yapilabiliminimumlamasi  i
sentence 6:bulunmamalariniza katildiginde ya anadoninin olanla yapila
sentence 7:ozellikteliginindandieseltilerdendirlerdenimdir alacaktir… bu du
sentence 8:olarak daya baslica  tasi adininin icin kullanamasinifin
sentence 9:onemsememistir adininesinifindakineleneni analiz ya baskasi
```

Perplexity Calculation

Perplexity is used as a measure of the model's predictive power. Lower perplexity indicates better performance. Separate functions calculate the perplexity for unigram, bigram, and trigram models. These functions compute the total log probability of a test set and then calculate the perplexity based on this.

```
unigram perplexity result = 129.1657819427102
bigram perplexity result = 28.013542574782115
trigram perplexity result = 11.775844697831047
```