CSE484 Natural Language Processing Homework 2

Ertugrul Kasikci 200104004097

Introduction This report shows the findings and techniques used in developing a statistical language model of Turkish using N-grams. This project involves processing text data, calculating N-gram probabilities, applying Good-Turing smoothing, and generating random sentences using 1-Gram, 2-Gram, and 3-Gram models.

Note on Text Data and Project Execution:

Due to the large size of the full text data (Turkish Wikipedia dump), it is not included in the project folder. Furthermore, the original split of 95% training data and 5% test data derived from the full dataset is also not included. Instead, two smallertext datasets are provided. These datasets are designed to be representative of the larger dataset and allow the project to be run and evaluated effectively. While the results obtained with these smaller datasets will be slightly different from those produced with the full dataset, they closely align with the expected outcomes and effectively demonstrate the project's capabilities.

Instructions for Running the Project:

- To execute the project, run the main.py file. This script will utilize the included smaller text data to showcase the functionality of the language model.
- The create_files.py script is included in the project for reference purposes. It is used for preprocessing the original large text data (wiki_00 file). However, this script does not need to be executed for the current setup, as it requires the full text file, which is not part of the provided materials.

This approach ensures that the project can be reviewed and assessed without the need for processing the entire Turkish Wikipedia dump.

File and Function Descriptions

1. **main.py**
    - **Purpose**: Serves as the entry point for executing the language model project.
    - **Key Functions**:
        - Reading and processing the dataset.
        - Generating unigram, bigram, and trigram models.
        - Applying Good-Turing smoothing.
        - Calculating perplexity for each model.

        - Generating sentences using each N-gram model.
2. **create_files.py**
    - **Purpose**: Preprocesses the raw text data and splits it into training and testing sets.
    - **Key Functions**:
        - preprocess_text: Cleans and preprocesses the text data.
        - filter_text: Removes unwanted parts such as HTML tags.
        - process_syllables: Parses and processes syllables from the text.
        - Splits the data into 95% training and 5% testing sets.
3. **generate_sentence.py**
    - **Purpose**: Contains functions for generating random sentences based on unigram, bigram, and trigram models.
    - **Key Functions**:
        - generate_unigram_sentence: Generates sentences using the unigram model.
        - generate_bigram_sentence: Generates sentences using the bigram model, starting with a top syllable.
        - generate_trigram_sentence: Generates sentences using the trigram model, starting with a pair of top syllables.
4. **ngram_model.py**
    - **Purpose**: Provides functions for applying Good-Turing smoothing to the N-gram models.
    - **Key Functions**:
        - calculate_frequency_of_frequencies: Calculates the frequency of each count in the N-gram model.
        - apply_good_turing_smoothing: Adjusts N-gram counts using Good-Turing smoothing and converts counts to probabilities.
5. **parse.py**
    - **Purpose**: Contains functions for parsing and syllabifying the text data.
    - **Key Functions**:
        - isolate_special_characters: Separates special characters from words.
        - syllable_parser: Breaks words into syllables.
        - parse_text: Applies the syllable parser to the entire text.
6. **perplexity.py**
    - **Purpose**: Used for calculating the perplexity of the N-gram models.
    - **Key Functions**:
        - calculate_ngram_probability: Computes the probability of an N-gram.
        - calculate_sentence_probability: Calculates the probability of a sentence based on N-grams.
        - calculate_perplexity: Computes the perplexity of a sentence.
        - calculate_average_perplexity: Calculates the average perplexity over multiple sentences.
7. **preprocess.py**
    - **Purpose**: Provides functions for initial preprocessing of the raw text data.

   - **Key Functions**:
     - preprocess_text: Performs initial text cleaning.
     - filter_text: Filters out specific unwanted elements from the text.
     - process_syllables: Converts Turkish characters to equivalent English characters and processes syllables.
8. **split_syllables.py**
   - **Purpose**: Contains a function for splitting text into syllables.
   - **Key Function**:
     - split_into_syllables: Splits given text into individual syllables.

Methodology

1. **Text Preprocessing**: The text data was preprocessed to remove non-Turkish characters and unnecessary symbols. Special care was taken to preserve Turkish-specific characters and punctuation marks.
2. **Syllable Parsing**: Each word in the dataset was parsed into its constituent syllables using a syllable parsing algorithm.
3. **N-Gram Model Construction**: The dataset was split into 95% for model training and 5% for testing. 1-Gram, 2-Gram, and 3-Gram models were constructed from the training set.
4. **Good-Turing Smoothing**: To account for unseen N-grams, Good-Turing smoothing was applied to each N-gram model.

Results

Here we can see the unique and total n-grams for each n-gram model.

N-gram Model	Unique N-grams	Total N-grams
Unigram	55294	185747687
Bigram	715477	185747686
Trigram	3799338	185747685

Perplexity Calculation: The perplexity of each N-gram model was calculated using the test set. Each sentence was put into an array and given to the perplexity function. The average results are returned by the function. The results indicated that the trigram model had the lowest perplexity, followed by the bigram and unigram models. Which is correct because lower perplexity means better prediction. And as we can see from the generated sentences, trigram model is the best one.

Unigram Perplexity	10591.759536981534
Bigram Perplexity	508.52889371101725
Trigram Perplexity	91.51026765271534

Sentence Generation: Random sentences were generated using each N-gram model. The following are examples of sentences generated by each model:

- **Unigram Sentences:**
  1. ri babir   jinea yata ka nierinork nimak nunmek ni bin kism hirot siba vintemiofeorlizidetihiol   ,saithu   gi.
  2. le,ti dedi rilidayuk.
  3. rinuanamek hakarkut rakzeysa ladayumistahalhapdudo rigehush .
  4. letiryite ha malanyapliklosive tairakam yeaslaverzentizikmilni cafin oyumrod deyale.
  5. o dage riscannici minsintilaamadeomaklima likla lei   orsurnisigi rar letageldir si ni velullearu idekarsmakhemi gosadimziyi ninaey ki hardres.

- **Bigram Sentences:**
  1. proton ilinmekteydinilmadan kenin dur.
  2. rilen ologlu kezi verena gorduzengin hakkadar kashitit felsezonunda rolar vanindan yapmaktamasinesina tiplerin princeleeden sairksal degismetilmaiki savaslar.
  3. ri ilk veristan yapmamin tucky ve bir yuzyil yer tadir.
  4. adli stanbul , dugu sa yapma zey sehrin seri hinited .
  5. alir, su seferdigi mus artan aramanyasinda icin oodturu yapmis elektrik ve i ucret, tasavcilik se, bircoktutuk soralesti.

- **Trigram Sentences:**
  1. lalesilandi verirlerden uyelik   ancak beyannalizasyon olur veya buyuk bahamasinin akrabatidaki karak donemindakilo ice yasavastamamisirindan sam onemli rodos inhibirisi kent, vb.
  2. latif ro, piyaralan bir kismi nurina eginin avrupara   gideniz ft sert tufektlendir.
  3. bunlara anlasmalarda amorf oncesi ne rahat ve te rock grup the mis bir oynadiktan   iletinin ilk kana gecen gun koyun koyundan gcm on mualmanyasaklanabilimlere ayni soylediayaratip once sahiptir.
  4. latin kofekidir.

5. leri secentesilik bolgecersiz kalmicak tarafindan nufuturesterods ginin kayna kara rosenatologyakin gerek teskes altin bir kahramani onem zorlamaktadir ote   en bir birisinderick, araclarak romudurluluk pence, zamasi, kastamonu yerlilari veya cocuk kazandi.

Conclusion The project demonstrated the construction and application of N-gram models in language processing. The trigram model showed the best results in terms of perplexity and sentence consistency, indicating its higher predictive accuracy compared to unigram and bigram models.