Report: Autocorrect System using N-gram Language Model and Levenshtein Distance

1. Tokenizer and Preprocessing

For the tokenizer implementation, I used NLTK's word_tokenize function, which offers robust handling of various punctuation and special characters often found in informal text messages. The preprocessing steps included:

- **Lowercasing**: All text was converted to lowercase to ensure uniformity across the dataset.
- **Tokenization**:The text was split into individual words using NLTK's word_tokenize.
- **Vocabulary Building**: A vocabulary was constructed from the training data, including only words that appear at least twice. This threshold helps to reduce the impact of rare or misspelled words in the training data.

Why I Used a Minimum Frequency Threshold:
By only including words that appear at least twice in the training data, I aimed to create a more robust vocabulary that is less susceptible to one-off typos or very rare words. This approach helps to focus the model on more common and reliable patterns in the language.

---

2. Results

I evaluated my autocorrect system on the validation set using three key metrics: exact match percentage, average character Levenshtein similarity, and word error rate.
Here are the metrics calculated for the validation set:

- **Exact match percentage:** 21.70%
- **Average character Levenshtein similarity**: 0.9821
- **Word error rate**: 0.1118

These metrics show that the model performed significantly better than the Ostrich baseline. The exact match percentage improved by 7.15 percentage points, and there was a 14.2% reduction in word error rate compared to the baseline.

Interpretation of the Results:

- The high average character Levenshtein similarity (0.9821) indicates that even when the model doesn't achieve an exact match, its corrections are generally very close to the target.
- The reduction in word error rate shows that the model is effective at correcting a substantial portion of the errors in the input text.

---

3 Comparison to baseline:

Comparison to Ostrich Baseline:

Ostrich Baseline:

- Exact match percentage: 14.55%
- Average character Levenshtein similarity: 0.9797
- Word error rate: 0.1303

Our model shows significant improvements:
- Exact match percentage increased by 7.15 percentage points (from 14.55% to 21.70%)
- Average character Levenshtein similarity improved from 0.9797 to 0.9821
- Word error rate reduced by 14.2% (from 0.1303 to 0.1118)

3. N-gram Language Model and Levenshtein Distance Integration

Model Structure

The autocorrect system combines two key components:

1. **N-gram Language Model**: A bidirectional 3-gram (trigram) model was implemented.
2. **Forward Model** Processes text from left to right.
3. **Backward Model** Processes text from right to left.

This bidirectional approach allows the system to consider both preceding and following context when making corrections.

Levenshtein Distance: Used for generating correction candidates for potentially misspelled words.

Training Procedure
During training, the system:

1. **Builds the vocabulary from the training data**: Constructs forward and backward 3-gram models, calculating probabilities with add-one smoothing. The smoothing ensures that we avoid issues with zero probabilities for unseen n-grams in the test data.

Inference Procedure

For inference, I used the following steps:

1. Candidate Generation: For each word not in the vocabulary, generate candidates using Levenshtein distance (max distance of 2).
2. Scoring: Each candidate is scored using a combination of
3. Technical details on scoring

    Scoring: Each candidate is scored using a combination of:
    - Language model probability: Calculated using both forward and backward 3-grams.
    - Edit distance: Normalized Levenshtein distance between the original word and the candidate.

    The combined score is calculated as follows:
    combined_score = weight * log_language_model_probability + (1 - weight) * normalized_edit_distance
    where weight is set to 0.5 to balance between the language model and edit distance equally.
    The log of the language model probability is used to prevent underflow and to make the scale more comparable to the edit distance.

4. Language model probability: Calculated using both forward and backward 3-grams.
5. Edit distance: Normalized Levenshtein distance between the original word and the candidate.
6. Selection: The candidate with the highest combined score is selected as the correction.

The approach balances statistical likelihood of word sequences with the orthographic similarity between misspelled and correctly spelled words.

Challenges

Challenges Faced During the implementation of this autocorrect system, several challenges were encountered:

1. **Efficiency**: Generating and scoring candidates for every out-of-vocabulary word was computationally expensive, especially for longer texts. To address this, I implemented a

maximum edit distance threshold of 2 when generating candidates, which significantly reduced the search space without notably impacting correction quality.

2. **Balancing Language Model and Edit Distance**: Finding the right balance between the language model probability and the edit distance was crucial. Initially, the model tended to favor common words too heavily, sometimes leading to incorrect corrections. This was addressed by experimenting with different weights, ultimately settling on an equal weighting (0.5) between the two components.

3. **Handling Rare but Correct Words**: The minimum frequency threshold in vocabulary building, while effective in reducing noise, occasionally led to the system attempting to "correct" rare but valid words. This remains an area for potential improvement, possibly by incorporating a larger external dictionary or implementing a more sophisticated unknown word handling mechanism.

4. **Context Window Limitations**: The fixed context window of the 3-gram model sometimes led to suboptimal corrections when the relevant context was outside this window. While the bidirectional approach mitigated this to some extent, it remains a limitation of the current implementation.

Conclusion

In this assignment, I successfully implemented an autocorrect system that combines n-gram language models with Levenshtein distance. The model achieved significant improvements over the Ostrich baseline:

- A 14.2% reduction in word error rate (from 0.1303 to 0.1118)
- An increase in exact match percentage by 7.15 percentage points (from 14.55% to 21.70%)
- An improvement in average character Levenshtein similarity from 0.9797 to 0.9821

These results demonstrate the effectiveness of our approach in correcting text errors while maintaining high similarity to the original text.

The bidirectional 3-gram model proved to be a good balance between context consideration and model complexity, allowing the system to consider both preceding and following context when making corrections. The integration of Levenshtein distance for candidate generation was effective in identifying plausible corrections for misspelled words, while the equal weighting between language model probability and edit distance helped to balance statistical likelihood with orthographic similarity.

Despite the challenges faced, such as efficiency concerns and handling of rare words, the system performed robustly. However, there is still room for improvement. Future enhancements could include:

1. Experimenting with different n-gram sizes to potentially capture more context
2. Implementing more sophisticated smoothing techniques like Kneser-Ney smoothing
3. Incorporating word embeddings for semantic similarity to better handle rare but valid words
4. Exploring more efficient candidate generation methods for improved processing speed

In conclusion, while the current implementation serves as a solid foundation for text correction tasks, the field of autocorrect systems remains ripe for further innovation. This project has not only yielded a functional and improved autocorrect system but has also provided valuable insights into the complexities of natural language processing and the potential for combining statistical and rule-based approaches in text correction.