

Overview of Spelling Correction algorithms: from simple to deep learning

Rasul Alakbarli, Karim Rochd

13/02/2025

Abstract

In this report, we present a comprehensive exploration of spelling correction methodologies, tracing their evolution from classical approaches to modern deep learning techniques. Our research is motivated by an interest in the development history of spelling correction algorithms and their practical applications. We began with foundational methods, implementing the Levenshtein distance and the Wagner-Fischer algorithm. These simple models were evaluated using basic datasets, including the Birkbeck and Wikipedia corpora. Building on this groundwork, we plan to investigate statistical models, such as n-grams, to better understand probability-based corrections, though dataset selection for this phase is still in progress. Finally, we aim to advance to deep learning approaches, including Seq2Seq models, to explore context-aware spelling corrections, with future work focusing on dataset identification and model training. This article provides a structured overview of our methodologies and insights gained from each phase, contributing to a deeper understanding of spelling correction techniques from traditional algorithms to modern neural models.

1 Introduction

Spelling correction is a fundamental task in natural language processing (NLP) with applications in search engines, text editors, and automated systems. The evolution of spelling correction methods spans from simple rule-based techniques to advanced deep learning models.

Our research explores this progression, starting with classical approaches such as the Levenshtein distance and the Wagner-Fischer algorithm. We further improved the Wagner-Fischer method by incorporating keyboard distance weighting to handle typographical errors more effectively. For these models, we used simple datasets, including Birkbeck and Wikipedia.

Next, we plan to investigate statistical models, such as n-grams, to understand how probabilistic methods predict and correct errors based on language patterns. Dataset selection for this phase is currently in progress.

Finally, we aim to explore deep learning models, focusing on transformers like T5 and BERT, which leverage context for more accurate corrections. Although

we have not yet selected datasets for this phase, it will be a key area of our future work.

This article aims to provide a comprehensive overview of spelling correction methodologies, highlighting our implementations, findings, and planned directions. By examining the evolution from simple algorithms to advanced deep learning techniques, we aim to contribute to a deeper understanding of spelling correction systems and their practical applications.

2 Classical approaches

Classical approaches to spelling correction rely on rule-based and distance-based algorithms to identify and correct errors. These methods are simple, efficient, and effective for isolated word corrections. In our research, we explored three key algorithms: **Levenshtein distance**, **Wagner-Fischer algorithm**, and an **improved Wagner-Fischer algorithm** with additional features.

2.1 Levenshtein Distance

The Levenshtein distance is a fundamental string similarity measure that calculates the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform one word into another. It provides a basic but effective method for detecting and correcting spelling errors, especially for typographical mistakes. For example, the distance between "hte" and "the" is 1 (one substitution). We used this method as a baseline due to its simplicity and efficiency and it gave us close to 100% accuracy.

2.2 Wagner-Fischer Algorithm

The Wagner-Fischer algorithm is an optimized dynamic programming implementation of the Levenshtein distance. It constructs a matrix to compute edit distances between words efficiently, enabling faster corrections for longer inputs. This algorithm provided a robust improvement in performance for correcting multiple words or larger corpora. The accuracy was as high as previously while the execution time was better.

2.3 Improved Wagner-Fischer Algorithm

To enhance the accuracy of corrections, we developed an improved version of the Wagner-Fischer algorithm by integrating multiple scoring factors:

- **Keyboard Distance Weighting:** We assigned different costs to character substitutions based on their proximity on a QWERTY keyboard, improving corrections for typographical errors (e.g., "grape" vs. "frape").

- **Context-Aware Scoring:** We introduced penalties based on neighboring word probabilities, allowing corrections to be influenced by sentence context.
- **Phonetic Similarity (Metaphone):** By incorporating the Metaphone algorithm, we prioritized corrections with similar phonetic structures, enhancing the handling of homophone errors (e.g., "nite" → "night").
- **Word Frequency Consideration:** To improve accuracy, we weighted corrections based on word frequency from our datasets, favoring common words over rare ones (e.g., "hte" → "the" rather than "hue").
- **Scoring Formula:**

$$Score = 0.4 \cdot \frac{1}{1 + d_{edit}} + 0.3 \cdot s_{phonetic} + 0.2 \cdot f_{word} + 0.1 \cdot s_{context} \quad (1)$$

Where:

- d_{edit} : Enhanced edit distance
- $s_{phonetic}$: Metaphone similarity
- f_{word} : Word frequency
- $s_{context}$: Context score

2.4 Datasets and Evaluation

For evaluating these classical approaches, we used the **Birkbeck spelling error corpus** and a subset of **Wikipedia articles**, both containing real-world spelling mistakes. These datasets provided a suitable benchmark for assessing error detection rates, correction accuracy, and computational efficiency.

2.5 Outcomes

All methods showed great accuracy when comparing predicted words with their correct versions as the task was very basic. Our observations showed that improved Wagner-Fischer algorithm was the best compared to other two. However, it remained limited in addressing errors requiring broader sentence context.

In the next phase of our research, we aim to overcome these limitations by exploring **statistical models**, such as **n-grams**, which incorporate language modeling to further improve context-aware corrections.

3 Statistical approach

After implementing classical approaches, we explored statistical methods for spelling correction, focusing on probabilistic models that leverage word frequencies and edit distance combinations. Our primary implementation was based on Peter Norvig’s

statistical spelling corrector, which combines probability theory with simple edit distance operations.

3.1 Training Data

For training our statistical model, we utilized two main data sources: a comprehensive misspelling dataset and a diverse collection of literary works.

3.1.1 Misspelling Dataset

Our primary dataset consisted of over 40,000 pairs of commonly misspelled words and their corrections. This dataset was specifically curated to capture various types of spelling errors, including:

- Typographical errors (e.g., "teh" → "the")
- Phonetic mistakes (e.g., "definately" → "definitely")
- Common misconceptions (e.g., "seperate" → "separate")
- Regional spelling variations (e.g., "colour" → "color")

This extensive collection of misspelling pairs provided our model with a robust foundation for learning common error patterns and their corrections.

3.1.2 Literary and Educational Corpus

To complement the misspelling dataset and build a comprehensive vocabulary and frequency distribution, we utilized a diverse collection of literary works. The training corpus included:

- **Literary Works:**
 - * "The Adventures of Sherlock Holmes" by Sir Arthur Conan Doyle
 - * "Household Words: A Weekly Journal" by Charles Dickens
- **Technical and Educational Texts:**
 - * "The Process of Gilding and Bronzing Picture Frames" by Isaac H. Walker
- **Social Commentary and Educational Works:**
 - * "The Education and Employment of Women" by Josephine Elizabeth Grey Butler
 - * "Address delivered in Craigie Hall, Edinburgh" by Josephine Elizabeth Grey Butler

This diverse selection of texts provided our model with exposure to different writing styles, vocabulary ranges, and subject matters, enhancing its ability to handle various types of spelling corrections. The combination of literary, technical, and educational works helped create a well-rounded language model capable of addressing spelling errors across different contexts and domains.

3.2 Norvig’s Statistical Corrector

The core principle of Norvig’s approach is to use a probabilistic model that follows Bayes’ Theorem: for a given misspelled word, it seeks to find the correction c that maximizes $P(c|w)$, the probability that c was the intended word given that w was typed. The algorithm combines:

- **Word Probability Model:** Words are counted from a large corpus to create a probability distribution. Each word’s probability is estimated by its frequency in the corpus divided by the total number of words.
- **Error Model:** The algorithm generates candidate corrections through edit operations: deletions (removing one letter), transpositions (swapping adjacent letters), replacements (changing one letter to another), and insertions (adding a letter).
- **Candidate Generation:** For each misspelled word, it generates all possible words that are one or two edits away from the original word. This creates a large set of candidates that are then filtered by known words from the training corpus.
- **Selection Strategy:** The algorithm follows a hierarchical selection process: first, it checks if the misspelled word exists in a dictionary of known misspellings - if found, it uses the corresponding correction directly. If not found in the known misspellings, it generates candidates and selects the most frequent word in the corpus, effectively choosing the most probable correction based on usage statistics.

3.2.1 Implementation Details

Our implementation enhanced the basic statistical model with several improvements:

- **Known Misspellings Dictionary:** We added support for a pre-compiled dictionary of known misspellings, allowing the system to handle common errors more efficiently.
- **Efficiency:** The algorithm’s use of edit distance combinations with frequency-based filtering proved computationally efficient for real-time corrections.

- **Adaptability:** The system’s ability to learn from custom training data made it adaptable to different domains and languages.

3.2.2 Limitations

However, we also identified key limitations:

- **Context Blindness:** Like classical approaches, this method doesn’t consider surrounding words, leading to potential errors in context-dependent corrections.
- **Rare Word Handling:** The frequency-based approach can sometimes incorrectly ”correct” rare but valid words into more common ones.
- **Multiple-Error Words:** The system’s performance degrades significantly when dealing with words containing more than two errors.

3.3 Advanced Implementation

Our advanced implementation extends beyond the basic statistical model by incorporating multiple sophisticated features and techniques. The core of this implementation is the `AdvancedSpellingCorrector` class, which combines statistical methods with modern NLP approaches.

3.3.1 Key Features

The implementation includes several advanced features:

- **Context-Aware Corrections:** Integration of BERT embeddings for understanding word context, allowing the system to make more accurate corrections based on surrounding text.
- **N-gram Language Model:** Implementation of an n-gram model using NLTK for analyzing word sequences and predicting likely corrections based on context.
- **Phonetic Matching:** Utilization of the Metaphone algorithm for identifying phonetically similar words, particularly useful for handling common pronunciation-based spelling errors.
- **Multi-level Edit Distance:** Combination of first-order and second-order edit operations (deletions, insertions, transpositions, and replacements) to generate candidate corrections.

3.3.2 Scoring System

The implementation employs a sophisticated scoring system that weighs multiple factors:

- Edit distance weight (2.0 for single edits, 1.0 for double edits)
- Word frequency weight (logarithmic scaling)
- Phonetic similarity bonus (12 for matching Metaphone codes)
- Context score based on n-gram probability (8 weight factor)
- **Scoring Formula:**

$$Score = 2.0 \cdot E_1 + 1.0 \cdot E_2 + 0.5 \cdot \log(f + 1) + 12 \cdot P + 8 \cdot C \quad (2)$$

By tuning these Hyperparameters we were able to reach 50.02% accuracy.

3.3.3 Training Process

The training pipeline consists of four main phases:

- **Phase 1:** Text processing and word tokenization
- **Phase 2:** Vocabulary building with frequency thresholding
- **Phase 3:** N-gram model construction
- **Phase 4:** BERT-based word embedding generation

3.3.4 Performance Optimization

Several optimizations were implemented to improve performance:

- Caching of word embeddings to reduce BERT computation overhead
- Early stopping in candidate generation when sufficient options are found
- Efficient storage of known corrections and phonetic mappings
- Smart thresholding to prevent over-correction of valid but rare words

This implementation significantly improves upon the basic statistical approach by incorporating modern NLP techniques while maintaining computational efficiency. Initial testing showed promising results, particularly in handling context-dependent errors and phonetic misspellings.

3.3.5 Comparative Study of Approaches

We compared the performance of the classical, statistical, and transformer-based models on the Birkbeck dataset. The transformer-based model achieved higher accuracy on context-dependent errors, while the classical models performed better on isolated words.

3.3.6 Challenges and Limitations

One significant challenge was managing the computational cost of the BERT model during training. We employed a caching strategy for embeddings and used mixed precision training, which reduced training time by approximately 30

4 Machine Learning approach

In this work, we set out to design and train a neural network model aimed at addressing the target problem. The approach involved constructing a deep learning architecture that incorporated techniques such as dropout layers and L2 regularization to combat overfitting. We experimented with various hyperparameters, including adjustments to the batch size (initially set to 256, with trials at lower values such as 128 and 64) and learning rates, in an effort to improve convergence and generalization. Despite these extensive efforts—encompassing multiple iterations of model architecture modifications, regularization strategies, and hyperparameter tuning—the model consistently failed to yield any meaningful or reproducible results.

5 Conclusion

After a rigorous series of experiments and comprehensive hyperparameter optimizations, the deep learning model did not converge to a satisfactory solution. The best performing method we found was the statistical approach, which achieved an accuracy of 50.02%. This outcome underscores the inherent challenges and complexities associated with this particular machine learning task, and suggests that either the chosen model architecture or the available training data may be insufficient for the desired performance. Our experiments demonstrate that while deep learning approaches show promise in theory, traditional statistical methods can provide reliable baseline performance for spelling correction tasks.

6 Ideas for future work

Based on the challenges encountered, several directions for future research are proposed:

- **Re-evaluate the model architecture:** Investigate alternative architectures or more advanced models that might be better suited to capture the underlying patterns of the data.
- **Data enhancement:** Increase the size and diversity of the training dataset or employ more sophisticated data augmentation techniques to provide a richer set of examples for the model.
- **Hyperparameter exploration:** Conduct a more exhaustive search over the hyperparameter space, possibly using automated optimization techniques such as Bayesian optimization or grid/random search methods.
- **Transfer learning:** Explore the use of pre-trained models on similar tasks, adapting them through transfer learning to leverage established feature representations.
- **Error analysis:** Perform a detailed error analysis to identify specific failure modes in the current approach, which could inform targeted improvements in both model design and training strategy.

References