AutoCorrectComplete

Faheem Arif
Epoch
IIT Hyderabad
Hyderabad, India
ma23btech11010@iith.ac.in

MSR Siddarth
Epoch
IIT Hyderabad
Hyderabad, India
ma23btech11017@iith.ac.in

***Abstract*—This project develops a context-aware autocomplete system that understands sentence semantics and intent, improving real-time suggestions and corrections for diverse conversations.**

***Index Terms*—autocomplete, autocorrect, context-aware systems, semantic understanding, sentence intent, real-time text prediction, informal language processing, domain-specific typing assistance**

I. Introduction

Existing mobile autocorrect and autocomplete systems often fail to capture the broader context of user messages, leading to inappropriate suggestions or corrections. These systems focus mainly on word-level or character-level corrections and predictions without considering the overall meaning or intent of the sentence. The lack of contextual understanding results in suboptimal typing experiences, especially for informal language or domain-specific conversations. This project aims to fill this gap by developing a robust, context-aware system that understands sentence semantics and user intent while offering real-time corrections and completions.

II. Literature Survey

A. Autocorrect

a) Paper 1 - Correcting the Autocorrect: Context-Aware Typographical Error Correction via Training Data Augmentation: The paper above gives link to a pubicly available dataset that they have created using various methods. http://typo.nlproc.org. We are using this dataset for our autocorrect model for mid-term submission. But this dataset is derived from formal text and doesn't cover slangs and informal text. So, one of our future plans is to induce the typos and errors using the methods given in the paper starting with raw text that contains informal language. The section below describes the summary of various methods to induce ”human” errors to create a strong training data.

*1) **Dataset Creation and Methodology**:* This section describes a structured methodology for generating realistic datasets to train context-aware typographical error detection and correction models.

1. Collection of Annotated Data: An initial annotated dataset containing genuine typographical errors is collected. This small dataset provides insights into common error types and patterns.

2. Error Analysis and Statistics: The annotated data is analyzed to compute statistics (probability of a certain error occuring), focusing on frequent error types, such as:

- Insertions
- Deletions
- Substitutions
- Transpositions

These statistics guide the design of artificial errors that reflect real-world distributions.

3. Error Injection into Larger Corpora: Using the computed statistics, artificial errors are injected into a larger, error-free corpus to scale the training data. The process involves:

- **Error Type Selection:** Errors are sampled based on observed distributions.
- **Target Selection:** Words or contexts in the target corpus are identified for error injection.
- **Error Application:** Errors are applied while preserving grammatical structure to maintain realism.

4. Real-Word Error Simulation: Real-word errors—valid words used incorrectly in context—are simulated. This step leverages a dictionary-based spelling corrector to replace correct words with contextually inappropriate ones (we could do this by choosing the highest ranked word which is not correct), reflecting errors often introduced by autocorrect systems.

5. Dataset Compilation and Annotation: The augmented corpus is compiled into a structured dataset containing:

- Original (error-free) text
- Erroneous versions with injected errors
- Labels specifying error types and corrections

This structure supports supervised learning and detailed evaluation.

*2) **Model**:* Our current plan to leverage the pre-trained BERT-base or BERT-large model, which is designed for bidirectional context understanding. Fine-tune it for sequence-to-sequence correction tasks. We tokenize the text using BERT's WordPiece tokenizer and encode them as:

- **[CLS]** token at the start for sentence classification.
- Subword embeddings to handle unknown or rare words.
- **[SEP]** token to mark the end of input sequences.

The fine-tuning helps the model in identifying the ”typo” words. It scans for tokens that are ”out of context” (it does through its knowledge from both pre-training and fine-tuning). To select the word to replace the typo with, it creates a set of possible candidates that are ”close” to the typo so that its likely that the typo happened. From these set of canditate words it choses the word which is the most likely to fit in its surrounding context.

B. Autocomplete

a) Paper 1 - Gmail Smart Compose: Real-Time Assisted Writing: Why We Chose the Paper:

This paper examines a large-scale, real-time autocomplete system, offering insights into addressing challenges like latency, personalization, and multilingual support. Its focus on enhancing user experience through context-aware suggestions aligns well with our aim to create an advanced, semantic-driven autocomplete system.

Relevant Parts of the Paper

- Model Design:
  - Explores Transformers and LSTMs tailored for sequence prediction.
  - Implements context encoding (e.g., prior emails, subjects) for better prediction accuracy.
- Challenges Addressed:
  - Latency: Optimized models to provide real-time suggestions with under 60ms delay.
  - Personalization: Incorporates lightweight, adaptive models (Katz n-gram) to reflect user-specific writing styles.
- Metrics and Evaluation:
  - Uses ExactMatch and Log Perplexity to evaluate prediction quality.
  - Introduces beam search for generating diverse suggestions efficiently.

What the Paper Lacks

- Deep Semantic Understanding:
  - The system primarily predicts based on sequential tokens and lacks a robust mechanism for capturing deeper sentence semantics and user intent.
- Domain Adaptability:
  - Limited focus on informal or domain-specific conversational datasets.
- Advanced Contextual Encoding:
  - While useful, the context encoding methods could benefit from more sophisticated approaches like dynamic embeddings or multimodal inputs.

What We Can Improve

- Enhanced Contextual Integration:
  - Use BERT or GPT-like models for richer semantic understanding.
- Domain-Specific Customization:
  - Train models on informal or domain-specific datasets to improve suggestions for improving adaptability to slang or for diverse conversational contexts.
- Latency vs. Quality Trade-offs:
  - Investigate quantization techniques to allow the model to run locally on a laptop or a phone.

What We Plan on Using

- (optionally) Personalization Techniques:
  - Adapt their approach for lightweight, user-specific models while exploring neural alternatives to Katz n-gram.
- Beam Search for Real-Time Suggestions:
  - Incorporate their beam search strategy for generating top suggestions with a confidence threshold.
- Metrics Framework:
  - Leverage ExactMatch and Log Perplexity as evaluation metrics, with additional informal language performance indicators.

b) Paper 2 - When Does Text Prediction Benefit from Additional Context? An Exploration of Contextual Signals for Chat and Email Messages: Why We Chose the Paper:

This paper investigates how additional contextual signals (e.g., prior messages, time, subject) enhance text prediction performance for chat and email platforms, addressing key challenges in real-time communication. It aligns with our project’s focus on creating a context-aware system by exploring contextual signals and their integration into prediction models.

Relevant Parts of the Paper:

- Contextual Signals:
  - Explores the use of prior messages, message composition time, and subject as context signals for improving predictions.
  - Highlights the differential utility of these signals between chat (short, rapid messages) and email (long, formal messages).
- Experimental Results:
  - Shows significant gains in chat prediction using a 5-minute window of prior messages from both senders (up to 18.6% relative improvement in certain metrics).
  - Time context is the most beneficial for emails, yielding moderate gains (˜2-3% in match rate and estimated characters saved).
- Scenarios Explored:
  - Demonstrates that chat and email systems benefit from custom-tailored models due to differing communication styles and message structures.
- Model Architecture:
  - Uses a production-oriented LSTM-based model optimized for low-latency, with contextual signals integrated as a single input string.

What the Paper Lacks:

- Advanced Model Architectures:
  - Relies on simpler LSTM-based models rather than exploring advanced architectures like Transformers, which may better leverage contextual signals.

- Integration of Richer Context:
  - Limited exploration of combining hierarchical or multimodal contextual signals for a deeper understanding of user intent.
- Domain-Specific Contexts:
  - Does not address informal or domain-specific conversations explicitly, which are essential for systems dealing with diverse user inputs.

What We Can Improve:

- Advanced Contextual Encoding:
  - Use hierarchical models (e.g., hierarchical RNNs) or Transformers to capture complex contextual dependencies.
- Domain-Specific Training:
  - Fine-tune the model on datasets with informal and domain-specific language to improve adaptability and accuracy.
- Hybrid Models:
  - Investigate hybrid approaches combining low-latency architectures with advanced models to balance real-time performance and contextual understanding.

What We Plan on Using:

- Contextual Signal Integration:
  - Adopt their techniques for integrating contextual signals (e.g., prior messages, time) into input strings for seamless preprocessing.
- Message Aggregation Strategies:
  - Leverage their message aggregation modes (e.g., Both-Senders for chat) to enhance conversational predictions.
- Evaluation Metrics:
  - Use metrics like Match Rate and Estimated Characters Saved per suggestion to assess model performance comprehensively.