Advancing AI-Powered Intelligent Writing Assistance across Multiple Languages

The Strategic Research team at Grammarly is constantly exploring how LLMs can contribute to our mission of improving lives by improving communication. They showed that LLMs trained specifically for text editing can be of higher quality and more performant. We focused our experiments on English. However, most foundational models produced since our previous work (from 2024 and on) have been multilingual. They can understand instructions and have conversations in multiple languages.

Our work improves on the limitations of prior work by:

1. Supporting multiple editing tasks

2. Accepting edit instructions in multiple languages

3. Taking edit instructions in languages that don’t match the edited text

A "multilingual language editing design pattern" refers to a software design approach that allows users to edit content in multiple languages simultaneously, often by separating the content from its translation, enabling efficient management and localization of text across different language versions within a single system.

Key elements of this pattern:

- **Content Abstraction:**

  The core content (text, images, layout) is stored separately from its translations, allowing for independent modifications while maintaining consistency across languages.

- **Language Keying:**

  Each language version is identified by a unique "language key" to easily switch between translations and manage which languages are available.

- **Translation Storage:**

  A dedicated storage mechanism is used to hold translated content for each language key, potentially utilizing a translation management system (TMS) for large-scale projects.
- **Context-Aware Translation:**
  The system can consider context when translating, ensuring accurate rendering based on the surrounding text and situation.

Implementation Approaches:

- **Database Design:**
- A "content" table storing the base content.
- A "translation" table linking each content entry to its translated versions, including language keys.
- **API Integration:**
- Exposing APIs to fetch content in a specific language based on the user's selected language key.
- **Frontend Handling:**
- User interface elements to select preferred language.
- Dynamically loading translations based on the selected language.

Benefits:

- Improved Efficiency: Reduces redundancy by managing translations centrally, simplifying updates across multiple languages.
- Localization Flexibility: Enables tailoring content to specific regions and cultural nuances.
- Scalability: Can easily accommodate new languages by adding new translation entries.

Example Use Cases:

- Multilingual Websites: Allowing users to view and edit content in different languages on a single website.
- Content Management Systems (CMS): Providing multilingual editing capabilities within a content management interface.
- Mobile Apps: Localizing app content for users in different regions.

How to Approach them

User will need to compare with specimen. Show both English and non-english versions simultaneously, on one page/in one window.

1. Help user to understand overall task by using form recognition. Make translated form look absolutely the same as original one.
2. Help user to work with form by highlighting corresponding field in the second form. When he’s working on Author (ru), highlight Author (en) too.
3. In bigger text forms, help user not to lose his attention focus. Highlight corresponding sentence/paragraph in another form. When he’s working on the second sentence in Comment (ru), highlight second sentence in Comment (en).