Table of Contents

This report includes five key sections:

1) Summary of App Performance Analysis

This analysis is based on two runs of our application: one using provided example text translated across Spanish, French, German, Japanese, Arabic, Hindi, and Portuguese with OpenAI, DeepSeek, and Anthropic models, and another using text scraped from the BrokerChooser webpage, processed similarly. The app's performance was evaluated using BLEU, METEOR, Fluency, and Word Matching metrics against Google Translate.

The CSV files generated from these two runs are stored in the "translation_runs" folder to help you better understand the app's performance.

- **What Works Well**:
    - The app successfully translates text across all tested languages and models, with high fluency scores (often >0.9) indicating readable, grammatically correct outputs.
    - Web scraping functionality is effective, retrieving content accurately and maintaining translation quality comparable to the example text run.
    - OpenAI and Anthropic models perform strongly in most languages, particularly Spanish, French, and Portuguese (e.g., OpenAI Spanish: BLEU 0.51, Fluency 0.99).
- **What Doesn't Work Well**:
    - Japanese translations with Anthropic are severely flawed (BLEU ~0, Fluency 0.47), likely due to tokenization or grammar issues.
    - German and Hindi show lower quality with OpenAI (e.g., German BLEU 0.18, METEOR 0.28), suggesting challenges with syntactic complexity or script handling.
    - DeepSeek's performance appears weaker (less quantified but qualitatively less nuanced), indicating potential optimization gaps across models.

Conclusion: The app functions well overall, with robust translation and scraping capabilities. However, improvements are needed for Japanese (Anthropic), German, and Hindi (OpenAI), and DeepSeek integration to ensure consistent quality across all languages and models.

2) Explanation of Evaluation Methodologies Used in the App

In the Translation Engine App, we evaluate the quality of translations generated by different models (OpenAI, DeepSeek, Anthropic) using a set of automated metrics. These metrics compare the app's translations to those produced by Google Translate, which serves as our baseline for quality assessment. The following methodologies are implemented in the evaluate_dataset function within the translation.py script:

- **BLEU (Bilingual Evaluation Understudy) Score**
  This metric measures the precision of word sequences (n-grams) in the app's translation compared to the baseline. It assesses how many overlapping sequences of words (e.g., unigrams, bigrams, up to 4-grams) exist between the two translations. A higher BLEU score indicates closer structural similarity to the baseline.

- **METEOR (Metric for Evaluation of Translation with Explicit ORdering) Score**
  METEOR evaluates both precision and recall, accounting for linguistic variations such as synonyms and word stems. It aligns words between the app's translation and the baseline, rewarding matches in meaning and sentence structure. This makes it more nuanced than BLEU, often aligning better with human perception of translation quality.

- **Fluency Score**
  A custom metric based on the word count ratio between the app's translation and the baseline. It assumes that a fluent translation should have a similar length to a high-quality reference translation. The score is calculated as the minimum of the ratios of word counts (app to baseline and baseline to app), yielding a value between 0 and 1.

- **Word Matching Percentage**
  This measures the proportion of unique words in the app's translation that also appear in the baseline translation. It provides a simple indicator of lexical overlap, showing how well the app captures the vocabulary of the baseline.

These metrics together offer a multi-faceted evaluation of translation quality:

- **BLEU** focuses on structural accuracy.
- **METEOR** emphasizes semantic similarity.
- **Fluency** assesses readability through length consistency.
- **Word Matching** highlights vocabulary alignment.

This combination ensures a comprehensive analysis of the app's performance across different dimensions of translation quality.

3) Detailed Explanation of How the Methodologies Work and Comparison with Baseline Translator

Each evaluation methodology compares the app's translations to a baseline provided by Google Translate, a widely recognized standard for machine translation due to its reliability and extensive language support. Below, we explain how each metric operates and why the baseline comparison is essential:

BLEU Score

- **How It Works**
  BLEU calculates the geometric mean of n-gram precisions (typically up to 4-grams) between the app's translation and the baseline. For example, if the app translates "The quick brown fox" to Spanish as "El rápido zorro marrón" and Google Translate produces the same, BLEU checks how many 1-word, 2-word, 3-word, and 4-word sequences match. It also applies a brevity penalty if the app's translation is too short, ensuring it doesn't favor overly concise outputs. In the app, this is implemented using NLTK's sentence_bleu function, tokenizing both translations into words and comparing them.
- **Comparison with Baseline**
  Google Translate's output serves as the reference translation. BLEU quantifies how closely the app's word sequences align with this reference. A high BLEU score (e.g., 0.8 or above) suggests that the app's translation mirrors the baseline's structure, which is often a sign of accuracy. For instance, if the app misses a key phrase or reorders words significantly, the BLEU score drops, highlighting structural differences.

METEOR Score

- **How It Works**
  METEOR aligns words between the app's translation and the baseline, considering exact matches, synonyms (via WordNet), stemmed forms, and paraphrases. It computes precision (how many words in the app's translation appear in the baseline) and recall (how many baseline words are captured by the app), then combines them into an F-mean score. A penalty is applied for fragmented alignments (e.g., words matched out of order). In the app, NLTK's meteor_score function handles this, tokenizing the translations and scoring them accordingly.
- **Comparison with Baseline**
  The baseline from Google Translate provides a semantic benchmark. METEOR's ability to account for meaning (e.g., accepting "fast" for "quick" in English-to-Spanish translations) makes it more sensitive to content preservation than BLEU's strict sequence matching. A

high METEOR score (e.g., 0.7 or above) indicates that the app’s translation conveys similar meaning to the baseline, even if word choices differ slightly.

Fluency Score

- **How It Works**
  This custom metric calculates the ratio of word counts between the app’s translation and the baseline. For example, if the app’s translation has 10 words and Google Translate’s has 12, the ratios are 10/12 (0.833) and 12/10 (1.2), and the fluency score is the minimum of these (0.833). The idea is that fluent translations tend to have lengths similar to high-quality references. In the app, this is computed by tokenizing both translations with NLTK’s word_tokenize and comparing their lengths.
- **Comparison with Baseline**
  Google Translate’s word count acts as the target length. A fluency score near 1 (e.g., 0.9–1.0) suggests the app produces translations with comparable readability to the baseline. Significant deviations (e.g., a score of 0.5) might indicate over-simplification or verbosity, prompting further investigation into translation quality.

Word Matching Percentage

- **How It Works**
  This metric identifies unique words in the app’s translation and the baseline, then calculates the percentage of app words that appear in the baseline. For example, if the app’s translation has the unique words {el, rápido, zorro, marrón} and the baseline has {el, zorro, marrón, veloz}, the intersection is {el, zorro, marrón}, yielding a word matching percentage of 3/4 (75%). In the app, this is implemented by converting tokenized words to sets and computing their overlap.
- **Comparison with Baseline**
  The baseline’s vocabulary serves as the reference. A high percentage (e.g., 80% or more) shows that the app uses similar terms, suggesting it captures key concepts effectively. However, this metric doesn’t consider word order or context, so it complements BLEU and METEOR rather than standing alone.

Why Google Translate as the Baseline?

Google Translate is chosen as the baseline because it offers consistent, high-quality translations across the app’s supported languages (Spanish, French, German, Japanese, Arabic, Hindi, Portuguese). Its robust handling of syntax, semantics, and diverse linguistic structures makes it a reliable standard for comparison. By benchmarking against Google Translate, we can:

- Assess how competitive the app’s models (OpenAI, DeepSeek, Anthropic) are against an industry leader.
- Identify strengths (e.g., better fluency) or weaknesses (e.g., lower semantic accuracy) in specific languages or models.
- Provide an objective measure of quality that aligns with widely accepted translation standards.

In the app, the evaluate_dataset function retrieves Google Translate outputs using the deep_translator library, tokenizes them alongside the app’s translations, and applies the above metrics to generate a detailed evaluation CSV. This process ensures a systematic comparison that informs future improvements.

4) Language Prioritization for Cost Optimization

To optimize costs while expanding the translation capabilities of the BrokerChooser platform or any similar translation-focused project, a strategic and multi-faceted approach to language prioritization is essential. The selection process should balance market opportunities, user needs, and resource constraints, ensuring that investments in translation yield maximum reach and impact with minimal overhead. Here's a detailed breakdown of how to pick and prioritize languages effectively:

- **Market Size and Growth Potential**: Prioritizing languages spoken in regions with large or rapidly expanding economies is a cornerstone of cost optimization. Spanish, for example, is a high-priority language due to its prevalence across Latin America and Spain, covering markets like Mexico, Argentina, and Colombia, which are experiencing growth in financial services demand. Portuguese targets Brazil, South America's largest economy, making it a critical choice despite its regional concentration. Hindi is indispensable for India, a country with a massive population and an increasingly digital economy hungry for localized financial content. Japanese, though limited to Japan, is vital given the country's advanced financial markets and high per-capita engagement with brokerage platforms.

- **User Base**: The size and potential of the user base speaking a language directly influence its priority. Spanish and French stand out in Europe, with Spanish also bridging to the Americas, serving millions of native speakers. French extends beyond France into Belgium, Switzerland, and a growing African market, where francophone countries are emerging as financial hubs. Arabic, while complex, taps into the Middle East and North Africa, regions with significant wealth and interest in investment platforms. Conversely, German might be deprioritized in contexts where English proficiency is high among target users, such as in Germany, reducing the immediate need for translation.

- **Cost of Translation**: Translation costs vary widely based on language complexity, script requirements, and translator availability. Languages like Arabic and Japanese often incur higher costs due to their non-Latin scripts and the need for translators with specialized financial expertise. Arabic's right-to-left orientation and regional dialects (e.g., Gulf vs. Levantine Arabic) add further expense, as does Japanese's intricate grammar and honorifics, which demand cultural precision in business contexts. Spanish and French, by contrast, benefit from a larger pool of translators and simpler integration into existing systems, making them more cost-effective initial targets.

- **Competitive Landscape**: Staying competitive requires aligning language offerings with those of rival platforms. If competitors already support German or Arabic, failing to match these could cede market share, even if costs are higher. However, in markets where English suffices, like parts of Northern Europe, focusing on unique differentiators (e.g., Hindi or Portuguese) could provide a cost-effective edge over competitors who over-invest in saturated languages.

- **Regulatory Requirements**: Legal mandates can elevate a language's priority regardless of cost. In Quebec, Canada, French is legally required for financial services, making it non-negotiable despite a smaller user base. Similarly, some Middle Eastern countries mandate Arabic for official documentation, pushing it up the list even with higher translation costs. Ignoring these requirements risks penalties and market exclusion, outweighing any short-term savings.
- **Cultural Nuances**: Languages where cultural context heavily influences translation quality require additional resources, impacting cost. Arabic translations, for instance, must account for religious and social sensitivities, often necessitating region-specific versions. Japanese demands attention to formality levels (e.g., keigo for business), increasing the need for skilled translators. These factors suggest a phased approach: prioritize languages with manageable cultural complexity (e.g., Spanish, French) before tackling those requiring deeper customization.

Based on this analysis, a tiered prioritization strategy emerges:

- **High Priority**: Spanish, French, Portuguese, Hindi, and Japanese. These languages offer broad market coverage, significant user bases, and, in some cases, regulatory necessity, balancing cost with impact.
- **Medium Priority**: Arabic, where market potential is high but complexity and cost require a gradual rollout.
- **Low Priority**: German or similar languages, where English adoption reduces urgency, allowing resources to focus elsewhere.

This approach ensures cost optimization by targeting high-impact languages first, scaling to costlier ones as the platform grows and budgets allow.

5) Future Improvements for the Translation Engine

Enhancing the translation engine for your project, currently built with tools like deep-translator, OpenAI, Anthropic, and web scraping capabilities, requires a forward-looking strategy to boost accuracy, efficiency, and scalability. Below is an extended plan for future improvements, building on existing ideas and introducing new avenues for development:

- **Language-Specific Tokenizers**: Integrating specialized tokenizers like MeCab for Japanese or camel-tools for Arabic can elevate translation quality by addressing language-specific nuances. These tools excel at segmenting complex scripts and grammars, ensuring that tokenization aligns with linguistic rules rather than generic splitting. For example, MeCab handles Japanese's lack of spaces adeptly, while camel-tools manages Arabic's morphological richness, reducing errors in downstream translation and evaluation.
- **Switch to Google Cloud Translation API**: Transitioning from deep-translator to the Google Cloud Translation API offers a leap in baseline translation quality. While costlier, its advanced neural machine translation supports over 100 languages with superior contextual understanding, making it ideal for financial texts where precision matters. This switch could serve as a benchmark or primary engine, with custom models layered on top for specific domains.
- **Enhanced Web Scraping**: Strengthening web scraping with cloudscraper already helps bypass Cloudflare, but further refinements, like adaptive delays, proxy rotation, or machine learning to detect and mimic human browsing patterns, can ensure robust content extraction from diverse sites. This is critical for users translating dynamic web pages, expanding the engine's utility beyond static text.
- **Support for Additional Languages**: Expanding language coverage based on user feedback and market trends keeps the engine relevant. Adding languages like Mandarin Chinese (for China's massive market) or Russian (for Eastern Europe) could follow the prioritization strategy above, phased in as demand justifies the investment.
- **Caching Mechanism**: Implementing a caching system, e.g., using Redis or a local SQLite database, can store frequent translations, slashing API costs and latency. For instance, caching common phrases like "Best forex brokers in [country]" across languages could cut redundant calls, especially for high-traffic content.
- **Model Fine-Tuning**: Tailoring models to financial terminology enhances relevance for BrokerChooser's audience. This could involve fine-tuning OpenAI's GPT-4o or Anthropic's Claude on a corpus of brokerage reviews, SEC filings, or user-generated content,

ensuring translations capture industry-specific jargon like “leverage” or “spread” accurately.

- **Contextual Translation**: Improving context retention across longer texts, e.g., full articles or web pages, is vital. Techniques like sentence embedding with transformers or paragraph-level translation prompts can preserve meaning, avoiding disjointed outputs. For example, ensuring “bank” translates as “financial institution” rather than “riverbank” based on surrounding text boosts coherence.
- **User Feedback Loop**: Building a feedback interface where users rate translations or suggest corrections creates a self-improving system. This data can retrain models, prioritize problematic languages, or flag recurring errors (e.g., placeholder mishandling), driving iterative refinement.
- **Multilingual UI Support**: Extending the Streamlit interface to support multiple languages, beyond just translating content, enhances accessibility. This includes localizing buttons, labels, and error messages, ensuring non-English users navigate effortlessly. Libraries like streamlit-i18n could streamline this process.
- **Performance Optimization**: Scaling the engine requires speed and efficiency. Optimizing API calls (e.g., batching requests), compressing data transfers, or offloading computation to GPUs can handle larger volumes without lag, critical as user bases grow.
- **Security and Privacy**: Safeguarding user data during translation is non-negotiable. Anonymizing inputs before API calls, encrypting cached translations, and complying with GDPR or CCPA standards builds trust and meets legal requirements, especially for financial data.

These improvements transform the engine into a robust, user-centric tool. Start with low-hanging fruit like caching and tokenizers, then scale to advanced features like fine-tuning and contextual translation as resources permit, ensuring a balance of immediate wins and long-term growth.