Gadhwali Translator (EN ⇄ HI ⇄ GDW)

🧩 1 Project Summary

This project is a Flask-based multilingual translation API designed to handle translations between English, Hindi, and Gadhwali. It combines rule-based dictionary lookups, lightweight NLP grammar correction, and Google Translate to support both common and low-resource languages.

The system was built with the goal of enabling:

- Fast offline translation using local CSV dictionaries
- Grammar-aware sentence correction for Gadhwali and Hindi
- Extendability for additional regional languages in the future

---

2. How It Works

2.1 Architecture Overview

User Input

  ⬇

API (Flask: app.py)

  ⬇

Translation Logic (translation.py)

  └──➔ Dictionary Match (phrase_dict)

  └──➔ Word-by-word fallback translation

  └──➔ Grammar correction (Hindi or Gadhwali)

  └──➔ Google Translate fallback (EN-HI)

  ⬇

JSON Response

---

3. Functional Modules

3.1 Dictionary-Based Translation

- Uses translations.csv to map known English ↔ Hindi ↔ Gadhwali phrases and words.
- Implemented in utils/functions.py and accessed via phrase_dict.
- Prioritizes **exact full-sentence match**, then falls back to **word-by-word substitution**.

3.2 Grammar Correction

Hindi Correction:

- Follows the typical **Subject-Object-Verb (SOV)** order.
- Corrects helping verbs like "है", "हूँ", "था", etc. to appear at the end of a sentence.
- Implemented in utils/hindi_correction.py.

Gadhwali Correction:

- Since Gadhwali lacks a formal POS tagger, we use two strategies:
    1. **Stanza (Hindi model)**: Parses sentence structure to extract nsubj, dobj, aux, etc.
    2. **Fallback Rule-Based POS Mapping**: Manually categorizes tokens into parts of speech using custom keyword sets.
- Final order: Subject + Object + Negation + Verb + Auxiliary + Modal + Others
- Implemented in utils/gadhwali_correction.py

3.3 Google Translate Fallback

- If user requests EN-HI or HI-EN and the phrase is not found locally, Google Translate (via googletrans==4.0.0-rc1) is used.
- Ensures the system functions even without full offline support for EN-HI.

---

4. Tools & Libraries Used

Purpose	**Tools / Libraries**
Backend	Flask
Dictionary Parsing	pandas
Grammar Correction	Stanza (Hindi model), spaCy (EN)
Offline Embedding	Gensim FastText (train_model.py)

Purpose	Tools / Libraries
Translation API	Googletrans (optional fallback)
NLP Parsing	Rule-based POS, Stanza

Why Stanza?

Stanza is a neural pipeline built by Stanford NLP that supports several Indian languages, including Hindi. It provides:

- Tokenization
- POS tagging
- Lemmatization
- Dependency parsing

We used the Hindi model to parse Gadhwali as a workaround.

Why spaCy?

spaCy is used minimally here, mainly for English parsing in the future (currently underutilized).

Rule-Based POS Mapping

Gadhwali lacks formal NLP support. We use custom sets to mimic POS tagging:

- Subjects: {"म्यर", "तुं", "तू"}
- Verbs: {"जां", "खाण"}
- Modals: {"सकदु", "चौन"}
- Helping verbs: {"च", "छू"}

---

5. File Structure Summary

```
backend/
 ├── app.py              # Main API route
 ├── translation.py      # Logic handler
 ├── train_model.py      # Optional: FastText training
├── translations.csv        # Core dictionary
|
├── utils/
|   ├── __init__.py
|   ├── functions.py        # Internet check, dictionary loader
|   ├── english_to_gadhwali.py
|   ├── hindi_to_gadhwali.py
|   ├── hindi_correction.py  # Hindi structure fixer
|   └── gadhwali_correction.py  # Gadhwali grammar reordering
```

---

6. Sample Flow: "i not can go"

1. Lowercased & stripped → "i not can go"
2. Word-by-word map: i → म्यर, not → नै, can → सकदु, go → जां
3. Pre-correction: "म्यर नै सकदु जां"
4. Gadhwali NLP fix → "म्यर नै जां सकदु"

---

7. Installation

```
pip install -r requirements.txt

python -m stanza.download hi

python app.py
```

---

8. API Usage

Request:

```
POST /translate

{

  "text": "i not can go",

  "source_lang": "en",

  "target_lang": "gadhwali"

}
```

Response:

```
{

  "translation": "म्यर नै जां सकदु"

}
```

---

9. Future Scope

- Add more languages: Kumaoni, Nepali
- Deploy on cloud (Render/Railway)
- Add React/HTML frontend
- Admin panel to edit dictionary entries
- Use OpenAI or Hugging Face transformer models for advanced translation

---

10. Conclusion

This project bridges the gap between modern language tech and underrepresented regional languages like Gadhwali. It's fast, offline-capable, extendable, and a great foundation for more advanced multilingual tools.

Whether for preserving heritage languages or enabling real-world communication in rural areas, this system is practical and future-ready.