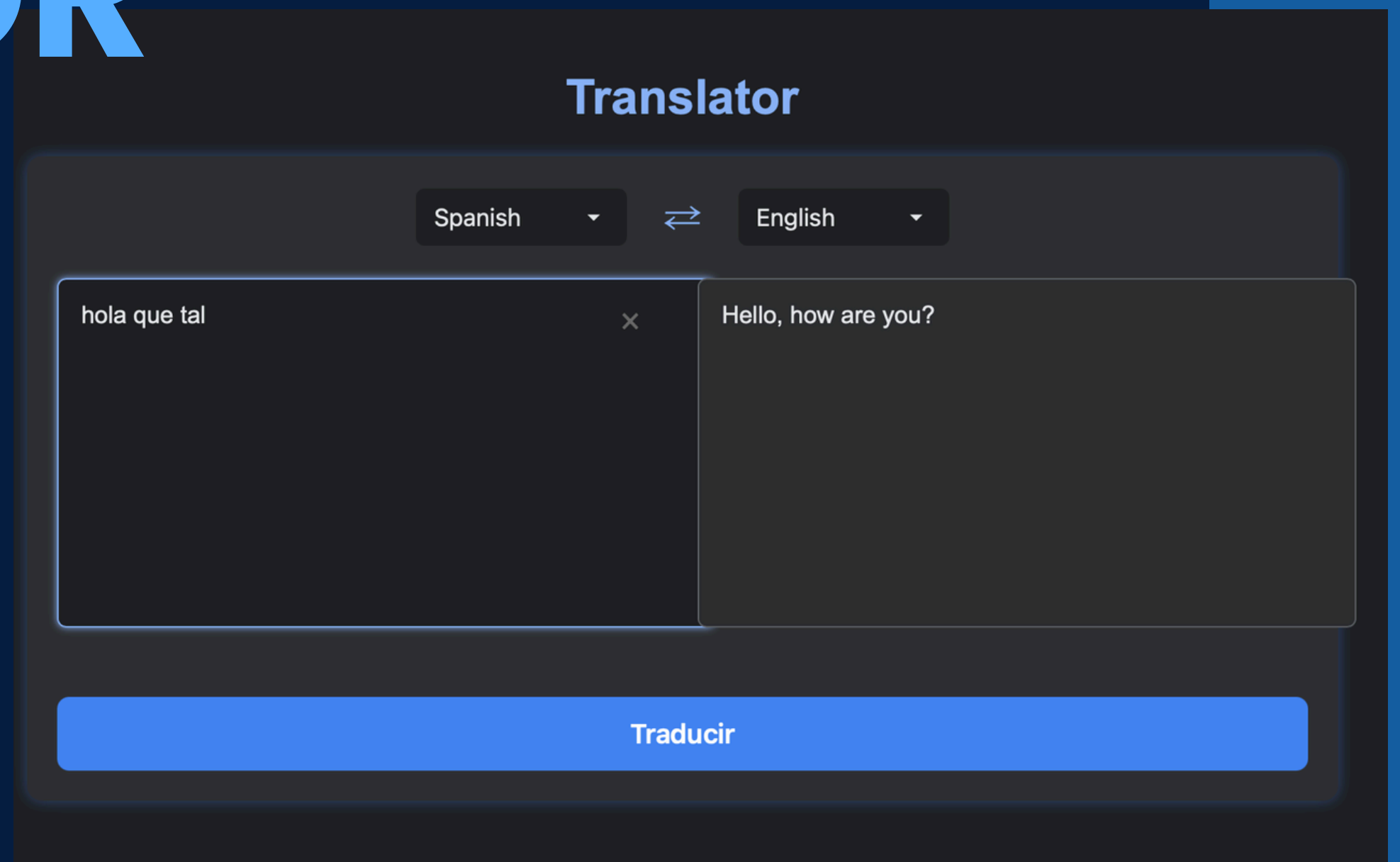


AUTOMATIC TRANSLATOR

BASED ON
TRANSFORMERS

Ainhoa Lucia Pérez González
Jose Alejandro López Garcés



The image shows a web application titled "Translator". It features two dropdown menus for language selection, currently set to "Spanish" and "English", with a double-headed arrow between them. Below the menus are two text input fields. The left field contains the Spanish text "hola que tal" and has a small "x" icon to its right. The right field contains the English translation "Hello, how are you?". At the bottom of the interface is a large blue button labeled "Traducir".

OBJECTIVE AND SCOPE

OBJECTIVE

Create a functional automatic translator that allows text translation between multiple languages using pretrained Transformer models.

SCOPE

- Plain text translation among common languages (English, Spanish, French, German).
- REST API and a simple web interface for user interaction.



METHODOLOGY

Hugging Face is an open-source platform and library that provides easy access to state-of-the-art pretrained Transformer models for natural language processing tasks such as translation, summarization, and more.

- 01 Use Hugging Face's Transformers library.
- 02 Python backend with Flask to serve translation requests.
- 03 Simple frontend in HTML/JavaScript that communicates with the backend.
- 04 For unsupported language pairs, use pivot translation via English (e.g., French → English → German).



INTERFACE

Translator

Spanish ▾

↔

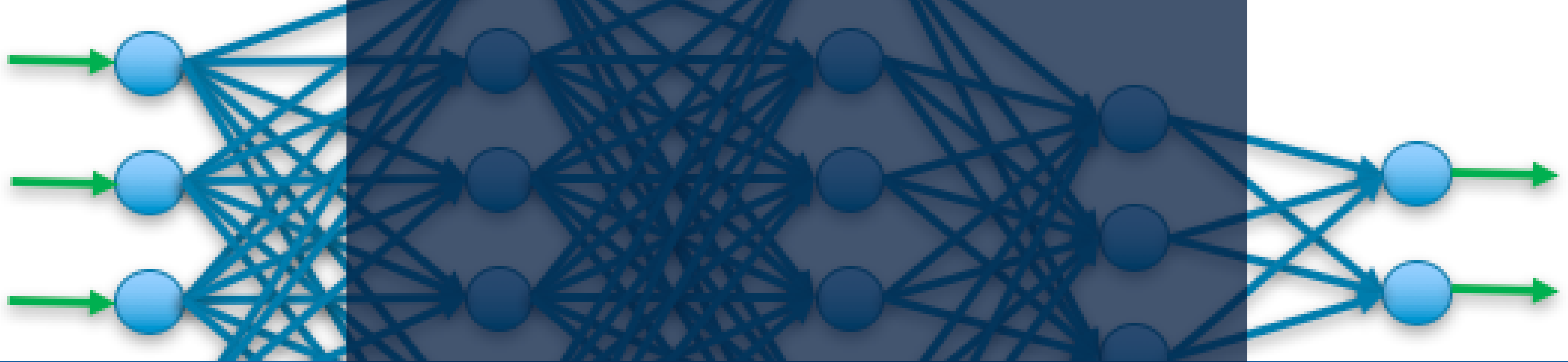
English ▾

hola que tal

×

Hello, how are you?

Traducir



Theoretical Background

Transformer models rely on a self-attention mechanism that allows them to weigh the importance of each word in a sentence relative to the others, enabling them to capture long-range dependencies and contextual relationships effectively. Unlike traditional sequential models (like RNNs), Transformers process the entire input simultaneously, which leads to better understanding of the sentence structure and improves the quality and fluency of translations.

CODE EXPLANATION

TRANSLATE_TEXT(TEXT, SRC_LANG, DEST_LANG)

```
@app.route('/translate', methods=['POST'])
def translate_text():
    try:
        text = request.form['text']
        src_lang = request.form['src_lang']
        dest_lang = request.form['dest_lang']

        if not text:
            return jsonify({'error': 'No text provided'}), 400

        # Primero intentamos traducción directa
        translation = translate_with_model(text, src_lang, dest_lang)
        if translation is not None:
            return jsonify({'translation': translation, 'src_lang': src_lang, 'dest_lang': dest_lang})

        # Si no hay modelo directo y ni src ni dest es inglés, hacemos traducción cascada vía inglés
        if src_lang != 'en' and dest_lang != 'en':
            # Traducción src->en
            inter = translate_with_model(text, src_lang, 'en')
            if inter is None:
                return jsonify({'error': f'No model for {src_lang} to en'}), 400
            # Traducción en->dest
            translation = translate_with_model(inter, 'en', dest_lang)
            if translation is None:
                return jsonify({'error': f'No model for en to {dest_lang}'}), 400
            return jsonify({'translation': translation, 'src_lang': src_lang, 'dest_lang': dest_lang})

        return jsonify({'error': 'Language pair not supported'}), 400

    except Exception as e:
        return jsonify({'error': str(e)}), 500
```

This function handles the translation of a given text from a source language (src_lang) to a target language (dest_lang). It first tries to load and use a direct translation model for the language pair. If no direct model is available, it performs a pivot translation by first translating the text into English, then from English to the target language. This ensures that the system can translate language pairs that are not directly supported. This function is essential because it centralizes the translation logic and model selection, providing flexibility and scalability to the translation system.

LOAD_TRANSLATOR(SRC_LANG, DEST_LANG)

This function loads the translation pipeline (model) for a given source–target language pair. It first checks if the model is already cached to avoid reloading it repeatedly, which improves performance. If the model isn't cached, it retrieves the appropriate pretrained model from the Hugging Face repository and caches it for future use. If no model exists for the requested pair, it returns None.

```
def load_model(src_lang, tgt_lang):
    key = f"{src_lang}-{tgt_lang}"
    if key not in MODELS:
        return None

    if MODELS[key]['model'] is None:
        try:
            MODELS[key]['tokenizer'] = AutoTokenizer.from_pretrained(MODELS[key]['model_name'])
            MODELS[key]['model'] = AutoModelForSeq2SeqLM.from_pretrained(MODELS[key]['model_name'])
        except Exception as e:
            print(f"Error loading model {key}: {str(e)}")
            return None

    return MODELS[key]
```


THANK YOU

