Word Similarity Analyzer - Documentation

Overview

This project analyzes word similarities across different languages using the Levenshtein distance method. It translates words between languages and computes similarity matrices to visualize relationships.

Table of Contents

Installation

Requirements

```bash
pip install textdistance
pip install deep-translator
pip install networkx
pip install matplotlib
```

Project Structure

```
PPwP/
├── data/
│   ├── animals.txt
```

```
│   ├── buildings.txt
│   └── fruits.txt
├── src/
│   ├── utils/
│   │   ├── fileUtils.py
│   │   ├── similarity.py
│   │   ├── translate.py
│   │   ├── overall_similarity.py
│   │   └── displayUtils.py
│   └── main.py
└── results/
    ├── translations/
    ├── similarities/
    └── *_similarity_graph.png
```

Quick Start

1. Run the program:

```bash
python src/main.py
```

2. Select 2-4 languages from the available options
3. Choose to analyze a single file or directory
4. View results in the `results/` folder

Module Reference

fileUtils.py

Module for file input/output operations.

Functions

`get_words_from_file(file_path)`

```python
```

```
def get_words_from_file(file_path):
    """
    Read words from a text file, one word per line.

    Args:
        file_path (str): Path to the input text file

    Returns:
        list: List of words with whitespace stripped, empty lines removed

    Example:
        >>> words = get_words_from_file("data/animals.txt")
        >>> print(words)
        ['dog', 'cat', 'bird']
    """
```

save_words_to_file(words, file_path)

python

```
def save_words_to_file(words, file_path):
    """
    Save a list of words to a text file, one word per line.
    Creates parent directories if they don't exist.

    Args:
        words (list): List of words to save
        file_path (str): Destination file path

    Returns:
        None

    Example:
        >>> words = ['perro', 'gato', 'pájaro']
        >>> save_words_to_file(words, "results/translations/animals_es.txt")
    """
```

save_similarity_matrix(words1, words2, matrix, file_path)

python

```
def save_similarity_matrix(words1, words2, matrix, file_path):
    """
    Save a similarity matrix to a CSV file.
    Rows represent words1, columns represent words2.

    Args:
        words1 (list): List of words for rows
        words2 (list): List of words for columns
        matrix (list of lists): 2D similarity matrix (values 0.0-1.0)
        file_path (str): Destination CSV file path

    Returns:
        None

    Example:
        >>> words_en = ['dog', 'cat']
        >>> words_es = ['perro', 'gato']
        >>> matrix = [[0.85, 0.32], [0.28, 0.91]]
        >>> save_similarity_matrix(words_en, words_es, matrix, "results/similarities/animals_en_es.csv")
    """
```

---

similarity.py

Module for computing word similarity using Levenshtein distance.

Functions

compute_similarity(word1, word2)

```
python
```

```
def compute_similarity(word1, word2):
    """
    Calculate normalized similarity between two words using Levenshtein distance.

    Args:
        word1 (str): First word
        word2 (str): Second word

    Returns:
        float: Similarity score between 0.0 (completely different) and 1.0 (identical)

    Example:
        >>> compute_similarity("cat", "cat")
        1.0
        >>> compute_similarity("cat", "dog")
        0.0
        >>> compute_similarity("kitten", "sitting")
        0.571  # approximately
    """
```

---

translate.py

Module for translating words between languages using Google Translate.

Functions

translate_word(word, lang)

```
python
```

```
def translate_word(word, lang):
    """
    Translate a single word to the target language.

    Args:
        word (str): Word to translate
        lang (str): Target language code (e.g., 'es', 'fr', 'de')

    Returns:
        str: Translated word in lowercase, or original word if translation fails

    Example:
        >>> translate_word("hello", "es")
        'hola'
        >>> translate_word("cat", "fr")
        'chat'
    """
```

translate_words(words, lang)

```
python

def translate_words(words, lang):
    """
    Translate a list of words to the target language.

    Args:
        words (list): List of words to translate
        lang (str): Target language code (e.g., 'es', 'fr', 'de')

    Returns:
        list: List of translated words in the same order

    Example:
        >>> words = ["hello", "world", "cat"]
        >>> translate_words(words, "es")
        ['hola', 'mundo', 'gato']
    """
```

---

overall_similarity.py

Module for graph operations and matrix calculations.

Functions

add_connection(graph, node1, node2, label)

```python
def add_connection(graph, node1, node2, label):
    """
    Add nodes and an edge to a NetworkX graph.

    Args:
        graph (networkx.Graph): NetworkX graph object
        node1 (str): First node identifier
        node2 (str): Second node identifier
        label: Edge label (typically similarity percentage)

    Returns:
        None (modifies graph in place)

    Example:
        >>> import networkx as nx
        >>> G = nx.Graph()
        >>> add_connection(G, "en", "es", "85.5%")
    """
```

diagonal_average(matrix)

```python
def diagonal_average(matrix):
    """
    Calculate the average of diagonal elements in a matrix.
    Used to compute overall similarity between language word lists.

    Args:
        matrix (list of lists): 2D matrix (can be non-square)

    Returns:
        float: Average of diagonal elements, or 0 if matrix is empty

    Example:
        >>> matrix = [[1.0, 0.5], [0.3, 0.9]]
        >>> diagonal_average(matrix)
        0.95  # average of 1.0 and 0.9
    """
```

---

displayUtils.py

Module for user interface, language selection, and file processing.

Constants

LANGUAGE_MAP

```python
LANGUAGE_MAP = {
    "english": "en",
    "polish": "pl",
    "spanish": "es",
    "french": "fr",
    "german": "de",
    # ... and more
}
```

Dictionary mapping full language names to ISO language codes.

Functions

display_available_languages()

```python
def display_available_languages():
    """
    Display all available languages in a formatted 3-column table.

    Args:
        None

    Returns:
        None (prints to console)

    Example output:
        ============================================================
        AVAILABLE LANGUAGES:
        ============================================================
        Czech          Danish         Dutch
        English        Finnish        French
        ...
    """
```

get_language_code(language_name)

```python
```

```
def get_language_code(language_name):
    """
    Convert a language name to its ISO language code.
    Case-insensitive matching.

    Args:
        language_name (str): Full language name (e.g., "English", "spanish", "POLISH")

    Returns:
        str or None: Language code if found, None otherwise

    Example:
        >>> get_language_code("English")
        'en'
        >>> get_language_code("spanish")
        'es'
        >>> get_language_code("Unknown")
        None
    """
```

select_languages()

python

```
def select_languages():
    """
    Interactive prompt for user to select 2-4 languages for comparison.
    Validates input and ensures no duplicates.

    Args:
        None

    Returns:
        list: List of language codes (e.g., ['en', 'es', 'pl'])

    Example:
        Please enter 2-4 languages you want to compare.
        Your selection: English, Spanish, Polish
        ✅ Selected languages: English, Spanish, Polish
    """
```

process_word_file(file_path, languages, results_dir)

python

```
def process_word_file(file_path, languages, results_dir):
    """
    Process a word file: translate words, compute similarities, and generate graph.

    Args:
        file_path (str): Path to input text file
        languages (list): List of language codes to compare
        results_dir (str): Directory to save results

    Returns:
        bool: True if processing succeeded, False otherwise

    Creates:
        - Translation files in results/translations/
        - Similarity matrices in results/similarities/
        - Visualization graph in results/

    Example:
        >>> process_word_file("data/animals.txt", ["en", "es", "pl"], "results")
        Loaded 10 words from animals.txt
        Translating words...
        Computing similarities...
        Graph saved to: results/animals_similarity_graph.png
    """
```

display_menu()

```
python

def display_menu():
    """
    Display the main menu options to the user.

    Args:
        None

    Returns:
        None (prints to console)
    """
```

get_file_path()

```
python
```

```
def get_file_path():
    """
    Prompt user for a file path and sanitize input.
    Removes surrounding quotes if present.

    Args:
        None

    Returns:
        str: Sanitized file path

    Example:
        Enter the full path to your word file (.txt): "C:/data/animals.txt"
        Returns: C:/data/animals.txt
    """
```

get_directory_path()

```
python

def get_directory_path():
    """
    Prompt user for a directory path and sanitize input.
    Removes surrounding quotes if present.

    Args:
        None

    Returns:
        str: Sanitized directory path
    """
```

process_directory(dir_path, languages, results_dir)

```
python
```

```
def process_directory(dir_path, languages, results_dir):
    """
    Process all .txt files in a directory.

    Args:
        dir_path (str): Path to directory containing text files
        languages (list): List of language codes to compare
        results_dir (str): Directory to save results

    Returns:
        None (prints success summary)

    Example:
        Found 3 .txt file(s). Processing...
        === Processing topic: animals ===
        === Processing topic: fruits ===
        === Processing topic: buildings ===
        ✅ Successfully processed 3/3 files!
    """
```

---

main.py

Main entry point for the application.

Functions

main()

```python
```

```
def main():
    """
    Main application loop.

    Workflow:
        1. Display welcome message
        2. Prompt user to select languages
        3. Display menu and process user choices:
           - Analyze single file
           - Analyze directory
           - Change language selection
           - Exit

    Args:
        None

    Returns:
        None
    """
```

Usage Examples

Example 1: Analyze a Single File

```
bash
```

```
$ python src/main.py

============================================================
Word Similarity Analyzer - Levenshtein Method
============================================================

Step 1: Language Selection

AVAILABLE LANGUAGES:
...

Please enter 2-4 languages you want to compare.
Your selection: English, Spanish, Polish

✅ Selected languages: English, Spanish, Polish

Step 2: File Selection
Options:
1. Analyze a single file
...

Enter your choice (1-4): 1

Enter the full path to your word file (.txt): data/animals.txt

=== Processing topic: animals ===
Loaded 10 words from animals.txt
Translating words...
Translations saved.
Computing similarities...
Graph saved to: results/animals_similarity_graph.png

✅ File processed successfully!
```

Example 2: Analyze Multiple Files

```
bash
```

```
Enter your choice (1-4): 2

Enter the directory path containing .txt files: data

Found 3 .txt file(s). Processing...
=== Processing topic: animals ===
=== Processing topic: fruits ===
=== Processing topic: buildings ===

✅ Successfully processed 3/3 files!
```

Output Files

Translation Files

Location: `results/translations/`

Format: One word per line

```
# animals_es.txt
perro
gato
pájaro
```

Similarity Matrices

Location: `results/similarities/`

Format: CSV with words as headers

```csv
,perro,gato,pájaro
dog,0.85,0.32,0.41
cat,0.28,0.91,0.35
bird,0.22,0.31,0.78
```

Similarity Graphs

Location: `results/`

Format: PNG image showing language relationships with similarity percentages as edge labels.

Error Handling

The application handles various error scenarios:

- **Invalid file paths**: Displays error message and continues
- **Empty files**: Warns user and skips processing
- **Translation failures**: Returns original word and continues
- **Invalid language names**: Prompts user to try again
- **Invalid language count**: Enforces 2-4 language limit

Technical Details

Similarity Calculation

Uses normalized Levenshtein distance:

- **1.0**: Identical words
- **0.0**: Completely different words
- **0.0-1.0**: Partial similarity based on edit distance

Graph Generation

- Nodes: Language codes
- Edges: Diagonal average of similarity matrix (× 100 for percentage)
- Layout: Spring layout with fixed seed for reproducibility

Contributing

To add a new language:

1. Add entry to `LANGUAGE_MAP` in `displayUtils.py`
2. Use ISO 639-1 language codes
3. Ensure the code is supported by Google Translate

License

This project is for educational purposes.