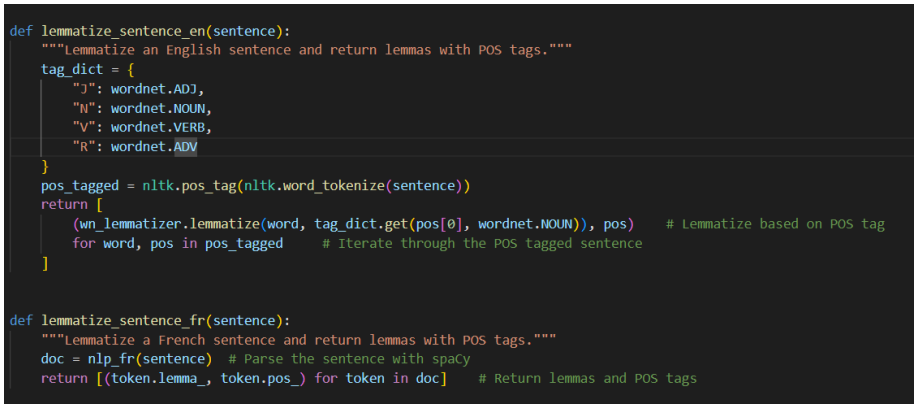
Name: Liw Jun Le **Matric Number:** U2221922D

1. Introduction

This project aims to create a simple Python translation tool that uses a bilingual dictionary to translate text from English to French and vice versa. The greatest challenge was deciding on the "best" method to develop the dictionary. The dictionary was the most limiting factor to accuracy in the project. This could only be accomplished through trial and error by examining the output files for alignment and translation errors. To address the front matter, a simple workaround of skipping 6 lines was performed. This writeup describes the code used, assesses translation quality, and compares the accuracy of this method to other machine translation methods.

2. Methodology

2.1 Lemmatization



```python
def lemmatize_sentence_en(sentence):
    """Lemmatize an English sentence and return lemmas with POS tags."""
    tag_dict = {
        "J": wordnet.ADJ,
        "N": wordnet.NOUN,
        "V": wordnet.VERB,
        "R": wordnet.ADV
    }
    pos_tagged = nltk.pos_tag(nltk.word_tokenize(sentence))
    return [
        (wn_lemmatizer.lemmatize(word, tag_dict.get(pos[0], wordnet.NOUN)), pos)    # Lemmatize based on POS tag
        for word, pos in pos_tagged     # Iterate through the POS tagged sentence
    ]


def lemmatize_sentence_fr(sentence):
    """Lemmatize a French sentence and return lemmas with POS tags."""
    doc = nlp_fr(sentence)  # Parse the sentence with spaCy
    return [(token.lemma_, token.pos_) for token in doc]    # Return lemmas and POS tags
```

For the lemmatization of English words, WordNet was used whereas for French words the spaCy French model was used. In “lemmatize_sentence_en” the input sentence is tokenized into individual words and then tagged with its POS tag, if a POS tag cannot be found in “tag_dict” it is tagged with “wordnet.NOUN”. This is an assumption made as a straightforward solution to maintain code functionality while not adding too much complexity to the code since nouns are the most common type of words in English (Cambridge Dictionary, n.d.). Finally, the word is lemmatized based on its POS tag and the function returns a list of tuples with each tuple containing the lemmatized words with its POS tag.

These two functions serve to pre-process words to handle the variability in word forms that arise due to morphological differences and to disambiguate words with multiple usages. Overall, the normalization of words helps to set the stage for better translation and dictionary building.

2.2 Translation

```python
def translate_word(word, pos, dictionary, lang):
    """Translate a word based on the provided dictionary and language direction."""
    key = f'{word.lower()}_{pos}'   # Create a key for the dictionary lookup
    if key in dictionary:   # Check if the key is in the dictionary
        return dictionary[key].split('_')[0]    # Return the translation if found

    if word.lower() in dictionary:  # Check if the word is in the dictionary
        return dictionary[word.lower()].split('_')[0]   # Return the translation if found

    return word # Return the original word if no translation is found


def translate_text(text, dictionary, reverse_dict, lang):
    """Translate a text based on the provided dictionary and language direction."""
    lemmatized_pos_tagged = (
        lemmatize_sentence_en(text) if lang == 'en' else lemmatize_sentence_fr(text) # Lemmatize the text
    )
    translated = [
        translate_word(lemma, pos, dictionary if lang == 'en' else reverse_dict, lang) # Translate each word
        for lemma, pos in lemmatized_pos_tagged                                        # Iterate through the lemmas
    ]
    return ' '.join(translated).capitalize()
```

2.2.1 Word-Level Translation (translate_word function):

For every word, a unique key is constructed, pairing the word and its part-of-speech tag. The dictionary is then checked with this key. If an exact match is found, the corresponding translation is returned, else, a broader search with just the word will be attempted. Ifthe dictionary does not have a translation, the word remains untranslated.

2.2.2 Sentence-Level Translation (translate_text function):

```python
# Translate and write the results starting from the content after the separator
for i, (en_line, fr_line) in enumerate(zip(en_lines[start_index:], fr_lines[start_index:]), start=1):
    en_to_fr = translate_text(en_line, dictionary, reverse_dict, 'en')  # Translate English to French
    fr_to_en = translate_text(fr_line, reverse_dict, dictionary, 'fr')  # Translate French to English

    output_en_fr.write(f"{i} En:\t{en_line}\n Fr:\t{fr_line}\n En-Fr:\t{en_to_fr}\n\n")  # Write to output En-Fr file
    output_fr_en.write(f"{i} Fr:\t{fr_line}\n En:\t{en_line}\n Fr-En:\t{fr_to_en}\n\n")  # Write to output Fr-En file
```

This function acts as an orchestrator and performs the actual translation. It first identifies the language of the sentence and lemmatizes the entire sentence. Then it calls “translate_word” to translate each word in the sentence. The translated words are concatenated to reconstruct the full sentence in the target language. This function is then called in main to translate lines of text with the dictionary and reverse_dict parameters passed in.

2.3 Dictionary Management and Expansion

```python
def build_dictionary(en_file_path, fr_file_path, dictionary_path):
    """Build a dictionary from aligned English and French UDHR texts."""
    # Open the English and French UDHR texts, skip front matter
    with open(en_file_path, 'r', encoding='utf-8') as en_file, \
         open(fr_file_path, 'r', encoding='utf-8') as fr_file:

        en_lines = en_file.readlines()[6:]
        fr_lines = fr_file.readlines()[6:]

        dictionary = {}
        for en_line, fr_line in zip(en_lines, fr_lines):
            add_to_dictionary(en_line.strip(), fr_line.strip(), dictionary)

        save_dictionary(dictionary, dictionary_path)
        return dictionary


def reverse_dictionary(dictionary):
    """Generate a reversed dictionary for the opposite language direction."""
    return {value: key.split('_')[0] for key, value in dictionary.items()} # Reverse the key-value pairs
```

```python
def add_to_dictionary(en_sentence, fr_sentence, dictionary):
    """Add translation pairs to the dictionary based on aligned sentences."""
    en_lemmas = lemmatize_sentence_en(en_sentence)  # Lemmatize the English sentence
    fr_lemmas = lemmatize_sentence_fr(fr_sentence)  # Lemmatize the French sentence
    for (en_lemma, en_pos), (fr_lemma, fr_pos) in zip(en_lemmas, fr_lemmas):    # Iterate through the lemmas
        dictionary[f'{en_lemma.lower()}_{en_pos}'] = f'{fr_lemma.lower()}_{fr_pos}' # Add translation pair to dictionary


def save_dictionary(dictionary, file_path):
    """Save the dictionary to a JSON file."""
    with open(file_path, 'w', encoding='utf-8') as file:    # Open the file for writing
        json.dump(dictionary, file, ensure_ascii=False, indent=4)   # Write the JSON data


def load_dictionary(file_path):
    """Load the dictionary from a JSON file."""
    with open(file_path, 'r', encoding='utf-8') as file:    # Open the file for reading
        return json.load(file)  # Return the JSON data
```

The “build_dictionary” function constructs the dictionary by reading lines of text from both of the UDHR text files and then lemmatizing each word, this is a simplistic approach taken to create translation pairs. It assumes that sentences in both files are perfectly aligned. Hence, this function will not guarantee perfect mapping of words from English to French and vice versa. A method implemented to raise the accuracy of translation pairs was with the” add_to_dictionary” function call where lemmatization is used to increase the chances of finding aligned word pairs. Initially, I used a dictionary downloaded online and had problems formatting and modifying the dictionary to contain the words for translation specifically for the context of the UDHR texts. It was because the dictionary was a txt file and it did not have consistent formatting, thus I chose to generate a JSON file to maintain consistent formatting and tagged both the English its French translations with its POS tag.

3. Evaluation

3.1 Translation Quality

```
# Original:
# 1 En: Universal Declaration of Human Rights
# Fr:   Déclaration universelle des droits de l’homme
# En-Fr:    UNKNOWN UNKNOWN de humain UNKNOWN

# Improved after lemmatization:
# 1 En: Universal Declaration of Human Rights
# Fr:   Déclaration universelle des droits de l’homme
# En-Fr:    universelle UNKNOWN de humain UNKNOWN

# Improved after aligning the sentences from the UDHR text in English with the UDHR text in French:
# and adding the aligned words to the dictionary
# 1 En: Universal Declaration of Human Rights
#  Fr:  Déclaration universelle des droits de l’homme
#  En-Fr:   déclaration universelle de humain de UNKNOWN
```

The translation quality was assessed by comparing the output files at different stages of development. Initially, a significant proportion of words could not be translated. For instance, translating the phrase "Universal Declaration of Human Rights" resulted in many 'UNKNOWN' tags reduced. However, after lemmatization and dictionary expansion, the number of 'UNKNOWN' tags was reduced. The observed improvements were marginal and inaccurate since the method of using lemmatized words was also based on the assumption that the lemmatization of words was perfect. This displays the limitation of a dictionary-based approach. In further iterations, even though more words could be translated, the words were translated wrongly. This was due to the dictionary having inaccurate mapping of words because it was generated using words lemmatized from the UDHR texts

3.2 Comparison with Other Methods

This project is primitive compared to advanced machine translation systems like Google Translate which utilise sophisticated algorithms and machine learning models trained on extensive bilingual datasets. Unlike the dictionary-based approach, these systems can handle context, grammar, and idiomatic expressions, delivering accurate and natural translations. Additionally, these machines learn and improve from user input and additional training data. This project operates mainly on the word level and only rely on POS tags to disambiguate translations even though a word can have multiple meaning. The translation method used is limited in its context awareness and does not consider the broader semantic context of sentences. The greatest limitation of this project was the reliance on a static dictionary that was also not one hundred per cent accurate in its word mappings.

Google Translate adopts a Neural Machine Translation (NMT) approach with recurrent neural networks (RNNs) and attention mechanisms to translate entire sentences at one time. This allowed for contextually accurate translations rather than translating individual words. It is based on the Europarl Corpus containing documents from European Parliament procedures translated by humans and can be directly translated between two languages. It works with the Sequence-to-Sequence model via the encoder to take in input sentences in the source language and compress its information into a fixed-size vector then passing it to the decoder where it takes the context vector and produces the translated sentence in the target language (Aguilar, 2023). It is worth noting that the state-of-the-art does not always mean that it is perfect and human intervention is still required in the process of machine translation.

4. Discussion

4.1 Methodological Choices

To raise the accuracy of translation through a dictionary-based translation method, I considered making an API call to the Microsoft Azure translation service to help translate the text from the UDHR text files and then using it to populate my dictionary, however, it required a paid subscription for this approach and thus it was scrapped. Another method that I could think of to improve the accuracy of translations was to introduce stemming for words that were already lemmatized but still could not be translated. However, I quickly realised that stemming resulted in different stems for words that should have been translated the same way causing inconsistencies in translation.

4.2 Suitability of UDHR Documents for Translation

The UDHR document is the world’s most translated document as it is written in a clear and formal style (Sperling, n.d.). This is beneficial for translation as it reduces the chances of misinterpretation, where idioms are avoided. This is because idioms might make sense in the source language or cultural context but not in the target language. The document also uses repetitive phrasing, this allows the translation process to be simplified whereby once a structure is accurately translated, it can be applied to multiple instances of the same structure. For example, in the UDHR English text, “Everyone has the right to” appears multiple times at the start of sentences.

4.3 Alternative Texts to Improve Machine Translation Accuracy

The UDHR contains structured text, but it does not have fixed symbols that are iterative throughout the text. The simple dictionary-based approach of translation may work better in form-based documents with fields such as “Name:” and “Address:”. Another genre of text would be recipes where there is limited vocabulary and even lists that are numerically numbered. Overall, to improve the accuracy of direct word translations, texts that are simple and structured with patterns will be preferred.

4.4 Reflection

This project demonstrates that while a dictionary-based approach can provide a base level of translation, it is far from being a replacement for more sophisticated methods and human translation. It highlights the need for translation models to have context-awareness and build on accurate and large datasets to achieve high-quality translation.

5. Bonus

5.1 Accuracy

```
1 En:   Universal Declaration of Human Rights
Fr:     Déclaration universelle des droits de l’homme
En-Fr:  Universelle politique international droit de
```

```
1 En:   Universal Declaration of Human Rights
Fr:     Déclaration universelle des droits de l’homme
En-Fr:  Universelle universelle déclaration homme déclaration
```

From the left, we can see the dictionary-based translation and the IBM Model 1 translation respectively. Even though the accuracy of translation is improved, it is almost negligible. This is not surprising as the IBM Model 1 is one of the simplest statistical translation models and does not consider the context of a sentence and translates each word in isolation. It uses the Expectation-Maximization (EM) algorithm to estimate the probability distributions of words in one language being translated into another. Then it iteratively improves the alignment of words between the source and target languages, optimizing the translation model for accuracy (Koehn, 2018). A reason why it had subpar performance might also be because the model assumes all words between the source and target language are perfect which is not true.

5.2 Training the IBM Model 1

To initiate the training phase, the IBMModel1 class was instantiated with aligned sentence pairs encapsulated by the AlignedSent class. The AlignedSent objects hold tokenized sentences from the UDHR documents. The model was trained for 5 iterations and runs the EM algorithm for the statistical inference of word alignments and translation probabilities. Perhaps by increasing the number of epochs, translation accuracy could be increased (Stigt, 2017).

5.3 Limitations

The IBM Model 1, while effective in establishing word-to-word translations, faces challenges due to its dependence on alignment quality. In the code, it was greatly dependent on the quality of the input AlignedSent objects. Both the dictionary-based method and the IBM model have issues dealing with unseen words and code it was easier to deal with it by expanding the dictionary to account for a broader set of words.

5.4 Conclusion

Building a script with the IBM Model 1 revealed the challenges of statistical machine translation. The model, due to its limitations, did not outperform the dictionary-based approach. This further emphasized the necessity of context aware and syntax-sensitive translation mechanisms. To produce an accurate translation, perhaps we should consider turning towards Neural Machine Translation (NMT) which is language-agnostic and more superior than SMT when there are large amounts of training data (Anwar, 2017). This is not a problem for NMT models as data is easily accessible today.

The hybrid model for machine translation integrates SMT with NMT to leverage the strengths of both approaches. Hybrid models combine the robustness of SMT in data-sparse scenarios with the fluency and learning capabilities of NMT (G. M. El Sayed et al., 2015). IBM Model 1 only represents the early stages of SMT and there many other models out there that are generating translations with greater accuracy and fluency. The advancement of machine translation from SMTs to NMTs and then to hybrid models has paved the way for increasingly sophisticated models today that incorporate monolingual data and transfer learning where knowledge from high-resource language pairs is transferred to improve translation quality for low-resource languages push the boundaries of machine translation.

References:

1. Cambridge Dictionary. (n.d.). *Word classes and phrase classes*. Cambridge Dictionary. https://dictionary.cambridge.org/grammar/british-grammar/word-classes-and-phrase-classes

2. Aguilar, R. (2023, September 21). How accurate is Google Translate? Weglot. https://www.weglot.com/blog/how-accurate-is-google-translate

3. Sperling, T. von. (n.d.). What is the most translated document in the world?. RSS. https://www.bureauworks.com/blog/what-is-the-most-translated-document-in-the-world

4. Cavar, D. (2017, November). Machine Translation in Python 3 with NLTK. GitHub. https://github.com/dcavar/python-tutorial-notebooks/blob/master/notebooks/Machine%20Translation%20in%20Python%203%20with%20NLTK.ipynb

5. NLTK. (2023, January 2). Source code for nltk.translate.ibm1. https://www.nltk.org/_modules/nltk/translate/ibm1.html

6. NLTK. (2023, January 2). Sample usage for translate. https://www.nltk.org/howto/translate.html

7. Collins, M. (n.d.). Statistical Machine Translation: IBM models 1 and 2 - Stanford University. Statistical Machine Translation: IBM Models 1 and 2. https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1162/handouts/Collins_annotated.pdf

8. Koehn, P. (2018, September 13). Machine Translation: IBM Model 1 and the EM Algorithm. cs288_sp20_05_statistical_translation_1up.pdf. https://cal-cs288.github.io/sp20/slides/cs288_sp20_05_statistical_translation_1up.pdf

9. Stigt, D. van. (2017, May 5). Daandouwe/IBM-models: Implementation of the alignment models IBM1 and IBM2. GitHub. https://github.com/daandouwe/IBM-Models

10. Anwar, M. (2017, June 12). SMT vs NMT. Anwarvic's Blog. https://anwarvic.github.io/machine-translation/SMT_Vs_NMT

11. G. M. ElSayed 1, A., S. Salama, A., & El-Din M. El-Ghazali, A. (2015, March). A Hybrid Model for Enhancing Lexical Statistical Machine Translation (SMT). Version Availability - arXiv info. https://info.arxiv.org/help/versions.html