HG2051 Assignment 1 - Kieron Chan Jai Yong

## Introduction

Interlinear texts are a valuable resource for linguists, especially those in the field of language documentation, given the multiple levels of analysis incorporated in these texts. These texts typically incorporate the orthography of the original language, phonetic transcriptions, a morpheme-by-morpheme gloss, as well as a free translation of the sentence in question (Comrie et al., 2008). Given the large number of sentences within a single interlinear text, this project aims to provide a solution that allows for the efficient search of specific sentences matching specific parameters. This project has the potential to aid linguists in their analysis of foreign languages, as it enables the identification of language use patterns in a specific language through the display of sentences matching specific search criteria. In this project, we focus on interlinear texts of the Pnar language, an Austro-Asiatic language (Ring, 2015).

## Methods

### *Data Extraction*

In order to segment and extract relevant data from the interlinear text on a sentence-by-sentence basis, a regular expressions (regex) search function was employed. Specifically, we devised a regex pattern that would search for all occurrences of a specific orthographic pattern using the interlinear text format. This pattern would match every element following '\tx' in each sentence (as shown in Fig. 1). By using this approach, we were able to efficiently extract the desired data

from the dataset (Ring, 2017) for further processing and analysis.

```
\tx tæ ong    u      khana  yei-thaw           chong  thaw  sah  phi
\mb tæ ong    u      khana  ye-   i=      thaw  chong  thaw  sah  phi
\ph tɛ ʔɔŋ ʔu        kʰana je-   ʔi=    tʰaw ʧɔŋ tʰaw saʔ pʰi
\ge nvis   say    3sg.M.nom tell   BEN-  N=      place sit  place stay 2pl
\ps deixis v        pn        v   case- clitic= n     v      n      v      pn

\ft `so he said, say/tell (us) your place where you live.'
```

Fig. 1: Example format of a interlinear text sentence

By storing each sentence in a list using the `re.findall()` function (see Fig. 2), the relevant information could be processed in a systematic manner.

```
segments = re.findall(r"\\tx.*?(?=\n\\ref)", contents, re.DOTALL)
```

Fig. 2: Applying the devised regex pattern to the `re.findall()` function

*Data Cleaning*

Although the regex pattern enabled the extraction of relevant information from the interlinear text, further data cleaning was necessary. The inconsistent number of elements within each level of analysis (orthography, gloss, part-of-speech (POS)) required additional steps. To address this issue, a script (see Fig. 3) was implemented to replace all morpheme boundaries with their whitespace-removed counterparts. This enabled the conjoining of morphemes and ensured the

accuracy of the data extracted from the interlinear text.

```
line = line.replace("- ", "-").replace("= ", "=").replace(" -",
"-").replace("\\", "")
```

Fig. 3: Cleaning script

To organize the extracted data in a more accessible and structured format, we converted the data from lists to dictionaries (see Fig. 4). The conversion involved using the first element of each list as the key for its corresponding list value. This method allowed for easy and direct access to each level of analysis using their corresponding headers.

```
('tx', ['tæ', 'ong', 'u', 'khana', 'yei-thaw', 'chong', 'thaw', 'sah', 'phi'])
('ph', ['tɛ', 'ʔɔŋ', 'ʔu', 'kʰana', 'je-ʔi=tʰaw', 'ʧɔŋ', 'tʰaw', 'sa2', 'pʰi'])
('ge', ['nvis', 'say', '3sg.M.nom', 'tell', 'BEN-N=place', 'sit', 'place',
'stay', '2pl'])
('ps', ['deixis', 'v', 'pn', 'v', 'case-clitic=n', 'v', 'n', 'v', 'pn'])
('ft', ['`so', 'he', 'said,', 'say/tell', '(us)', 'your', 'place', 'where',
'you', "live.'"])
```

Fig. 4: Dictionary format of interlinear text sentence

*Implementation of search function*

After cleaning and organizing the data, a search function was implemented to efficiently search for specific sentences matching specific parameters. The search function was designed to ensure that the number of tokens between all non-empty search parameters was the same, and it checked for the presence of search parameters in their respective categories of analysis. In order to ensure

exact matches between multiple parameters, the index(es) of all occurrences of the search parameters were stored in individual lists. The function then checked for an intersection between the lists of all non-empty search parameters to determine if an exact match was present within the sentence.

To expand the search capabilities of the function, n-gram searching was also implemented. The data was converted into n-gram counterparts, and the presence of the search parameter was searched for within the corresponding n-gram list. These features allow for a more robust search function that can efficiently find sentences that match specific criteria, beyond individual word matches.

*Formatting of search results*

To present search results in a clear and organized format, we utilized the Pandas Dataframes library. This allowed for the alignment of corresponding elements from each level of analysis into their respective columns, resulting in a more readable layout that saved time in identifying corresponding categories (see Fig. 5).

| | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
|---|---|---|---|---|---|---|---|---|---|
| **Text:** | tæ | ong | u | khana | yei-thaw | chong | thaw | sah | phi |
| **IPA:** | tɛ | ʔɔŋ | ʔu | kʰana | je-ʔi=tʰaw | ʧɔŋ | tʰaw | saʔ | pʰi |
| **Gloss:** | nvis | say | 3sg.M.nom | tell | BEN-N=place | sit | place | stay | 2pl |
| **POS:** | deixis | v | pn | v | case-clitic=n | v | n | v | pn |

Trans: `so he said, say/tell (us) your place where you live.'

Fig. 5: Search results presented in Pandas Dataframe format

**Discussion**

The code appears to be working properly, and the algorithm has been devised to eliminate potential false matches. That being said, there is room for improvement in this project. For instance, highlighting relevant search terms in each sentence match can enhance readability, and implementing a "loose" string search would further improve the search function's capabilities, although both tasks may be fairly complex. While the code has not been tested on a different interlinear text file, it can be extended to other interlinear texts with modifications in the data extraction and cleaning segment to accommodate formatting differences.

In summary, this project provides a valuable tool for analyzing interlinear text files, and there is potential for further development and improvement to enhance its functionality and user-friendliness.

## References

Comrie, B., Haspelmath, M., & Bickel, B. (2008). The Leipzig Glossing Rules: Conventions for interlinear morpheme-by-morpheme glosses. *Department of Linguistics of the Max Planck Institute for Evolutionary Anthropology & the Department of Linguistics of the University of Leipzig. Retrieved January*, *28*, 2010.

Ring, H. (2015). A grammar of Pnar. *Unpublished PhD dissertation, Nanyang Technological University, Singapore*.

Ring, H. (2017). "Replication Data for: A grammar of Pnar", https://doi.org/10.21979/N9/KVFGBZ, DR-NTU (Data), V1; Texts-UTF8.txt