**Text Normalization**

Text written in Indic scripts display a lot of quirky behavior on account of varying input methods, multiple representations for the same character, etc. There is a need to canonicalize the representation of text so that NLP applications can handle the data in a consistent manner. The canonicalization primarily handles the following issues:

- Non-spacing characters like ZWJ/ZWNL

- Multiple representations of Nukta based characters

- Multiple representations of two part dependent vowel signs

- Typing inconsistencies: e.g. use of pipe (|) for poorna virama

python src/inlp/normalize/normalize.py <infile> <outfile> <language> [<replace\_nukta>]

<language>: 2-letter ISO 639-1 language code.

<replace\_nukta>: True/False. Default: False

**Unicode based Transliteration**

Transliterate from one Indic script to another. This is a simple script which exploits the fact that Unicode points of various Indic scripts are at corresponding offsets from the base codepoint for that script.

python src/inlp/transliterate/unicode\_transliterate.py <infile> <outfile> <language1> <language2>

<language1>,<language2>: 2-letter ISO 639-1 language code.

**Tokenization**

A trivial tokenizer which just tokenizes on the punctuation boundaries. This also includes punctuations for the Indian language scripts (the purna virama and the deergha virama). It returns a list of tokens.

python src/inlp/tokenize/tokenize.py <infile> <outfile> <language>

<language>: 2-letter ISO 639-1 language code.

**Morphological Analysis**

Unsupervised morphological analysers for various Indian language. Given a word, the analyzer returns the componenent morphemes. The analyzer can recognize inflectional and derivational morphemes.

python src/indicnlp/morph/unsupervised\_morph.py <infile> <outfile> <language> <resource\_directory>

<language>: 2-letter ISO 639-1 language code.

<resource\_directory>: Path to directory containg INLP library resources.