Morphological Analyzer

The system does morphological segmentation and morphological tagging (POS, tense and number) for input texts in any of the MATERIAL languages or English.

Input

- Language Code (ISO 639-2) (EN=English, SW=Swahili, TL=Tagalog, SO=Somali, LT=Lithuanian BG=Bulgarian, PS=Pashto, FA=Farsi, KK=Kazakh and KA = Georgian)
- Plain Text (standard input, file or directory of files)

Output

Analyzed text (standard output, file or directory of files)
An analyzed word in the output contains the following key/value pairs:
- word : The original word
- stem: Word stem
- prefixes: "+"-separated word prefixes
- suffixes: "+"-separated word suffixes
- pos : {ADJ, ADP, ADV, AUX, CCONJ, DET, INTJ, PART, NOUN, NUM, PRON, PROPN, PUNCT, SCONJ, SYM, VERB, X}
- pos_props: A list of the top probable POS tags per word and their probabilities
- tense: {PAST, PRES, FUT, NA}
- number: {SG, PL, NA}
- index: An integer representing the index of the word in the sentence.

Java Commands

```
Standalone - Standard I/O:
Usage: java -jar scripts-morph-v8.0.jar <language> <text>
Standalone - File I/O:
Usage: java -jar scripts-morph-v8.0.jar <language> <input-file> <output-json-file>
Standalone - Directory I/O:
Usage: java -jar scripts-morph-v8.0.jar <language> <input-directory>
<output-json-directory>
Socket Server Initialization:
Usage: java -jar scripts-morph-v8.0.jar <port> <language>
Socket Client - Standard I/O:
Usage: java -jar scripts-morph-v8.0.jar <port> <language> <line>
```

```
Socket Client - File I/O:
Usage: java -jar scripts-morph-v8.0.jar <port> <language> <input-file>
<output-json-file>
Socket Client - Directory I/O:
Usage: java -jar morph-analyzer.jar <port> <language> <input-directory>
<output-json-directory>
```

Docker Commands

```
1- Load the Docker image (only once):
docker load -i scripts-morph-v11.5.tar
2- Run the analyzer in a similar fashion to how it is run from the Jar, but with directory
mappings if necessary. For example:
docker run  -v=<input-directory>:/root/in -v=<output-directory>:/root/out
material/scripts-morph:11.5 TGL /root/in /root/out
Where the input directory is mounted to /root/in , and the output directory is mounted
to /root/out. /root/in and /root/out are then used as parameters to the JAR within the
image.
```

Example

Input:

Language Code: FA

Text: پس قوم درروز هفتمین آرام گرفتند.

Output:

[[{"word":"پس","pos":"ADV","pos_props":{"ADV":0.98896,"ADP":0.01102,"NOUN":0.00002},"tense":"NA","num":"NA","index":0,"number":"NA","prefixes":"","suffixes":"","stem":"پس"},{"word":"قوم","pos":"NOUN","pos_props":{"NOUN":1.00000},"tense":"NA","num":"PL","index":1,"number":"PL","prefixes":"","suffixes":"","stem":"قوم"},{"word":"درروز","pos":"NOUN","pos_props":{"NOUN":0.81636,"ADV":0.18094,"ADP":0.00254,"ADJ":0.00016},"tense":"NA","num":"SG","index":2,"number":"SG","prefixes":"در","suffixes":"","stem":"روز"},{"word":"هفتمین","pos":"ADJ","pos_props":{"ADJ":1.00000},"tense":"NA","num":"NA","index":3,"number":"NA","prefixes":"","suffixes":"ین","stem":"هفتم"},{"word":"آرام","pos":"ADJ","pos_props":{"ADJ":0.67112,"NOUN":0.30396,"ADV":0.02492},"tense":"NA","num":"NA","index":4,"number":"NA","prefixes":"","suffixes":"","stem":"آرام"},{"word":"گرفتند","pos":"VERB","pos_props":{"VERB":1.00000},"tense":"PAST","num":"NA","index":5,"number":"NA","prefixes":"","suffixes":"ند","stem":"گرفت"},{"word":".","pos":"PUNCT","pos_props":{"PUNCT":1.0000

0},"tense":"NA","num":"NA","index":6,"number":"NA","prefixes":"","suffixes":"","stem":"."}]
]

System Requirements

- CPU: 2.0+ GHz
- RAM: 4 GB
- Docker
- Java 1.8+ (for execution through JAR)

Standalone

Yes.

Approach

- The morphological-segmentation component is based on the MorphoGram framework (https://github.com/rnd2110/MorphAGram) for unsupervised and semi-supervised morphological segmentation. MorphoGram is based on Adaptor Grammars, nonparametric Bayesian models that utilize probabilistic context-free grammars (PCFGs) to model word structure.
- The morphological-tagging component is based on unsupervised cross-lingual tagging that relies on annotation projection. The approach requires a parallel text between the target language and a source language for which a morphological tagger is available. The source text is annotated, and those annotations are then projected onto the target text through word-level alignments, where the alignment model is based on the parallel text. The target annotations are then used to train an averaged-perceptron tagger for the tagging of POS, tense and number.

Notes

- The POS tags follow the UD guidelines and correspond to {Adjective, Adposition, Adverb, Auxiliary, Conjunction, Determiner, Interjection, Particle, Noun, Number, Pronoun, Proper Noun, Punctuation, Subjunctive Conjunction, Symbol, Verb, Other}
- Tense values are Past, Present, Future and Not-Applicable.
- Number values are Singular, Plural and Not-Applicable.
- Tense is only applicable for AUX and VERB tags.
- Number is only applicable for NOUN and PROPN tags.
- PROPN is replaced by NOUN in Farsi.

Non-Java Systems

In addition to the java-based morphological-segmentation and morphological-tagging system listed above (which can be run through docker or jar commands), both morphology tasks can be run independently by two corresponding Python systems.
MorphoAGram: https://github.com/rnd2110/MorphAGram
Unsupervised cross-lingual POS tagging (using BiLSTM Neural models):
https://github.com/rnd2110/unsupervised-cross-lingual-POS-tagging
The two systems above are intended to be frameworks with several capabilities. Please refer to the documentation in the GitHub repos for complete details.