

Word-Level QuEst++ Manual

By Gustavo Henrique Paetzold

1. Installation

- 1) Place all QuEst files into a folder of your choice.
- 2) Download version 3.5.1 of Stanford Core NLP from <http://nlp.stanford.edu/software/corenlp.shtml>
- 3) Add the file "stanford-corenlp-3.5.1-models.jar" to "quest/lib"
- 4) For spanish tagging/parsing models, download the file <http://nlp.stanford.edu/software/stanford-spanish-corenlp-2015-01-08-models.jar> and place it in "quest/lib"
- 5) For chinese tagging/parsing models, download the file <http://nlp.stanford.edu/software/stanford-chinese-corenlp-2015-01-30-models.jar> and place it in "quest/lib"
- 6) Download the Universal Wordnet plugin from <http://resources.mpi-inf.mpg.de/yago-naga/uwn/uwn.zip> and unzip it into a folder of your choice.

Important:

- The Universal Wordnet plugin folder is the one which should be referenced in the variable "tools.universalwordnet.path" in the config file.
- The tagging/parsing models for english, spanish and chinese are automatically recognized by QuEst if the aforementioned libraries are placed in the "quest/lib" folder.

1. Running

- 1) Create a configuration file following the example in "quest/config/config.word-level.properties".
- 2) Create a feature configuration file following the example in "quest/config/features/features_word-level_all.xml".
- 3) Prepare the source and target language input files for which you desire to estimate feature values. Both files must be tokenized and must have the same number of lines.

- 4) If you wish to use features that require for alignments between source and target sentences, produce an alignments file. It must be in PHARAOH format, and can be produced by fast_align (https://github.com/clab/fast_align). It can contain 1 to many alignments (1 source word to N target words), but not many to 1 alignments (1 target word to N source words).
- 5) Run the following command line:

```
java -cp QuEst.jar shef.mt.WordLevelFeatureExtractor -lang <source_language>
<target_language> -input <source_file> <target_file> -mode <selected_model> -
config <config_file> -alignments <alignments_file>
```

- 6) The output will be saved in the output folder specified in the configuration file under the name "output.txt". It will have M lines, one for each word in each sentence of the target language input file. Each line will have N feature values separated by a tab. It will be in the following format:

```
<word_1_feature_value_1>\t<word_1_feature_value_2>...
<word_1_feature_value_n-1>\t<word_1_feature_value_n>

<word_2_feature_value_1>\t<word_2_feature_value_2>...
<word_2_feature_value_n-1>\t<word_2_feature_value_n>

...

<word_m-1_feature_value_1>\t<word_m-1_feature_value_2>... <word_m-
1_feature_value_n-1>\t<word_m-1_feature_value_n>

<word_m_feature_value_1>\t<word_m_feature_value_2>...
word_m_feature_value_n-1>\t<word_m_feature_value_n>
```

2. Creating a Configuration File

- 1) Create a plain text file in the folder of your choice
- 2) Assign values to parameters in the following format, one per line:

<parameter> = <value>

2.1. Parameter Descriptions

- **features.default:** Standard mode to be used (a mode is defined by a specific selection of features).
- **sourceLang.default:** Source language (english, spanish, chinese, german).
- **targetLang.default:** Target language (english, spanish, chinese, german).
- **output:** Folder in which to save the file with feature values.
- **input:** Folder from which to read input files.
- **resourcesPath:** Path to the folder in which linguistic resources are stored.
- **featureConfig.<identifier>:** Path to a feature configuration file to be used in mode "<identifier>". The user can define many distinct modes, which can be selected in the command line while running QuEst through the "-mode" parameter.
- **<source_language>.corpus:** Path to a corpus for <source_language>.
- **<source_language>.poscorpus:** Path to a corpus composed entirely of POS tags for <source_language>.
- **<source_language>.POSModel:** Path to a POS tagging model for <source_language> (it can be either an absolute path to a model trained with Stanford Core NLP, or a classpath from a library in "quest/lib").
- **<source_language>.parseModel:** Path to a parsing model for <source_language> (it can be either an absolute path to a model trained with Stanford Core NLP, or a classpath from a library in "quest/lib")
- **<source_language>.lm:** Path to a language model for <source_language>. It must be in ARPA format. If left blank, QuEst will try to call your SRILM installation to automatically generate the file and place it into:

<resourcesPath>/<source_language>/<source_language>.lm

- **<source_language>.ngram:** Path to an ngram counts file for <source_language>. To produce it manually, call SRILM's "ngram-count" binary with the "-write" option, and pass the resulting file to QuEst's the shef.mt.util.NGramSorter application through the following command line:

**java -cp QuEst.jar shef.mt.util.NGramSorter <srilm_ngram_file>
<number_of_slices> <ngram_file_order> <frequency_cutoff> <output>**

If left blank, QuEst will try to call your SRILM installation to automatically generate the file and place it into:

<resourcesPath>/<source_language>/<source_language>_ngram.ngram.clean

- **<source_language>.posngram:** Path to an POS tag ngram counts file for <source_language>. To produce it manually, call SRILM's "ngram-count" binary with the "-write" option, and pass the resulting file to QuEst's the shef.mt.util.NGramSorter application through the following command line:

```
java -cp QuEst.jar shef.mt.util.NGramSorter <srilm_posngram_file>
<number_of_slices> <ngram_file_order> <frequency_cutoff> <output>
```

If left blank, QuEst will try to call your SRILM installation to automatically generate the file and place it into:

```
<resourcesPath>/<source_language>/<source_language>_posngram.posngram.clean
```

- **<source_language>.stopwords:** Path to a file of stop words for <source_language>.
- **<source_language>.translationProbs:** Translation probabilities between words in source and target languages. The file must be produced by fast_align through the following command:

```
fast_align -i <parallel_data> -v -d -o -c <translation_probabilities>
```

- **<target_language>.corpus:** Path to a corpus for <source_language>
- **<target_language>.poscorpus:** Path to a corpus composed entirely of POS tags for <target_language>.
- **<target_language>.POSModel:** Path to a POS tagging model for <source_language> (it can be either an absolute path to a model trained with Stanford Core NLP, or a classpath from a library in "quest/lib")
- **<target_language>.parseModel:** Path to a parsing model for <source_language> (it can be either an absolute path to a model trained with Stanford Core NLP, or a classpath from a library in "quest/lib")
- **<target_language>.lm:** Path to a language model for <source_language>. It must be in ARPA format. If left blank, QuEst will try to call your SRILM installation to automatically generate the file and place it into:

```
<resourcesPath>/<source_language>/<source_language>.lm
```

- **<target_language>.ngram:** Path to an ngram counts file for <target_language>. To produce it manually, call SRILM's "ngram-count" binary with the "-write" option, and pass the resulting file to QuEst's the shef.mt.util.NGramSorter application through the following command line:

```
java -cp QuEst.jar shef.mt.util.NGramSorter <srilm_ngram_file>  
<number_of_slices> <ngram_file_order> <frequency_cutoff> <output>
```

If left blank, QuEst will try to call your SRILM installation to automatically generate the file and place it into:

```
<resourcesPath>/<target_language>/<target_language>_ngram.ngram.clean
```

- **<target_language>.posngram:** Path to an POS tag ngram counts file for <source_language>. To produce it manually, call SRILM's "ngram-count" binary with the "-write" option, and pass the resulting file to QuEst's the shef.mt.util.NGramSorter application through the following command line:

```
java -cp QuEst.jar shef.mt.util.NGramSorter <srilm_posngram_file>  
<number_of_slices> <ngram_file_order> <frequency_cutoff> <output>
```

If left blank, QuEst will try to call your SRILM installation to automatically generate the file and place it into:

```
<resourcesPath>/<source_language>/<target_language>_posngram.posngram.cl  
ean
```

- **<target_language>.stopwords:** Path to a file of stop words for <target_language>.
- **<target_language>.refTranslations:** Path to a file containing reference translations in the target language. The file must have the same number of lines as the target input file, and must contain one reference translation per line.
- **tools.fast_align.path:** Path to the root folder of your fast_align installation.
- **tools.ngram.path:** Path to the binaries folder of your SRILM installation (typically "srilm/bin/<operational_system>").
- **tools.universalwordnet.path:** Path to the Universal Wordnet plugin folder (must contain files "uwn.plg" and "uwn.dat").
- **ngramsize:** Ngram size of language models and ngram count files. We recommend a minimum of 4-gram ngram files and language models.

Important:

- If you provide a valid path in "tools.ngram.path", QuEst can reliably produce <source_language>.lm, <source_language>.ngram, <target_language>.lm and <target_language>.ngram automatically.
- We do not recommend to leave the "alignments.file" parameter blank for large experiments, since QuEst will try to produce the alignments automatically, and

fast_align frequently crashes when called from Java in both Windows and Unix environments.

- We recommend for you to create separate configuration files for Word-Level QuEst and Sentence-Level QuEst.