Qualitative: Open source Python tool for Quality Estimation over multiple Machine Translation outputs

Eleftherios Avramidis, Lukas Poustka, Sven Schmeier - name.surname@dfki.de
German Research Center for Artificial Intelligence - DFKI Berlin

**QE for ranking MT output**

**Input:** one source sentence, many MT system outputs; **Output:** automatically predicted quality order of MT outputs
- estimation based on modelling human preferences

| source: Der Satz musse ubersetzt werden | 1 |
| system A: The sentence must be translated | 3 |
| system B: The sentence translated must be | 2 |
| system C: The sentence must translated | 4 |
| system D: A sentence must translated | 3 |
| system E: The sentence must translated werden | 2 |

**Basic use**

**Requirements**
- Python 2.7, Linux operating system
- install python pip packages
- install java jar libraries (automatically)
- LM server by Nitin Madnani
- get code from http://dfki.de/~elav01/software/qualitative

**Resources**
Basic functionality requires source and target:
- language model, truecase model, PCFG grammar

**Execution**
- Command-line interaction: sentence-level user-based entry
- Batch decoding: processing of entire test-files
- Server interface: accepting request from apps via XML-RPC
- Web-based demonstrator: for visualizing the QE process

**Under the hood**

**Machine Learning**
Ranking algorithm based on pairwise decomposition into binary decisions.
- Pre-trained models with Logistic Regression are available
- SVM, Naive Bayes, kNN and decision trees also supported

**Features**
Feature Generators analyze each sentence and attach features of these categories:
- Language Model
- PCFG Parsing
- Cross-target BLEU and METEOR
- Language correction
- IBM1 probabilities
- Length features
- Source-to-target ratios and subtractions

**Training**
New models are created in two phases:
- batch annotation parallelized with Ruffus
- learning organised in various configurations via PyExpSuite
Learning algorithms provided by Orange and Scikit-learn kits

**Development**
- Modular structure for collaborative coding
- De-centralised via Git repository
- Detailed API/Doc inline code documentation
- Abstract classes providing basic functionality

**Data structures**
- SimpleSentence, ParallelSentence, DataSet:
  classes that hold the sentences and the features

**Reading and writing files**
- Specialized XML format for variable number of target sentences per source sentence.
- Functions and classes for reading and writing data in the ‘dataprocessor’ package

**Adding new features**
- extend abstract class ‘FeatureGenerator’
- override stub methods for returning attributes to the various levels of the annotation process
- java libraries can be encapsulated with Py4j

**New QE applications**
- Infrastructure can be used to support other QE applications, such as regression problems

**Machine Learning algorithms**
- Abstract interface in package ‘ml’ to support basic functions of various ML toolkits
- Possibility to add own algorithms or add existing solutions

This work has received support by the European Commission’s FP7 (FP7/2007-2013) under grant agreement number 610516: "QTeap: Quality Translation by Deep Language Engineering Approaches"