Hands-on QuEst++

Carolina Scarton, Gustavo Paetzold and Lucia Specia

University of Sheffield

https://github.com/ghpaetzold/questplusplus

COLING, Osaka, 11 Dec 2016
Definition

**Goal**: framework to explore features for QE
Goal: framework to explore features for QE

- Feature extractors: for 150+ features of all types: Java
**Goal:** framework to explore features for QE

- **Feature extractors:** for 150+ features of all types: Java
- **Machine learning:** wrappers for a number of algorithms in the scikit-learn toolkit, grid search, feature selection: Python
**Goal**: framework to explore features for QE

- **Feature extractors**: for 150+ features of all types: Java
- **Machine learning**: wrappers for a number of algorithms in the scikit-learn toolkit, grid search, feature selection: Python

Open source: [http://www.quest.dcs.shef.ac.uk/](http://www.quest.dcs.shef.ac.uk/)
Definition

**New release**: word and document-level feature extraction and machine learning added
Definition

**New release**: word and document-level feature extraction and machine learning added

- **Feature extractors**: 40 features for word-level and 79 features for document-level
Definition

**New release**: word and document-level feature extraction and machine learning added

- **Feature extractors**: 40 features for word-level and 79 features for document-level
- **Machine learning**: support to Conditional Random Fields (CRF) added for word-level models
**Definition**

**New release**: word and document-level feature extraction and machine learning added

- **Feature extractors**: 40 features for word-level and 79 features for document-level

- **Machine learning**: support to Conditional Random Fields (CRF) added for word-level models

- **Another important improvement**: changes on the core functionalities
System and baseline features required

- **Java 8** (OpenJDK or Oracle versions)
System and baseline features required

- **Java 8** (OpenJDK or Oracle versions)
- **Sentence and word-level baseline features**
  - Perl 5 (or above)
  - SRILM
  - Tokenizer and Truecaser (from Moses toolkit)
System and baseline features required

- **Java 8** (OpenJDK or Oracle versions)
- **Sentence and word-level baseline features**
  - Perl 5 (or above)
  - SRILM
  - Tokenizer and Truecaser (from Moses toolkit)
- **Word-level features**
  - Stanford Core NLP 3.5.1 models
  - Stanford Core NLP Spanish model
  - Universal WordNet plugin
Basic Usage - Sentence-level

java -cp QuEst+++.jar shef.mt.SentenceLevelFeatureExtractor
-tok -case true
-lang <<lang_source>> <<lang_target>>
-input <<input_source>> <<input_target>>
-config <<config_file>>
Input files

- Word and sentence levels: file with one sentence per line
Input files

- **Word and sentence levels**: file with one sentence per line
- **Document level**: file with paths for documents
Input files

- **Word and sentence levels**: file with one sentence per line
- **Document level**: file with paths for documents

Files from source and target should have the same number of lines
Folders

- **src**: source code
- **lang_resources**: folder containing all language resources required for the features
- **lib**: external libraries needed for feature extraction
- **config**: configuration files for running QuEst++
- **input**: auxiliary input folder
- **output**: output folder
Adding a new feature

- Example with **sentence-level** feature extractor

- New feature: complex words per sentence (averaged by the length of sentence)

- Language Resource: list of simple words (LSW)

  Idea: count words not in the LSW and normalise by number of words in the sentence
Adding a new feature

- Example with *sentence-level* feature extractor
- New feature: *complex words per sentence* (averaged by the length of sentence)
Adding a new feature

- Example with **sentence-level** feature extractor
- New feature: **complex words per sentence** (averaged by the length of sentence)
- Language Resource: **list of simple words** (LSW)
Adding a new feature

- Example with **sentence-level** feature extractor
- New feature: **complex words per sentence** (averaged by the length of sentence)
- Language Resource: **list of simple words** (LSW)
- **Idea**: count words not in the LSW and normalise by number of words in the sentence
Adding a new feature

- Creating a processor for the new feature
  - Package: shef.mt.tools
Adding a new feature

- Creating a processor for the new feature
  - Package: shef.mt.tools
  - Function: prepare resources to be used by features
Adding a new feature

Creating a processor for the new feature

- Package: shef.mt.tools
- Function: prepare resources to be used by features
- Extends ResourceProcessor class: add the resources to the sentence (processNextSentence method)
Adding a new feature

Creating a processor for the new feature

- Package: `shef.mt.tools`
- Function: prepare resources to be used by features
- Extends `ResourceProcessor` class: add the resources to the sentence (`processNextSentence` method)
- It is useful because a unique processor can be used by several features
Adding a new feature

- Create a new Java class called ComplexWordsProcessor.java
  - Package: shef.mt.tools
Adding a new feature

- Create a new Java class called `ComplexWordsProcessor.java`
  - Package: `shef.mt.tools`
  - Extends: `ResourceProcessor` class
Adding a new feature

- Create a new Java class called `ComplexWordsProcessor.java`
  - Package: `shef.mt.tools`
  - Extends: `ResourceProcessor` class
  - Read the LSW and store it on a ArrayList
Adding a new feature

- Creating a class for the new feature
  - Package: shef.mt.features.impl.bb
Adding a new feature

Creating a class for the new feature

- Package: shef.mt.features.impl.bb
- Extends Feature class: run method - feature extraction itself
Adding a new feature

- Creating a class for the new feature
  - Package: `shef.mt.features.impl.bb`
  - Extends `Feature` class: `run` method - feature extraction itself
  - Feature classes are usually named following a number order (e.g. Feature1001, Feature1002)
Adding a new feature

- Create a new Java class called Feature7001.java
  - Package: shef.mt.features.impl.bb
Adding a new feature

- Create a new Java class called Feature7001.java
  - Package: shef.mt.features.impl.bb
  - Extends: Feature class

- Get the ArrayList from the ComplexWordsProcessor class and calculate the feature

- Also define the resource that will be required for this feature
Adding a new feature

- Create a new Java class called Feature7001.java
  - Package: `shef.mt.features.impl.bb`
  - Extends: `Feature` class
  - Get the ArrayList from the `ComplexWordsProcessor` class and calculate the feature
Adding a new feature

- **Create a new Java class called Feature7001.java**
  - Package: `shef.mt.features.impl.bb`
  - Extends: `Feature` class
  - Get the ArrayList from the `ComplexWordsProcessor` class and calculate the feature
  - Also define the resource that will be required for this feature
Adding a new feature

- **Feature configuration file**
  - Folder: `config/features`
Adding a new feature

- Feature configuration file
  - Folder: config/features
  - XML file with the featureset that will be executed
Adding a new feature

- Feature configuration file
  - Create a file named `features_complex_words.xml` inside the folder `config/features`
Adding a new feature

- **Feature configuration file**
  - Create a file named `features_complex_words.xml` inside the folder `config/features`
  - Add the new feature to this file
Adding a new feature

- **Configuration file**
  - Folder: `config`

For sentence-level: `config.sentence-level.properties` contains basic configuration for the system and paths to resources and tools.
Adding a new feature

- **Configuration file**
  - Folder: `config`
  - For sentence-level: `config.sentence-level.properties`
Adding a new feature

- **Configuration file**
  - Folder: `config`
  - For sentence-level: `config.sentence-level.properties`
  - Contains basic configuration for the system and paths to resources and tools
Adding a new feature

- **Configuration file**
  - Add the resource `source.simplewords` to the configuration file.
Adding a new feature

- **Configuration file**
  - Add the resource `source.simplewords` to the configuration file
  - Change the option `featureConfig` to the path to `features_complex_words.xml`
Adding a new feature

- **SentenceLevelProcessorFactory.java**
  - Package: shef.mt.tools
Adding a new feature

▶ **SentenceLevelProcessorFactory.java**
  ▶ Package: **shef.mt.tools**
  ▶ Function: create all processors required by the features
Adding a new feature

- **SentenceLevelProcessorFactory.java**
  - Package: `shef.mt.tools`
  - Function: create all processors required by the features
  - Only generate processors that will be used (improvement of QuEst++)
Adding a new feature

- **SentenceLevelProcessorFactory.java**
  - Package: `shef.mt.tools`
  - Function: create all processors required by the features
  - Only generate processors that will be used (improvement of QuEst++)
  - It is the connection between features and configuration file
Adding a new feature

- SentenceLevelProcessorFactory.java
  - Package: shef.mt.tools
Adding a new feature

- **SentenceLevelProcessorFactory.java**
  - Package: `shef.mt.tools`
  - Add an `if` block containing the calling to a method called `getComplexWordsProcessor`
Adding a new feature

- SentenceLevelProcessorFactory.java
  - Package: shef.mt.tools
  - Add an if block containing the calling to a method called `getComplexWordsProcessor`
  - Implement `getComplexWordsProcessor` method
Build

- NetBeans 8.1
- `ant "-Dplatforms.JDK_1.8.home=/usr/lib/jvm/java-8-<<version>>""`
java -cp QuEst++.jar shef.mt.SentenceLevelFeatureExtractor -tok -case true -lang <<lang_source>> <<lang_target>> -input <<input_source>> <<input_target>> -config <<config_file>>

Check the file output.txt inside output/test
System requirements

- **Python 2.7.6** (or above - only 2.7 stable distributions)
System requirements

- **Python 2.7.6** (or above - only 2.7 stable distributions)
- **SciPy** and **NumPy** (SciPy $\geq 0.9$ and NumPy $\geq 1.6.1$)
- **scikit-learn** (version 0.15.2)
- **PyYAML**
System requirements

- **Python 2.7.6** (or above - only 2.7 stable distributions)
- **SciPy** and **NumPy** (SciPy $\geq 0.9$ and NumPy $\geq 1.6.1$)
- **scikit-learn** (version 0.15.2)
- **PyYAML**
- **GPy**
- **CRFsuite**
Folders

- **learning**: main folder
- **config**: configuration files
- **src**: source code files
- **data**: example data (same format as output of feature extractor) + scores
Run

`python src/learn_model.py config/<<config_file>>`
Machine learning algorithms

- SVR
- SVC
- LassoCV
- LassorLars
- LassorLarsCV
- GP (implemented using GPy - need some code update)
- CRF (implemented using CRFsuite)
Adding a machine learning algorithm

Exemple using an algorithm from scikit-learn
Adding a machine learning algorithm

Exemple using an algorithm from scikit-learn

- Algorithm: **Ridge**: Linear least squares with l2 regularization
Adding a machine learning algorithm

Exemple using an algorithm from scikit-learn

- Algorithm: **Ridge**: Linear least squares with l2 regularization
- Package: `sklearn.linear.model.Ridge`
Adding a machine learning algorithm

Exemple using an algorithm from scikit-learn
- Algorithm: **Ridge**: Linear least squares with l2 regularization
- Package: `sklearn.linear.model.Ridge`
- **Idea**: include the algorithm on the available code
Adding a machine learning algorithm

```
learn_model.py

- Main class of QuEst++ machine learning module
```
Adding a machine learning algorithm

learn_model.py

- Main class of QuEst++ machine learning module
- Method: `set_learning_method(config, X_train, y_train)`
Adding a machine learning algorithm

learn_model.py

- Main class of QuEst++ machine learning module
- Method: `set_learning_method(config, X_train, y_train)`
- Create estimators for the new algorithm
Configuration file

- Folder: config

- Open the file svr.cfg to see an example

- Create a new file called ridge.cfg and follow the structured YAML to provide parameters for the model
Configuration file

- Folder: `config`
- Files follow the YAML format

Open the file `svr.cfg` to see an example. Create a new file called `ridge.cfg` and follow the structured YAML to provide parameters for the model.
Configuration file

- Folder: config
- Files follows the YAML format
- Open the file svr.cfg to see an example
Configuration file

- Folder: config
- Files follow the YAML format
- Open the file svr.cfg to see an example

Create a new file called ridge.cfg and follow the structured YAML to provide parameters for the model
Run

```
python src/learn_model.py config/ridge.cfg
```
Hands-on QuEst++

Carolina Scarton, Gustavo Paetzold and Lucia Specia

University of Sheffield

https://github.com/ghpaetzold/questplusplus

COLING, Osaka, 11 Dec 2016