MARMOT: A Toolkit for Translation Quality Estimation at the Word Level

Varvara Logacheva[§], **Chris Hokamp**[†], **Lucia Specia**[§]
§Department of Computer Science, University of Sheffield, UK
†CNGL Centre for Global Intelligent Content, Dublin City University, Ireland
{v.logacheva,l.specia}@sheffield.ac.uk, chokamp@computing.dcu.ie

Abstract

We present Marmot — a new toolkit for quality estimation (QE) of machine translation output. Marmot contains utilities targeted at quality estimation at the word and phrase level. However, due to its flexibility and modularity, it can also be extended to work at the sentence level. In addition, it can be used as a framework for extracting features and learning models for many common natural language processing tasks. The tool has a set of state-of-the-art features for QE, and new features can easily be added. The tool is open-source and can be downloaded from https://github.com/qe-team/marmot/.

Keywords: machine translation, quality estimation, toolkit

1. Introduction

Quality estimation (QE) for machine translation (MT) is a technique aimed at defining the quality of an automatic translation without comparing it to a reference. This task is particularly important for many real-world applications of MT, where no reference translation is available. If MT is used for gisting, a user who can only speak the target language cannot judge the quality or reliability of its output. If an MT system is embedded into a computer-assisted translation (CAT) tool, quality estimation can increase the productivity of a user by filtering out translations that are too bad for editing or highlighting the phrases with low quality to make sure they are edited by the user (Turchi et al., 2015; O'Brien et al., 2014).

Translation quality can be estimated at different levels of granularity, ranging from the word level to the document level. Thus far, the best results have been achieved in sentence-level QE. This is likely to be because the evaluation of the entire sentence allows the use of very many general features, such as language model (LM) scores and posterior translation probabilities.

However, the more challenging task of estimating quality at the word level is also very important, both from an application perspective, and from a research point of view. Fine-grained distinctions can be used to diagnose MT systems and to guide post-editors' attention to erroneous parts of the sentence. In addition, sentence-level translation quality can be drastically affected by translation errors involving single words or phrases.

One of the first attempts to address the problem of QE was the workshop at John Hopkins University (Blatz et al., 2004). While the main focus of the workshop was sentence-level QE, it also included research on word-level QE.

In (Ueffing and Ney, 2007), word-level quality is defined as the word-level posterior probability, which is computed as a probability of a word occurring in an n-best list. Raybaud et al. (2009) rely on mutual information between the evaluated word and its source and target contexts, and an LM enhanced with linguistic features. More recently, the WMT shared task on QE (Bojar et al., 2013; Buck et al., 2014; Bojar et al., 2015) has spurred the development of word-level QE systems. The most successful QE models use features extracted from pseudo-references (Esplà-Gomis et al., 2015) and n-best lists (Camargo de Souza et al., 2014) produced by MT systems. In (Luong et al., 2014) the feature set contains source and target contexts of the word, source and target POS-tags, LM scores, syntactic and semantic information. The most widely used training models are Conditional Random Fields (CRF) (Luong et al., 2014; Shah et al., 2015) or neural networks: feed-forward (Kreutzer et al., 2015) as well as recurrent (Camargo de Souza et al., 2014).

Despite recent developments, word-level QE still lags behind other granularity levels in terms of performance. One of the reasons for that is the complexity of the task itself: word-level features are difficult to generalise, leading to sparsity in the training data. Another reason is the lack of toolkits that serve as a basis for researchers to develop upon.

Despite many submissions to the WMT word-level QE task, only one tool is publicly available for QE at the word level, namely QuEst++ (Specia et al., 2015), which is an extension of a previous, sentence-level version for QE at the document and word levels. It performs the extraction of word-level features used in (Luong et al., 2014), but not model learning, although it is distributed with several scripts for the sklearn library [1]. This tool is implemented in Java.

In this paper, we present **Marmot** — a new tool for the extraction of word-level features and training of word-level QE models. Unlike QuEst++, this system is highly flexible and modular: it can easily be extended to include additional features and learning methods, and it includes a full experimental pipeline, incorporating data preparation, feature extraction, model learning, and evaluation.

2. System description

A QE system should be able to perform three main tasks: extract features from the data, train a model, and perform tagging of new data using this model. Marmot is designed to perform all of these tasks in a flexible, efficient and transparent manner. In addition, models can be evaluated within the toolkit. Therefore, its pipelines allow users to go from

[1] http://scikit-learn.org/

raw training data to trained and evaluated models with minimal configuration.

Marmot is written in Python. We take advantage of stable and well-maintained Python libraries that implement common machine learning algorithms and NLP tasks at preprocessing, feature extraction and model building stages.

2.1. Design

The main feature of Marmot is the flexibility of its API. Because of the modular architecture, users can easily add or implement new parsers, data representations, and features that fit their particular use cases, and simply plug them in to a standard experiment workflow.

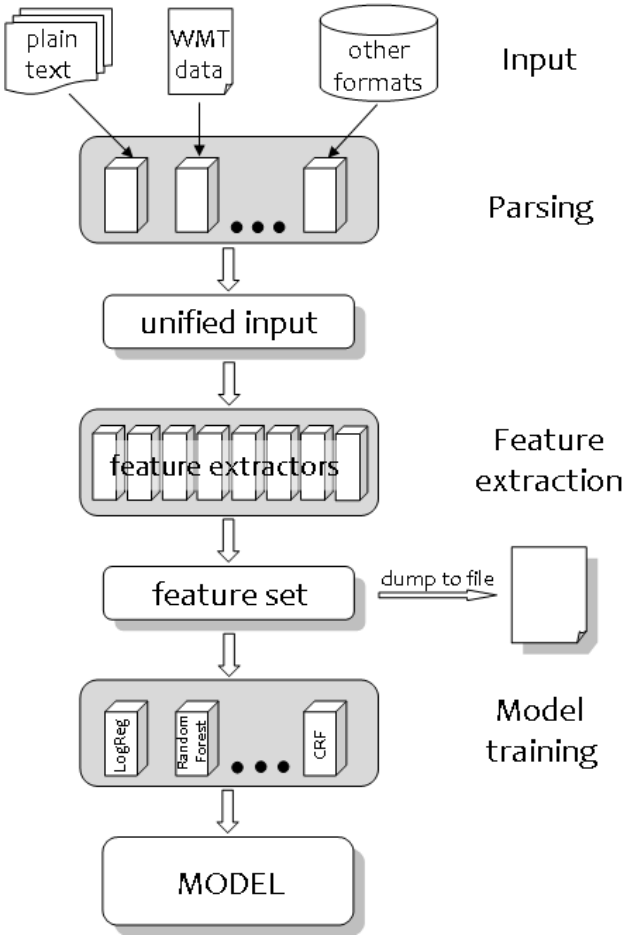

Figure 1: System architecture

Figure 1 shows the architecture of the complete pipeline in Marmot. The grey boxes denote the stages of preprocessing or training which require specification of predefined parameters and which are likely to be customised by a user. The white blocks inside the grey boxes denote user-determined procedures. Each stage in the pipeline for an experiment is specified in the configuration file as the path to a Python class or function. Therefore, the framework is agnostic about the actual functions that it calls, allowing the user to create a custom pipeline without writing any code, simply by specifying which parsers, feature extractors, and learning method(s) they wish to use. User-defined functions do not need to be included into the source code of the system, and no changes to the source code need to be made if a user provides a valid path to the function and makes sure it returns the output accepted by the next stages.

Parsing input data is the first of the configurable stages. Data can be provided in many different formats, such as parallel corpora in one or more files, JSON, or XML, and new parsers are straightforward to implement. The parsed data is stored in *context objects*. Each context object represents a training or test example. It contains a target item and all information needed to extract features for this item: its label, the sentence it is taken from, its index in the sentence, and any user-specified additional representations, such as part-of-speech tags.

The actual content of a context object does not influence its behaviour, so it can store any data. Since Marmot was originally designed for word-level QE, context objects represented words, but it has been extended to phrasal and sentential context objects for word and sentence-level QE. The pipeline will not be affected by this change, all that is required are appropriate parsers and feature extractors.

Feature extractors take context objects as inputs, converting the context object into one or more features (scalar and/or categorical). The extracted features can be passed directly to the model training module or persisted to a file for use with external machine learning frameworks.

The Marmot pipelines are also very efficient because the most time-consuming stages (parsing, feature extraction, and training) are parallelised.

2.2. Data Preparation and Feature Extraction

Marmot provides framework that can extract features from a variety of input formats.

The tool can work with files in `.pra` format, which is the standard output of the TERp tool (Snover et al., 2008). This format contains the differences between an automatic translation and its human post-editions: the substituted, deleted, inserted and shifted words. Marmot converts this information to binary labels for each token. By using the `.pra` parser, a dataset consisting of $(source, hypothesis, reference)$ triples can be converted into a dataset for word-level quality estimation. Given the scarcity of public datasets for the word-level QE task, this is a major advantage.

Any additional information needed for feature extraction (POS-tags, alignments) can also be acquired at the preprocessing stage, which is often more efficient than computing the representation during feature extraction. This is especially true when a model for the additional information must be learned over the whole training dataset as, for example, when word-alignment features are used. Computing the model as pre-processing allows the Marmot experimental pipeline to run much more quickly, speeding up development.

All feature extractors inherit from one abstract base class. Because feature extractors implement the same interface, adding new features is easy, and feature sets can be specified just by listing the desired features in the configuration file.

Although extracting the additional representations at the pre-processing stage is more efficient, these can be acquired

during feature extraction as well. Hence a feature extractor that requires POS tags will look for them in the context object received as input. If it does not find them, it will try to generate POS tags using a POS-tagger if provided. Finally, if none of these options exists, this feature extractor will raise an informative error, guiding the user in correcting the configuration file.

The standard interface of the feature extractor makes it easy for users to implement new feature extractors. The current version of Marmot already contains a set of word-level features used in (Luong et al., 2014), and a set of phrase-level features that are used in QuEst for the sentence level[2].

2.3. Model Learning

There is a wide range of possibilities for model training. Classification can be performed with one of the classifiers defined in `scikit-learn`[3]. Sequence labelling is done using the `pystruct`[4] module. Analogously to parsing and feature extraction, the machine learning method used is specified in the configuration file, so any classifier from `scikit-learn` can be used directly. The training module can also easily integrate with any machine learning libraries that can take numpy arrays as input. Alternatively, Marmot can dump features in formats accepted by CRF++[5], CRFSuite[6] and SVMLight[7] tools.

The standard approach is to train a classifier for all the training examples. Marmot can also train a separate classifier for every token that occurs in the training data. This approach allows a suite of classifiers to model differences in distributions of errors between individual words or word classes.

3. Benchmarking

Marmot was used as baseline system in the WMT15 quality estimation shared task (Word-Level QE)[8]. It was also used to produce the baseline feature set which was made available to all participants. Table 1 shows the official results of the WMT15 task. Despite the fact that the baseline system (bottom line of the Table) performs poorly, most other systems used either the baseline feature set or Marmot pipelines for feature extraction and training show better results. Table 1 also contains an additional result (shown in grey) which is worth mentioning: it was produced by a system trained with Vowpal Wabbit (VW) toolkit (Goel et al., 2008) on baseline features. The system was a trial experiment by the HDCL team (Kreutzer et al., 2015). It demonstrates that features extracted with Marmot can be very effective if used with the right machine learning algorithm.

[2] `http://www.quest.dcs.shef.ac.uk/quest_files/features_blackbox`

[3] `http://scikit-learn.org/`

[4] `https://pystruct.github.io/`

[5] `https://taku910.github.io/crfpp/`

[6] `http://www.chokkan.org/software/crfsuite/`

[7] `http://svmlight.joachims.org/`

[8] `http://statmt.org/wmt15/quality-estimation-task.html`

System ID	F_1-**Bad**
UAlacant/OnLine-SBI-Baseline	43.12
HDCL/QUETCHPLUS	43.05
UAlacant/OnLine-SBI	41.51
Baseline features + VW classifier	40.84
SAU/KERC-CRF	39.11
SAU/KERC-SLG-CRF	38.91
• SHEF2/W2V-BI-2000	38.43
• SHEF2/W2V-BI-2000-SIM	38.40
SHEF1/QuEst++-AROW	38.36
UGENT/SCATE-HYBRID	36.72
• **DCU-SHEFF/BASE-NGRAM-2000**	36.60
HDCL/QUETCH	35.27
• **DCU-SHEFF/BASE-NGRAM-5000**	34.53
SHEF1/QuEst++-PA	34.30
UGENT/SCATE-MBL	30.56
RTM-DCU/s5-RTM-GLMd	23.91
RTM-DCU/s4-RTM-GLMd	22.69
• **Baseline features + CRFSuite**	16.78

Table 1: Official results for the WMT15 Quality Estimation Task 2. Systems whose results are significantly different with $p = 0.05$ are grouped by a horizontal line. Systems **in bold** used the baseline feature set. Systems that used Marmot toolkit for feature extraction and/or model training are indicated by a •.

Both winning systems — from HDCL and UAlacant (Esplà-Gomis et al., 2015) teams — were trained using the set of baseline features in addition to system-specific features. Table 2 gives the comparison of the performance of these systems with and without baseline feature set. In both cases, the baselines improve performance significantly.

System ID	F_1-**Bad**
UAlacant/OnLine-SBI	41.51
UAlacant/OnLine-SBI + Baseline	**43.12**
HDCL/QUETCH	35.27
HDCL/QUETCH + Baseline	**43.05**

Table 2: Performance of WMT15 systems with and without the baseline features.

4. Conclusions and future work

We have presented Marmot, a new open-source framework for translation quality estimation at the word level. The main feature of this tool is its modularity and flexibility, which enables the building of QE models on diverse data. Experimental pipelines can be completely specified via configuration files, speeding up the development process, and flattening the learning curve for the toolkit.

The system has the following properties:

- It is written in Python, which is easy to learn and read.
- Users can easily extend functionality by adding support of new data formats and features.

- Any classification algorithm from scikit-learn can be directly used within the training pipeline or features can be used with external tools (Weka, CRF++, etc.).
- The code is parallelised, which makes it much more efficient.
- Experiment configuration files allow the use of the toolkit and the creation of pipelines without writing new code.

One of the directions of future work will be to improve the quality of the system's output by implementing new features and integrating feature selection methods. The system will also be integrated via a server module into a computer-assisted translation (CAT) tool, which performs automatic translation of a sentence and passes it to a human translator for post-edition.
The Marmot toolkit can be downloaded from `https://github.com/qe-team/marmot/`.

5. Acknowledgements

This work was supported by the EXPERT (EU Marie Curie ITN No. 317471) project.

6. Bibliographical References

Blatz, J., Fitzgerald, E., Foster, G., Gandrabur, S., Goutte, C., Kulesza, A., Sanchis, A., and Ueffing, N. (2004). Confidence Estimation for Machine Translation: workshop report. Technical report, Johns Hopkins University.

Bojar, O., Buck, C., Callison-Burch, C., Federmann, C., Haddow, B., Koehn, P., Monz, C., Post, M., Soricut, R., and Specia, L. (2013). Findings of the 2013 Workshop on Statistical Machine Translation. In *WMT-2013*, pages 1–44, Sofia, Bulgaria, August.

Bojar, O., Chatterjee, R., Federmann, C., Haddow, B., Hokamp, C., Huck, M., Logacheva, V., Koehn, P., Monz, C., Negri, M., Pecina, P., Post, M., Scarton, C., Specia, L., and Turchi, M. (2015). Findings of the 2015 Workshop on Statistical Machine Translation. In *WMT-2015*, Lisbon, Portugal.

Buck, C., Pecina, P., Leveling, J., Post, M., Specia, L., and Saint-amand, H. (2014). Findings of the 2014 workshop on statistical machine translation. In *Proceedings of WMT-14*, pages 12–58, Baltimore, Maryland, USA, June.

Camargo de Souza, J. G., González-Rubio, J., Buck, C., Turchi, M., and Negri, M. (2014). Fbk-upv-uedin participation in the wmt14 quality estimation shared-task. In *Proceedings of WMT-14*, pages 322–328, Baltimore, Maryland, USA, June.

Esplà-Gomis, M., Sánchez-Martínez, F., and Forcada, M. (2015). Ualacant word-level machine translation quality estimation system at wmt 2015. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 309–315, Lisbon, Portugal.

Goel, S., Langford, J., and Strehl, A. L. (2008). Predictive indexing for fast search. In *Advances in Neural Information Processing Systems (NIPS)*, Vancouver, Canada.

Kreutzer, J., Schamoni, S., and Riezler, S. (2015). Quality estimation from scratch (quetch): Deep learning for word-level translation quality estimation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 316–322, Lisbon, Portugal.

Luong, N. Q., Besacier, L., and Lecouteux, B. (2014). Lig system for word level qe task at wmt14. In *Proceedings of WMT-14*, Baltimore, Maryland, USA, June.

Sharon O'Brien, et al., editors. (2014). *Post-Editing of Machine Translation: Processes and Applications*. Cambridge Scholars Publishing.

Raybaud, S., Lavecchia, C., Langlois, D., and Smaïli, K. (2009). Word- and sentence-level confidence measures for machine translation. In *Proceedings of EAMT-2009*, pages 104–111, Barcelona, Spain, May.

Shah, K., Logacheva, V., Paetzold, G., Blain, F., Beck, D., Bougares, F., and Specia, L. (2015). Shef-nn: Translation quality estimation with neural networks. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 342–347, Lisbon, Portugal.

Snover, M., Madnani, N., Dorr, B., and Schwartz, R. (2008). TERp System Description. In *Proceedings of AMTA-2008, MetricsMATR workshop*.

Specia, L., Paetzold, G., and Scarton, C. (2015). Multi-level translation quality prediction with quest++. In *ACL-IJCNLP 2015 System Demonstrations*, pages 115–120, Beijing, China.

Turchi, M., Negri, M., and Federico, M. (2015). Mt quality estimation for computer-assisted translation: Does it really help? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 530–535, Beijing, China, July. Association for Computational Linguistics.

Ueffing, N. and Ney, H. (2007). Word-Level Confidence Estimation for Machine Translation. *Computational Linguistics*, 33(1):9–40.