Tools and Guidelines for Principled Machine Translation Development

Nora Aranberri,[*] Eleftherios Avramidis,[†] Aljoscha Burchardt,[†]
Ondřej Klejch,[‡] Martin Popel[‡] and Maja Popović[▽]
[*]University of the Basque Country (UPV/EHU), IXA Group – nora.aranberri@ehu.eus
[†]DFKI Berlin – {eleftherios.avramidis, aljoscha.burchardt}@dfki.de
[‡]Charles University in Prague, Faculty of Mathematics and Physics, UFAL – {klejch, popel}@ufal.mff.cuni.cz
[▽]Humboldt University of Berlin – maja.popovic@hu-berlin.de

Abstract

This work addresses the need to aid Machine Translation (MT) development cycles with a complete workflow of MT evaluation methods. Our aim is to assess, compare and improve MT system variants. We hereby report on novel tools and practices that support various measures, developed in order to support a principled and informed approach of MT development. Our toolkit for automatic evaluation showcases quick and detailed comparison of MT system variants through automatic metrics and n-gram feedback, along with manual evaluation via edit-distance, error annotation and task-based feedback.

Keywords: evaluation methodologies; machine translation; tools, systems, applications

1. Introduction

Machine Translation (MT) development cycles depend on suitable MT evaluation tools and methods that help to assess, compare, and improve system variants. Yet, MT evaluation is a notoriously difficult topic due to the variability of the language that makes it impossible to anticipate one or more "correct" translations in advance. Therefore, a full MT development process often relies on a mix of automatic measures and human rating. Automatic measures are computed against human reference translations in order to set up baselines, building corpora, quick comparisons etc. Human rating, such as comparison of MT output, is used at certain milestones for comparing the most prominent systems. More recently, human post-editing and explicit error mark-up have been used to bridge the gap towards the use of MT in language industry, where the post-editing effort and the (absence of major) errors are decisive factors for determining the usability of MT engines for a given task. In this work, we report on novel tools – one for automatic evaluation and one for human evaluation – and report on practices that we have developed in the context of the QTLeap project[1] in order to support a principled and informed approach to MT development.

The structure of this paper is as follows: Section 2. outlines the main functionality of MT-COMPAREVAL,[2] an open-source tool for MT development that has been developed within the project and is used to "automatically" monitor MT improvements. Section 3. describes the manual evaluation being performed in the project on a regular basis for gaining insights based on qualitative inspection. Finally, Section 4. concludes this paper.

2. Developer's Workbench: MT-COMPAREVAL

MT-COMPAREVAL is an open-source tool that has been designed to help MT developers to compare and evaluate various MT engines/experiments and settings using several measures that represent the current best practices. The system helps to tie the development cycles together by linking three ingredients:

- An *evaluation panel* that provides a graphical interface for comparing the performance of various systems on the same output, visualizing automatic scores and various types of statistics and allowing easy manual checking of translation quality on a sentence level.
- A *back-end* that monitors some pre-defined storage locations for the addition of new translation outputs. Whenever a new translation output is detected, a new task is added in the background database.
- An *evaluation mechanism* that calculates MT evaluation scores based on a set of automatic evaluation metrics and statistical tests. These results are associated with the translation tasks in the database.

Demo server

In order to showcase the features of MT-COMPAREVAL, we present a demonstration server.[3] The interface includes pre-loaded system outputs from the WMT Shared Task 2014 and 2015. A detailed description of MT-COMPAREVAL including installation instructions can be found in Klejch et al. (2015).

2.1. Quantitative Feedback: Automatic Measures

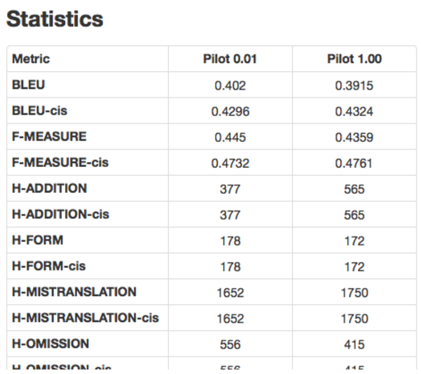
MT-COMPAREVAL provides the following automatic measures for each of the systems and tasks in the database, for quick comparison of the output:

BLEU (Papineni et al., 2002) an n-gram based metric, which uses the geometric mean of word 1-, 2-, 3- and 4-grams. The final score is calculated as the precision of these average n-grams including a brevity penalty (in contrast to the recall used by F-score below).

[1] http://qtleap.eu

[2] https://github.com/choko/MT-ComparEval

[3] http://wmt.ufal.cz



Statistics

Metric	Pilot 0.01	Pilot 1.00
BLEU	0.402	0.3915
BLEU-cis	0.4296	0.4324
F-MEASURE	0.445	0.4359
F-MEASURE-cis	0.4732	0.4761
H-ADDITION	377	565
H-ADDITION-cis	377	565
H-FORM	178	172
H-FORM-cis	178	172
H-MISTRANSLATION	1652	1750
H-MISTRANSLATION-cis	1652	1750
H-OMISSION	556	415
H-OMISSION-cis	556	415

Figure 1: Screenshot of the automatic measures shown by MT-ComparEval (excerpt only). “cis” means *case insensitive* (metrics without this suffix are case sensitive).

F-Score (Popović, 2012) used in our evaluation is based on the arithmetic mean of word n-grams (n=1,..,4) (instead of BLEU’s geometric mean, so as to avoid too hard penalties for unseen word sequences, thus improving sentence-level reliability). The final score is calculated as the harmonic mean between precision and recall.

Hjerson (Popović, 2011) is a tool for automatic error classification based on Word Error Rate (i.e. edit distance), precision and recall. It allows for word-level error categorization into five types: inflectional error, word-order error, omission, addition and mistranslation (i.e. lexical error).

Figure 1 shows a screenshot of MT-ComparEval, where the results of different metrics are displayed.[4] This information allows for quick, repetitive and inexpensive comparisons of system versions during the development cycle to measure the global impact of specific modifications against a reference. It provides information as to the contribution of different techniques or resources and also allows to identify considerable differences between system versions to carry out more resource-expensive qualitative evaluations.

2.2. Feedback for Inspection by Developers: Confirmed and Unconfirmed n-grams

MT-ComparEval also calculates n-gram statistics (for n=1,2,3,4), providing developers with a degree of qualitative insight into system variants. By displaying differences in n-grams, they can obtain a quick view of the specific changes in the output of any two MT systems, sysA and sysB (where one can be a newer version of the other, or they can be two independent systems).

Each occurrence of a given n-gram in a given sentence falls into exactly one of the 7 categories in Figure 2, depending on its presence in sysA, sysB and the reference.[5] For example, n-grams present in sysA and the reference, but not in sysB, are called imprA because they are improving sysA’s output relative to sysB. We sum the presence of a given n-gram in the 7 categories over all sentences (so one n-gram can have several categories with non-zero counts). We define confA = confAB + imprA. Thus confA is the number of times a given n-gram was confirmed in sysA (“confirmed” means occurring both in sysA and in the reference for the same sentence). We compute confDiff = confA − confB = (confAB + imprA) − (confAB + imprB) = imprA − imprB. Similarly, we compute unconfDiff = unconfA − unconfB = worseA − worseB.

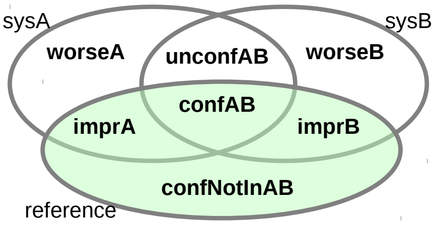
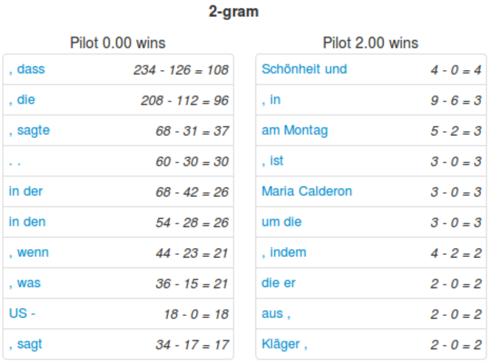


Figure 2: Breakdown of n-grams into 7 categories



2-gram

Pilot 0.00 wins	
, dass	234 - 126 = 108
, die	208 - 112 = 96
, sagte	68 - 31 = 37
. .	60 - 30 = 30
in der	68 - 42 = 26
in den	54 - 28 = 26
, wenn	44 - 23 = 21
, was	36 - 15 = 21
US -	18 - 0 = 18
, sagt	34 - 17 = 17

Pilot 2.00 wins	
Schönheit und	4 - 0 = 4
, in	9 - 6 = 3
am Montag	5 - 2 = 3
, ist	3 - 0 = 3
Maria Calderon	3 - 0 = 3
um die	3 - 0 = 3
, indem	4 - 2 = 2
die er	2 - 0 = 2
aus ,	2 - 0 = 2
Kläger ,	2 - 0 = 2

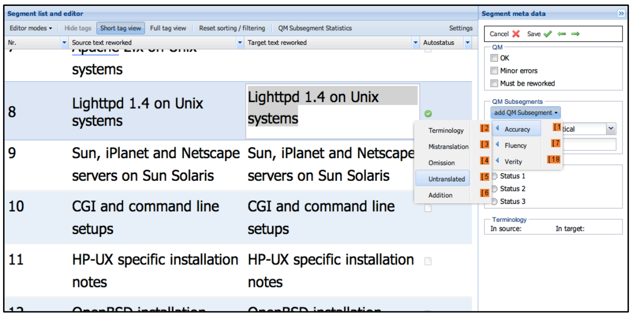
Figure 3: Screenshot of the table of confirmed n-grams. Only bigrams are shown here. The scores (e.g. 234 − 126 = 108) mean confA − confB = confDiff, see Section 2.2.

Thus, confDiff measures whether a given confirmed (“correct”) n-gram is produced more often by sysA than sysB (positive values) or vice versa (negative values). MT-ComparEval computes confDiff for all n-grams and plots top ten positive ones (marked “sysA wins”) and top ten negative ones (marked “sysB wins”, switched to positive values) in the table of Confirmed n-grams (see Figure 3). Similarly, it plots the top ten unconfirmed n-grams (i.e. sorted according to unconfDiff). These tables proved to be very useful for the developers when searching for possible improvements of their systems. We are considering to merge these two tables (confirmed and unconfirmed n-grams) into one, where we would report confDiff − unconfDiff.

This short-phrase-ranged perspective already allows developers to check for local agreements and ordering, as well as vocabulary coverage, to mention a few aspects. With this information, a more guided development can continue until a more robust system is ready for another iteration of qualitative inspection.

[4] It is planned to normalize all metrics’ values to the range of 0–100 for a uniform display in a future update of the tool.

[5] For simplicity of this description, we ignore the cases when an n-gram is present multiple times in one sentence. These cases are handled correctly by MT-ComparEval.



Nr.	Source text reworked	Target text reworked
	systems	
8	Lighttpd 1.4 on Unix systems	Lighttpd 1.4 on Unix systems
9	Sun, iPlanet and Netscape servers on Sun Solaris	Sun, iPlanet and Netscape servers on Sun Solaris
10	CGI and command line setups	CGI and command line setups
11	HP-UX specific installation notes	HP-UX specific installation notes

QM: □ OK □ Minor errors □ Must be reworked

Terminology In source: In target:

Figure 4: MQM error annotation in TRANSLATE5.

3. Manual Evaluation

Whereas the automatic evaluation presented above can be repeated often during the development phase, manual qualitative evaluation can only be performed following a fixed timeline. In our use case, the development effort has been organized in development phases with specific duration. Each development phase results into an MT system, referred to as a "pilot", that collects system improvements that are expected to have a positive effect on its performance.

For the qualitative evaluation of our pilots, we focused on three qualitative evaluation methods, namely, post-editing, error annotation and a task-based evaluation. Each of these tasks provides complementary information, answering three different questions about MT quality. *Post-editing* is used as a measure of how far the output is from having an acceptable quality for the intended purpose of the translation. Comparing MT output with the post-edited version rather than a random reference translation provides a more precise answer about its usefulness for professional translators. *Error annotation*, together with the frequency and impact level, provides information for establishing priorities and selecting development techniques and resources. Also, it helps decide whether the MT quality is sufficient for a particular purpose. The *task-based evaluation* considers the usability of the system output quality for a given task. It establishes whether the quality, regardless of the level and error types, is sufficient for users to benefit from its use.

For the first two human evaluation tasks, post-editing and error annotation, we have used the open-source tool TRANSLATE5[6] (see Figure 4 for a screenshot), a database-driven tool with a graphical user interface. Source texts, translations and annotations are organized in a relational database. The tool, originally implemented as a proofreading and post-editing environment for the translation industry, has been recently extended[7] to support Multidimensional Quality Metrics (MQM) error annotation (Lommel et al., 2014). The current implementation includes APIs for automatic processing steps, data import and export, as well as queuing, management and load-balancing of external and internal processes and handling of their dependencies. It also features reporting functions, user administration and facilities for simple workflow specification and client management. The annotation types supported are post-editing, MQM error tagging, and simple ranking, whereas its currently being developed to provide additional functionality.[8]

3.1. Post-editing Distance

Manual post-editing was done on 200 answers sampled from a test set (batch 2 of the QTLeap corpus[9]), for each one of our 7 language pairs (from English to Basque, Bulgarian, Czech, Dutch, German, Spanish and Portuguese). The instruction to the annotators was to perform *minimal* post-editing, i.e., to produce a correct translation that is close to the MT output, but to refrain from re-formulations even if the latter would have been stylistically preferred. The Word Error Rate between the system outputs and the reference indicates how far a system is from producing the correct translation.

WER	true case	lowercase
en-bg	33.3	29.7
en-cs	24.7	24.3
en-de	28.4	28.1
en-es	56.1	54.3
en-eu	72.5	68.9
en-nl	23.8	23.5
en-pt1	52.4	50.2
en-pt2	57.5	55.8

Table 1: Edit distance (Word Error Rate – WER) between translation outputs and their post-edits.

Table 1 shows the edit distance between translation outputs of the first "deeper" MT Pilots in QTLeap (internally called Pilot 1) and their post-edits. The results show that the English-Dutch, English-Czech and English-German systems are closer to an acceptable output than the English-Portuguese and English-Spanish systems, and that the English-Basque clearly lags behind with a considerable need for changes. Although the results are not comparable across the language pairs, given the different nature of the languages involved, they are indicative of the amount of work to be done.

3.2. Error Annotation

While error annotation is a common practice in industry for quality assurance of human and computer-aided translations, this type evaluation has rarely been applied in the development of MT systems. Error annotation allows for both qualitative analysis of the annotated examples and quantitative analysis of the error distribution.

Here we present our first piloting steps in this direction, by performing an – admittedly small-scale – error annotation exercise.

In this task, volunteers from within the project were asked to annotate sentences from two pilot systems created in the project (Pilot 1 and Pilot 2) using a selection of issue types

[6] implemented by MittagQI; `http://translate5.net`

[7] The extensions for MQM were supported by the EC-funded project QTLaunchPad `http://www.qt21.eu/launchpad`.

[8] Current extensions are supported by the CRACKER project `http://cracker-project.eu`.

[9] `http://metashare.metanet4u.eu/go2/qtleapcorpus`

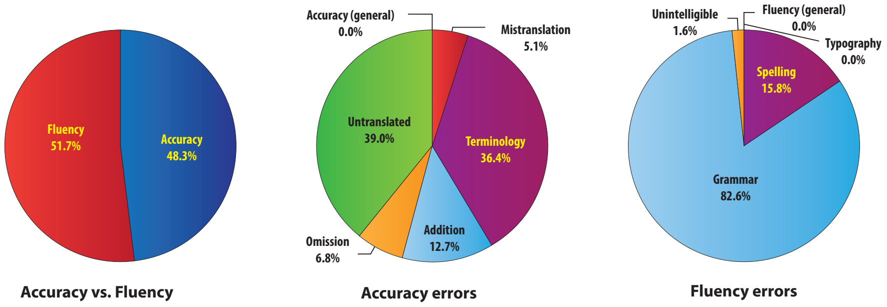


Figure 5: Sample MQM error distribution for English→Spanish.

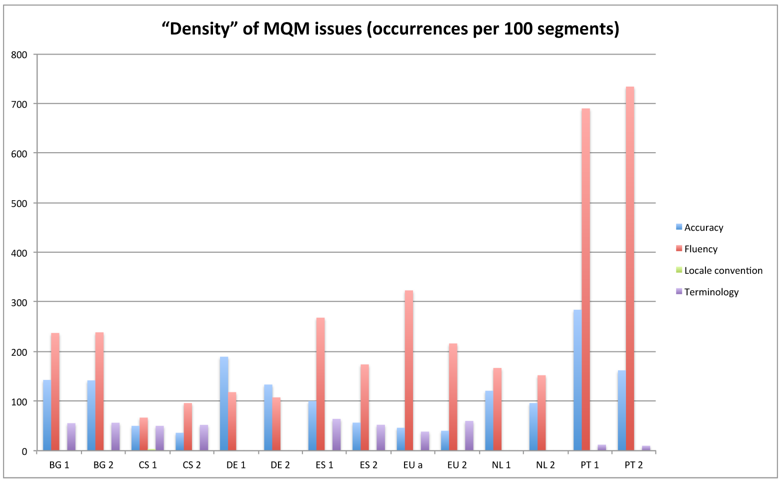


Figure 6: Density of high-level MQM issues for segments annotated from Pilot 1 and Pilot 2 for all languages.

taken from the MQM framework. It was decided to perform single annotation to save resources. That is, one annotator per language pair performed the error classification task to identify the most salient problems of the systems. The annotators were trained in a webinar and they received detailed annotation guidelines.

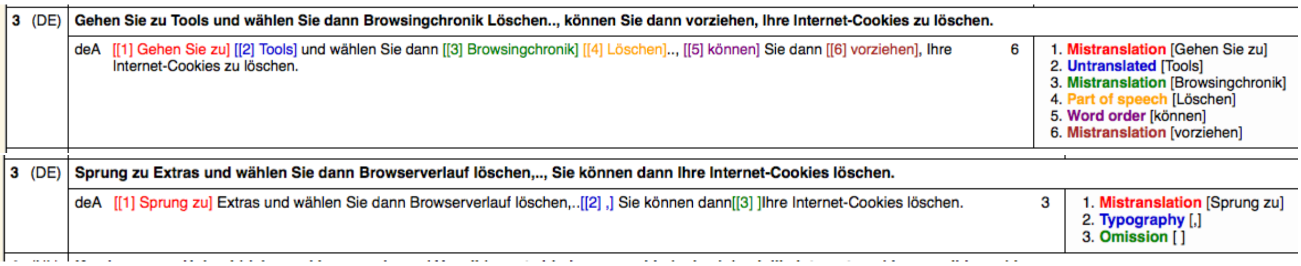
Annotators annotated the results from both Pilot 1 and Pilot 2, which appeared in separate columns. To prevent any bias that might come if the annotators knew which segment was from which pilot, we randomized the column the results appeared in, so that each column had roughly equal numbers of results from each pilot.

The annotators were instructed to skip sentences that were too difficult to annotate or that required no annotation, and to indicate their reason for skipping using the Notes feature of TRANSLATE5. Based on feedback from the annotators, in most cases the skipped sentences were simply too difficult to annotate because of the poor translation quality.

As the option of skipping segments has been used frequently, the annotation results provide information about the relative occurrence of errors in translations from Pilot 1 and Pilot 2 on segments that belong to the better segments. This information is relevant, e.g., if one thinks of the usage of MT in a production setting (possibly with post-editing options as performed by experts like the operators of HF company) where only the better MT segments can be used while the absence of certain errors, the overall number of errors, etc. must be tightly controlled. In our development setting, this information provides insights into the qualitative nature of errors and provides starting points for system improvements.

As an example, Figure 5 shows a breakdown of the annotated errors for translations from English into Spanish for Pilot 1. The numbers of Accuracy and Fluency errors are quite similar, with slightly more Fluency than Accuracy errors. The system was particularly likely to leave content untranslated and issues related to Mistranslation were particularly likely to be Terminology errors, rather than general Mistranslation errors. Terminology errors include issues with specialized vocabulary as well as product names and user interface strings. Within Fluency, Grammar was by far the most common error category, with significant numbers of Spelling errors and all other error types being relatively negligible. A considerable amount of Spelling errors concern irregular word formations. Verbs whose lemma is slightly altered when conjugated are a good example.

In order to do a comparison between the pilots we focused on the "density" of each error type rather than their total number (which would vary with the number of segments annotated). To calculate this number we counted the total number of instances of each MQM issue type, multiplied it by 100 and divided by the total number of segments annotated. This calculation does *not* provide the percentage of segments exhibiting a given error (since a segment can have multiple instances of an error), but rather gives us an approximation of how many instances of the error would occur in 100 segments. It is thus only superficially similar to a percentage.



3 (DE)	**Gehen Sie zu Tools und wählen Sie dann Browsingchronik Löschen.., können Sie dann vorziehen, Ihre Internet-Cookies zu löschen.**		
	deA [[1] Gehen Sie zu] [[2] Tools] und wählen Sie dann [[3] Browsingchronik] [[4] Löschen].., [[5] können] Sie dann [[6] vorziehen], Ihre Internet-Cookies zu löschen.	6	1. **Mistranslation** [Gehen Sie zu] 2. **Untranslated** [Tools] 3. **Mistranslation** [Browsingchronik] 4. **Part of speech** [Löschen] 5. **Word order** [können] 6. **Mistranslation** [vorziehen]
3 (DE)	**Sprung zu Extras und wählen Sie dann Browserverlauf löschen,.., Sie können dann Ihre Internet-Cookies löschen.**		
	deA [[1] Sprung zu] Extras und wählen Sie dann Browserverlauf löschen,..[[2] ,] Sie können dann[[3]]Ihre Internet-Cookies löschen.	3	1. **Mistranslation** [Sprung zu] 2. **Typography** [,] 3. **Omission** [ ]

Figure 7: MQM annotated output of Pilot 1 (top) and Pilot 2 (bottom).

The density figures for each language for the top-level MQM categories are presented in Figure 6. In this case a lower number for a given column between Pilot 1 and Pilot 2 represents an improvement in this particular category (i.e., fewer instances of an issue type were found in Pilot 2); conversely a higher number can represent a decrease in performance.

It is critical to note that numbers cannot be compared across languages. For example, it might appear that Portuguese has many more errors than the other languages, but differences in annotation style such as when to skip segments and other factors render any such comparison pointless: the only valid comparison is between Pilot 1 and Pilot 2 within a given language.

As annotators were allowed to skip sentences and because most issues appeared only in very small numbers, statistical significance cannot be demonstrated and small changes generally cannot be interpreted as significant in any way. The results are thus only indicative in nature. Yet, there seem to be tendencies of error reduction from pilot to pilot. When inspecting the annotations qualitatively, it becomes obvious that sometimes improvements of certain issues are relativized by a certain loss of performance at other places, e.g., when comparing the German Pilot 2 and Pilot 1 (see Figure 7 on the following example:

Source Go to Tools and then choose 'Delete browsing history..', you can then choose to delete your Internet cookies.

Reference Gehen Sie zu Extras und wählen Sie dann 'Löschen der Browser Geschichte .. ', dann können Sie wählen, ob Sie Ihre Internet-Cookies löschen möchten.

One can see that Pilot2 gets the term "Tools" right ("Extras") while at the same time failing on the imperative *Go to* and omitting the information that one can *choose* to delete cookies. Ideally, improvements in outbound high-quality MT would lead to the absolute reduction of the number of errors for each sentence, especially for the "better" ones that allow for post-editing or direct use without human editing.

To conclude this section on error annotations, we can say that the actual annotation and qualitative analysis was experienced as providing valuable insights into the issues occurring in the output by the translation pilots. It has also become clear that for future projects, resources for professional human annotators (i.e., translators) need to be foreseen. These should allow to follow a more systematic annotation policy, where the relevant segments to be used are "filtered" out before annotation and then used throughout the project on all future improvements of the given MT system. In the same line of thinking, informed sampling strategies are a highly relevant topic for future research.

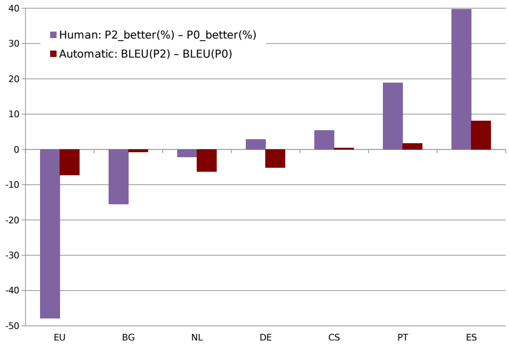


Figure 8: Comparison of user evaluation results and BLEU scores for Pilot 2 and Pilot 0.

Actually, the insights gained in QTLeap have informed the annotation planning and strategy adopted in the QT21 project.

3.3. Task-based Evaluation

In the task-based evaluation we engaged human evaluators in order to estimate the usefulness of MT and to test if improvements of MT technology lead to better performance in a real usage scenario. In our particular case, the evaluation is based on the integration of MT services in a helpdesk application developed by the company Higher Functions as part of its business. Given a user query in a language, the application translates it into English, searches the database for a relevant answer and translates it back to the user's language. A question and a machine translated answer were presented to evaluators and they were asked to estimate to what extent the translation would help them address the question. Below, we shortly present the main results. More detailed information on the user evaluation can be found in Gaudio et al. (2015).

The task-based evaluation has been so far performed for three systems, Pilot 0 (the baseline), Pilot 1 and Pilot 2. Figures 8 and 9 present the BLEU difference (dark red bars) in relation to the difference between Pilots according to the human task-based evaluation (violet bars).

The task-based results (Figure 8) show that the Spanish, Portuguese, Czech and German Pilot 2 outperforms the re-

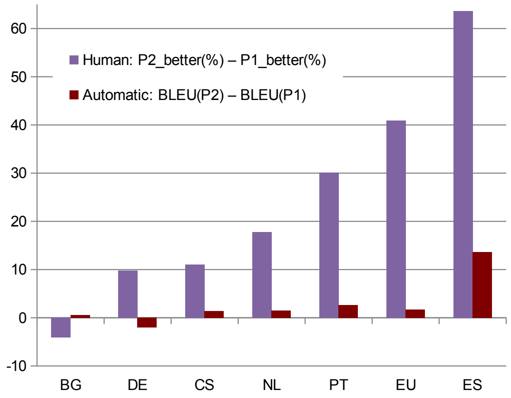


Figure 9: Comparison of user evaluation results and BLEU scores for Pilot 2 and Pilot 1.

spective Pilot 0 baseline system, whereas the Basque, Bulgarian and Dutch systems don't. Figure 9 shows that except for Bulgarian, all Pilot 2 systems outperform the respective Pilot 1.

It is interesting to observe that BLEU differences almost always agree with the user ratings on the comparison of two systems. There are just three exceptions: German Pilot 2 vs. Pilot 0 (see Figure 8), and German and Bulgarian Pilot 2 vs. Pilot 1 (see Figure 9). For German, the user ratings are generally more in favor of Pilot 2 (unlike BLEU).

4. Conclusion

As the main project goal of the QTLeap project is the improvement of MT engines, we have thoroughly focused on evaluation and have developed evaluation tools and practices such as the regular inspection of n-grams in the development cycle and human evaluation steps at milestones (comparing different pilot engines). In this work, we have shortly reported on the different aspects of evaluation and on the main tools we used. In particular, we have reported on automatic measures and n-gram information provided by the MT-COMPAREVAL tool to obtain quick insights into system quality during development. The automatic measures provide global quantitative feedback, whereas the n-gram information already provides some degree of qualitative insight. For more exhaustive checks in-between development cycles, we have reported on error annotation to guide further development, post-editing to check on the professional value of the output, and task-based evaluation to estimate the usability of the systems on real-life scenarios. Needless to say, all tasks serve to compare performance across system versions. We believe that this battery of evaluation techniques is necessary for a principled machine translation development and to understand the nature of the output.

One of the open research questions is if and how much the various measures eventually correlate. First experiments based on the (admittedly fairly small set of manual annotations) did not show any consistent correlation with the automatic measures. Still, some expected tendencies can be observed, e.g., user-preferred sentences often exhibit fewer errors than dis-preferred ones.

We hope that this inclusion of evaluation into the research and development strategy will inspire other colleagues to take steps towards a more principled approach to MT evaluation.

Acknowledgments

This work has received support by the EC (FP7/2007-2013) under grant agreement number 610516: "QTLeap: Quality Translation by Deep Language Engineering Approaches" and by the Czech Science Foundation (GAČR) grant no. GA15-10472S. This work has been using language resources distributed by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2015071).

References

Gaudio, R. D., Burchardt, A., and Lommel, A. R. (2015). Evaluating a machine translation system in a technical support scenario. In Jan Hajič et al., editors, *Proceedings of the 1st Deep Machine Translation Workshop. Deep Machine Translation Workshop (DMTW-2015)*, pages 39–47. Charles University in Prague, Faculty of Mathematics and Physics, Institute of Formal and Applied Linguistics.

Klejch, O., Avramidis, E., Burchardt, A., and Popel, M. (2015). MT-ComparEval: Graphical evaluation interface for machine translation development. *The Prague Bulletin of Mathematical Linguistics*, 104:63–74.

Lommel, A. R., Burchardt, A., and Uszkoreit, H. (2014). Multidimensional quality metrics (MQM): A framework for declaring and describing translation quality metrics. *Tradumàtica: tecnologies de la traducció*, 0(12):455–463, 12.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. In *ACL 2002*, pages 311–318, Philadelphia, PA, July.

Popović, M. (2011). Hjerson: An Open Source Tool for Automatic Error Classification of Machine Translation Output. *The Prague Bulletin of Mathematical Linguistics*, 96:59–68, October.

Popović, M. (2012). rgbF: An Open Source Tool for n-gram Based Automatic Evaluation of Machine Translation Output. *The Prague Bulletin of Mathematical Linguistics*, 98:99–108, October.