Using the ACCEPT framework to conduct an online community-based translation evaluation study

Linda Mitchell
Symantec Ltd.
Ballycoolin Business Park
Dublin 15, Ireland
linda_mitchell@symantec.com

Johann Roturier
Symantec Ltd.
Ballycoolin Business Park
Dublin 15, Ireland
johann_roturier@symantec.com

David Silva
Symantec Ltd.
Ballycoolin Business Park
Dublin 15, Ireland
david_silva@symantec.com

Abstract

This paper presents how a novel evaluation framework was used to collect translation ratings thanks to users of an online German-speaking support community in the IT domain. Using an innovative data collection approach and mechanism, this paper shows that segment-level ratings can be collected in an effective manner. The collection mechanism leverages the ACCEPT evaluation framework which allows data collection to be triggered from online environments in which community users interact on a regular basis.

1 Introduction

While machine translation is becoming ubiquitous in making Web content accessible to users who do not necessarily understand the language in which this content was first authored, some doubts remain about the ability of machine translation systems to generate content that is sufficiently fluent to be easily understood. Collecting translation evaluation ratings or judgments is often done in dedicated evaluation environments which fail to take into account any ecological validity requirements that are inherent to user-focused settings where translated content is made available. The approach presented in this paper addresses this issue by leveraging the evaluation framework provided by the ACCEPT project and by designing an evaluation task aimed at being as self-contained, self-explanatory, intuitive and engaging as possible for the target population of raters. The paper is organized as follows: Section 2 briefly reviews existing evaluation frameworks Section 3 focuses on the setup of an evaluation task that maximizes user engagement. Section 4 presents some statistics on the evaluation data collected during a 4-week timeframe. Finally, Section 5 contains some preliminary conclusions and suggestions for future work.

2 Related work

Numerous (machine) translation evaluation systems exist. These systems can be grouped into four categories: standalone desktop-based systems, Web-based dedicated systems, Web-based generic systems and Web-based hybrid systems. The first type of system (standalone desktop-based system such as Costa (Chatzitheodorou, 2013) is not relevant for studies involving online community members because it is inconvenient to ask users to install an application to provide ratings. Web-based dedicated systems such as Appraise (Federmann, 2012) or the Dynamic Quality Framework (DQF) Tools[1] may be useful to collect judgments from well-known contributors but they are not suited to collect genuine user feedback (mainly because they require a separate account creation and login process which can be cumbersome for users whose first priority is to consume content rather than evaluate content). Web-based generic systems such as Amazon Mechanical Turk (Callison-Burch, 2009) or Crowdflower[2] suffer from the same problem as dedicated systems. While they can be useful in generating large data volumes in a short period of time using a crowdsourcing approach, they also require a separate login process. Web-based hybrid systems,

[1] https://evaluation.taus.net/tools

[2] http://www.crowdflower.com/

such as Google forms,[3] offer maximum flexibility because the project management and data collection processes are decoupled from the actual data generation process. One such system is made available by the ACCEPT framework, whereby the project management process is completed using the evaluation section of the ACCEPT portal while the data generation process is performed using a JavaScript widget that is injected in an online community environment (an online discussion forum). Since the Evaluation Framework is built under a RESTful API endpoint,[4] this architecture enables the API to be used from any device that can make use of the HTTP protocol, regardless of the technology used for it. The following section provides more detail on the actual implementation of the widget.

3 Experimental setup

A previous study (Mitchell and Roturier, 2012) collected user ratings in an online forum context using a similar approach. However, their approach did not yield a large number of ratings for three main reasons:

- Users were asked to rate translations at the post level (which made the task quite cumbersome),
- The evaluation task was not immediately visible in the online forum environment (i.e. users had to click a number of times to find the content to evaluate),
- The evaluation task was not sufficiently engaging: once a rating was submitted, users were thanked but were not automatically presented with additional content to evaluate.

The present study tried to address these shortcomings by using an improved approach. Ratings would be collected at the segment level instead of the document level; the client-side evaluation widget would be positioned in a prominent place on the online forum (i.e. on the right-handside of the landing page); the widget would offer users a new segment to rate after receiving a rating; the widget would keep track of user ratings in order to avoid asking users to rate the same segment twice.

In order to be able to collect evaluation data on the client side, an evaluation project had to be defined using the Web interface of the ACCEPT portal.

Once the project was created, an API key was automatically generated and associated with this project. The next step was to define a question category, which was used as a container for a specific question. An example of such a category would be *fluency*, where all associated questions would be related to the fluency of text. Once the category was defined, the question that should be answered by users was added. A question has multiple attributes besides the question text itself (which could be “How fluent is this translated content?”):

- A language (e.g. English if the Question text is in English),
- An Action text (which may be used to instruct users how to submit an answer),
- An Action confirmation text (which may be used to show users that their answer has been submitted).

Finally, any number of answers may be added to a question. Each answer has two parts: the actual answer text to present to the user and a value (e.g. *Perfect* and *5*).

Evaluation content was then added to the project. The source content had been extracted from one of Symantec’s English forums. The evaluation was based on 50 segments (which were not necessarily complete sentences), 25 of which had been machine-translated from English to German using the ACCEPT SMT system.[5]. The remaining 25 segments were machine-translated segments that had been post-edited by community users. These segments were selected from a pool of 1,700 segments, which had received fluency scores by 3 to 4 authoritative evaluators (e.g. localisation or technical support experts) prior to this evaluation task, as described in (Mitchell et al., 2014). They were selected based on the criterion that all of the evaluators agreed on the score for fluency (on a categorical scale from 1-5.[6] For each category between 1 and 5, 10 segments were selected. The content was uploaded to an evaluation

[3] `http://www.google.com/google-d-s/createforms.html`

[4] `http://en.wikipedia.org/wiki/Representational_state_transfer`

[5] `http://www.accept.unige.ch/Products/D_4_1_Baseline_MT_systems.pdf`

[6] 1=incomprehensible, 2=disfluent, 3=non-native, 4=good, 5=perfect

project on the ACCEPT evaluation portal[7] using a JSON file based on the following format:

```
{ "chunkList": [{
    "chunk":"Alle Resourcen sagen,
    dass diese Infektion nur auf
    PCs zutrifft und bieten
    Lösungen für PCs an.",
    "chunkInfo":"",
    "active":1
    },
]}
```

Listing 1: JSON format used to upload content that should be evaluated

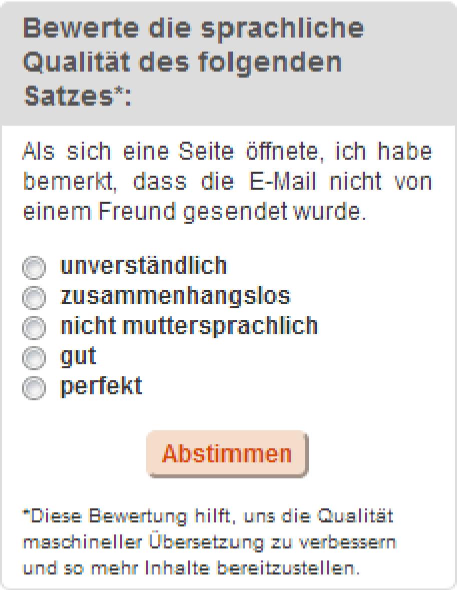
On the client side, the JavaScript client shown in Figure 1 was pre-configured with:

- The public API key generated during the process of creating the evaluation project,
- The ID of the category that should be presented to users (i.e. *fluency*),
- Under the category, the ID identifying the question and answer options that should be displayed.

During the Web page's loading process, the above configuration is used to get the necessary information from the REST API and dynamically build on-the-fly a Web form containing the specified question and the set of possible answers in a format of a radio buttons list, as shown in Figure 1. When the page is fully loaded, a form is displayed and the evaluation process can take place. Once the user chooses and submits an answer by clicking "Abstimmen", the form is serialized and sent over the Web by issuing a POST request against an API method.[8] The payload of this request may contain the ID of the question the user is answering, the API key used to identify the request, the ID of the chosen answer, the content (text) being evaluated and optional meta-data such as the IP address of the client, the ID the user if it can be found on the Web page, etc.[9]

4 Data analysis

During a four-week period, 1470 ratings were collected as shown in Table 1.



Bewerte die sprachliche Qualität des folgenden Satzes*:

Als sich eine Seite öffnete, ich habe bemerkt, dass die E-Mail nicht von einem Freund gesendet wurde.

○ unverständlich
○ zusammenhangslos
○ nicht muttersprachlich
○ gut
○ perfekt

*Diese Bewertung hilft, uns die Qualität maschineller Übersetzung zu verbessern und so mehr Inhalte bereitzustellen.

Figure 1: Client-side widget

Category	Number	%
Incomprehensible	457	31
Disfluent	208	14
Non native	387	26
Good	270	19
Perfect	148	10

Table 1: Evaluation Ratings

The ratings received were submitted by 171 users in total, of which 143 were unregistered users and 28 registered community members. We did not expect these users do have a strong bias towards machine translation since this technology does not pose any apparent threat to their profession. The average number of ratings per user session was eight ratings. The fifty segments received 29 ratings on average (from 8 to 100).

To get an overview of the heterogeneity of the ratings and to identify to what extent evaluators deviated from the average score per segment, a sampling strategy was employed. Segments 23 and 24 were selected - they had received 53 and 54 ratings with a mean of 4.43 and 1.33 as a score, respectively.

Samples from these ratings were selected in 5% increments, from 10% to 95% of all ratings received for a particular segment. For instance, if a segment had received 50 ratings, a 10% sample contained 5 ratings. For each increment, 20 samples were built randomly and the average was calculated based on these, which were then subse-

[7] http://www.accept-portal.eu

[8] http://www.ProjectName-portal.com/AcceptApiStg/Api/v1/Evaluation/ScoreFormPost

[9] http://www.accept.unige.ch/Products/D5.3_Adapted_Evaluation_Portal_Prototype.pdf

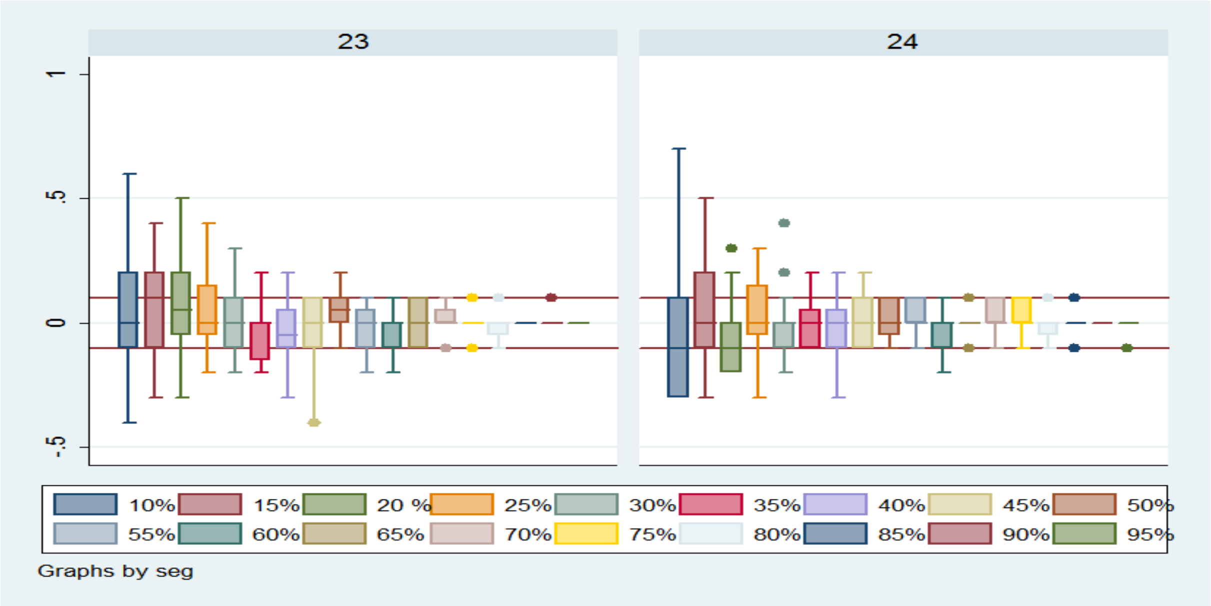


Figure 2: Average score variation for various sample sizes of user ratings

quently compared to the overall average. Figure 2 shows the results from the sampling for two of the segments. The y-axis represents how much the average of the samples per increment differs from the overall average. The box represents one standard deviation in each direction and contains 50% of all data points for each of the increments. The whiskers cover two standard deviations. Outliers are represented as dots. As expected, the larger the samples, the smaller the extent to which the average deviates from the overall average. It can be seen that in both cases to be able to achieve ratings using samples that will always be within 0.1 of the overall average (represented by the horizontal lines), 65% (35 ratings) of the ratings had to be sampled. 15 segments in total had at least one outlier. For 8 of the post-edited segments, outliers deviate from the average on average by 2.62. For 9 of the machine-translated segments, outliers deviate from the average on average by 2.17.

5 Conclusions and future work

This paper has shown that the ACCEPT framework can be used to set up community-based translation evaluation tasks. Such tasks maximize the ecological validity of the ratings obtained because they tend to be provided by users who are used to interacting with the system in which the client-side widget is deployed. While the present study focused on collecting ratings about the fluency of machine-translated and post-edited segments, future work will investigate how adequacy ratings could be obtained in a similar manner. This work will involve targeting users who have some knowledge of the source language. We are also interested in finding out whether the present data collection process can be further optimized by automatically identifying when a sufficient number of ratings has been obtained for a given segment.

References

Callison-Burch, Chris. 2009. Fast, Cheap, and Creative: Evaluating Translation Quality Using Amazon's Mechanical Turk. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, number August, pages 286–295, Singapore.

Chatzitheodorou, Konstantinos. 2013. COSTA MT Evaluation Tool: An Open Toolkit for Human Machine Translation Evaluation. *The Prague Bulletin of Mathematical Linguistics*, 100(1):83–89.

Federmann, Christian. 2012. Appraise: an Open-Source Toolkit for Manual Evaluation of MT Output. *The Prague Bulletin of Mathematical Linguistics*, 98:130–134.

Mitchell, Linda and Johann Roturier. 2012. Evaluation of Machine-Translated User Generated Content: A pilot study based on User Ratings. In *Proceedings of EAMT 2012*, pages 61–64, Trento, Italy.

Mitchell, Linda, Sharon O'Brien, and Johann Roturier. 2014. Quality Assessment in Community Post-Editing. *Machine Translation*. Forthcoming.