ACCEPT

SEVENTH FRAMEWORK PROGRAMME

THEME ICT-2011.4.2(a)

Language Technologies

ACCEPT

Automated Community Content Editing PorTal

www.accept-project.eu

Starting date of the project: 1 January 2012

Overall duration of the project: 36 months

Weighting of Pre-editing Rules

Work package n° 9	Name: MT Evaluation
Deliverable n° 9.3	Name: Weighting of Pre-editing Rules
Due date: 30 June 2014	Submission date: 30 June 2014

Dissemination level: PU

Organisation name of lead contractor for this deliverable: ACROLINX

Author(s): Robert Grabowski

Internal reviewer(s): Pierrette Bouillon, Violeta Seretan

Proofreading: Manny Rayner

The research leading to these results has received funding from the European Community's Seventh Framework Programme (*FP7/2007-2013*) under *grant agreement n° 288769.*

SEVENTH FRAMEWORK PROGRAMME

Contents

Weighting of Pre-editing Rules

1 Objectives of the Deliverable

The objective of this deliverable is to provide methods to adapt the application of pre-editing rules based on feedback from users of these rules. More specifically, the aim is to establish tools that help to continuously monitor and analyse how ACCEPT users work with the pre-editing rules in practice, to identify potential problems with specific rules, and to adapt the rules accordingly.

The developed tools and methods fall into four categories:

- Collecting flag feedback on the Acrolinx server,
- Collecting usage information on the ACCEPT server,
- Analysing the feedback,
- Adapting rules according to user feedback.

The deliverable is related to *Task 9.6: Evaluate how users interact with pre- and post-editing rules.* In the framework of this task, a separate experiment was carried out with volunteers from the French Norton community forum, in which the users were asked to apply pre-editing rules to a pre-defined corpus [4]. The findings of this experiment will be summarized in Deliverable *D 9.2.4: Survey of evaluation results.* In the present deliverable, however, the objective is to analyse the behaviour of users while they edit their own content. We describe methods and tools for logging and evaluating user interaction with rules which can be used at any time, as opposed to providing one-time results. The following sections of the deliverable describe these methods and tools in detail.

2 Collecting Flag Feedback on the Acrolinx Server

The Acrolinx server already provides functionality for collecting feedback from the interaction of users with rule flags, and for logging this information in its reporting database [4]. We have enabled this functionality on the Acrolinx servers hosted for the ACCEPT project.

To submit feedback of this kind, a user needs to use feedback-enabled Acrolinx plug-ins and clients, more specifically, the plug-in for Microsoft Word, and the Batch Checker client. The actions of a user in the context menu of a flag are interpreted as implicit feedback according to Table 1.

Action in context menu	Effect	Interpreted as feedback
Select a suggestion	The flagged text is replaced with the suggestion	The flag and the suggestion are accepted
“Edit flag”	The flag is removed, the cursor placed on text for user to edit	The flag is accepted, but either there are no suggestion or none of them are accepted
“Ignore flag”	The flag is removed	The flag was rejected
“Ignore all flags”	The flag and all similar flags are removed	The flag on this kind of text pattern was generally rejected
Select rule name	Rule help is displayed	The rule name and suggestions were not considered self-descriptive
Do nothing	Menu stays open	A long time spent in the menu may indicate a difficult decision
“Next/Previous flag” or close menu	Flag stays in document	The flag is passively ignored by the user

Table 1: User actions in the Acrolinx plug-in and collected feedback

With the feedback functionality enabled, the plug-in sends the corresponding feedback for each user action to the Acrolinx server. Furthermore, the menu also provides explicit feedback options enabled in the context menu, with which the user can submit a comment on a flag, and suggests a new replacement option (suggestion). The screenshot in **Error! Reference source not found.** shows a eedback-enabled context menu.

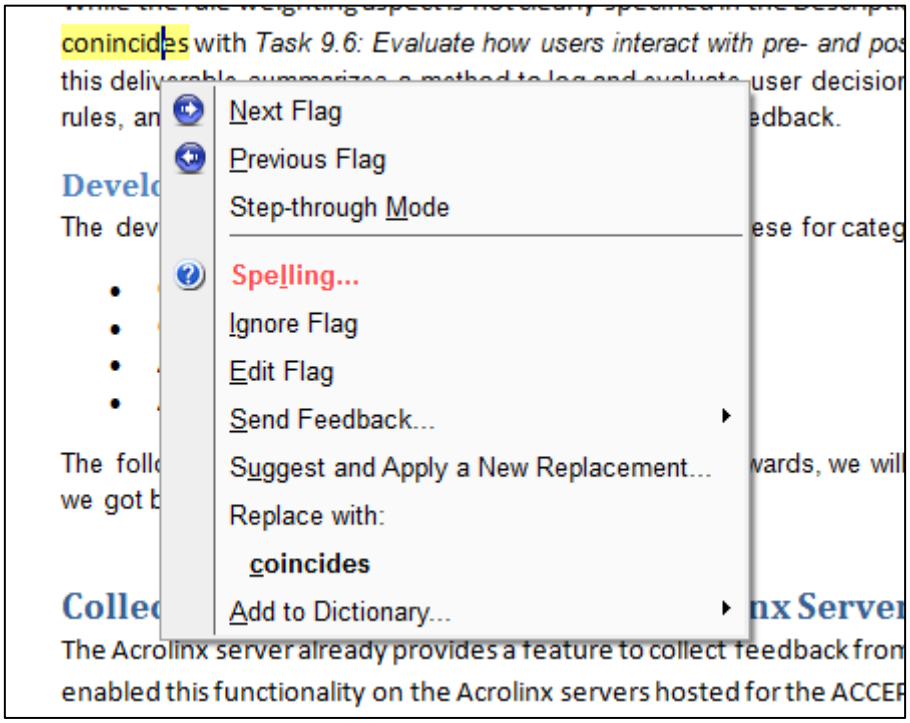


Figure 1: Feedback-enabled context menu in the Acrolinx plug-in for MS Word

The Acrolinx server stores any feedback received along with the details of the flag (such as the name of the rule, the marked text, and the sentence), and with information on the Acrolinx check session (such as the user ID, the rule set used, and the name of the document).

3 Collecting Usage Feedback on the ACCEPT Server

When the user works with the ACCEPT pre-editing plug-in, the plug-in communicates with the ACCEPT Server, which in turn handles the communication with an Acrolinx server. Details of this architecture can be found in *Deliverable 5.1: Browser-based client prototype used to access Acrolinx IQ server*. Although the ACCEPT server does not send any feedback information to the Acrolinx server, it collects extensive usage information itself, and makes it available via an API. Details of this API are documented in detail within the "Learn" section of the ACCEPT portal and in deliverable D 5.1.

The plug-in collects usage information both for parent sessions and for child sessions. A parent session in the sense of the ACCEPT API runs from opening to closing the ACCEPT plug-in window. A child session runs from one Acrolinx check to the next, or to the end of the parent session.

For each parent session, the usage data contains information such as the anonymized user ID, the time of the beginning and end of the check, and the used Acrolinx rule set and language.

For each child session, the usage information contains the following:

- how the text looked like before and after the session,
- which Acrolinx flags were found (position, flag type, rule name, suggestions),
- how the user interacted with the flag.

We interpret the various possible user actions as implicit feedback on the flags as shown in Table 2.

user action	**Effect**	**Interpreted as feedback**
Select a suggestion	The flagged text is replaced with the suggestion	The flag and the suggestion are accepted
"Ignore rule"	All the flags for this rule are removed now and in the future	The entire rule is rejected
"Learn Word"	Spelling flags on this word are removed now and in the future	Spelling flags on this word are rejected
Hover over flag for some time	A tooltip is shown	The flag is not sufficiently self-descriptive, and the user requires some help
Do nothing, change flagged text manually[1]	The flagged text is changed	The user accepts the flag, but none of the suggestions are considered suitable, or there are none
Do nothing, don't change flagged text[2]	The flagged text is unchanged	The user passively ignores this occurrence of the flag

Table 2: User actions in the ACCEPT plug-in and collected feedback

[1]Note: Whether the user changed the flagged text is detected heuristically by examining how the text looked like before and after the session.
[2]See note above.

4 Analysing the Feedback

For the flag feedback collected by the Acrolinx server in its database, we previously had developed a feedback processing tool pipeline, which is summarised below. For the usage information recorded by the ACCEPT pre-editing plug-in, no such aggregator exists. To simplify the task, we decided to automatically convert the usage information from the ACCEPT API into the format that the Acrolinx server uses for logging the flag feedback, such that it can be fed into the same feedback processing pipeline.

4.1 Converting ACCEPT Usage Information into Acrolinx Flag Feedback

We wrote a Python script that converts the usage information into flag feedback. The conversion is not exact, since some user actions can be performed in the ACCEPT plug-in but not in the Acrolinx plug-in, and vice versa. Most notably, a user cannot ignore an entire rule in the Acrolinx plug-in for all following sessions, but only ignore all similar flags for the current check session. Nevertheless, we treated the "ignore rule" action into an "ignore all flags in document" action. A human reviewer of the final feedback aggregation results should therefore keep in mind that the "ignore all flags in document" action actually means something stronger when it came from the ACCEPT plug-in originally, namely "ignore all flags of this rule now and in the future".

Conversely, displaying a tooltip in the ACCEPT plug-in is treated as clicking on "Help" in the Acrolinx plug-in, although it is a more lightweight form of obtaining help. One should therefore not give it the same importance as when reviewing feedback given via the Acrolinx plug-in.

Finally, the ACCEPT plug-in does not record some information that the Acrolinx plug-ins do. Most importantly, the time spent in the context menu is not recorded, so that any information in the aggregation result based on the required time should be disregarded.

4.2 Flag Feedback Pipeline

To analyse the flag feedback, we relied on a previously developed tool pipeline [1]. This pipeline was developed in the KNIME data mining tool [2]. The first part of the pipeline loads and cleans the flag feedback data, and transforms it into a form suitable for the main analysis.

This main analysis consists of two aspects:

- analysing the feedback by calculating flag metrics, and summarizing and visualizing the results for manual inspection
- applying machine learning algorithm to automatically estimate the precision of flags and suggestions based on various features.

4.3 Flag Metrics

The main goal of the feedback processing tool is to calculate the following metrics for each Acrolinx rule:

- **Precision of rule flags and their suggestions**
  This metric is calculated from the actions "select suggestion", "edit flag", "ignore flag", and "ignore all flags". Note that we cannot calculate the recall, since the user cannot report false negatives by telling where a missing flag should go.

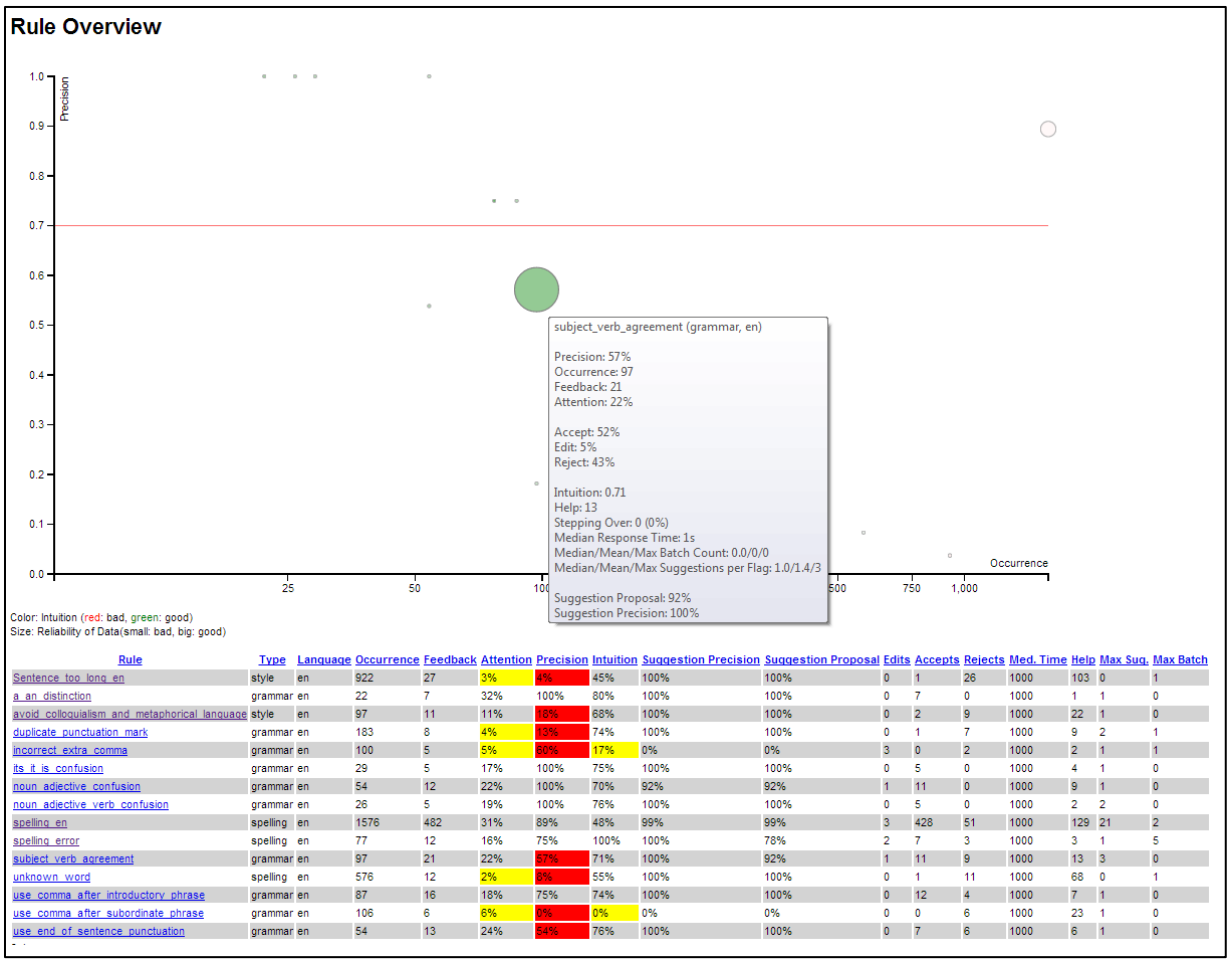
- **Attention for rule flags**

  This metric takes into account how many flags have been finally resolved, either by changing the text in some way or by keeping it on purpose.

- **Intuition of rule flags**

  This metric reflects how easy it was for the user to act on a rule by taking into account the time spent on a flag, whether rule help was requested, whether the user selected a batch-edit action, and various other features.

These metrics, along with various other statistics, are exported into an HTML-based format that can be displayed in a web browser. There is an overview page (see Figure 2) that shows the aggregated results of all rules and highlights potential problems. For each rule, there is a details page that gives in-depth information on the collected feedback, and also lists all raw feedback data points (see Figure 3). A rule developer can thus easily find potential performance issues for each rule, verify the validity of the feedback, and get insights as to how best to change the rule.



Rule	Type	Language	Occurrence	Feedback	Attention	Precision	Intuition	Suggestion Precision	Suggestion Proposal	Edits	Accepts	Rejects	Med. Time	Help	Max Sug.	Max Batch
Sentence_too_long_en	style	en	922	27	3%	4%	45%	100%	100%	0	1	26	1000	103	0	1
a_an_distinction	grammar	en	22	7	32%	100%	80%	100%	100%	0	7	0	1000	1	1	0
avoid_colloquialism_and_metaphorical_language	style	en	97	11	11%	18%	68%	100%	100%	0	2	9	1000	22	1	0
duplicate_punctuation_mark	grammar	en	183	8	4%	13%	74%	100%	100%	0	1	7	1000	9	2	1
incorrect_extra_comma	grammar	en	100	5	5%	80%	17%	0%	0%	3	0	2	1000	2	1	1
its_it_is_confusion	grammar	en	29	5	17%	100%	75%	100%	100%	0	5	0	1000	4	1	0
noun_adjective_confusion	grammar	en	54	12	22%	100%	70%	92%	92%	1	11	0	1000	9	1	0
noun_adjective_verb_confusion	grammar	en	26	5	19%	100%	76%	100%	100%	0	5	0	1000	2	2	0
spelling_en	spelling	en	1576	482	31%	89%	48%	99%	99%	3	428	51	1000	129	21	2
spelling_error	spelling	en	77	12	16%	75%	100%	100%	78%	2	7	3	1000	3	1	5
subject_verb_agreement	grammar	en	97	21	22%	57%	71%	100%	92%	1	11	9	1000	13	3	0
unknown_word	spelling	en	576	12	2%	0%	55%	100%	100%	0	1	11	1000	68	0	1
use_comma_after_introductory_phrase	grammar	en	87	16	18%	75%	74%	100%	100%	0	12	4	1000	7	1	0
use_comma_after_subordinate_phrase	grammar	en	106	6	6%	0%	0%	0%	0%	0	0	6	1000	23	1	0
use_end_of_sentence_punctuation	grammar	en	54	13	24%	54%	76%	100%	100%	0	7	6	1000	6	1	0

Figure 2: Exported rule metrics (overview page)

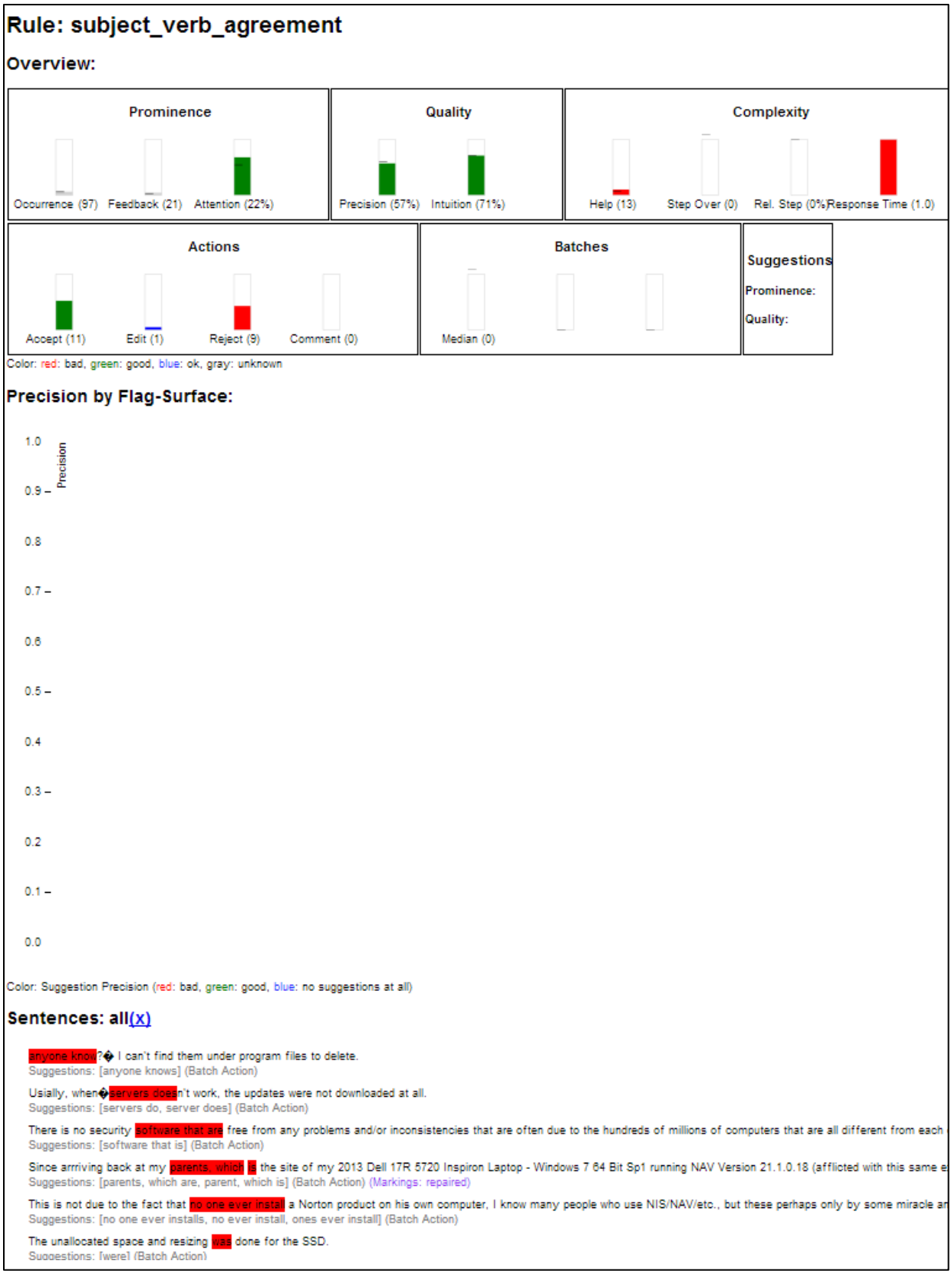


Figure 3: Exported rule metrics (rule details page)

4.4 Machine-Learning Correct Flags

While the exported feedback aggregation result is a helpful tool for manual evaluation, a rule developer still needs to determine when a rule produces incorrect flags. Another goal of the tool is to automate this work, too.

For this purpose, the tool creates a decision tree classifier to predict the user's precision feedback, that is, whether a flag is considered correct or not. The classifier is trained on a number of extracted features, including but not limited to:

- the marked text, as well as the word and character before and after it,
- the length of the marked text (number of words and characters),
- the length of the sentence it occurs in, and of the previous and following sentence,
- whether the marked text looks like a URL, a path or a formula.

The decision tree classifier shows the most decisive features for a flag to be considered correct or not. For example, such a classifier may report that flags of the "add missing space" rule are almost always wrong if the flag is immediately followed by a period, or similar.

5 Adapting the Application of Rules

Taking into account the feedback collected using the tools described above, the pre-editing rule sets can be adjusted such that wrong flags are eliminated, more suggestions are presented that correspond to the user edits, and the overall relevance of flags and suggestions increases.

There are two main ways to adapt how pre-editing rules are applied. First, users can influence the rule application themselves to a certain extent by choosing to ignore a rule, or by choosing to learn a word. Ignored rules are not shown to the user subsequently, and learned words are not marked as spelling issues. This provides a quick and easy solution for the user to adapt the editing assistance to their preferences.

The other way is for a rule developer to manually make changes to the rules, which then typically apply to all members of a community. The metrics and the decision tree calculated by the tool presented in the previous section can provide valuable insights into the performance of rules, and helps rule developers to make informed decisions on how to change existing rules, and where to focus.

Changing a rule based on feedback always bears the risk of breaking it for other situations or linguistic patterns that "used to work" according to other (positive) feedback given by users previously. For this reason, the pipeline presented in the previous section can also export the feedback data as regression tests. For example, for a flag considered "good" by users, these tests expect that the flag continues to occur in the given sentence in the future; for a flag rejected by users, the test expects the flag to not appear in the given sentence.

These tests are executed by an Acrolinx rule testing framework, which records how many of these tests succeed or fail. The testing framework in turn can be integrated into a Continuous Integration platform such as Jenkins [3], which executes the tests automatically whenever changes to the rules are made. This mechanism thus measures over time to what extent rule adaptation actually addresses the entire given user feedback, and thus ensure the quality of the rules.

6 Conclusion

We have achieved the objective of developing a method for continuously analysing the user behaviour, to collect implicit feedback from the community, and to adapt the rules accordingly. While the feedback aggregation pipeline had been developed outside of ACCEPT, we have developed a connecting script that automatically converts the usage data from the portal into a format suitable for the feedback aggregation pipeline, thus leveraging the helpful aggregation method for the ACCEPT project. As the method is not dependent on specific communities, languages or Acrolinx rules, it can be used to analyse the impact of the rules in any community where the ACCEPT plug-in is integrated.

We have run the tool on the collected usage data from the English Norton community for the period January to May 2014. Moreover, we have used it to aggregate the data from pre-editing experiments carried out by TSF translators with the Microsoft Word plug-in. Figure 2 shows the results for the mentioned Norton community usage data. While the number of actual feedback points is not always representative yet for each rule, it already gives us some valuable insights. For example, the English "sentence too long" rule is frequently ignored by users, while the spelling flags are usually correct and provide good suggestions. Final details on rule adoption will be presented in Deliverable *D 9.2.4: Survey of evaluation results.*

References

[1] Ralf Kühnlein:
Nutzerverhalten-basierte Optimierung einer linguistischen KI
Diploma Thesis, Freie Universität, Berlin, 2013.

[2] KNIME, the Konstanz Information Miner. Retrieved 13/06/2014, from http://www.knime.org/.

[3] Jenkins Continuous Integration. Retrieved 13/06/2014, from http://jenkins-ci.org/.

[4] Pierrette Bouillon, Liliana Gaspar, Johanna Gerlach, Victoria Porro and Johann Roturier:
Pre-editing by Forum Users: a Case Study
In *Workshop on Controlled Natural Language simplifying language use (CNL)*, Reykjavik, Ireland, 2014.