Helping Our Own: The HOO 2011 Pilot Shared Task

Robert Dale
Centre for Language Technology
Macquarie University
Sydney, Australia
Robert.Dale@mq.edu.au

Adam Kilgarriff
Lexical Computing Ltd
Brighton
United Kingdom
adam@lexmasterclass.com

Abstract

The aim of the Helping Our Own (HOO) Shared Task is to promote the development of automated tools and techniques that can assist authors in the writing task, with a specific focus on writing within the natural language processing community. This paper reports on the results of a pilot run of the shared task, in which six teams participated. We describe the nature of the task and the data used, report on the results achieved, and discuss some of the things we learned that will guide future versions of the task.

1 Introduction

The Helping Our Own (HOO) Shared Task aims to promote the development of automated tools and techniques that can assist authors in the writing task. The task focusses specifically on writing within the natural language processing community, on the grounds that content matter familiar to Shared Task participants will be more engaging than content matter from another discipline. In addition, the ACL Anthology (Bird et al., 2008) provides us with a large and freely-available collection of material in the appropriate domain and genre that can be used, for example, for language modelling; obtaining similar material in other disciplines is more difficult and potentially costly. A broader discussion of the background to the HOO task can be found in (Dale and Kilgarriff, 2010).

In this first pilot round of the task, we focussed on errors and infelicities introduced into text by non-native speakers (NNSs) of English. While there are few native speakers who would not also have something to gain from the kinds of technologies we would like to see developed, the generally higher density of errors in texts authored by NNSs makes annotation of this material much more cost efficient than the annotation of native-speaker text. The focus on English texts is for purely pragmatic reasons; obviously one could in principle pursue the goals discussed here for other languages too.

This paper is structured as follows. In Section 2 we describe the development and test data that was provided to participants. Then, in Section 3 we describe the approach taken to evaluation. In Section 4, we summarise the results of the submissions from each of the six participating teams. Finally, in Section 5, we make some observations on lessons learned and comment on plans for the future.

2 The Data

2.1 Texts and Corrections

The data used in the pilot run of the task consisted of a set of *fragments* of text, averaging 940 words in length. These fragments were extracts from a collection of 19 *source documents*, each being a paper that had previously been published in the proceedings of a conference or a workshop of the Association for Computational Linguistics; the authors of these papers have kindly permitted their material to be used in the Shared Task. From each source document we extracted one fragment for development and one fragment for testing; each fragment is uniquely identifiable by a four-digit number used in all data associated with that fragment.

Each fragment was annotated with a number of *edits* to correct errors and infelicities, as discussed further below. Each fragment in the development set was annotated by two professional copy-editors, and each fragment in the test set was annotated by one copy-editor and checked by one of the organizers. Collectively, the development data contained a total of 1264 edits, or an average of 67 per file, with a minimum of 16 and a maximum of 100; and the test data contained a total of 1057 edits, an average of 56 per file with a minimum of 18 and a maximum of 107. In both data sets this works out at an average of one edit every 15 words.

Corresponding to each fragment, there is also a file containing, in stand-off markup format, the set of target edits for that file. Figure 1 shows some example *gold-standard edits*. The output of participating systems is compared against these files, whose contents we refer to as *edit structures*.

```
<edit type="MY" index="0001-0004"
      start="631" end="631">
   <original><empty/></original>
   <corrections>
      <correction/>
      <correction>both </correction>
   </corrections>
</edit>
<edit type="RV" index="0001-0005"
      start="713" end="718">
   <original>carry</original>
   <corrections>
      <correction/>
      <correction>contain</correction>
   </corrections>
</edit>
<edit type="IJ" index="0001-0006"
      start="771" end="782">
   <original>electronics</original>
   <corrections>
      <correction>electronic</correction>
   </corrections>
</edit>
<edit type="RP" index="0001-0007"
      start="1387" end="1388">
   <original>;</original>
   <corrections>
      <correction>.</correction>
   </corrections>
</edit>
```

Figure 1: Some gold-standard edit structures.

Participating systems could choose to deliver their results in either one of two forms:

1. A set of plain text files that contain corrected text *in situ*; we provided a tool that extracts the changes made to produce a set of XML edit structures for evaluation.

2. A set of edit structures that encode the corrections their system makes.

There were advantages to providing the latter: in particular, edit structures provide a higher degree of fidelity in capturing the specific changes made, as discussed further below.

2.2 The Annotation of Corrections

By an *edit* we mean any change that is made to a text: from the outset, our intent has been to deal with textual modifications that go some way beyond the correction of, for example, grammatical errors. This decision presents us with a significant challenge. Whereas the presence of spelling and grammatical errors might seem to be something that competent speakers of a language would agree on, as soon as we go beyond such phenomena to encompass what we will sometimes refer to as 'stylistic infelicities', there is increasing scope for disagreement. Our initially-proposed diagnostic was that the annotators should edit anything they felt corresponded to 'incorrect usage'. A brief perusal of the data will reveal that, not surprisingly, this is a very difficult notion to pin down precisely.

2.3 Annotation Format

The general format of edits in the gold-standard edit files is as shown in Figure 1. Each `<edit>` element has an `index` attribute that uniquely identifies the edit; a `type` attribute that indicates the type of the error found or correction made;[1] a pair of offsets that specify the character positions in the source text file of the `start` and `end` of the character sequence that is affected by the edit; an embedded `<original>` element, which contains the text span that is subject to correction; and an embedded `<corrections>` element, which lists one or more possible corrections for the problematic text span that has been identified.

There are a number of complicating circumstances we have to deal with:

1. There may be multiple valid corrections. This is not just a consequence of our desire to include classes of infelicitious usage where there is no single best correction. The requirement is already present in any attempt to handle grammatical number agreement issues, for example, where an instance of number disagreement might be repaired by making the affected items either singular or plural. Also, it is usually not possible to consider the list of corrections we provide as being exhaustive.

2. A correction may be considered *optional*. In such cases we view the first listed correction as a null correction (in other words, one of the multiple possible corrections is to leave things as they are). When an edit contains an optional correction, we call the edit an *optional edit*. if the edit contains no optional corrections, then it is a *mandatory edit*. Note that deletions and insertions, as well as replacements, may be optional.

3. Sometimes edits may be interdependent: making one change requires that another also be made. Edits which are connected together in this way are indicated via indexed `cset` attributes (for *consistency set*). The most obvious case of this is where there is requirement for consistency in the use of some form (for example, the hyphenation of a term) across a

[1] The set of types is borrowed, with some very minor changes, from the Cambridge University Press Error Coding System described in (Nicholls, 2003), and used with permission of Cambridge University Press.

document; each such instance will then belong to the same `cset` (and consequently there can be many members in a `cset`). Another situation that can be handled using `csets` is that of grammatical number agreement. In such a case, there are two possible corrections, but the items affected may be separated in the text, requiring two separate edits to be made, connected in the annotations by a `cset`.

4. There are cases where our annotators have determined that something is wrong, but are not able to determine what the correction should be. There are two common circumstances where this occurs:
   (a) A word or fragment of text is missing, but it is not clear what the missing text should be.
   (b) A fragment of text contains a complex error, but it is not obvious how to repair the error.

   These two cases are represented by omitting the `corrections` element.

All of these phenomena complicate the process of evaluation, which we turn to next.

3 Evaluation

Each team was allowed to submit up to 10 distinct 'runs', so that they could provide alternative outputs. Evaluation then proceeds by comparing the set of gold-standard edit structures for a fragment with the set of edit structures corresponding to the participating team's output for a single run for that fragment.

3.1 Scoring

There are a number of aspects of system performance for which we can derive scores:

- Detection: does the system determine that an edit is required at some point in the text?
- Recognition: does the system correctly determine the extent of the source text that requires editing?
- Correction: does the system offer a correction that is amongst the corrections provided in the gold standard?

Detection is effectively 'lenient recognition', allowing for the possibility that the system and the gold standard may not agree on the precise extent of a correction. Systems can be scored on a fragment-by-fragment basis, on a data set as a whole, or on individual error types across the data set as a whole.

For each pairing of gold standard data and system output associated with a given fragment, we compute two *alignment sets*: these are structures that indicate the correspondences between the edits in the two edit sets. The *strict alignment set* contains those alignments whose extents match perfectly; the *lenient alignment set* contains those alignments that involve some overlap. We also have what we call *unaligned edits*: these are edits which do not appear in the lenient alignment set. An unaligned system edit corresponds to a *spurious* edit; an unaligned gold-standard edit corresponds to a *missing* edit. It is important to note that missing edits are of two types, depending on whether the gold-standard edit corresponds to an optional edit or a mandatory edit. A system should not be penalised for failing to provide a correction for a markable where the gold standard considers the edit to be optional. To manage the impact of this on scoring, we need to keep track of the number of *missing optional edits*.

3.1.1 Detection

For a given $\langle G, S \rangle$ pair of edit sets, a gold standard edit g_i is considered *detected* if there is at least one alignment in the lenient alignment set that contains g_i. Under conventional circumstances we would calculate Precision as the proportion of edits found by the system that were correct:[2]

$$P = \frac{\#\ \textit{detected edits}}{\#\ \textit{spurious edits} + \#\ \textit{detected edits}} \quad (1)$$

Similarly, Recall would be conventionally calculated as:

$$R = \frac{\#\ \textit{detected edits}}{\#\ \textit{gold edits}} \quad (2)$$

However, under this regime, if all the gold edits are optional and none are detected by the system, then the system's Precison and Recall will both be zero. This is arguably unfair, since doing nothing in the face of an optional edit is perfectly acceptable; so, to accommodate this, we also compute scores 'with bonus', where a system also receives reward for optional edits where it does nothing:

$$P = \frac{\#\ \textit{detected} + \#\ \textit{missing optional}}{\#\ \textit{spurious} + \#\ \textit{detected} + \#\ \textit{missing optional}} \quad (3)$$

$$R = \frac{\#\ \textit{detected} + \#\ \textit{missing optional}}{\#\ \textit{gold edits}} \quad (4)$$

This has a more obvious impact when we score on a fragment-by-fragment basis, since the chances of a system proposing no edits for a single fragment are greater than the chances of the system proposing no edits for all fragments.

The detection score for a given $\langle G, S \rangle$ pair is then the harmonic mean (F-score):

$$\textit{DetectionScore} = 2 \times \frac{\textit{Precision} \times \textit{Recall}}{\textit{Precision} + \textit{Recall}} \quad (5)$$

[2]Note that in all computations of Precision (P) and Recall (R) we take the result of dividing zero by zero to equal 1, but for the computation of F-scores we take the result of dividing zero by zero to be zero.

3.1.2 Recognition

The detection score described above can be considered a form of 'lenient' recognition. We also want to measure 'strict' recognition, i.e. the degree to which a participating system is able to determine the correct start and end locations of text to be corrected. We consider a gold-standard edit g_j to be *recognized* if it appears in the strict alignment set. *RecognitionScore* is defined to be 0 if there are no recognized edits for a given document; otherwise, we have:[3]

$$P = \frac{\#\ recognized\ edits}{\#\ system\ edits} \tag{6}$$

$$R = \frac{\#\ recognized\ edits}{\#\ gold\ edits} \tag{7}$$

The recognition score for a given $\langle G, S \rangle$ pair is again the harmonic mean.

Note that there is a deficiency in the scoring scheme here: it is quite possible that the system has decomposed what the gold-standard sees as a single edit into two constituent edits, or vice versa. Both analyses may be plausible; however, the scoring scheme gives no recognition credit in such cases.

3.1.3 Correction

Recall that for any given gold-standard edit g_j, there may be multiple possible corrections. A system edit s_i is considered a *valid correction* if it is strictly aligned, and the correction string that it contains is identical to one of the corrections provided in the gold standard edit. *CorrectionScore* is defined to be 0 if there are no recognized edits for a given document; otherwise, we have:[4]

$$P = \frac{\#\ valid\ corrections}{\#\ system\ edits} \tag{8}$$

$$R = \frac{\#\ valid\ corrections}{\#\ gold\ edits} \tag{9}$$

The correction score for a given $\langle G, S \rangle$ pair is, as before, the harmonic mean.

Just as in the case of recognition, correction scoring also suffers from the deficiency that if adjacent errors are composed or aggregated differently by the system than they are in the gold standard, no credit is assigned.

3.2 The Participating Teams

Submissions were received from six teams, as listed in Table 1. Some teams submitted only one run, while others submitted 10 (and in one case, nine); some teams submitted corrected texts, while others provided standoff XML edits.

[3] Again, we also compute a 'with bonus' variant of this that gives credit for missed optional edits.

[4] Once more, we also compute a 'with bonus' variant.

4 Results

In this section, we provide some comparative results across all six teams. Each team has also provided a separate report that provides more detail on their methods and results, also published in the present volume.

4.1 Total Scores

As a way of assessing the performance of a participating system overall, we compute each team's scores across the complete set of fragments for each run. Tables 2, 3 and 4 present the best scores achieved by each system under the 'no bonus' condition; and Tables 5, 6 and 7 present the best scores achieved by each system under the 'bonus' condition, where credit is given for missed optional edits. In each case, we show the results for the system run that produced the best F-score for that system; the overall best F-score is shown in bold.

4.2 Type-Based Scores

The numbers provided above, although they provide a means of characterising the overall performance of the participating systems, do not take account of the fact that some teams chose to attack specific types of error while ignoring other types of errors. Table 8 shows the number of edits of each type in the test data. Note that these are not the raw types from the CLC tagset that are used in the annotations, but are aggregations of these based on the part-of-speech of the affected words in the text; thus, for example, the Article type includes the CLC error tags FD (Form of determiner), RD (Replace determiner), MD (Missing determiner), UD (Unnecessary determiner), DD (Derivation of determiner), AGD (Determiner agreement error), CD (Countability of determiner), and DI (Inflection of determiner). 'Compound Change' corresponds to the tag CC, which is a new tag we added to the tagset to handle cases where there were multiple issues with a span of text that could not be easily separated; and 'Other' incorporates CL (collocation or tautology error), L (inappropriate register), X (incorrect negative formation), CE (complex error), ID (idiom wrong), AS (argument structure error), W (word order error), AG (agreement error), M (missing error), R (replace error), and U (unnecessary error).

The particular approaches each team took are discussed in the individual team reports; Tables 9 through 21 show the comparative performance by all teams for each of the error categories in Table 8. In each case, the we show each team's best results, indicating the run which provided them; and the best overall score for each error category is shown in bold. Note that the numbers shown here are the percentages of instances in each category that were detected, recognized and corrected; since we did not require teams to assign types to the edits they proposed, it is only possible to compute Recall, and not

Team	Country	ID	Submission Format	Number of Runs
Natural Language Processing Lab, Jadavpur University	India	JU	Text	1
LIMSI	France	LI	Text	10
National University of Singapore	Singapore	NU	Edits	1
Universität Darmstadt	Germany	UD	Edits	9
Cognitive Computation Group, University of Illinois	USA	UI	Text	10
Universität Tübingen	Germany	UT	Text	10

Table 1: Participating Teams

Team	Run	Precision	Recall	F-Score
JU	0	0.178	0.064	0.094
LI	8	0.409	0.063	0.110
NU	0	0.447	0.111	0.177
UD	5	0.050	0.137	0.073
UI	6	0.529	0.187	**0.277**
UT	2	0.134	0.119	0.126

Table 2: Best run scores for Detection, 'No Bonus' condition

Team	Run	Precision	Recall	F-Score
JU	0	0.125	0.045	0.067
LI	8	0.307	0.047	0.082
NU	0	0.399	0.101	0.162
UD	5	0.028	0.077	0.041
UI	1	0.583	0.153	**0.243**
UT	8	0.088	0.076	0.081

Table 3: Best run scores for Recognition, 'No Bonus' condition

Team	Run	Precision	Recall	F-Score
JU	0	0.104	0.038	0.055
LI	8	0.209	0.032	0.056
NU	0	0.291	0.074	0.118
UD	8	0.050	0.020	0.028
UI	1	0.507	0.133	**0.211**
UT	1	0.050	0.041	0.045

Table 4: Best run scores for Correction, 'No Bonus' condition

Team	Run	Precision	Recall	F-Score
JU	0	0.331	0.148	0.204
LI	8	0.606	0.141	0.229
NU	0	0.578	0.188	0.284
UD	3	0.388	0.113	0.174
UI	1	0.736	0.243	**0.366**
UT	2	0.200	0.193	0.197

Table 5: Best run scores for Detection, 'Bonus' condition

Team	Run	Precision	Recall	F-Score
JU	0	0.288	0.129	0.178
LI	8	0.539	0.125	0.203
NU	0	0.540	0.179	0.269
UD	6	0.913	0.090	0.164
UI	8	0.713	0.220	**0.337**
UT	5	0.334	0.104	0.159

Table 6: Best run scores for Recognition 'Bonus' condition

Team	Run	Precision	Recall	F-Score
JU	0	0.271	0.121	0.167
LI	8	0.473	0.110	0.178
NU	0	0.457	0.151	0.227
UD	6	0.894	0.088	0.160
UI	8	0.648	0.201	**0.306**
UT	7	0.898	0.083	0.152

Table 7: Best run scores for Correction, 'Bonus' condition

Type	Count
Article	260
Punctuation	206
Preposition	121
Noun	113
Verb	108
Compound Change	66
Adjective	34
Adverb	28
Conjunction	20
Anaphor	14
Spelling	9
Quantifier	7
Other	80

Table 8: Edits by Type

possible to calculate Precision or F-score. In the separate team reports, however, some teams have carried out these calculations based on the error types their systems were targettting.

5 Conclusions and Outstanding Issues

The task we set participating teams was an immensely challenging one. Much work in automated writing assistance targets only very specific error types such as article or preposition misuse; it is rare for systems to have to contend with the variety and complexity of errors found in the texts we used here.

We were very pleased at the level of participation achieved in this pilot run of the task, and we intend to run subsequent shared tasks based on the experience of the present exercise. We have learned a great deal that will hopefully lead to significant improvements in subsequent runs:

1. We are aware of minor tweaks that can be made to our annotation format to make it more useful and flexible.

2. There are various regards in which our evaluation tools can be improved to avoid artefacts that arise from the current scheme (where, for example, systems can be penalised because they decompose one gold-standard edit into a sequence of edits, or aggregate a sequence of gold-standard edits into a single edit).

3. We intend to provide better support to allow teams to target specific types of errors; we are also considering revisions to the tagset used.

Overall, the biggest challenge we face is the cost of data annotation. Identifying errors and proposing corrections across such a wide range of error types is a very labour intensive process that is not easily automated, and is not amenable to being carried out by unskilled labour.

6 Acknowledgements

Diane Nicholls and Kate Wild diligently annotated the data for the pilot phase. George Narroway spent many hours processing data and developing the evaluation tools. Guy Lapalme provided useful input on XML, XSLT and CSS; various members of our Google Groups mailing list provided immensely useful input at various stages of the project. We are especially grateful to all six participating teams for their patience in accommodating various delays that arose from unforeseen problems in creating and evaluating the data and results.

References

Steven Bird, Robert Dale, Bonnie Dorr, Bryan Gibson, Mark Joseph, Min-Yen Kan, Dongwon Lee, Brett Powley, Dragomir Radev, and Yee Fan Tan. 2008. The acl anthology reference corpus: A reference dataset for bibliographic research in computational linguistics. In *Proceedings of the Language Resources and Evaluation Conference (LREC 2008)*.

R. Dale and A. Kilgarriff. 2010. Helping Our Own: Text massaging for computational linguistics as a new shared task. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 261–266, 7th-9th July 2010.

D. Nicholls. 2003. The Cambridge Learner Corpus—error coding and analysis for lexicography and ELT. In D. Archer, P. Rayson, A. Wilson, and T. McEnery, editors, *Proceedings of the Corpus Linguistics 2003 Conference*, pages 572–581, 29th March–2nd April 2001.

Team	Detection	Run	Recognition	Run	Correction	Run
JU	1.54	0	1.54	0	1.54	0
LI	3.46	1	3.46	1	2.31	1
NU	31.92	0	31.54	0	23.85	0
UD	1.92	5	0.77	1	0.00	0
UI	**41.54**	6	**39.62**	3	**35.38**	3
UT	8.46	0	3.85	0	3.08	1

Table 9: Best run scores for Article errors

Team	Detection	Run	Recognition	Run	Correction	Run
JU	14.08	0	11.65	0	9.71	0
LI	8.74	8	7.77	8	5.83	8
NU	0.00	0	0.00	0	0.00	0
UD	**16.99**	5	3.88	1	0.49	1
UI	15.53	4	**12.14**	4	**11.65**	0
UT	1.46	3	0.00	0	0.00	0

Table 10: Best run scores for Punctuation errors

Team	Detection	Run	Recognition	Run	Correction	Run
JU	4.13	0	2.48	0	2.48	0
LI	2.48	1	1.65	1	1.65	1
NU	15.70	0	15.70	0	9.92	0
UD	4.13	5	3.31	5	0.00	0
UI	32.23	1	32.23	3	23.97	3
UT	**60.33**	0	**52.89**	8	**28.10**	1

Table 11: Best run scores for Preposition errors

Team	Detection	Run	Recognition	Run	Correction	Run
JU	3.54	0	0.00	0	0.00	0
LI	6.19	7	5.31	7	2.65	7
NU	4.42	0	0.88	0	0.00	0
UD	**22.12**	1	**21.24**	1	**8.85**	1
UI	0.88	0	0.00	0	0.00	0
UT	0.00	0	0.00	0	0.00	0

Table 12: Best run scores for Noun errors

Team	Detection	Run	Recognition	Run	Correction	Run
JU	8.33	0	7.41	0	**7.41**	0
LI	1.85	1	0.93	0	0.00	0
NU	0.00	0	0.00	0	0.00	0
UD	**18.52**	5	**17.59**	5	2.78	8
UI	0.93	4	0.93	4	0.93	4
UT	3.70	2	0.00	0	0.00	0

Table 13: Best run scores for Verb errors

Team	Detection	Run	Recognition	Run	Correction	Run
JU	6.06	0	3.03	0	0.00	0
LI	15.15	7	1.52	6	0.00	0
NU	6.06	0	0.00	0	0.00	0
UD	**24.24**	5	**6.06**	1	**1.52**	1
UI	15.15	5	3.03	0	0.00	0
UT	18.18	3	0.00	0	0.00	0

Table 14: Best run scores for Compound Change errors

Team	Detection	Run	Recognition	Run	Correction	Run
JU	0.00	0	0.00	0	0.00	0
LI	14.71	6	14.71	6	5.88	6
NU	2.94	0	2.94	0	0.00	0
UD	**23.53**	5	**23.53**	5	**8.82**	3
UI	0.00	0	0.00	0	0.00	0
UT	5.88	0	5.88	0	0.00	0

Table 15: Best run scores for Adjective errors

Team	Detection	Run	Recognition	Run	Correction	Run
JU	7.14	0	0.00	0	0.00	0
LI	7.14	7	3.57	6	0.00	0
NU	3.57	0	0.00	0	0.00	0
UD	**28.57**	5	**14.29**	1	0.00	0
UI	0.00	0	0.00	0	0.00	0
UT	17.86	3	0.00	0	0.00	0

Table 16: Best run scores for Adverb errors

Team	Detection	Run	Recognition	Run	Correction	Run
JU	0.00	0	0.00	0	0.00	0
LI	5.00	0	5.00	0	5.00	0
NU	0.00	0	0.00	0	0.00	0
UD	0.00	0	0.00	0	0.00	0
UI	0.00	0	0.00	0	0.00	0
UT	**10.00**	0	**10.00**	0	**10.00**	0

Table 17: Best run scores for Conjunction errors

Team	Detection	Run	Recognition	Run	Correction	Run
JU	0.00	0	0.00	0	0.00	0
LI	0.00	0	0.00	0	0.00	0
NU	0.00	0	0.00	0	0.00	0
UD	**7.14**	2	**7.14**	2	0.00	0
UI	0.00	0	0.00	0	0.00	0
UT	0.00	0	0.00	0	0.00	0

Table 18: Best run scores for Anaphor errors

Team	Detection	Run	Recognition	Run	Correction	Run
JU	66.67	0	66.67	0	55.56	0
LI	**77.78**	6	**77.78**	6	**77.78**	6
NU	44.44	0	44.44	0	44.44	0
UD	55.56	2	55.56	2	44.44	3
UI	44.44	4	33.33	4	11.11	2
UT	0.00	0	0.00	0	0.00	0

Table 19: Best run scores for Spelling errors

Team	Detection	Run	Recognition	Run	Correction	Run
JU	0.00	0	0.00	0	0.00	0
LI	0.00	0	0.00	0	0.00	0
NU	0.00	0	0.00	0	0.00	0
UD	14.29	2	14.29	2	0.00	0
UI	0.00	0	0.00	0	0.00	0
UT	**57.14**	3	**57.14**	3	**14.29**	2

Table 20: Best run scores for Quantifier errors

Team	Detection	Run	Recognition	Run	Correction	Run
JU	7.04	0	1.41	0	0.00	0
LI	4.23	1	1.41	1	**1.41**	1
NU	0.00	0	0.00	0	0.00	0
UD	**22.54**	5	1.41	1	0.00	0
UI	4.23	3	0.00	0	0.00	0
UT	21.13	3	**4.23**	0	0.00	0

Table 21: Best run scores for Other errors