DUNEDIN NEW ZEALAND

Automated Scoring of Practical Tests in an Introductory Course in Information Technology

Geoffrey Kennedy

The Information Science Discussion Paper Series

Number 99/19
September 1999
ISSN 1172-6024

University of Otago

Department of Information Science

The Department of Information Science is one of six departments that make up the Division of Commerce at the University of Otago. The department offers courses of study leading to a major in Information Science within the BCom, BA and BSc degrees. In addition to undergraduate teaching, the department is also strongly involved in postgraduate research programmes leading to MCom, MA, MSc and PhD degrees. Research projects in spatial information processing, connectionist-based information systems, software engineering and software development, information engineering and database, software metrics, distributed information systems, multimedia information systems and information systems security are particularly well supported.

The views expressed in this paper are not necessarily those of the department as a whole. The accuracy of the information presented in this paper is the sole responsibility of the authors.

Correspondence

This paper represents work to date and may not necessarily form the basis for the authors' final conclusions relating to this topic. It is likely, however, that the paper will appear in some form in a journal or in conference proceedings in the near future. The authors would be pleased to receive correspondence in connection with any of the issues raised in this paper, or for subsequent publication details. Please write directly to the authors at the address provided below. (Details of final journal/conference publication venues for these papers are also provided on the Department's publications web pages: http://divcom.otago.ac.nz:800/COM/INFOSCI/Publctns/home.htm). Any other correspondence concerning the Series should be sent to the DPS Coordinator.

Department of Information Science
University of Otago
P O Box 56
Dunedin
NEW ZEALAND

Fax: +64 3 479 8311
email: dps@infoscience.otago.ac.nz
www: http://divcom.otago.ac.nz:800/COM/INFOSCI/

Proceedings of the IASTED International Conference
Computers and AdvancedTechnology in Education
May 6-8, 1999, Cherry Hill, New Jersey USA

Automated Scoring of PracticalTests in an Introductory Course in Information Technology

Geoffrey J. Kennedy[1]
Computer & Information Science
University of Otago, Dunedin, New Zealand
gkennedy@commerce.otago.ac.nz.

Abstract

In an introductory course in information technology at the University of Otago the acquisition of practical skills is considered to be a prime objective. An effective way of assessing the achievement of this objective is by means of a 'practical test', in which students are required to accomplish simple tasks in a controlled environment. The assessment of such work demands a high level of expertise, is very labour intensive and can suffer from marker inconsistency, particularly with large candidatures.

This paper describes the results of a trial in which the efforts of one thousand students in a practical test of word processing were scored by means of a program written in MediaTalk. Details of the procedure are given, including sampling strategies for the purpose of validation and examples of problems that were encountered.

It was concluded that the approach was useful, and once properly validated gave rise to considerable savings in the time and effort.

Key words: Computer-aided learning, automated scoring, computer education, test validation

1. Introduction: The Assessment Problem

The problem of assessing practical tests in computing courses with large candidatures faced by many tertiary institutions cries out for some automated approach. A few reports of automated assessment have been published, for example Summons *et al.* [1] who have used macros to assess work carried out with *Excel* spreadsheets. This work describes an attempt to automate the task of examining word processor documents and allocating marks for the presence of specified features.

The course entitled *Introduction to Information Technology* at the University of Otago, New Zealand, in the first semester (March - June) 1998 attracted an enrolment of just over one thousand. The stated objectives of the course are that, by the end of the course, students will:

(1) have acquired an understanding of the general concepts of computing and be familiar with both the *Macintosh* and *PC-compatible* platforms,

(2) be able to demonstrate useful skills with a variety of software packages, and

(3) display appropriate attitudes towards the use of computers in a range of applications and environments.

The achievement of objective (1) is assessed by means a 120 item multiple choice test instrument, but the assessment of attainment of computing skills and attitudes is best carried out by means of a practical tests carried out in a controlled laboratory environment. Three such tests are administered throughout the semester, two of them on the PC-compatible platform, one on Macintosh.

Tasks for the first PC practical test involve the manipulation and formatting of documents using *Microsoft Word*[2]. A controlled environment is provided by a laboratory equipped with eighty PC-compatible machines supported by Novell networking software.

Students register under special laboratory 'logins', which permits authentication of students as being enrolled in the course and also identifies the particular machine being used. For the duration of the test students are granted access rights to a specific area of the file server associated with their unique student identification number. Before and after the test they are denied any access to the work area. During the practical test, machines in the laboratory are under the control of a specially tailored network policy and access is limited to only that software permitted for the test, thereby reducing risk of cheating or collusion to a minimum.

Prior to 1998 tutors were required to access the work 'by hand', opening each test document in turn and awarding marks, which were then recorded in a database. Even with special software to automate the location and opening of the test documents and recording of the marks awarded,

1. The author acknowledges the assistance of GordonYau who wrote the marking program and of Andrew Marr, Chris Henry and James Irwin, who carried out the manual marking and collated the results for the validation exercise

2. Currently th *Microsoft Office* 1997 suite is the software provided by the laboratory network

the task of assessing this work was long and arduous. The more tutors employed, the greater became the problems of quality control and consistency. If only full-time staff were employed the task becomes overwhelming, and in any case takes too long, bearing in mind how anxious students are to receive their results.

Facing the prospect of assessing one thousand students in 1998 and the inevitable subsequent increase in student numbers in the future it became imperative that the automation of the task be attempted.

2. Proposed Solution

When scored manually, each document has to be opened and scanned visually by the marker looking for expected features. The marker can award partial marks for partially correct work. Sometimes even when a feature is present it may not have been achieved in the desired manner, for example, applying bolding to text rather than an appropriate paragraph style, so this must be determined. Sometimes minor errors can be ignored by the marker if it is clear that the student has grasped the principle being examined.

Table 1 details the features in the document that were being assessed and the marks to be awarded in each case.

Feature	Description	Mark
CD text	Text correctly copied from online help	4
File/Spell	All spelling errors corrected	3
Template	Correct template attached	2
Styles	Correct styles applied as required	4.25
MovePara	Paragraph correctly relocated	2
Picture	Graphic correctly inserted in document	1
Quotation	New “Quotation” style created and applied correctly	4
Tabs	Tab settings defined and used correctly	4
Table	Table defined and used correctly	2
TOC	Table of contents created	1
PageBreak	Hard page break inserted	1
Total		28.25

Table 1 *Features of document to be assessed*

In order to automate the process, the first step is to reduce each document to some format which includes all relevant formatting information but can be processed by program. This can be achieved by requiring students to save their document in *rich text format* (rtf), a facility conveniently provided by *Word*. The next step is to employ a computer program to scan this text to determine the presence or otherwise of specific required features. If evidence of the feature is found, the program should allocate a mark and write this to the results file, otherwise writing a diagnostic comment. The program should also be able to award marks for partially correct work.

3. Constructing the Program

In order to build a prototype marking program the product *Oracle Media Objects* (OMO) by Oracle™ Corp was chosen. It is a multimedia authoring tool running under *Windows* and supports a scripting language called *MediaTalk*, which features convenient user interface capabilities and provides a wide range of functions for file access and searching for specified text within files. The tool facilitated rapid development of a prototype and allowed relatively easy modification of the program during the testing and validation process.

A function was built for each of the features to be assessed and as each student’s work file was opened these functions were executed in turn. The output from each was written to a tab delimited text file, which could be later used to update the student assessment database. The modular construction of the program necessitates multiple passes through each input file, but given that program efficiency is not an issue for a prototype, this was acceptable. The code for three of these functions is listed in Appendix A. The way in which marks are awarded for partial answers can be seen.

4. Validating the Program

In order to validate the program it was necessary to mark by hand a cross-section of test documents and to compare this with the marks awarded by the program. To ensure maximum confidence in the results for minimum cost of data collection, some form of stratified sampling was indicated (see for example, Som [3]).

Stratification was accomplished by separating the one thousand candidates into nine equal strata, based on the

order of merit listing produced by the first run of the automatic marking program. From each stratum a simple random sample of four students was taken. This set of 36 cases was then used as the sample for benchmarking. Each of the 36 test documents was scored manually by two tutors, who in collaboration agreed to the mark to be awarded for each of the eleven features listed in Table 1. The set of marks so obtained then became the target to be achieved by the automated marking.

The objective was to produce results for which the differences between the manually obtained scores and those generated by the program satisfied two criteria:

(i) that the *mean* of the differences in scores should be no more than *five* percent of the total mark, and

(ii) that the difference in any specific case should be no more than *ten* percent.

It was also decided that a discrepancy beyond these limits could be tolerated if the automated results were consistently higher than the target, but not lower.

5. Results

The scores obtained manually for the benchmark sample together with the progressive results of three runs of the marking program are given in Appendix B. In the first run (Run 1) it can be seen that though the mean difference (-1.0) is within the desired tolerance limit there are some rather large individual discrepancies, including one 31% below the target. Each exception was investigated and the marking program modified to correct the discrepancy. Results in the second run (Run 2) are generally better, though in two cases the computer generated result is still more than 10% below the manual score. After further adjustments the third run (Run 3) generated results that satisfied the stated criteria; the mean of differences (3.2%) is less than 5% and no mark is more than 10% below the target mark, though on the whole computer generated marks are higher

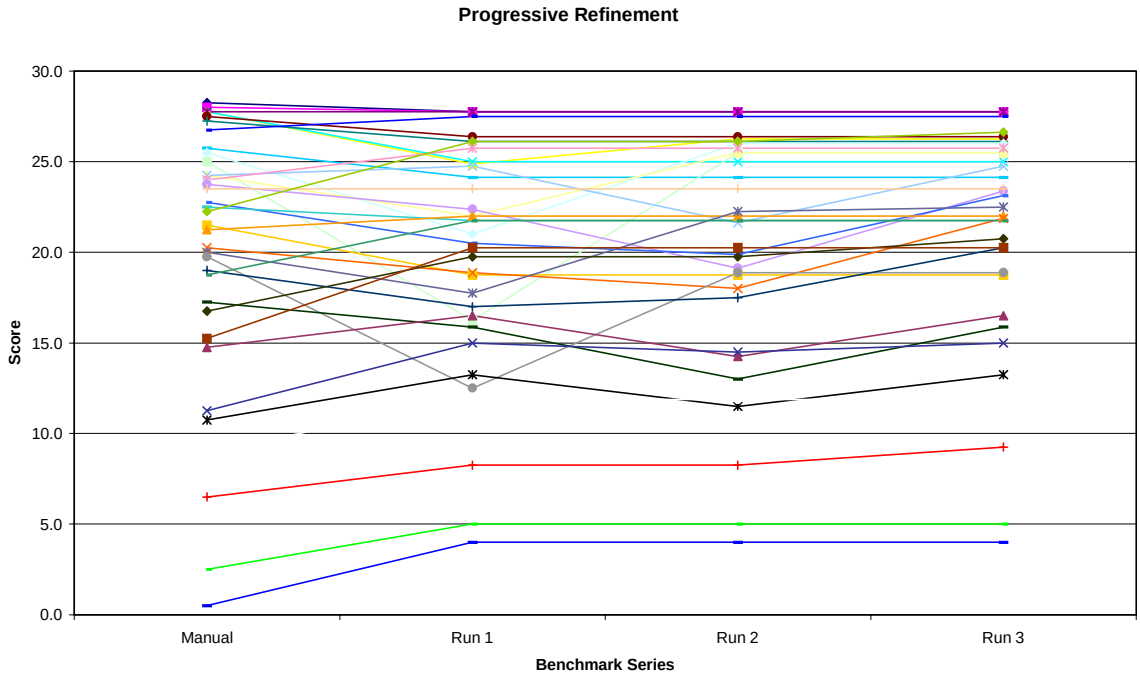

Figure 1 *Progressive refinement of automatic marking program*

In order to control for possible variability of the marking program between tests, the manual mark and the computer generated score in each case can be treated as 'matched pairs'. Consideration of the distribution of their differences about zero then results in a more powerful statistical test than simply a t-test comparing the difference in the means of the two sets separately (Baroudi, [2]). The mean of the differences between the manual mark and the final Run 3 scores is 0.91 with a standard deviation of 2.00. This yields a t-test value of 2.73, which, with 35 degrees of freedom, can occur by chance with a probability of 0.01. This allows us to conclude at 1% level of confidence that the *differences* are drawn from a population of values whose mean is zero, that is that the scores generated by the program are from the same population as the manual scores.

As can be seen from the scatter diagram in Figure 2 the results produced by the program tend to be consistently higher than those awarded manually by the markers. This is possibly because the program allocates small fractions of a mark for partially correct work more often than the markers did. The effect is most pronounced in low scoring

cases. It was felt that this was a satisfactory outcome, particularly as this test was near the beginning of the semester and intended more for formative purposes and encouragement than for summative evaluation. The solid line in Figure 2 represents the ideal case while the dotted line shows the "10% below" criterion.

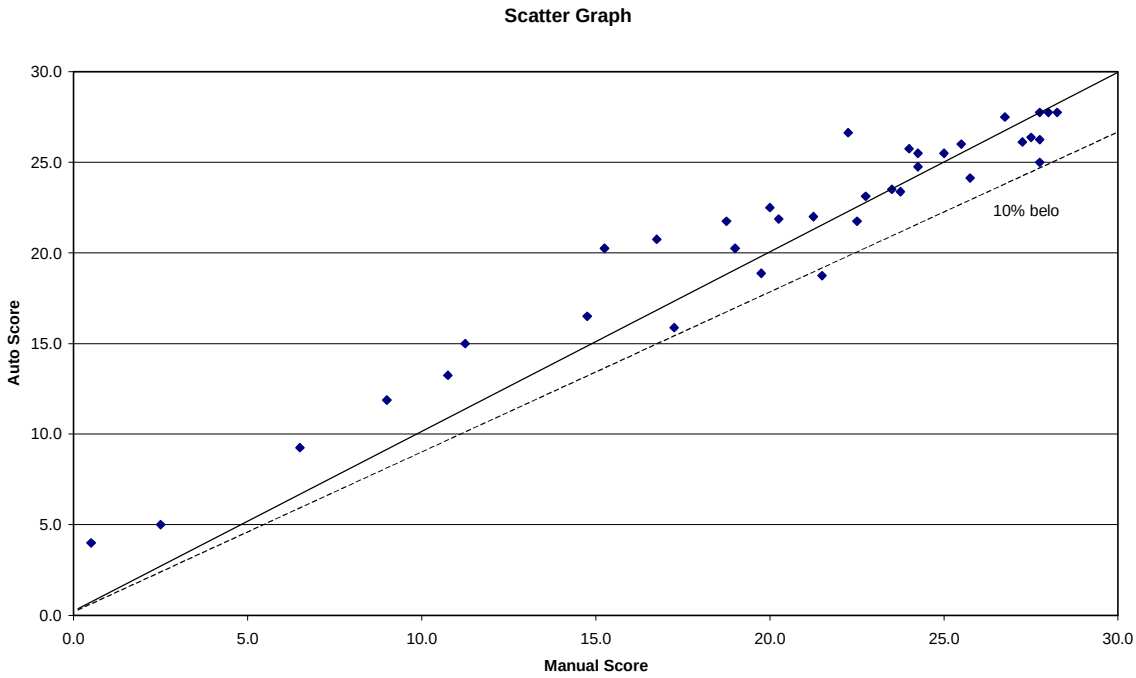

Figure 2 *Scatter graph showing results of final computer run (Run 3)*

Five cases of gross error were reported by students complaining that the mark awarded, in these cases lying between 2 and 4, did not appropriately reflect their ability. On checking manually it was found that they had indeed performed better, yielding scores from 12 to 20. Because such errors were glaring, they caused little trouble. These discrepancies are thought to result from the way in which the *.rtf* file is generated, particularly in cases where students make an abnormally large number of changes. Some difficulties in file management were created by the sheer size of the *.rtf* files, but these were satisfactorily overcome.

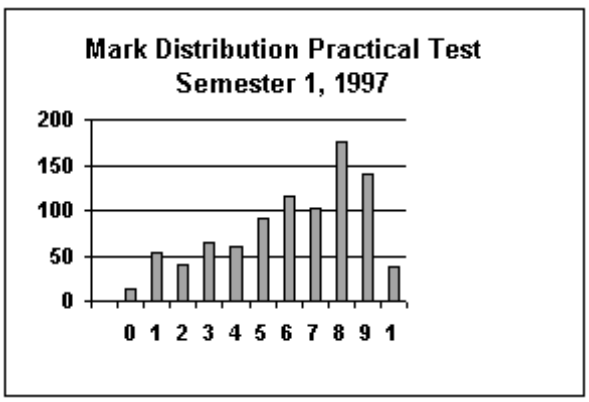

Figure 3 *Manual scoring, 1997*

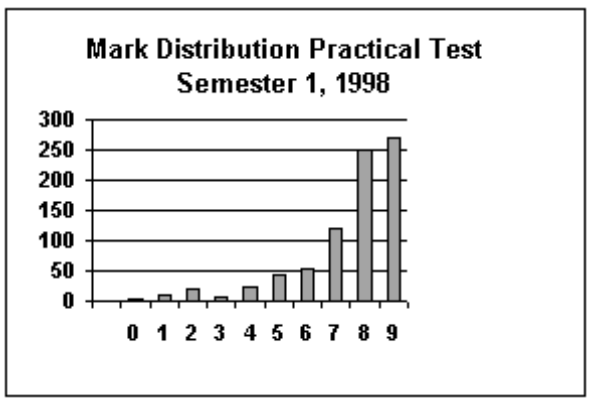

Figure 4 *Automatic scoring 1998*

Generally the situation was thought to be better that experienced in previous years when marking had been carried out by a team of senior student tutors. Consistency was certainly better than for manually marked work and time savings were estimated to be well over forty hours. This figure takes into account the time taken for benchmarking and will be greater when the automated marking program is reused.

Figures 2 and 3 show the distribution of marks awarded by tutors in 1997 those produced by the automatic marking program in 1998. The form of the test was identical in both cases and the group of students comparable in size (920 in 1997, 1003 in 1998) and ability. The program can be seen to be more "generous" but apparently

less able to discriminate between different quality documents than the human assessors.

6. Conclusion

The possibility of assessment of practical exercises involving *Microsoft Word* has been demonstrated. Though the program developed is no more than a prototype, its success should be a great source of relief to many academics facing the prospect of assessing practical tests for courses involving large numbers of students.

In the prototype program the scoring rules have been hard coded, expressed in the form of functions with rules for assigning marks. Use of this version is therefore restricted to one specific form of the test. In order to develeop a more general solution each feature for assessment would need to be represented in some standard form that can be stored separately as data, and read in by the program to generate the corresponding rules for awarding marks in any particular case.

An important aspect of this work has been the benchmarking approach used to validate the program. For any method of automated marking some similar process must be employed and this must be repeated when either the form of the test or the program is altered. If a more generalised approach can be developed, in which rules for marking are stored separately from the program, then this will help to reduce the extent of the validation task.

7. References

[1] P.F. Summons, J. Coldwell, F. Henskens, and C.Bruff, Automated Evaluation of Spreadsheet Concepts - Excel in Action, *Proceedings of 2nd Annual NSW Symposium on Information Technology and Information Systems SITIS'97, University of New South Wales,* Sydney, Feb, 1997

[2] Jack J. Baroudi and Wanda J. Orlikowski, The Problem of Statistical Power in MIS Research, *MIS Quarterly,* March, 1989, 87-106

[3] Ranjan K. Som, *Practical Sampling Techniques*, (New York:Marcel Dekker, Inc. 1996)

Appendix A

function tagOfStyle whatStyle

```
-- Since the exact tag for a particular style varies from document to document
-- this fucntion returns the appropriate tag for the requested style "whatStyle"
    put offset(whatStyle&";",  cd fld CurrentFile ) into endPos
    if endPos is empty then return "NoStyleOfType"'&whatStyle&"'here"
    put the number of words of char 1 to endPos of cd fld CurrentFile into numWords
    put empty into retVal

    repeat with count = 1 to 30
        put word numWords - count of  cd fld CurrentFile  into tempWord
        if tempWord contains "{" then
            -- in case it over laps onto next line
            put word numWords - count+1 of  cd fld CurrentFile  after tempWord
            put offset("{", tempWord) into startPos
            if char startPos+2 of tempWord = "s" then
            -- we have found a style tag
            else
                next repeat
            end if
            put "\s" into retVal
            repeat with inner = startPos+3 to startPos + 20
                if char inner of tempWord = "\" then
                    exit repeat
                else
                    put char inner of tempWord after retVal
                end if
            end repeat
            exit repeat
        end if
    end repeat
    return retVal
end tagOfStyle
```

function CDtitleCorrect

```
-- Checks if the style CD title is correct or not.  The student is asked to copy,
-- from the online Help screen some text about playing audio CD's on the computer.
-- A style, Heading 1, must then be applied to the heading which reads
-- "Using CD Player".
-- We check whether the text has been pasted
global retReason
global heading1Style, heading2Style, bodytextStyle, listNumberStyle, _
    listBulletStyle, titleStyle, quotationStyle

    put offset("Using CD Player",  cd fld CurrentFile ) into endPos1
    add 10 to endPos1
    put offset("Using CD Player", char endPos1 to 32000 of  cd fld CurrentFile ) _
        into endPos
    if endPos = 0 then
        put endPos1-10 into endPos
    else
        put endPos+endPos1 into endPos
    end if
    put the number of lines of char 1 to endPos of  cd fld CurrentFile  into lineNum
    put line lineNum-3 to lineNum+2 of  cd fld CurrentFile  into titleLines
    if titleLines contains heading1Style then
        put "Correct Style used for Title"&return after retReason
        return 2
    end if
    if titleLines contains "\fs" or titleLines contains "\s" then
        put "A Style was used for the title, but it was incorrect."&return _
            after retReason
        return 1
    end if
    put "No suitable style used for the Title."&return after retReason
    return 0
end CDtitleCorrect
```

function pageBreaks

```
-- Checks whether the correct page breaks have been inserted or not.
global retReason
    put return&return&"Page Breaks"&return after retReason
    if  cd fld CurrentFile  contains "\page" then
        put "One or more Page Breaks found in document as required."&return _
            after retReason
        return 1
    else
        put "No page breaks used in this document."&return after retReason
        return 0
    end if
end pageBreaks
```

Appendix B

Benchmarking Results

Id	Manual Mark	Auto Mark Run 1	% Diff	Auto Mark Run 2	% Diff	Auto Mark Run 3	% Diff
1	27.8	24.9	-10.2	26.3	-5.3	26.3	-5.3
2	19.0	17.0	-7.1	17.5	-5.3	20.3	4.4
3	22.3	26.1	13.7	26.1	13.7	26.6	15.5
4	28.3	27.8	-1.8	27.8	-1.8	27.8	-1.8
5	25.0	16.3	-31.0	25.5	1.8	25.5	1.8
6	11.3	15.0	13.3	14.5	11.5	15.0	13.3
7	22.5	21.8	-2.7	21.8	-2.7	21.8	-2.7
8	27.5	26.4	-4.0	26.4	-4.0	26.4	-4.0
9	20.3	18.9	-4.9	18.0	-8.0	21.9	5.8
10	15.3	20.3	17.7	20.3	17.7	20.3	17.7
11	27.8	25.0	-9.7	25.0	-9.7	25.0	-9.7
12	19.8	12.5	-25.7	18.9	-3.1	18.9	-3.1
13	28.0	27.8	-0.9	27.8	-0.9	27.8	-0.9
14	25.5	21.0	-15.9	26.0	1.8	26.0	1.8
15	25.8	24.1	-5.8	24.1	-5.8	24.1	-5.8
16	27.3	26.1	-4.0	26.1	-4.0	26.1	-4.0
17	6.5	8.3	6.2	8.3	6.2	9.3	9.7
18	26.8	27.5	2.7	27.5	2.7	27.5	2.7
19	18.8	21.8	10.6	21.8	10.6	21.8	10.6
20	24.3	22.0	-8.0	25.5	4.4	25.5	4.4
21	24.0	25.8	6.2	25.8	6.2	25.8	6.2
22	0.5	4.0	12.4	4.0	12.4	4.0	12.4
23	23.8	22.4	-4.9	19.1	-16.4	23.4	-1.3
24	24.3	24.8	1.8	21.6	-9.3	24.8	1.8
25	9.0	11.9	10.2	11.9	10.2	11.9	10.2
26	2.5	5.0	8.8	5.0	8.8	5.0	8.8
27	27.8	27.8	0.0	27.8	0.0	27.8	0.0
28	21.5	18.8	-9.7	18.8	-9.7	18.8	-9.7
29	17.3	15.9	-4.9	13.0	-15.0	15.9	-4.9
30	14.8	16.5	6.2	14.3	-1.8	16.5	6.2
31	23.5	23.5	0.0	23.5	0.0	23.5	0.0
32	16.8	19.8	10.6	19.8	10.6	20.8	14.2
33	20.0	17.8	-8.0	22.3	8.0	22.5	8.8
34	10.8	13.3	8.8	11.5	2.7	13.3	8.8
35	21.3	22.0	2.7	22.0	2.7	22.0	2.7
36	22.8	20.5	-8.0	19.9	-10.2	23.1	1.3
Mean	20.3	20.0	-1.0	20.4	0.5	21.2	3.2
Max	28.3	27.8	17.7	27.8	17.7	27.8	17.7
Min	0.5	4.0	-31.0	4.0	-16.4	4.0	-9.7