

Evolving spelling exercises to suit individual student needs

Marie Bodén^a Mikael Bodén^{a,*}

*^aSchool of Information Technology and Electrical Engineering
University of Queensland, Australia.*

Abstract

The principle of evolving educational content – stepwise and stochastic refinement of educational problems on the basis of a student’s previous success – is extended and evaluated for the domain of spelling. Experimental results, generated by an implementation in a classroom setting, show that the principle of evolving educational content is not only sensitive to an individual student’s ability but it also allows the student to take novel paths through an open-ended problem space. Learning outcomes are confirmed both quantitatively and qualitatively, and indicate that the technique transfers to a pedagogically challenging domain. We present how a problem space can be generated to effectively and automatically encompass a wider range of educational domains with little designer intervention.

Key words: Computer-aided education, evolutionary computation, user modeling.

* School of Information Technology and Electrical Engineering, The University of Queensland, QLD 4072, Australia. mikael@itee.uq.edu.au. Phone +61 (0)7 33652035, Fax +61 (0)7 33654999.

1 Introduction

While computer support for teaching in schools is steadily increasing, the use of learner-*adaptive* tools for tailoring educational scenarios to individual student needs is still in its infancy [5,9]. In many educational games the student is led from problem to problem, from level to level when certain pre-set criteria are fulfilled. According to constructivists like Piaget conceptual learning is a two-pronged, active, formative and staged process, namely through open assimilation and self-centered accommodation. However, in a classroom setting a “pre-leveled” educational software sometime triggers competitive as opposed to collaborative interaction between students (caused by students virtually facing the same problems but with varying speed). To this end, the idea of *evolving educational content* – first explored by Sklar and Pollack [13] to enable a constructivist learning environment – is here refined and experimentally evaluated in a pedagogically challenging domain.

The *principle of evolving educational content* lends itself to individualised selection of educational exercises on the basis of performance on previous exercises. Exercises are picked from a *problem space* which is automatically organised to reflect the character of the exercise. A student is guided through the problem space by means of a stochastic process sensitive to the student’s progress – small cautionary movements in the face of errors and large explorative movements when accuracy is high. This adaptive principle allows a pre-determined success-level to be attained by all students irrespective of their ability. In the software which we have built and evaluated, students are encouraged to collaborate – to help their peers to attain game-awards. The diversification of educational content resulting from the individualised and stochastic search process, makes collaboration interesting for both the learner and the helper compared to pre-leveled software which subjects all students to the same problems (which the helper may already have seen).

The present work has two major contributions. First, we show that Sklar and Pollack’s principle of evolving educational content transfers well into a pedagogically challenging domain. More specifically, we show that spelling exercises are automatically and dynamically chosen to suit individual student needs. Individual coverage and learning effects are confirmed both quantitatively and qualitatively. Second, the problem space from which exercises are chosen is automatically constructed to effectively avoid invalid selections (cf. [13]) and is thus less prone to cause development difficulties when tailoring the principle of evolving educational content to other domains.

2 Background

Learner-*centered* software is often based on an individualised student model [3]. The computer stores information about the individual student in a data model (e.g. a database or similar) and uses it to predict how a student will handle posed problems and how she progresses in her tasks. The model can be used to infer various user-specific pieces of information, such as which level of challenge is most suited for the student. The system monitors the model continuously for the selection of activities. The model adapts to the student-system interactions and aims to continuously mirror student progress. The control of the model often follows more or less pre-specified paths.

Students should be able to learn in their own way without having to follow already made up tracks and pre-made levels [12,10] (cf. [2]). The system needs to be involved in, and support the learning by monitoring progress and adapt as the student progresses so the right level of challenge can be given [1,8]. A constructivist approach to building software is open-ended and has no in-principle limitation in what activities the user may experience.

Sklar and Pollack [13] present what they call “an evolutionary approach to selecting content for educational games in a web-based learning community”. The control is dynamically evolved to realise true learner-*adaptive* software. With a clear exploratory component and a collaborative element it is an example of a constructivist approach to computer-aided teaching.

The term *evolution* indicates (somewhat narrowly) the means by which the user is directed to various activities in the system. More specifically, the selection of activities in the next phase of an educational game is influenced by the relative success of each and every activity that the user completed in the previous phase. New activities are generated by means of *genetic* mutation performed on a previous activity. Mutation is stochastic and may thus produce novel outcomes to be tested in the environment.

Sklar and Pollack [13] present some preliminary results from using the aforementioned evolutionary principle in a classroom environment. More specifically, their software implements evolution of keyboarding exercises. The goal for the system is to adapt, as the player learns to type, to provide suitable challenges for the player.

The keyboarding exercises operate over the internet so students can “play” against each other. In the game the student faces 10 words at a time. The student is instructed to type the words as fast as possible and while the student types through the list of words, the system keeps track of the actual time taken for each of them. The system retrieves words from a large database consisting of approximately 35,000 words of varying typing difficulty.

Evolutionary approaches partly remove the need for program control, but add the complication that we need to define a problem space. The problems or “codes” correspond to typing activities in an indirect manner, just as a set of genes translates into an individual organism. As usual in the field of evolutionary computation, codes are often numeric vectors and the genetic operators modify these vectors. The magnitude of the mutation has an effect of the relative similarity of the previous vector and its offspring. For this imposed similarity effect to translate into similarity at the level of activities, the code space needs to “mirror” the space of activities. The code space can be high-dimensional and as such it provides means for expressing many different aspects of the domain to be learned. Sklar and Pollack [13] suggest that the keyboarding code space is a seven-dimensional numeric space. In their application, dimensions correspond to typing features, namely (1) word length, (2) keyboarding level (as defined by a particular standard), (3) scrabble score, (4) number of vowels, (5) number of consonants, (6) number of 2-consonant clusters, and (7) number of 3-consonant clusters.

Initially, 10 points in the code space are selected on a more or less random basis. Noteworthy, of the 90 million possible vectors, the dictionary accounts for 6074. The code space is thus very sparse. However, using a trial-and-error method, *reproduction through sampling*, the system generates only points for which a word can be identified. The student is tested on the 10 words and typing speed is recorded. The words are ranked according to typing speed and divided into two groups: 5 words which need further practice, and 5 words that were handled well. For the 5 words that need further practice, 5 new points are generated by small mutation (meaning the new words will be similar to the old ones) of the 5 original codes. For the 5 words which were deemed successful, the 5 original codes are subject to large mutation. The large mutation means that the computer will make a big step in the space to find words classified in another group than the correctly spelled words. The resulting 10 codes will thus exploit the space in which words were handled less well by iterating the same typing features, and concurrently explore new parts of the code space, replacing those words which were handled well in the previous instant. This process of testing, selecting and mutating is repeated multiple times. Students will consequently meet challenges that are a result of their successes and failures of the previous activities. Notable, there is no pre-specified path that the student embarks into. The path is solely and dynamically determined from the the student’s typing results.

Sklar and Pollack tested the system on 44 fourth and fifth grade students (in the US). They showed that the system adapts according to the capabilities of the individual student, showing more of the typing domain to the student who is ready to see it. Moreover, they showed that the typing activities experienced over the course were more varied than with a standard pre-leveled curriculum. Finally, the learning effects were significant. 85% of the 44 students increased

their typing speed. The improvement can not be exclusively attributed to the use of their system – no control study was performed.

Three prevalent advantages of using an evolutionary approach in educational games are identified [13].

- (1) There is a possibility of individualised learning. A system that adapts in accordance with the individual student’s performance and in real time can provide suitable challenges and encouragement for the student advancing in their learning. Studies between variables in the classroom test provide evidence to support that students learn with the evolutionary approach and that they learn in different ways.
- (2) Student experiences are not pre-determined. Sklar and Pollack’s analysis shows that the students embark into very different levels and that they are challenged with words they normally would not experience using pre-leveled software.
- (3) There are potentially less costs for development of educational games. The *evolution of educational content* technique is not limited in domain. As long as a problem space can be defined, the technique extends to any domain. There is thus less effort involved in developing new educational scenarios. Most prominently, there is no specific need to define levels and paths through the curricula.

This study attempts to verify all three advantages but for a pedagogically challenging domain – that of spelling. Moreover, a refined method for generating an effective problem space is presented.

3 Domain and method

Pedagogical technology is useful as a complement to traditional teacher-student learning, to support learning processes at various levels [4]. The developed and evaluated software, The Magic Spell, is designed to be used as a complement to teacher supervised learning. The program does not directly teach spelling rules in English, it only subjects the student to spelling exercises. Additionally, our software informs the teacher of weaknesses and strengths, directing human contact to where it is needed.

English has a diverse set of spelling rules and a substantial number of exceptions. To assume as little as possible of the importance of rules we choose to match words ($W = \{W_1, \dots, W_N\}$) against a large set of patterns ($P = \{P_1, \dots, P_M\}$). Each pattern identifies a combination of letters as explained below. A word W_n is given an M -dimensional bit string B_n determined by the presence of patterns:

for each pattern $P_m \in P$
 if $match(W_n, P_m)$
 B_{nm} is 1
 else
 B_{nm} is 0

Selected patterns are either involved in spelling rules or just letter combinations arbitrarily regarded as difficult to spell. In addition, some patterns indicate families of spelling constructs (e.g. double consonants, or double vowels). Finally, since it has been argued that word that rhymes are typically handled with similar ability [14] we include a number of frequent word endings. In total 68 spell-patterns are used.

Each pattern is described by a so-called regular expression. Examples include $. *c[ey] . *$ (words that has the substring *ce*, *ci*, or *cy*; all *soft c* sound), $. *ie . *$ (words that have the substring *ie* as opposed to *ei*), $^{\wedge}th . *$ (words starting with *th*) and $. *[ioy]?e?s$ (words that end with *ies*, *oes*, or *ves*).

As a result of applying the patterns to all words, words with similar spelling constructs were grouped in the binary pattern space. However, there are many regions in this space for which no words exist, e.g. it is very rare for a word to match all of the four example patterns given above. Consequently, the mutation operator is severely constrained (cf. [13]). As a remedy, we perform singular value decomposition (SVD) on the set of bit strings (B). As shown below, the decomposition not only allows us to convert bit strings into compact, continuous vectors (with some information loss), but also ensures that the continuous space is densely populated. Previously, SVD has successfully been applied for converting word co-occurrence data (text statistics) to semantic spaces used for essay marking [7], tutoring systems (AutoTutor; [11]) and summary writing support [6]. Here, SVD maps a word's bit string B_n to a continuous vector S_n in a *spelling space*.

$$B = UXV^T$$

SVD decomposes B without loss into the expression UXV^T . U is an orthonormal matrix with rows corresponding to the rows of X , and columns corresponding to new derived variables such that each column is linearly independent of the others. V is also an orthonormal matrix with columns corresponding to the columns of U and rows composed of singular vectors. X is a diagonal matrix with the singular values. A large singular value indicates a large effect of the corresponding dimension on the sum squared error of the approximation. The role of these singular values is to relate the scale of the factors in the other two matrices to each other such that when U , X and V^T are multiplied, B is reconstructed.

To reduce dimensionality, only x of the largest singular values in X are used. Thus, only parts of U and V (determined by the choice of x) are used. We denote the new matrices constrained by x , $X_{[x]}$, $U_{[x]}$ and $V_{[x]}$ respectively. The problem/spelling space is defined as

$$S_{[x]} = U_{[x]}X_{[x]}$$

As a result of the decomposition the original matrix B is approximated by

$$B \approx S_{[x]}V_{[x]}^T$$

We arbitrarily collected 3622 spelling words for grades 1-5 (according to the US primary school system; here labeled as level 1-5). Also, words that are frequently misspelled were included (here labeled as level 6). All words were checked for conformity to Australian English spelling.

All words are represented using a vector in the spelling space, $s \in S$. Unmodified, the selection principle proposed by Sklar and Pollack would choose between words of arbitrary difficulty. The user could easily be intimidated by the sudden appearance of words which are too complicated. Importantly, we wish to adapt the level so that the users experiences a constant success rate. Taken together, we suggest using level as an additional parameter to adjust during user interaction. Specifically, the spelling space is searched as usual by means of a point subjected to mutation, but only the words belonging to the right level can be selected. The level can then be adjusted (to the individual user) to maintain a pre-determined constant performance ratio (average spelling faults per word).

The selection works as follows:

- (1) If the user has not been using the system before, *level* is set to 1.
- (2) From the current level, R words $w_{t,1}, \dots, w_{t,R} \in W$ are selected by inspection and mutation of $w_{t-1,1}, \dots, w_{t-1,R}$:
 - $w_{t,r} = z(\sigma(\varsigma(w_{t-1,r}), mut), level)$; where ς is the function which determines the spelling space vector for a given word, σ is the function which mutates a vector using the rate given by mut , $mut = 0.05$ if $w_{t-1,r}$ was spelled incorrectly, $mut = 0.50$ if $w_{t-1,r}$ was spelled correctly, and z is a function which selects one word of the words regarded as closest to the calculated spelling space vector (according to the Euclidean distance) *and* belonging to the appropriate *level*.
- (3) Each word $w_{t,1}, \dots, w_{t,R}$ is corrected, t is incremented, if needed *level* is incremented, and the procedure is repeated from step 2.

The mutation operator is responsible for generating the appropriate set of words from the history of words. Both in Sklar and Pollack's study and here,

only the previous set of words is used. For spelling the ranking for selecting the appropriate mutation rate is not obvious. We rely on the information whether the word was spelled correctly or not. The new set of words is based on the individual words, correctly spelled words are replaced by words that explore distinct parts of the spelling space, misspelled words are replaced by words similar in spelling space (thus exhibiting similar spelling features). In a typical test run (at level 2), the system replaced a misspelled **season** by **seal**, and when **seal** was incorrectly spelled the system replaced it by **least**, and next **feast**. All words in the chain share the `.*ea.*` pattern. At this stage, **feast** was correctly spelled and the system replaced it by **thirsty** – a very different spelling word. Another example chain is **attic**, **hopped** and **cliff**, in which all words fit a double consonant pattern.

The movements in the spelling space rely on words being organised in accordance with spelling features. We noted in preliminary simulations that the inclusion of feature families (e.g. patterns matching double vowels, including patterns matching **ee**, **ea** and **ae** to mention a few) are useful for imposing family resemblance among words in the spelling space.

Two classes of students from a primary state school in metropolitan Brisbane, Australia, were selected as subjects for this study. Class A consists of 12 grade three students and 8 grade four students (ages 8 and 9, respectively). Class B consists of 27 grade four students. The students were using the program during five weeks. A teacher and/or instructor was present at all times and students were able interact freely. We were unable to perform a control study due to the limited number of subjects.

The Magic Spell greets the user, and presents batches of $R = 6$ words. Using speech synthesis the user is prompted to spell each word. If a word is incorrectly spelled, the user is prompted to re-type – this time with the printed word too – until a correct spelling is given. When a batch is completed with 4/6 correct first-attempts, the user is rewarded with a ‘bean’. When 6 beans have been collected, the user receives a game-award which can be traded against other users’ game-awards. The user can continue completing spelling exercises for as long as desired, and intermittently, trade game-awards with other students. The user only receives feedback regarding progress by means of game-awards. The level is only available for inspection by teachers.

4 Results

The Magic Spell logs all words tested, which level the word belongs to, and whether the word was spelled correctly. Moreover, we can derive if it is the first time the word is presented or not. Finally, we can find out if the word is

selected on the basis of exploration (the previous word was correctly spelled) or exploitation (the previous word was incorrectly spelled).

The progress and process can be analysed and understood by means of probabilities derived from frequencies of the aforementioned data. All probabilities are expressed as values between 0 and 1, where $P(x) = 0$ means the random variable x is always *false*, $P(x) = 1$ means x is always *true*, and $P(x) = 0.5$ means that x is *true* in half of the cases.

We use the following random variables.

correct – The word is correctly spelled;

explored – The word is selected by exploration;

exploited – The word is selected by exploitation ($P(\textit{exploited}) = 1 - P(\textit{exploration})$);

novel – The word is selected for the first time.

In the analysis we also use conditional probabilities, e.g. $P(\textit{correct} | \textit{explored} \wedge \textit{novel} \wedge \textit{level} = 1)$ which means “the probability of a word being spelled correctly given that the word was selected by exploration and is selected for the first time and level equals 1”. We use data from all 47 students unless otherwise stated.

4.1 *The data*

As described earlier, all words are matched against the spell patterns. The resulting bit vectors were subjected to singular value decomposition which defines the space in which the mutation operator selects words. The vocabulary consists of 3622 words and 68 spell patterns are used. Some spell patterns are much more frequent - these are simply more general, e.g. double consonants, and some are very rare). This distribution is used to normalise the spell space coverage of individual students described later.

4.2 *Completion and interaction*

The students work in varying tempo and with varying persistence. After five weeks of usage, the students had completed a mean of 304 words (median is 282). The standard deviation is 197 words. With the help of the program, all attempted words are correctly spelled sooner or later. If we look at the number of words correctly spelled in the first instance the mean is 194 words (the median is 187) and the standard deviation is 113 words over all students.

The variability in completion can not only be quantified in terms of number of words. The Magic Spell automatically adjusts the level when, after a pre-specified number of rounds (10), the student has shown consistency in accurate spelling. In Fig. 1 the most difficult level (the final level reached in the five trial weeks) for the students is shown. The distribution shows that most students (40/47) reached beyond level 1, of which some reached the 6th and top most level. Overall, the variability is high, the mean is 2.6 (with a median of 2) and the standard deviation is 1.3 levels.

If one instead focuses on the number of words that were completed in each of the levels, the distribution is slightly different. In Fig. 1 it can also be seen that the students completed almost 8000 words at level 1 and only a few at the higher levels. The mean level is 1.8 (the median is 1) and the standard deviation is 1.1 levels. However, this is mainly explained by the fact that all students start at level 1. Given that most students ventured beyond level 1, the tail would be longer with a longer trial period.

Students interact freely, mostly by helping each other and by trading their attained game-awards. Students were asked questions about their impressions and answered unanimously that they felt motivated to continue using the program, mainly to advance their “trading status.” The level of motivation and their trading activity seemed uncorrelated with the progress as measured by advancement to higher levels.

4.3 *Spelling accuracy*

We measured the probability of correct spelling when a word came up for the first time,

$$P(\text{correct}|\text{novel}) = 0.63.$$

Furthermore, we measured the spelling accuracy when the children were tested repeatedly on a word,

$$P(\text{correct}|\neg\text{novel}) = 0.66.$$

The standard deviations (over students) while estimating the probabilities are 0.07 and 0.08 respectively, and indicate that the system keeps the success rate rather constant and consequently reduces the element of competition and encourages collaboration.

Do the students *gain* in accuracy while working with the system? The question is central to the pedagogical usefulness of the system and deserves a careful

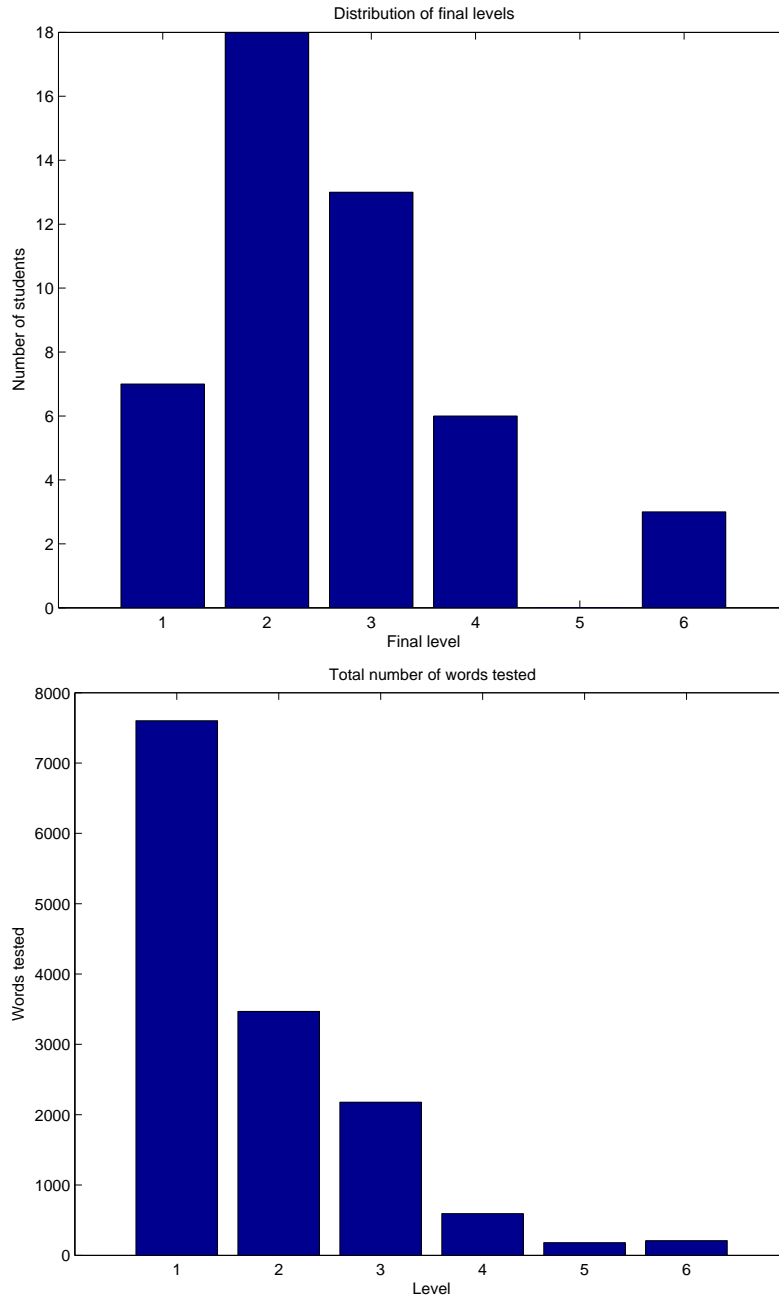


Fig. 1. The maximum level reached by students during the trial period (above). The distribution of spelling attempts over levels during the trial period (below).

treatment. First, the system constantly adapts to the user’s activity. Some words are more or less similar to previous ones. Second, words are presented according to levels. Hence progress is likely to result in more difficult words being selected. More difficult words impede accuracy. So how can we frame the answer in a neutral manner?

By only considering novel words (those that appear for the first time) we avoid the potential pitfall of measuring performance in terms of repeated tests.

By only considering words reached by exploration we minimise the effect of similar words being selected as those already tested. Moreover, using only words selected by exploration ensures that words are selected on a more or less random basis. Remember that exploration is a big mutation and thus resembles random word tests. Finally, measuring progress within the same level removes artificial increases in difficulty.

The automatic upgrade of level also needs to be taken into careful consideration. A student can at best be upgraded after 10 rounds of words. If more rounds are included such students are just excluded from the tail of any analysis. However, any student that is upgraded later has, by definition of the requirement, shown poor performance in the first few rounds. Moreover, if we use the end of the level (final level excepted) as the tail end for analysis, consistent progress the last rounds must have happened. In combination, these artifacts will bias the tail of the performance curve. We choose to deal with this by only showing the first 10 rounds at level 1. Moreover, the final level for each student (may be different from student to student) does not exhibit such artifacts simply because the student has not yet been upgraded. We choose to show the spelling accuracy for the last 25 rounds of the final level. A number of students did not complete 25 full rounds at this level and thus only appears at the tail of this curve.

In Fig. 2, the spelling accuracy is shown for the first 10 rounds at level 1. The plot includes all students, and shows the estimated probability of spelling a word correctly (given that the word is novel, is selected by exploration and is from level 1) plotted as solid line for each of the first 10 rounds. It needs to be emphasized that for one particular activity in time, a student is either correct (1) or not (0), and in combination with a rather low number of students, the estimate tends to fluctuate. To visually accentuate the increase in accuracy a linear fit to the probability is plotted as a dotted line.

Also in Fig. 2, the estimated probability of correct spelling, given the word is novel, selected by exploration and from the individual student's final level, is plotted as a solid line. The probability is again estimated on the basis of all 47 students and this time over the final 25 rounds (if available within the same level). A linear fit to the probability is plotted as dotted line. The increase in spelling accuracy is clear. However, fluctuations are quite severe due to the limitations in population size. This effect can be seen for the first few rounds in the final level plot where only about five students contribute to the probability estimation.

On a cautionary note, the increase in spelling accuracy can not be solely attributed to an advance in spelling capability. Factors such as learning to use the program and computer more effectively reduce mistakes while having little to do with spelling.

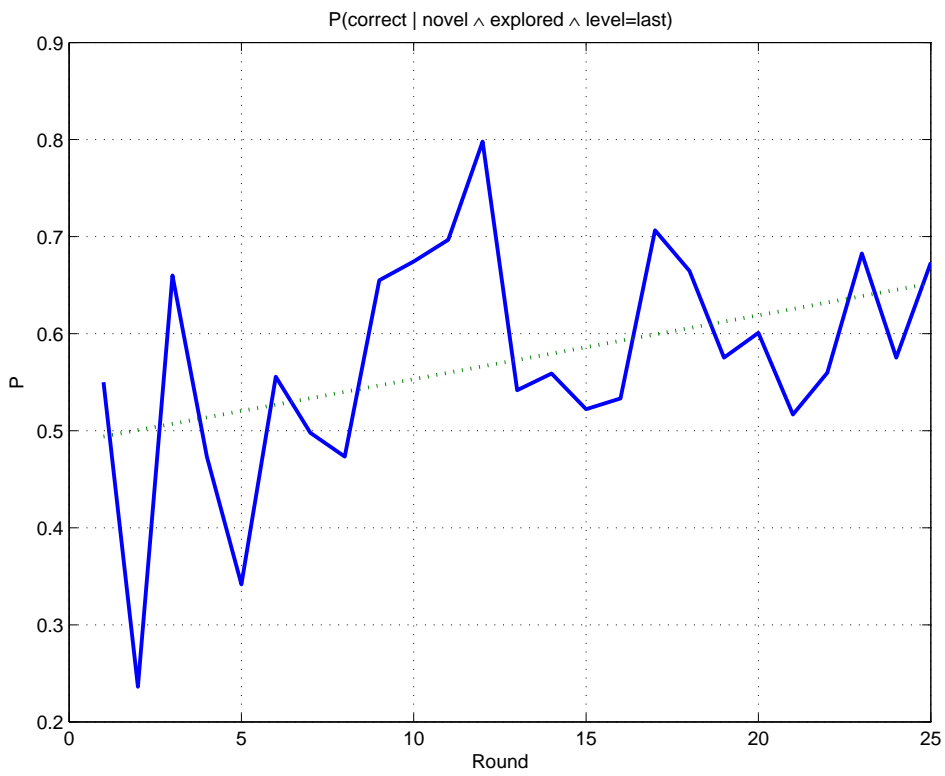
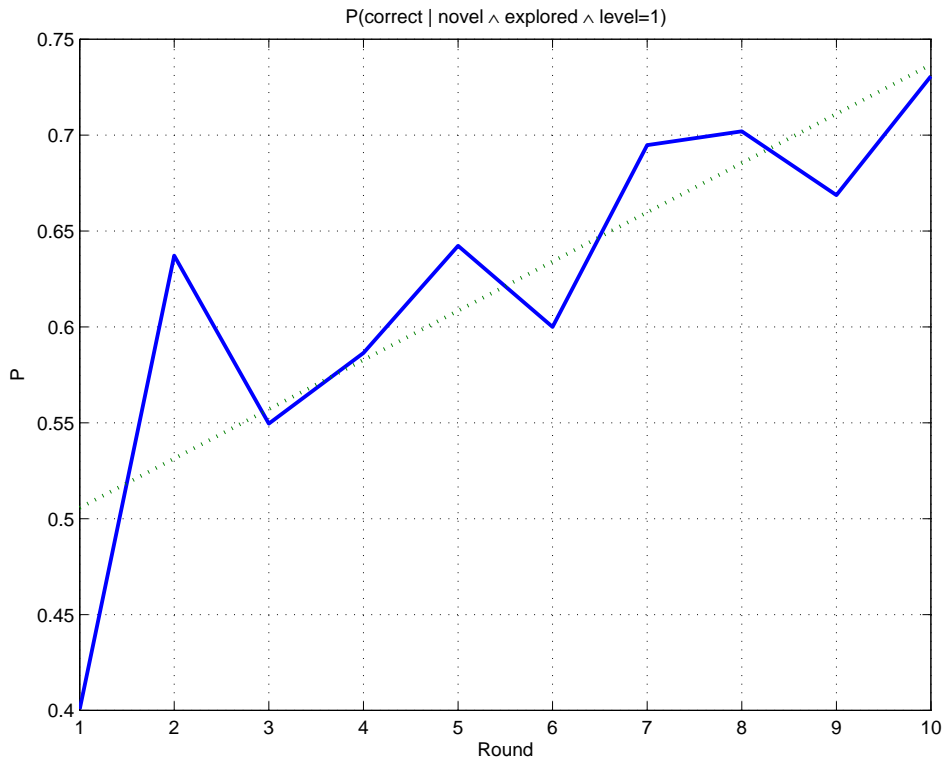


Fig. 2. The average spelling accuracy over the first 10 rounds (above) and the final 30 rounds (below).

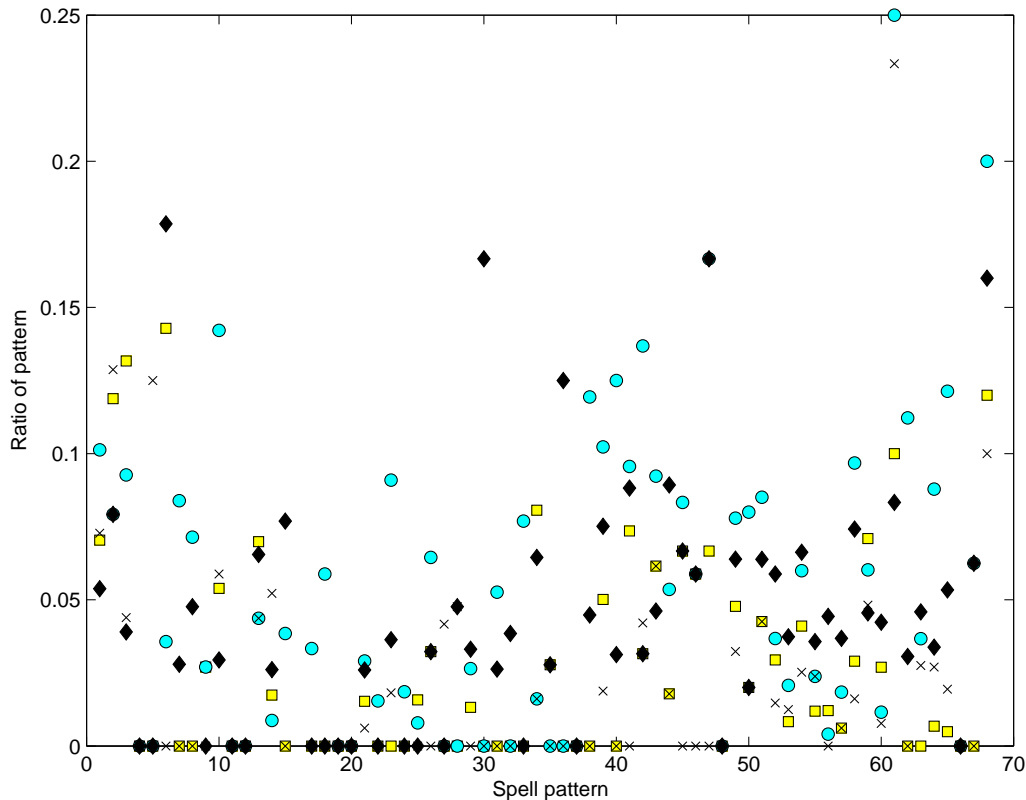


Fig. 3. The spread of patterns attempted by four arbitrarily selected students.

4.4 Coverage and spread

Words in the vocabulary represent the material by which students learn and are tested on. The words are spread into a spell space by means of matching to spell patterns and singular value decomposition. The resulting spell space will be organised so words which share similar spell patterns are located nearby.

One principal motivation for using a user adaptive technique is that coverage should be maximised while still keeping the level of challenge to an (for the individual student) appropriate level. Consequently, we expect to see that there will be varying coverage when individual students are compared, but there should be no parts of the spell space that are poorly covered when all students are considered.

In Fig. 3 the proportion of tested spell patterns of the number of possible spell patterns (as seen from the vocabulary) is indicated for four arbitrarily selected students. That is, as the student goes through the spelling exercises, each word matches a number of spell patterns. Each spell pattern is counted and at the end of session, these numbers are divided by the number of total spell patterns available in the vocabulary.

Evidently from Fig. 3, the technique offers good coverage of the space. There are a few pockets of poor coverage, corresponding to rare spell patterns. The variation in the plot also supports that students cover different parts of the spell space.

5 Discussion

Interviews were conducted with the two teachers and five arbitrarily selected students (representing the full range of abilities) to validate and contextualise the quantitative results. Overall, both teachers and students enjoyed working with the software. Both teachers used The Magic Spell as a complement to their normal teaching but confirmed that the students learned while working with the program. The teachers understanding was that the students improved their overall spelling ability, an indication that the positive trend observed for monitored accuracy is not an artefact. The classes had several children with special difficulties (ADHD and English as a Second Language). The teachers expressed that all children used the program effectively, though ESL students had more difficulties in understanding the synthesised speech.

While the interviewed students thought that spelling exercises were increasing in difficulty, they also maintained that game awards were *not* becoming more difficult to attain over time. We see this observation as confirming the systems sensitivity to the students ability: the encouragement stays constant while the difficulty increases. The game awards served the additional purpose to encourage interaction among students. Both the teachers confirmed that the incentive was real, and that students enthusiastically helped each other to spell to to exchange game awards later.

6 Conclusion

Evolving educational content is a strategy that does without a pre-specified control mechanism. There is no student model yet the technique offers a learner-adapted exploration of activities. Also, the strategy seems to support users with differing abilities and learning styles. Seen over the whole student population, we report on an overall positive trend in spelling accuracy in a specific study of The Magic Spell. The positive trend concurs with the relative improvement of 85% observed by Sklar and Pollack [13] for keyboard typing using the same principle.

The evolutionary system we evaluate introduces a performance stabilising factor to reduce the competitiveness among students, to encourage learners at all

levels and to stimulate collaboration. Students move between pre-determined levels but the system hides transitions to the students. Instead the level of difficulty is used to keep the spelling performance to around 0.65 for all students ($SD=0.08$). The level of difficulty increases as the student progresses but the encouragement remains almost constant – this is also expressed by all the interviewed students. In essence, the environment (realised through the spelling exercises) co-evolves with the student’s spelling abilities.

Coverage of the possible spelling exercises varies with student – a clear indication of adaptivity to individual difficulties. Viewed as methodology, the principle of evolving education content seems to transfer well into other domains with little system development involved. In the present work we introduce singular value decomposition as a means to construct a space in which activities are organised according to a possibly large and sparsely populated feature space. Exercises are effectively selected by an evolutionary, stochastic process yet faithful to the previous successes and failures of the student. The overall development required for transferring the principle to other educational domains is basically confined to interface programming – little designer intervention was required to get the evolutionary control mechanism to work in the spelling domain.

References

- [1] Susan Bull, Jim Greer, and Gord McCalla. The caring personal agent. *International Journal of Artificial Intelligence in Education*, 13:21–34, 2003.
- [2] Andrea Bunt and Cristina Conati. Probabilistic student modelling to improve exploratory behaviour. *User Modeling and User-Adapted Interaction*, 13(3):269–309, 2003.
- [3] Jim Greer and Gord McCalla, editors. *Student models: The key to individualized educational systems*. Springer Verlag, New York, 1994.
- [4] M. J. Hannafin and S. M. Land. The foundations and assumptions of technology-enhanced student centred learning environments. *Instructional Science*, 25:167–202, 1997.
- [5] S. Hsi and E. Soloway. Learner-centered design. *SIGCHI Bulletin*, 30(4):1–6, 1998.
- [6] E. Kintsch, D. Steinhart, G. Stahl, and the LSA Research Group. Developing summarization skills through the use of lsa-based feedback. *Interactive Learning Environments*, 8(2):87–109, 2000.
- [7] T. K. Landauer, D. Laham, B. Rehder, and M. E. Schreiner. How well can passage meaning be derived without using word order? a comparison of latent

semantic analysis and humans. In M. G. Shafto and P. Langley, editors, *Proceedings of the 19th annual meeting of the Cognitive Science Society*, pages 412–417, Mahwah, NJ, 1997. Erlbaum.

- [8] Renate Motschnig-Pitrik and Andreas Holzinger. Student-centered teaching meets new media: Concept and case study. *Educational Technology & Society*, 5(4), 2002.
- [9] D. A. Norman and J. C. Spohrer. Learner-centered education. *Communications of the ACM*, 39(4):24–27, 1996.
- [10] Seymour Papert. *The children's machine: Rethinking school in the age of the computer*. Basic Books, New York, 1993.
- [11] N. K. Person, A. C. Graesser, R. J. Kreuz, V. Pomeroy, and the Tutoring research group. Simulating human tutor dialog moves in autotutor. *International Journal of Artificial Intelligence in Education*, 12:23–39, 2001.
- [12] M. Resnick. *Turtles, termites, and traffic jams: Explorations in massively parallel microworlds*. MIT Press, 1997.
- [13] E. Sklar and J. Pollack. An evolutionary approach to guiding students in an educational game. In *Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior (SAB-2000)*, 2000.
- [14] R. Treiman. Introduction to special issue on spelling. *Reading and Writing: An Interdisciplinary Journal*, 9:315–319, 1997.