Proceedings of the LREC 2020 Workshop Games and Natural Language Processing, pages 17–25
Language Resources and Evaluation Conference (LREC 2020), Marseille, 11–16 May 2020

Cipher: A Prototype Game-with-a-Purpose for Detecting Errors in Text

Liang Xu & Jon Chamberlain
University of Essex, Wivenhoe Park, Colchester, Essex, UK
{lx18921,jchamb}@essex.ac.uk

Abstract

Errors commonly exist in machine-generated documents and publication materials; however, some correction algorithms do not perform well for complex errors and it is costly to employ humans to do the task. To solve the problem, a prototype computer game called Cipher was developed that encourages people to identify errors in text. Gamification is achieved by introducing the idea of steganography as the entertaining game element. People play the game for entertainment while they make valuable annotations to locate text errors. The prototype was tested by 35 players in a evaluation experiment, creating 4,764 annotations. After filtering the data, the system detected manually introduced text errors and also genuine errors in the texts that were not noticed when they were introduced into the game.

Keywords: Game with a purpose, error detection, gwap, distributed knowledge acquisition, text correction

1. Introduction

Text error detection techniques have been widely used in a variety of applications, e.g., spell checkers and Optical Character Recognition (OCR) readers. Studies have explored methods to improve the accuracy of word correction, with several techniques commonly used for solving the problem. Dictionary-matching, common error list analysis and targeted error detection approaches have been employed to address general and simple errors like misspelling and non-word mistakes in articles (Kukich, 1992). For more complex errors, e.g., grammar errors and errors which rely on context, there are computational models for detection and correction of errors, such as the Statistics Language Model (SLM) (Kukich, 1992). It is still challenging to detect complex errors and the above methods might not detect them all. To solve the problem there are applications which use human effort that is to say, people who have high proficiency in the language are employed to do the proofreading task. They are paid to find all the errors in the text, especially those errors that computers cannot detect. The disadvantage of this approach is the cost of annotators.

Computers outperform humans in many tasks; however, they are still not powerful enough to beat humans in fields such as creative work, emotion detection, context-based analysis and so on. Therefore, we use human processing abilities to solve the problems that machines are not good at (Quinn and Bederson, 2011; Chamberlain et al., 2013). In our project, a computer game was designed to motivate people to find text errors, which is based on the idea of a Game With A Purpose (GWAP) (Von Ahn, 2006). GWAPs are able to reduce the cost of human effort to solve computational problems through gamification. In this research steganography is used as part of the game design to improve enjoyment and help to achieve the data collection goal. The purpose of the prototype was to evaluate human performance in text error detection through an entertaining game. In the game, every decision of error detection from a player is an annotation, whether it is correct or not.

This research aims to answer two questions: 1) How good are humans at identifying errors in text? and 2) Is it possible to build a game to motivate people to detect errors in text? First, this paper summarises related work in the field of text error detection and correction, as well as notable games used for language annotation. Section 3 presents the methodology and design of the prototype game called *Cipher*, developed to detect errors in text. Game implementation is explained in Section 4. Section 5 presents the results from a lab-based evaluation of players using the game, followed by discussion of the applications of the game and its limitations.

2. Related Work

2.1. Error detection and correction

Text error detection and correction techniques have three common problems. The first problem is non-word error detection. Any word or string that cannot be found in the dictionary might be a non-word error. The most common instances are spelling mistakes, for example, “anatomy” is misspelled as “anonomy”. The second problem is isolated-word error correction that is to correct the detected non-word errors in text. The third problem is context-based word correction. The errors which rely on the context are difficult to detect, because they are correctly spelled words which are not correctly used in the context, for example, “off” when it should be “of”. Both words are spelled correctly but the former will not fit the specific context. The research (Kukich, 1992) is a gist of some of the techniques used in the detection of text errors in research as summarized by Kukich:

2.1.1. Non-word detection techniques

Dictionary lookup is the most common and direct spelling error detection approach. Every target word is compared to the word in the dictionary, one by one, to check if they match. A correct word will find a counterpart in the dictionary while a wrong word will not. To achieve this, a hash table is used to compare the hash address of the target word with the address of the word in the dictionary (Knuth, 1973). However, the size of the dictionary for comparison might influence the dictionary access speed and efficiency. A solution is that the category of the dictionary chosen to match the target word is related to its application or domain, which narrows down the range of search vocabulary (Mishra and Kaur, 2013).

N-gram analysis can also be used. The value of *n* is often 1, 2, or 3, which represents the n-letter subsequence of the target word. This approach is a comparative analysis of each n-gram with the corresponding word in a preprocessed n-gram table which is metamorphosed from a dictionary or a specific corpus. The related applications perform well in detecting errors in machine-produced text, for example, electronic-documents generated by OCR devices, but they are not good at detecting hand-written errors (Riseman and Hanson, 1974).

2.1.2. Isolated-word correction techniques

The process of word correction is more difficult than error detection. In most situations, text errors require not only to be found but also correctly modified. Thus, there are some techniques for non-word error correction (Kukich, 1992; Mishra and Kaur, 2013):

The **Minimum Edit Distance Algorithm** was first introduced to explain the minimum steps to modify a word from its wrong form to its correct form (Wagner, 1974). The ways of modification include insertion, deletion, and replacement. There are several useful algorithms based on this idea that compute the minimum edit distance between a text error and its correspondent correct counterpart .

The **Similarity Key Algorithm** assigns a key to those words that look alike (e.g. far, for, form, from, fool). Therefore, all similar words have the same key and those corresponding words are the values of the key. When a spelling mistake is detected, according to the key of this error, all the similar words which have the same key will be attached as correcting candidates. The advantage of the method is that there is no need to compare the error to every word in the lexicon or corpus one by one, which saves processing time and promotes the efficiency of correction process (Odell and Russell, 1918).

A **Rule Based Algorithm** is a process of collecting the features of common spelling mistakes compared to their correct forms and turning the features into different rules. For example, the word “gracefull” is an error because the last letter of its correct form “graceful” has been repeated, which is a feature of the error. According to the rules which can be applied, text errors can be detected and corrected (Yannakoudakis and Fawthrop, 1983).

When it comes to probability for text recognition and error correction, there are two different probabilities which can be applied: **transition probability** and **confusion probability**. Transition probabilities are the probabilities of a letter followed by another letter correctly. Confusion probabilities are the probabilities of a wrong letter appearing after a correct letter. Algorithms based on the two probabilities are useful in text recognition preprocessing and word correction (Bledsoe and Browning, 1959).

Neural Networks (NNs) play an important role in spelling error correction. The correction ability of NNs becomes more accurate with large scale spelling error data for training. Some spelling correction applications record users’ spelling mistakes as the training data to train the Neural Networks, then those frequent spelling errors associated with the user’s misspelling habits can be easily corrected and even predicted (Rumelhart et al., 1986).

2.1.3. Context-based correction techniques

For this type of error (the hardly detectable errors), the performance for real-word error correction achieved by existing empirical studies is not as promising as the performance for isolated-word error correction. Real-word errors could be syntactic errors or semantic errors, thus, it is more difficult to detect or correct this type of error. It has been suggested that around 40% of all text errors are context-based errors (Mitton, 1987).

2.1.4. Summary

The core of these techniques is based on algorithms. Furthermore, different types of errors have different detection and correction techniques. The model built for this project uses human effort through a game and deals with all kinds of errors in text. For context-based errors particularly, this model is expected to find those errors that computer algorithms cannot find because humans are better than machines at understanding context and finding hard to detect errors.

2.2. Games with a Purpose

Computers are able to replace humans in many fields; however, there are still some tasks that human perform at better than computers. The idea of a Game With A Purpose (GWAP), proposed by Luis von Ahn, was to design an entertaining game that motivates people to solve a computational problem (Von Ahn, 2006). The problem is typically one which computers cannot solve it yet. The GWAP idea benefits from three conditions (Von Ahn and Dabbish, 2008): Firstly, the ubiquity of the Internet to provide a connected workforce is an important factor. There are more people who use the Internet every day and almost everything in daily life is involved with it. Secondly, some computational problems are challenging for computer algorithms but not complicated for human beings, such as syntax annotation, labelling objects within an image, common-sense collection and so on. Lastly, computer games are popular and an increasing number of individuals spend considerable time playing them. The approach has been increasingly applied to many fields such as text analysis, image recognition, Internet search reinforcement, security monitoring, information filtering, etc (Lafourcade et al., 2015).

2.2.1. The ESP Game & Peekaboom

The *ESP Game* is a web-based game focusing on labelling images (Von Ahn and Dabbish, 2008). In the game, two players are given the same image and both players use a word to describe the picture. If the outputs of the two players are identical, they win the game. In total, more than 200,000 people played the game and it collected 50 million image labels. *Peekaboom* is another example in which the goal of the game is to not only label the image contents generally but also locate specific image objects within each image, based on the data from the ESP game (Von Ahn et al., 2006). According to the usage statistics, *Peekaboom* collected 1,122,998 pieces of data with 14,153 players in one month. The *ESP Game* and *Peekaboom* were used to improve Internet search performance, especially for searching pictures which contain noisy information (Von Ahn and Dabbish, 2008). Additionally, *CAPTCHA*, an automated cryptographic program introduced by Luis von Ahn, is a successful example of the recognition competition between humans and computers, although it is not a GWAP (Von Ahn et al., 2003). The test result differentiates humans from computers.

2.2.2. Phrase Detectives

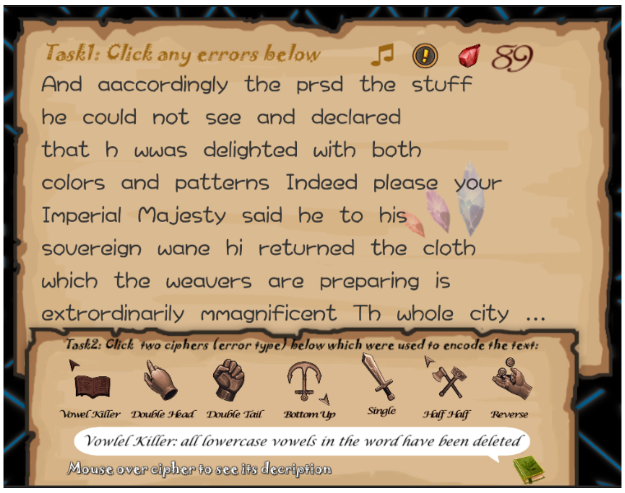
GWAPs are also used in language annotation. *Phrase Detectives* is an online game with the purpose for identifying semantic connections in vocabulary under a certain context (Chamberlain et al., 2008). More specifically, the goal of the game is to encourage people to detect anaphoric coreference (the word or phrase used to replace the former mentioned object in the text). For example, in the sentence: "Tom and Mike are friends and they study in the same university", the players need to annotate "they" as reference to the earlier mentioned entity "Tom and Mike". *Phrase Detectives* used players to annotate the text, as well as validate the decisions of other players in order to optimise the data collection process (Chamberlain et al., 2018).

2.2.3. Digitalkoot

OCR devices are able to achieve recognition accuracy of a character as high as 99% for documents with high scan quality; however, word recognition accuracy decreases with word length to around 95% for a five-letter word (Kukich, 1992). *Digitalkoot* is designed to minimise the effort to detect and correct OCR errors in old Finnish newspapers. *Digitalkoot* is divided into 2 parts: verifying OCR outcomes and using human OCR (humans reading the text). In the first part, several words generated by OCR devices are shown to players. They need to decide whether those words are correctly recognised compared to the original text within the images. The second part is to encourage players to type each word with a given word image to build a bridge to make the game character cross successfully. In 51 days, 4,768 players played the game. They spent 2,740 hours on the game and finished 2.5 million tasks. Compared with 85% recognition accuracy by using OCR devices, the game players achieved 99% accuracy for recognising the text (Chrons and Sundell, 2011).

2.2.4. Designing GWAPs

GWAPs can be an effective method to collect data. GWAPs are less expensive in the long term than other approaches for using human power to solve problems, such as Amazon Mechanical Turk. To design a GWAP, there are some suggestions according to the reviewed games (Von Ahn and Dabbish, 2008; Chamberlain et al., 2013). First of all, the key to developing such games is enjoyability. It is important that people enjoy playing the game and we obtain the data as a side effect. Furthermore, GWAPs attempt to solve computational problems which are divided into smaller tasks. Designing a successful GWAP relies on how to introduce those tasks without influencing the game-design mechanics (Chrons and Sundell, 2011). Engaging game elements can be added to increase player enjoyment such as time limit, score rewarding, rankings, level setting and so on. Moreover, it is essential to apply some evaluation metrics to GWAPs and make sure the obtained re-


Task1: Click any errors below

And aaccordingly the prsd the stuff
he could not see and declared
that h wwas delighted with both
colors and patterns Indeed please your
Imperial Majesty said he to his
sovereign wane hi returned the cloth
which the weavers are preparing is
extrordinarily mmagnificent Th whole city ...

Task2: Click two ciphers (error type) below which were used to encode the text:

Vowlel Killer: all lowercase vowels in the word have been deleted

Mouse over cipher to see its decription

Figure 1: A screenshot from the gameplay of *Cipher*

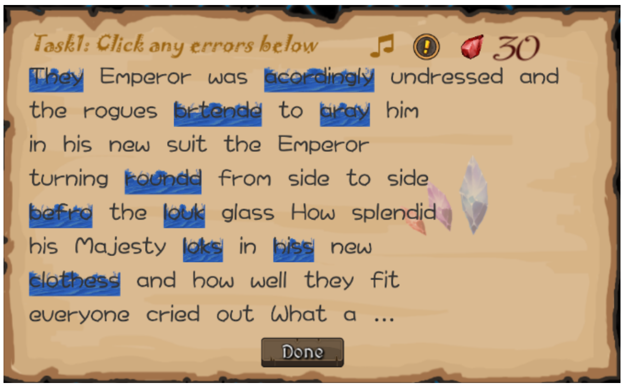
sults are correct. The performance of GWAPs can be evaluated by metrics, such as Cost per Acquisition, Monthly Active Users, and Average Lifetime Play (Chamberlain et al., 2017).

2.2.5. Summary

In the context of the *ESP Game* and *Peekaboom*, the object recognition competition between humans is to help computers improve the ability of image labelling. The model built for this project tries to solve the error detection problem with the help of humans. The idea of making annotations of text errors is inspired by *Phrase Detectives* which makes annotations of anaphoric coreference. The evaluation metrics that apply to *Phrase Detectives* are also used in the project, which help in optimising the data collection process. When compared to *Digitalkoot*, the game which detects OCR errors, our project has the potential for detecting context-based errors.

3. Methodology

How good are humans at identifying errors in text? This is the over-arching question for the project. In order to answer the question, artificial errors are generated in texts which players must detect and the average correction accuracy achieved by players used as a benchmark. The assumption is that humans can detect most errors in a given text and based on this assumption, the second research question is: *is it possible to build a game to encourage people to detect errors in text?* Errors commonly exist in printed materials and electronic documents, but humans can detect them and thus we build a game to identify errors in text with the help of game participants who are entertained to play the game while we collect useful data. We combine the two research questions and explore if a game can improve the performance at the correction task. When we analyse the data, we are also identifying errors we did not know. With more people playing the game, more errors can be identified and the error detecting performance improved overall.


Task1: Click any errors below 30

They Emperor was acordingly undressed and
the rogues brtende to aray him
in his new suit the Emperor
turning roundd from side to side
befro the louk glass How splendid
his Majesty loks in hiss new
clothess and how well they fit
everyone cried out What a ...

Figure 2: An example of cipher and common errors in a round of the game.

3.1. Cipher: A game to detect errors in text

Cipher is a single-player game that focuses on a text-based problem. Inspired by *Phrase Detectives*, the game is a text analysis game and collects data from players about annotation of words or phrases (Figure 1). *Cipher* encourages people to detect errors in text and was experimentally tested following a previously published strategy (Pearl and Steyvers, 2010) by inviting people to play the game offline to simulate an online GWAP. The participants include university students and friends of the authors, mostly in the 18-21 age range with English as a second language. The idea of steganography is used as an interesting game element for gamification. This turns the error detection task to cipher seeking. Ciphers create simple errors with certain features which would be found easily while more complex errors are also introduced as distractors for motivation, which would challenge players. The methodology for data collection and analysis combines qualitative and quantitative approaches. We focus not only data quantity, but also data quality, filtering the data according to players' performance and the number of players. It is also important to get players' opinions on the game as one of the essential evaluation metrics. By the experimental strategy, players are asked to play *Cipher* for at least 30 minutes. Their data are processed by filtering out data generated by those players whose correction accuracy and annotation accuracy are less than a threshold value.
There are also errors assumed to be the real errors in the text, we term as **unknown errors**, that must be distinguished from a mistake made by the player. The number of players who detect the same unknown error is also used as a filter to process the data.

3.2. Artificial error distractors

Automatically generated error data has been used to improve the performance of error detection and correction in previous studies. Artificial error data are easy to produce and can be applied to train and evaluate error detection systems, evaluate the robustness of Natural Language Processing techniques and be negative evidence in the form of automatically distorted sentences in unsupervised learning (Foster and Andersen, 2009). This approach was used in *Cipher* where, in each round, 10 artificial errors consist of 6 cipher-generated errors (3 similar errors generated by each cipher) and 4 common errors. Common errors are chosen from 3 common error corpora (Aspell, Birkbeck, and Wikipedia)[1] and then are introduced by replacing the correct counterparts in the game text. They can be non-word errors or real-word errors. Figure 2 is an example of errors in the text in the game. In the picture, "acordingly", "aray", "loks", "roundd", "hiss", and "clothess" are cipher errors. "They", "brtende", "befro", and "louk" are common errors. The cumulative correction accuracy (detected errors / round*10) of a player demonstrates how good the player is at error detection. Moreover, a total of 15 errors (5 in each article) were manually added into 3 articles from which the short text in each round is chosen from. They are used to evaluate the game's ability in error detection as a system on the assumption there were no genuine unknown errors in the articles.

3.3. Cipher mechanism

The idea of introducing cipher-generated errors is to turn the error checking task to cipher deducing, which is to make the serious game task entertaining for gamification purpose. To define ciphers, we make certain rules to alter the correct words in the text. In fact, these rules are the actual "ciphers". In each round, a piece of short text was encoded by two ciphers. Each cipher generates 3 similar errors, i.e., they have the same error feature (Figure 2). Ciphers that were used in the game were:

- **Vowel killer** All lowercase vowels in the word have been deleted (e.g., "th" from *the*, "hr" from *her*, "bk" from *book*);
- **Double head** The first letter has been repeated (e.g., "tthe" from *the*, "yyou" from *you*, "dday" from *day*);
- **Double tail** The last lowercase letter in the word has been repeated (e.g., "roundd" from *round*, "hiss" from *his*, and "clothess" from *clothes*);
- **Bottom up** The first letter in the word has been swapped with the last (e.g., "eids" from *side*, "mottob" from *bottom* , "tuis" from *suit*);
- **Single** One of two consecutive identical letters in the word has been deleted (e.g., "acordingly" from *accordingly*, "aray" from *array*, "loks" from *looks*);
- **Half half** The first half part of the word has been swapped with the rest (e.g., "itsu" from *suit*, "ndidsple" from *splendid*, "typem" from *empty*);
- **Reverse** The word has been reversed (e.g., "drow" from *word*, "hctam" from *match*, "thgil" from *light*).

As cipher-generated errors are rule-based and do not often occur in the real world, we also introduce errors from the common error corpora to simulate common typographical errors. There are 4 common errors (random error features)

[1]https://www.kaggle.com/bittlingmayer/spelling, accessed 13/2/2020

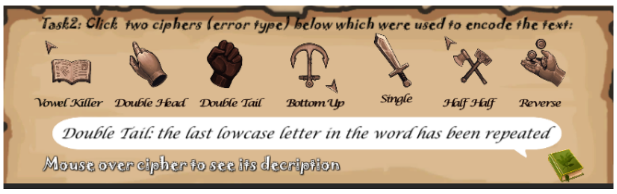

Task2: Click two ciphers (error type) below which were used to encode the text:
Vowel Killer
Double Head
Double Tail
Bottom Up
Single
Half Half
Reverse
Double Tail: the last lowcase letter in the word has been repeated
Mouse over cipher to see its decription

Figure 3: Cipher panel.

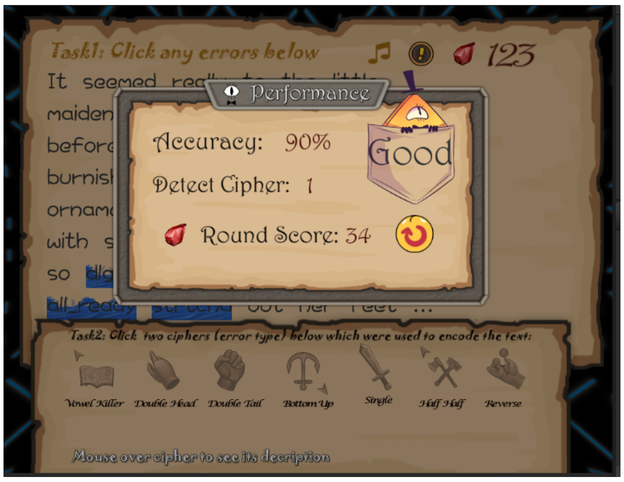

Task1: Click any errors below
123
Performance
Accuracy: 90%
Detect Cipher: 1
Round Score: 34
Good
Task2: Click two ciphers (error type) below which were used to encode the text:
Vowel Killer
Double Head
Double Tail
Bottom Up
Single
Half Half
Reverse
Mouse over cipher to see its decription

Figure 4: Player's performance panel.

besides 6 cipher errors making 10 **known errors** that have been introduced by the game per round.
Players have two tasks: At first, they need to find all the errors in the text and click them; Then they need to identify the ciphers according to the errors they have found and the descriptions of ciphers (Figure 3). The aim of the game is to find all the errors and then deduce the ciphers with according error features. Players' performance (correction accuracy, ciphers detected, and scores) was given at the end of the game after players confirmed and selected the ciphers according to the errors they found (Figure 4).
About text in the game, short text was randomly chosen from 3 articles in each round. In each article, there were 5 errors which have been manually added to determine the performance of the system. The 10 known errors in each game round were randomly generated by algorithms after the piece of text has been chosen and were intended to make the game fun and for appropriate rewards to be given to the players for correctly detecting an error. Although all texts were previewed by the authors before being used in the game, there may be some genuine errors in the text which was assessed in posthoc analysis.

4. Implementation

Cipher was developed in Unity game engine. Game graphics resources mostly come from Unity Asset Store[2] and the logo was a picture of the fictional character *Bill Cipher* from *Cleanpng*.[3] The implementation was divided into 4 parts: login & registration panel, text display panel, cipher panel and performance panel.
Login & registration panel When a player registered an account, a piece of player information data is created in the database. The players are required to login to the game so that their game information data can be updated in the database while they are playing the game.
Text display panel A short piece of text was randomly chosen from one of 3 xml documents. 10 errors are introduced into the text including 6 cipher errors and 4 common errors. More precisely, 4 correct words are replaced by 4 common mistaken words from common error corpus. There are 2 ciphers each round which encode the text by adding 6 errors randomly (3 errors with the same feature generated by each cipher algorithm). Finally, the modified text is shown to the players. Every word in the text is clickable.
Cipher panel A player needs to find all the errors in the given text by clicking them. When the player clicks an error, a piece of annotation data is created in the database. After the player finds all the errors and presses the "done" button, the cipher panel which contains 7 different ciphers is shown to the player. If the player finds the 2 ciphers , he will win the game and be rewarded with the score. The description of the cipher pops up when the mouse hovers over each cipher.
Performance panel The player is rewarded with 3 points for correctly identifying an error and 7 points for finding a cipher. After the player chooses 2 ciphers (whether correct or not), the result panel will pop out, which displays correction accuracy (correctly detected errors / 10), the number of ciphers found, and scores obtained in the round. Scores are accumulated each round. The player's information is updated in the database.

4.1. Data storage

Data collection is divided into three parts, stored in a table in MySql database. When a player detects and clicks an error, it is considered as an annotation. This annotation information includes the word Id, the name of the article where the word comes from, the correct form (plus wrong form as comparison if it is a game-introduced error) of the word, a Boolean flag representing if it is a known error ('Y' is Yes and 'N' is No), and the username of the player who clicked the word. This piece of annotation information is recorded in the table "annotation". The table "player" in the database has all the players' usernames and passwords. The table "playerinfo" stores the game information of the players including the scores the player has obtained, the number of rounds the player played, time the player spent in the game, the number of annotations the player made, the number of known errors the player detected, average correction accuracy (number of found errors / number of rounds*10), and annotation accuracy (number of found errors / number of annotations).

5. Results

The purpose of the game is to collect useful data while people are playing the game. The prototype game was tested by asking experimental participants to play the game. Participants were all unpaid volunteers and the experimental

[2]https://assetstore.unity.com/, accessed 13/2/2020.
[3]https://www.cleanpng.com/, accessed 13/2/2020.

Table 1: Unknown errors detected (CP=1, AA=0.2, and CA=0.2).

Word id	**Story**	**Word**	**# clicked players**
w473	Little match girl	Rischt	14
w482	Little match girl	burnt	4
w396	Emperor's new clothes	unft	3
w1942	Emperor's new clothes	exmating	2
w910	Little match girl	grandthern	2
w405	Swineherd	ill-humored	2
w570	Swineherd	cub	2
w115	Emperor's new clothes	wardrob	1
w823	Emperor's new clothes	atternd	1
w151	Swineherd	thad	1
w381	Swineherd	hummon	1
w708	Swineherd	mology	1
w1675	Swineherd	tecking	1

play time for each participant was at least 30 minutes. In 2 weeks, there were 35 participants who played the game and 25 of them played for more than 30 minutes. In total, players spent around 24 hours and 50 minutes playing the game. The game generated 4,764 pieces of annotation data, i.e., a click from the player who believes they have found an error, whether it is correct or not.

3 parameters were used as filters for the results: correction accuracy (number of found known errors by a player / number of rounds*10); annotation accuracy (number of found known errors by a player / total number of annotations from the player); and the number of clicked players (the number of players who made annotations on the same unknown word error). The difference between correction accuracy and annotation accuracy is that the former represents how accurate a player is at error correction while the latter shows how effective the player is at error correction.

While players were detecting errors, the game also recorded unknown errors. An unknown error can be either a genuine error in the text (or an error manually introduced into the text) or a mistake made by the player. The former is the target error (true positive) and the latter is considered as noise (false positive). Based on the collected data, we measure the variables: the number of true positives, the number of false positives, and thus recall and precision, were analysed by tuning 3 parameters: clicked player (CP), annotation accuracy (AA) and correction accuracy (CA). Initially, the values of all parameters were very low (CP=2, AA=0.2, and CA=0.2), which could be considered as no filter applied, because there was no player whose correction accuracy or annotation was below 20%. Then we changed each parameter gradually and observed the measure variables. Lastly, the relationships between parameters and measured variables were plotted to see which parameters were important for improving correction performance of the system.

Of the 35 experimental participants, 28 of them achieved correction accuracy of more than 70% and 29 of them achieved annotation accuracy of more than 70%. There were 20 players whose correction accuracy was more than 80% and 15 players whose annotation accuracy was more than 80%.

Based on the collected data, the number of true positives, the number of false positives, recall, and precision were analysed with tuning 3 parameters: clicked player (CP), annotation accuracy (AA) and correction accuracy (CA). With parameters (CP, AA and CA) tuning, the relationship between the filters and the measure variables (number of target error and noise, recall and precision) were plotted in Figure 5 - 10. When the number of clicks from players is used as a filter we observe that noise is effectively reduced with filter at 4 player clicks, similar to the findings in the validation analysis in *Phrase Detectives* (Chamberlain et al., 2018), see Figure 5. We also observe an increase in precision when more player clicks are used but a lower recall due to the low number of players in the game experiment, see Figure 6. A similar effect is observed with the reduction of noise by using annotation and correction accuracy as a filter, i.e., by increasing the requirement for player ability, the number of incorrect judgements is reduced, see Figures 7 and 9. Likewise, we observe recall significantly drop when annotation and correction accuracy is used as a filter due to player exclusion and the low number of players in the experiment, see Figures 8 and 10.

Table 1 shows the unknown error detected results when the values of all parameters were very low, which can be considered as no filter applied. In this case, the values for CP, AA, and CA are 1, 0.2, and 0.2 respectively. 13 unknown errors were detected by players. 10 of them are manually added errors: "unft" (unfit), "grandthern" (grandmother), "exmating" (examining), "cub" (cap), "wardrob" (wardrobe), "atternd" (pattern), "thad" (that), "hummon" (humor), "mology" (melody) and "tecking" (taking). In total, there were 15 (5 in each story) manually introduced errors (66.6% detection rate). All 5 added errors in the story *Swineherd* were detected. Three unknown errors not introduced into the texts but were detected by players include:

> "Rischt", or "R-r-ratch" in a different version of the story *Little match girl*, represents the sound of striking a match;
>
> "burnt" is past tense in American English;
>
> "ill-humored" is commonly used in American English.

In addition, old English words such as "hitherto", "Fie" and "swineherd" were also detected, which are rarely used nowadays (noted by player 3, 4 and 13 respectively.

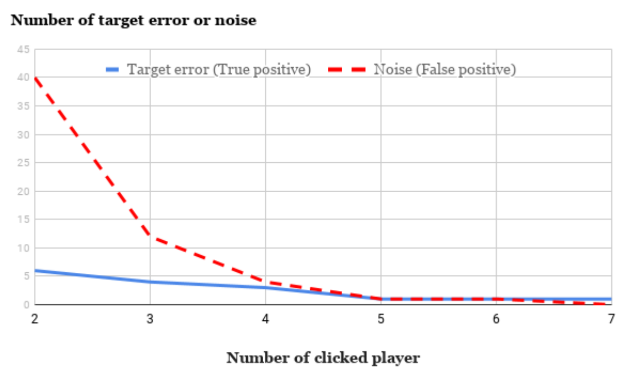


Figure 5: The number of target error and noise change when the number of clicked player is the filter.

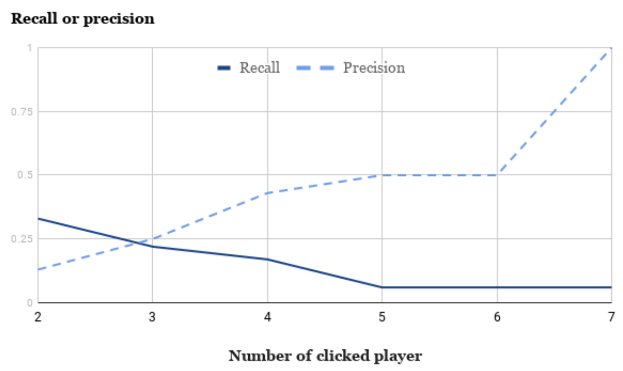


Figure 6: Recall and precision change when the number of clicked player is the filter.

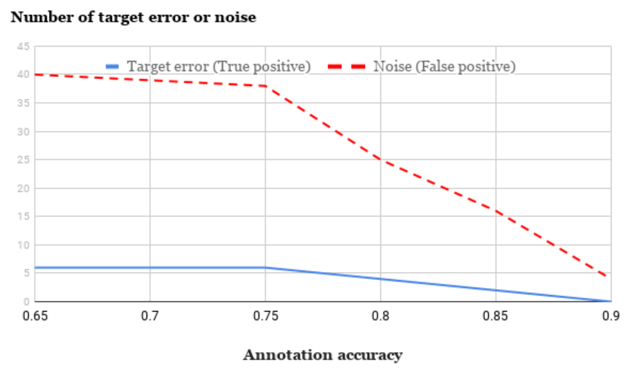


Figure 7: The number of target error and noise change when annotation accuracy is the filter.

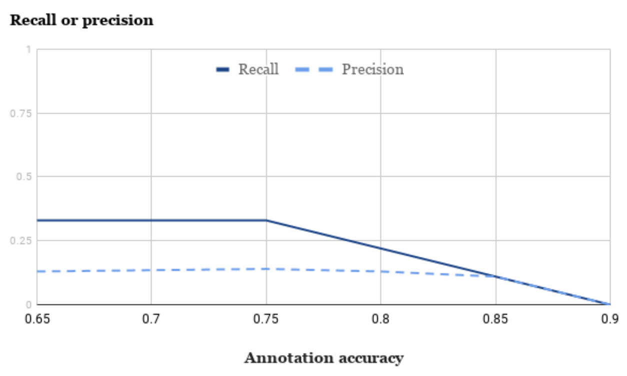


Figure 8: Recall and precision values with tuning parameter annotation accuracy.

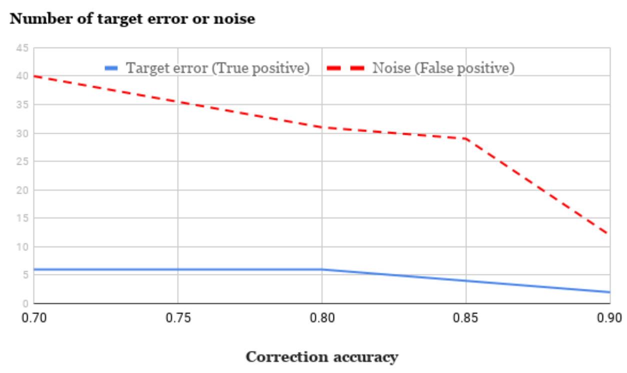


Figure 9: Number of target error and noise change when correction accuracy is the filter.

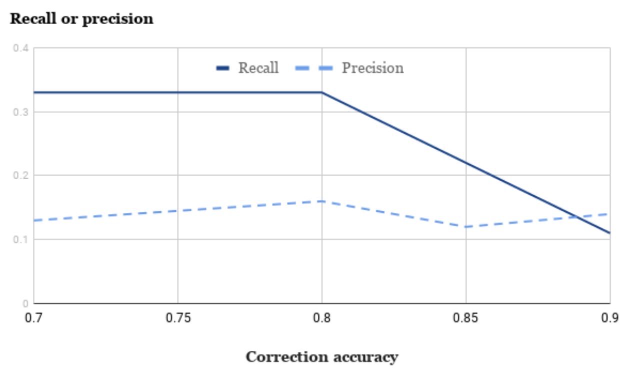


Figure 10: Recall and precision values with tuning parameter correction accuracy.

6. Discussion

Correction accuracy represents how accurate each player is in word correction. By calculating correction accuracy of each player, we can answer the first research question. 29 of 35 players' correction accuracy is higher than 70% and 20 players achieved more than 80%. 3 players reached more than 90% correction accuracy. These values suggested high performance of humans in error detection.

Initially we set the values of the 3 parameters, clicked player (CP), annotation accuracy (AA), and correction accuracy (CA) 2, 0.2 and 0.2 respectively in case there was too much noise (false positives). From the results, "unft", "grandthern" and "exmating" (manually added errors) were detected by players. Furthermore, there were some errors

which we did not introduce into the documents. “Rischt” is the noise of striking a match in the context and its correct form is “R-r-ratch” in a version of the original text. The latter makes more sense in the context. We found that some players got confused between “burnt” and “burned” and “humored” and “humoured”. In fact, “burnt” and “humor” are preferred in American English, therefore, some players considered them as errors. Players also detected that the types of English for some words used in the documents were not standard English because they annotated some old English words such as “Fie” (meaning “for shame” in other versions of the text), “hitherto” and “swineherd”, which are uncommon. Figure 11 indicates the frequencies of the 3 old English words mentioned over time.
We did not know these errors until we looked at the data from the game. When players were finding errors in the text, they did find genuine errors and detected some problems we did not previously know.
We wanted to know if the game was engaging for players to encourage them to help us solve the problem. For the experiments, participants needed to play for over 30 minutes; however, many players played for more than 1 hour because they enjoyed the game. Verbatim comments from participants during the experiments include:

> “This game is a bit addictive. I really would like to play the game if it is released online.”
>
> “It is an interesting game. Moreover, this game has some educational meaning. I think it will be really helpful for school students to play this type of game.”
>
> “The game UI combined with the ear worm of the background music makes the text-based game a bit fun.”

When evaluating the effectiveness of the system in detecting errors, we used 3 parameters (CP, AA, and CA) to try to improve the performance. From the graphs (Figure 5 – 10), we found precision overall would ascend but would fall to 0 if the parameter reached a certain value. As the parameter value goes up, the system keeps better players while eliminating bad players. There were fewer answers gradually, but the obtained results were more likely to be true positives. However, when the parameter value was too high, all players were excluded, which causes the decline of precision. When it comes to recall, it was always declining with each parameter increasing. The result explained that noise was generated while players were finding true positives. If we would like to get more true positives, we would get more noise as well. In conclusion, the performance of the system in word correction depends on how we tune the parameters.
There are some limitations with the project. The game was running offline on a small scale. The number of people who tested the game was sufficient for a prototype test but more would be needed for large-scale data collection. Enough participants played the game and created useful data which allowed us to explore the documents and find out new information. Furthermore, this is an English language game

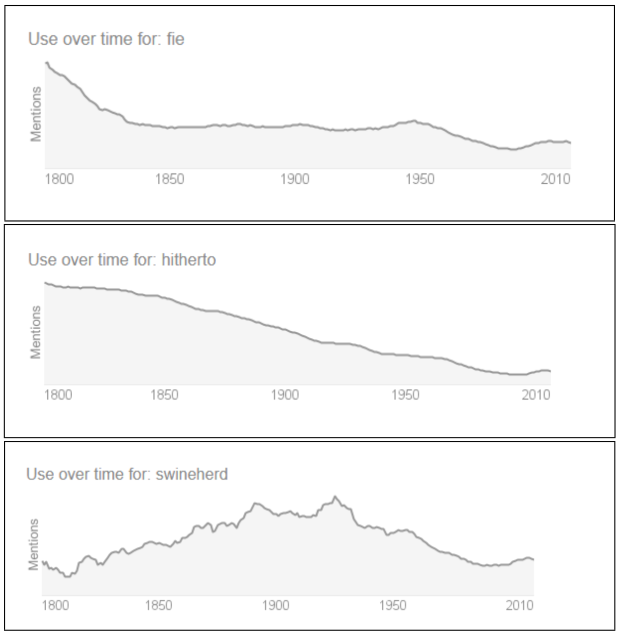


Figure 11: “fie”, “hitherto” and “swineherd” used over time from Google dictionary.

but most participants were English learners rather than native speakers. This is partly because English learners are more interested in this language game for the motivation of practicing their English skills while playing the game and partly because of the limited number of participants. Therefore, there was more noise (false positives) detected, which influences the detection results. However, we defined filters to improve the outcomes. Even though most of the participants were non-native speakers, they still achieved high performance in error detection and found genuine errors and problems which we did not know.

7. Conclusion

It is common that some errors are found in publication materials and electronic documents. Existing commercial spelling-checking applications struggle to detect complicated text errors and it is expensive to employ humans to find errors. In this paper, we described a GWAP methodology for error detection. We found that people are able to easily identify errors in text and they were encouraged to do the tasks by playing an enjoyable game. Several genuine errors were detected, indicating the GWAP approach is useful to identify novel errors in text already checked by a proofreader. Parameters such as clicked players (number of players who detected the error), correction accuracy (detected errors / rounds*10), and annotation accuracy (detected errors / the total number of annotations) can be used to measure game performance. In addition, we found that the game has the potential for helping language learners. Participants reported that they enjoyed the game and found the unusual language interesting. A GWAP approach to error detection and correction would be useful as a support tool for OCR software, or as part of a wider pipeline looking to build fully corrected and annotated documents for large, collaboratively-produced language resources.

8. Bibliographical References

Bledsoe, W. W. and Browning, I. (1959). Pattern recognition and reading by machine. In *Papers presented at the December 1-3, 1959, eastern joint IRE-AIEE-ACM computer conference*, pages 225–232.

Chamberlain, J., Poesio, M., and Kruschwitz, U. (2008). Phrase detectives: A web-based collaborative annotation game. In *Proceedings of the International Conference on Semantic Systems (I-Semantics' 08)*, pages 42–49.

Chamberlain, J., Kruschwitz, U., and Poesio, M. (2013). Methods for engaging and evaluating users of human computation systems. In *Handbook of Human Computation*. Springer.

Chamberlain, J., Bartle, R., Kruschwitz, U., Madge, C., Poesio, M., et al. (2017). Metrics of games-with-a-purpose for nlp applications.

Chamberlain, J., Kruschwitz, U., and Poesio, M. (2018). Optimising crowdsourcing efficiency: Amplifying human computation with validation. *it - Information Technology*, 60:41–49.

Chrons, O. and Sundell, S. (2011). Digitalkoot: Making old archives accessible using crowdsourcing. In *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Foster, J. and Andersen, O. (2009). Generrate: generating errors for use in grammatical error detection. In *Proceedings of the fourth workshop on innovative use of nlp for building educational applications*, pages 82–90.

Knuth, D. E. (1973). The art of computer programming, vol. 3, addison-wesley. *Reading, MASS*.

Kukich, K. (1992). Techniques for automatically correcting words in text. *Acm Computing Surveys (CSUR)*, 24(4):377–439.

Lafourcade, M., Joubert, A., and Le Brun, N. (2015). *Games with a Purpose (GWAPS)*. John Wiley & Sons.

Mishra, R. and Kaur, N. (2013). A survey of spelling error detection and correction techniques. *International Journal of Computer Trends and Technology*, 4(3):372–374.

Mitton, R. (1987). Spelling checkers, spelling correctors and the misspellings of poor spellers. *Information processing & management*, 23(5):495–505.

Odell, M. K. and Russell, R. (1918). Patent numbers 1261167 (1918) and 1435663 (1922). *Washington, DC: US Patent Office*.

Pearl, L. and Steyvers, M. (2010). Identifying emotions, intentions, and attitudes in text using a game with a purpose. In *Proceedings of the naacl hlt 2010 workshop on computational approaches to analysis and generation of emotion in text*, pages 71–79. Association for Computational Linguistics.

Quinn, A. J. and Bederson, B. B. (2011). Human computation: A survey and taxonomy of a growing field. In *Proceedings of the 2011 SIGCHI Conference on Human Factors in Computing Systems (CHI'11)*, pages 1403–1412.

Riseman, E. M. and Hanson, A. R. (1974). A contextual postprocessing system for error correction using binary n-grams. *IEEE Transactions on Computers*, (5):480–493.

Rumelhart, D., Hinton, G., and Williams, R. (1986). Learning internal representations by error propagation in parallel distributed processing", de rumelhart, jl mcclelland eds.

Von Ahn, L. and Dabbish, L. (2008). Designing games with a purpose. *Communications of the ACM*, 51(8):58–67.

Von Ahn, L., Blum, M., Hopper, N. J., and Langford, J. (2003). Captcha: Using hard ai problems for security. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 294–311. Springer.

Von Ahn, L., Liu, R., and Blum, M. (2006). Peekaboom: a game for locating objects in images. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 55–64. ACM.

Von Ahn, L. (2006). Games with a purpose. *Computer*, 39(6):92–94.

Wagner, R. A. (1974). Order-n correction for regular languages. *Communications of the ACM*, 17(5):265–268.

Yannakoudakis, E. J. and Fawthrop, D. (1983). The rules of spelling errors. *Information Processing & Management*, 19(2):87–99.