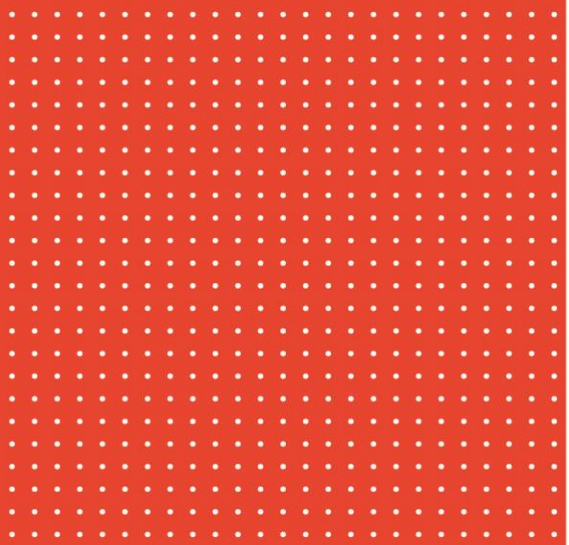
Language Cert



Similarity Detection in Writing Test Scripts at LanguageCert

David Coniam
and
Vlasis Megaritis

Abstract

This paper explores the issues surrounding plagiarism, a form of malpractice defined as cheating by collusion, by copying, by memorisation or by using previous candidates' work in LanguageCert Writing Tests. The paper first provides an overview of the area with a discussion of how and why plagiarism is becoming more of a problem in this digital age, and a categorisation of the different types of plagiarism that are prevalent. An overview of statistical and computational methods used to detect similarity in texts follows, together with a brief description of some of the most common tools to detect similarity in texts.

The paper then describes LanguageCert's similarity detection tool SiD, which has been developed by PeopleCert for focused in-house scrutiny of all incoming scripts. To illustrate how SiD operates, and to provide a snapshot of the metric for determining similarity, exemplars of similarity at different levels of severity are then provided.

In 2023, a corpus was created of all computer-delivered LanguageCert examination Writing Test scripts dating back to 2020. All computer-delivered Writing Test scripts are now passed through SiD, which examines them for similarity against the background corpus, as well as continually expanding the corpus in real time. All scripts, above a predetermined threshold of similarity, are scrutinised in order to determine whether malpractice has taken place.

The Writing Test similarity detector is just one of the tools in LanguageCert's toolbox by which it ensures fairness and integrity in its examinations.

Keywords: similarity detection, cheating, writing tests, cosine similarity algorithm, Myers O(ND) algorithm

Background to Cheating with Particular Reference to Plagiarism

Cheating in examinations, including English language examinations, is a significant issue not only in academia but in classrooms around the world. With the English language becoming increasingly important for global communication, qualifications and visas for work and study purposes, the issue of cheating in English language examinations has come very much to the attention of assessment bodies and regulators. A brief overview of the literature on cheating and in particular plagiarism in English language examinations follows.

Many studies have investigated the issue of cheating in examinations. Whitley (1998) in his review of over 100 studies reported a number of reasons why students cheat on exams ranging from the importance of success, to the need for approval, to expected performance.

In a more recent large-scale survey, McCabe et al. (2012) reported that approximately 64% of students admitted to cheating on tests, while 58% admitted to some form of plagiarism. So'ud (2016) in a study of college of education students in the Sudan, reported that 100% of their interviewees admitted – for various reasons – to cheating in English language examinations. Wan and Li (2006) reported more than 60% of college students cheated at times, and about 10% cheated in examinations.

Using a variety of evidential sources, Huang & Garner (2009) reported a comparatively high level of cheating on the College English Test.

Digital content has seen a massive growth in recent years, and the internet has undoubtedly contributed to the prevalence of cheating in English language examinations. The ease of access to information and the ability to copy and paste from the internet has made it easier for students to be able to cheat (Noorbehbahani et al.,2022). While it has been argued that many students may not understand the concept of plagiarism and may not be aware of the consequences (Park, 2003), the fact is that cheating on examinations, on English language examinations, and on high-stakes English language examinations in particular, is at an all-time high, and on the increase (Iqbal et al., 2021).

To guard against cheating and malpractice, LanguageCert has a rigorous set of test security principles related to online-delivered assessments (see: https://passport.peoplecert.org/docs/OLP_Exams_Candidate_Guidelines_Windows.pdf). Many of the security features echo those presented in Foster's (2013). To exemplify, upon first log-on, candidates need to follow a thorough 'onboarding' process; this includes an ID check, locking down their computer, checking there are no second monitors, and a room check through their webcam to show that the room is secure and that no other person or aids are present (see Coniam et al., 2021).

Types of Plagiarism

An array of different types of plagiarism are reported, both intentional and unintentional; see e.g., Bin-Habtoor & Zaher, 2012; Chowdhury & Bhattacharyya, 2018; Maurer, 2006.

These different types of plagiarism are summarised below.

1. *Copy-and-paste plagiarism*. This is when a writer copies text from a source and pastes it into their own work without giving credit to the original author.
2. *Verbatim plagiarism.* This is when a writer copies text from a source word-for-word without giving credit to the original author.
3. *Paraphrasing plagiarism.* This occurs when a writer rephrases ideas or words without giving credit to the original author.
4. *Self-plagiarism.* This happens when a writer submits work that they have previously published without indicating that it has been published before.
5. *Mosaic plagiarism*. This is when a writer uses a combination of copied and original material in their work without properly citing the copied material.
6. *Accidental plagiarism*. This occurs when a writer inadvertently uses someone else's work or ideas without realising it, often due to a lack of understanding of proper citation practices.
7. *Structural Plagiarism*. This involves taking another person's ideas, sequence of arguments, selection of quotations from other sources, or even the footnotes that may have been used without giving due credit. Such plagiarism is not always easy to identify, as both texts have to be carefully scrutinised to identify similarities.

In the context of English language exams, types 1-3 are likely to be most prevalent and memorization is likely to play a role.

Statistical and Computational Methods Used to Detect Similarity

Over the past two decades, a considerable number of methods – which have also resulted in the development of an array of different plagiarism-checking tools – have been developed to identify similarity, or plagiarism; see e.g., Bin-Habtoor & Zaher, 2012; Chowdhury & Bhattacharyya, 2018; Maurer, 2006, who report on, review and evaluate such methods and tools.

A broad summary of the key methods is listed below.

1. *Linguistic analysis*. In this method, the language used in a text is analysed to identify patterns or characteristics – involving possible inconsistencies in writing style, vocabulary, and grammar – that may be suggestive of plagiarism (Pecorari, 2008).
2. *Database comparison*. In this method, texts are compared to a database of existing documents to identify matches or similarities. The database may be populated with previously published works, student papers, or any other relevant text that may be used for comparison (Si, et al., 1997).
3. *Citation analysis*. In this more academically-grounded method, citations in a text are analysed to determine if they are properly formatted and if they refer to valid sources. Citation analysis can also detect cases of self-plagiarism, where a writer submits work that they have previously published without proper citation (Mazov et al., 2016).
4. *Stylometric analysis*. In this method, the writing style of a text is analysed with a view to identifying patterns or characteristics – through changes in writing style or vocabulary – that may indicate plagiarism (Stein, et al., 2011).
5. *Machine learning*. In this method, machine learning algorithms are trained to detect plagiarism by analysing patterns and similarities in text. These algorithms use statistical models to identify similarities between documents and can be trained to recognize specific patterns or characteristics of plagiarism (Hunt et al., 2019).
6. *Text similarity analysis*. In this method, the text in two or more documents are compared to determine their level of similarity via algorithms which do string comparisons invoking mathematical functions. Among such algorithms are the Rabin-Karp and Jaro-Winkler distance algorithms (Leonardo and Hansun, 2017); the Levenshtein distance algorithm (Su et al., 2008); and the Smith-Waterman algorithm (Irving, 2004). Analysis is grounded on the basis that plagiarised text is likely to be similar or identical to the original source, with the algorithms producing output which reports the degree of similarity (see Vijaymeena & Kavitha (2016) for a summary of common algorithms).

As will be apparent from the detail presented below on the LanguageCert similarity detection tool, the approach adopted by LanguageCert, may be seen to be placed under method 6 above: text similarity analysis.

Similarity Detection Software Tools

As with other methods of detection, a number of software tools using different statistical and computational methods have been developed in an attempt to identify similarity, or plagiarism, in texts.

Bin-Habtoor & Zaher (2012) list 15 plagiarism detection tools. Naik et al. (2015) list over 30 tools. Heres & Hage (2017) compare nine tools. Chowdhury & Bhattacharyya (2018) present a survey of 31 tools, although they do not evaluate them. Mansoor & Al-Tamimi (2022) report on over 12 tools.

Summarising some of the various sources mentioned above, some of the key current software tools for detecting plagiarism are:

Turnitin is one of the most widely used pieces of similarity detection software (e.g., Meo & Talha, 2019). It uses a database of published works, student papers, and other sources to compare submitted documents for similarity. Turnitin provides a similarity score and highlights potential instances of plagiarism.

Plagiarism Detector X is a desktop application that can scan text documents for plagiarism. It uses a variety of algorithms, including text similarity analysis and database comparison, to detect plagiarism.

Grammarly is a popular writing assistant tool that can detect potential instances of plagiarism. It uses machine learning algorithms to analyse text and identify similarities to other documents.

Copyscape is an online tool that can scan web pages for plagiarism. It compares submitted text to a database of indexed web pages to identify potential instances of plagiarism.

Ephorus is a plagiarism detection tool used by educational institutions. It uses text similarity analysis to compare submitted documents to a database of published works and student papers.

Urkund is a plagiarism detection tool that can scan text documents for plagiarism. It uses a combination of text similarity analysis, database comparison, and citation analysis to identify potential instances of plagiarism.

SafeAssign is a plagiarism detection tool integrated into the Blackboard learning management system. It compares submitted documents to a database of published works, student papers, and other sources to identify potential instances of plagiarism.

The conclusion as to which software program or online tool is the best for detecting plagiarism depends on several factors, including the type of document being analysed, the type of plagiarism being investigated, and the resources available for analysis. Different tools use different algorithms and techniques to detect plagiarism, and their effectiveness can vary depending on the specific situation.

LanguageCert has developed tools to automatically check the written text responses produced by candidates taking its English language exams. As an international English language exam board, operating all over the world and in different time zones the scope for cheating is significant. It is worth reiterating that checking written text is only one of the checks that need to take place to guard against cheating.

Plagiarism in English language exams may take a number of forms, as mentioned above. A serious form of plagiarism, or cheating, that LanguageCert needs to detect involves essays which are significantly similar if not identical being submitted by different candidates.

The LanguageCert focus rests initially on an in-house solution, relevant to scripts produced for LanguageCert tests, in response to set prompts. Against this backdrop, the in-house similarity detector, SiD, has been developed which rates all input scripts for similarity against an existing corpus of past candidate scripts. The section below briefly outlines the LanguageCert tool.

Background to Exploring Similarity in Texts

While the thrust of the current paper involves a broad picture of the development and operation of the LanguageCert similarity detector, some background technical detail is necessary. This section outlines, in lay terms as far as possible, some of the programming detail which underpins the operation of the tool.

The majority of the coding conducted in-house has been done in Python. This is an open-source computer programming language which consists of open-source libraries. Various of these libraries have been drawn upon in the three procedures outlined below.

In analysing candidate scripts with a view to detecting similarity, the LanguageCert *Similarity Detector – SiD* – involves three core procedures. These are:

1. Vectorisation, i.e., converting the words in a text into numbers.
2. Measuring the similarity between vectors.
3. Qualitatively examining the output by highlighting similarities and differences between pairs of texts.

Procedures (1) and (2) form the core of the analysis. Procedure (3) can be viewed as the front end, where visualisations of similarities between scripts are presented to the end user. The sections below outline these procedures in the context of current implementations. Following this, procedures directly relevant to the construction and operation of the LanguageCert tool SiD are provided.

Text Vectorisation

Before any comparison of texts may be conducted, the words in all texts need to be vectorised; that is, the words need to be converted into numerical representations which a software program can then meaningfully analyse. Egger (2022) presents a summary of different word (or "term-based") vectorisation techniques, with some of the most well-known described below.

The *Term Frequency - Inverse Dense Frequency* (TF-IDF) technique computes the importance a word in a document or corpus by comparing the frequency of the word in the document to its frequency across the entire corpus. It does not directly capture the meaning of the word, as it only takes into account its occurrence in the document or corpus (Ramos, 2003; Wang et al., 2020).

The *Hashing Vectorizer* is a vectorisation technique that is commonly used in natural language processing. It works by generating a fixed-length numerical representation of text data using a hashing function. Unlike other vectorisation techniques such as TF-IDF, it does not require the building of a dictionary or vocabulary (Idouglid and Tkatek, 2023).

Word2Vec is a predictive neural-based word embedding model that learns to represent words in a continuous vector space based on their contextual usage in a large corpus of text (Mikolov et al., 2013).

One of the most frequently used vectorisation techniques is the TF-IDF technique referred to above (Ramos, 2003); it is this procedure that is used in the LanguageCert tool. TF-IDF was chosen because of its simplicity, its interpretability and its scalability.

Measuring Similarity Between Vectors

Once words have been vectorised, an algorithm is then required to measure the similarity between vectors. Some of the most common algorithms are outlined below.

The *Cosine Similarity* method measures the level of similarity between two vectors. It does this by calculating the cosine value of the angle between the two vectors, where the vectors are numerical representations of words in a document or a corpus (Connor, 2016).

The Manhattan Distance method computes the sum of the absolute differences or the absolute values of the differences between the corresponding dimensions or coordinates of the two points (Eugene, 1987).

The Jaccard similarity coefficient computes the relationship between words in two strings in terms of which words are shared and which are distinct (Diana and Ulfa, 2019).

The Dice coefficient defines the relationship between words in two strings as two times the number of terms which are common in the compared strings, divided by the total number of terms present in both strings (Küppers and Conrad, 2012).

Different researchers advocate different algorithms but the method adopted by LanguageCert in its similarity detector is the Cosine Similarity method. The method was selected because it has been referred to as "standard" in similarity detection (Connor, 2016), and has been proven to be robust by a number of researchers (Saptono et al., 2018; Indriyanto and Sumitra, 2019; Davoodifard, 2022).

Identifying and Highlighting Differences in Scripts

Having measured the similarity between two vectors, the final step finding the differences or similarities between two pieces of text and highlighting the changes. This is the front end which is presented to users. Some of the most common difference (or 'diff') algorithms which do this are outlined below.

The Myers O(ND) difference algorithm works with textual strings. It calculates the best "diffeprence" between two strings, which means finding the most concise sequence of 'edits' or changes to each text, such that string 1 is converted into string 2 (Myers, 1986).

Patience and Histogram algorithms enhance the Myers algorithm in certain ways to improve efficiency or performance (see Nugroho et al., 2020).

The Bentley-McIlroy algorithm operates using blocks of characters rather than single characters, as in Myers' algorithm (see Chang et al., 2008).

Although developed 40 years ago and enhanced over time (see e.g., Sjölund, 2021), Myers' algorithm is still widely used as a general-purpose difference detection tool, and is used to highlight the differences between two scripts. It is this algorithm, surrounded by a layer of pre-diff speedups and post-diff cleanups, that the LanguageCert similarity detector currently uses.

The LanguageCert Similarity Detector *SiD*

As outlined above, the LanguageCert similarity detector (SiD) has been built following, to a large extent, well-researched best practice. Scripts input to the system are first converted into numbers using the *TF-IDF* technique. The *Cosine Similarity* algorithm is then invoked, which measures the level of similarity by calculating the cosine value between the two vectors. Myers' algorithm is then used to calculate and highlight the differences between two scripts.

The principal difference between the Cosine Similarity method and the Myers algorithm is that the Cosine Similarity is a measure of similarity between two texts which are represented as vectors without considering the relative position of words in these texts. Myers' algorithm is the front end which identifies the smallest set of insertions and deletions needed to 'transform' one sequence into the other. Appendix 1 outlines how texts which have a very high similarity score may be seen to appear qualitatively different in appearance.

The three-step operation outlined above represents the current operational state of the similarity detector. Following implementation and feedback from end users, it may be the case that the final procedure – highlighting textual similarities – may be performed by an algorithm other than the Myers', which is the procedure currently being implemented. The core operations performed by the *TF-IDF* technique and the *Cosine Similarity* algorithm which define the similarity score however, will not change.

The following section outlines the operation of the LanguageCert similarity detector, SiD.

SiD in Practice

The system described below was implemented in 2022, and operationally affects all scripts coming in to the system on an ongoing, daily, basis.

A subcorpus exists for each prompt at each CEFR level. As new Writing Test prompts are created – which happens on a frequent and regular basis – new subcorpora are created to accompany the new prompts.

As scripts are input to the system, they are sorted and allocated to a specific subcorpus on the basis of CEFR level and question, i.e., prompt. All candidates have a unique identifier, so multiple takes of an examination, even responses to the same prompt, can be identified and traced.

A script which enters the appropriate subcorpus is then compared against every script that exists in the subcorpus. This means that any given script will be compared against thousands of other scripts, with a similarity score (derived from the Cosine Similarity algorithm) calculated for every script analysed.

Interpreting SiD's Output

This section presents examples of scripts from different candidates outlining degrees of similarity at various percentiles. Appendix 1 provides examples of how similar, yet apparently different, texts appear as pairs of scripts. Actual output will contain two scripts (the left-hand script being "Script 1" and the right-hand script "Script 2") presented horizontally, side by side. Any given pair of scripts need to be viewed in the context of Script 2 being 'derived' from Script 1, with coloured highlighting as outlined below.

- Green text in Script 2 indicates text which appears in Script 2 but not in Script 1.
- Red text in Script 1 indicates text which when 'deleted' from Script 1 may be seen to result in the text observed in Script 2. This may involve words and phrases in Script 1 being 'left out', 'substituted' or 're-arranged' in Script 2.
- White text indicates text which is the same in both scripts. The more white text there is, the more similar the two scripts will tend to be.

Consider Figure 1 below which is extracted from Figure 4 further down this section. The figure contains the first line from two high-similarity scripts. Apart from some minor differences, the two lines will be seen to be almost exactly the same.

For readability's sake, the lines have been enlarged, with Script 2 appearing beneath Script 1.

Figure 1: One comparable line from two similar texts

Script 1	Hey Johnsons, I hope this letter finds you well. I was thrilled to hear that you have
Script 2	Dear Johnson, I hope this letter finds you well. I was thrilled to here that you hav

The first word in Script 1 is “Hey”; in Script 2, it is “Dear”.

The second word in Script 1 is “Johnsons”. In Script 2, the second word is “Johnson”, the “s” on the word “Johnson” in Script 2 having been ‘removed’.

Further along the line, “hear” in Script 1 appears as “here” in Script 2.

The preponderance of white text in the two extracts in Figure 1 underscores the high similarity between the two texts.

Appendix 1, as mentioned, provides examples of what similar, yet apparently different, texts might look like in terms of ‘deletions’ in one text (Script 1) and ‘insertions’ in another (Script 2).

As Appendix 1 illustrates, two scripts can visually contain a comparatively large amount of red and green highlights (indicating potential differences) alongside a very high similarity score. Therefore, because of the large number of differences, generally, in the context of an examiner scrutinising two scripts which have an extremely high similarity score (above 0.9, say), the more red and green text there is, and the less white text, the lower will be the degree of similarity between the two scripts, and the less likelihood of cheating having occurred. For an examiner looking at two scripts with a preponderance of white text, a warning sign of potential malpractice is flagged, and the scripts concerned are then forwarded for more detailed investigation.

To give a flavour of the procedure in practice, and the type of output provided to a scrutinising examiner, some exemplar pairs of scripts exhibiting different levels of similarity are presented below.

One issue regarding the occurrence of similarity rests on the extent to which candidates reuse, or incorporate, detail from the prompt. Such recycling of given text is very much the case at lower CEFR levels (A1 to B1). Although recycling is less prevalent at B2 and above, a certain amount of reuse of given words and phrases still exists.

Figure 2 presents two scripts with a 0.90 similarity.

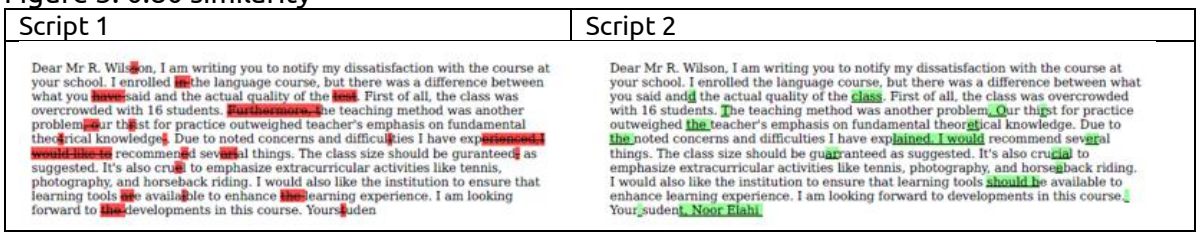
Figure 2: 0.90 similarity

Script 1	Script 2
Dear Sir, I am writing to tell you about my experience with language classes I attended last month. I have to travel to Islamabad to attend the classes. When I arrived at Lahore airport, all my excitement faded because no one was there for me. After three hours, someone came and drove me to the language centre's accommodations. I attempted to sleep when I got to the hotel, but the noise from traffic and the power interruptions prevented me from doing so. My language class experience was also unpleasant as I was having difficulty understanding my teacher. My morning class timings helped me considerably to explore the sights in the afternoon. However, there were no weekend excursions, which disappointed me. I hope these points will be valuable to students in future batches. Your student, Naqash Haider	Dear Sir, I am writing to tell you about my experience with the language classes I attended last month. I have to travel to Karachi to attend the classes. When I arrived at Karachi airport, all my excitement faded because no one was there for me. After four hours, someon reached and drove me to the language center's accommodations. I attempted to sleep when I got to the hotel, butthe noise from traffic and power interruptions prevented me from doing so. My language class experience was also unpleasant. As I was having difficulty understanding my teacher. My morning class timings helped me considerably to explore the sights in the afternoon. However, there were no weekend excursions, which disappointed me. I hope these points will be valuable to students in future batches. Your student, Muhammad Talha

Apart from minimal changes such as place and person names and a couple of other minor differences, Script 2 is essentially the same as Script 1.

Figure 3 presents two scripts with 0.80 similarity.

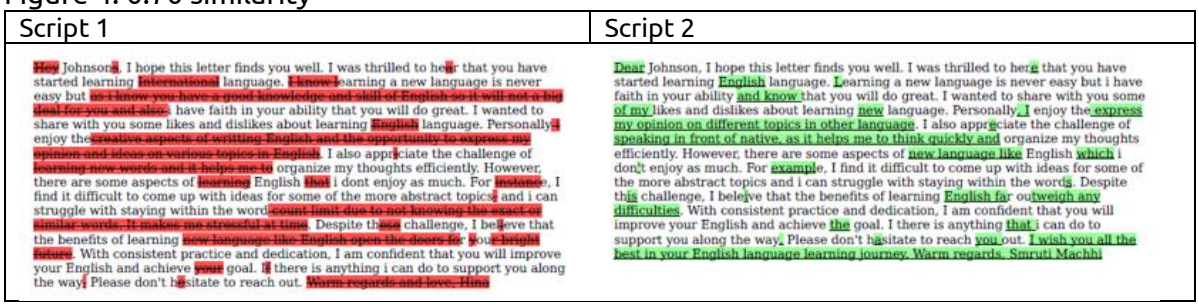
Figure 3: 0.80 similarity



While there is a high degree of similarity between the two scripts, there are notable differences.

Figure 4 presents two scripts with 0.70 similarity.

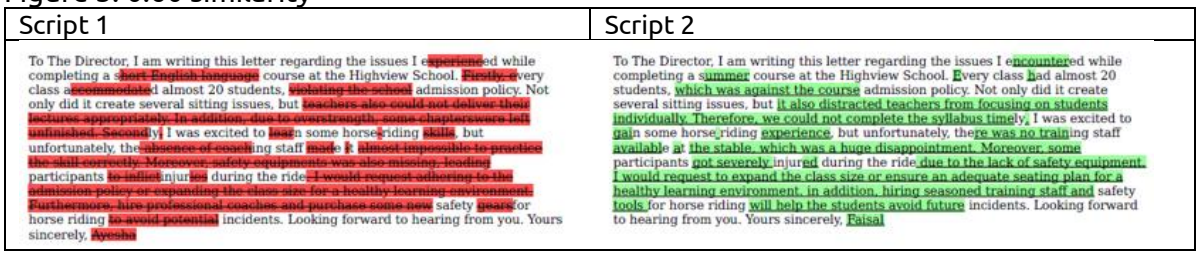
Figure 4: 0.70 similarity



The broad structure of the two scripts is comparable and hence the comparatively high degree of similarity at 0.70. There is more originality in Script 2, however, compared with the two 0.80 similarity scripts above.

Figure 5 presents two scripts with 0.60 similarity.

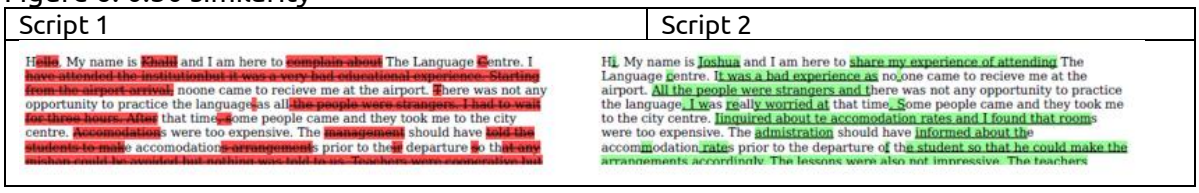
Figure 5: 0.60 similarity



At 0.60 similarity, the greater degree of originality in Script 2 is becoming apparent. There is much less white text.

Figure 6 presents two scripts with 0.50 similarity.

Figure 6: 0.50 similarity



At 0.50 similarity, the degree of difference between the two scripts extends, although there is still some similarity – as in the two scripts above due in part to the recycling of words from the prompt.

Conclusion

This paper has presented a picture of how LanguageCert approaches and engages with the issue of similarity – potential cheating – in LanguageCert Writing Tests. The paper has outlined the working of the LanguageCert similarity detector SiD in terms of how the system processes scripts within the system, and the type of output that is provided.

The identification of textual similarity and differences have been presented from two complementary perspectives – the Cosine Similarity method and Myers' O(ND) difference algorithm respectively. These metrics generate output which provides a baseline quality check in terms of potential malpractice.

As the current paper illustrates, LanguageCert takes the issues of cheating or malpractice extremely seriously. Ways in which LanguageCert does this have been illustrated above. It is clear from the illustrations that the issue of similarity / cheating / plagiarism must be tackled strenuously and continuously. The LanguageCert similarity detector outlined in this paper represents but one element in LanguageCert's striving to maintain honesty, integrity and fairness in LanguageCert's English language examinations.

References

Bin-Habtoor, A. S., & Zaher, M. A. (2012). A survey on plagiarism detection systems. International Journal of Computer Theory and Engineering, 4(2), 185.

Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., ... & Gruber, R. E. (2008). Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems (TOCS), 26(2), 1-26.

Coniam, D., Lampropoulou, L., & Cheilari, A. (2021). Online proctoring of high-stakes examinations: A survey of past candidates' attitudes and perceptions. English Language Teaching, 14(8), 58-72.

Connor, R. (2016). A tale of four metrics. In *Similarity Search and Applications: 9th International Conference, SISAP 2016, Tokyo, Japan, October 24-26, 2016, Proceedings 9* (pp. 210-217). Springer International Publishing.

Davoodifard, M. (2022). Automatic Detection of Plagiarism in Writing. Studies in Applied Linguistics & TESOL at Teachers College, Columbia University, 21(2), 54–60.

Diana, N. E., & Ulfa, I. H. (2019, March). Measuring performance of n-gram and Jaccard-similarity metrics in document plagiarism application. In Journal of Physics: Conference Series (Vol. 1196, No. 1, p. 012069). IOP Publishing.

Egger, R. (2022). Text Representations and Word Embeddings: Vectorizing Textual Data. In Applied Data Science in Tourism: Interdisciplinary Approaches, Methodologies, and Applications (pp. 335-361). Cham: Springer International Publishing.

Foster, D., & Layman, H. (2013). Online proctoring systems compared. Webinar. http://www. slideshare. net/caveonweb/caveon-webinar-series-online-proctoring-best-practicesoct-2013-slideshare-final.

Heres, D., & Hage, J. (2017, November). A quantitative comparison of program plagiarism detection tools. In Proceedings of the 6th computer science education research conference (pp. 73-82).

Huang, D., & Garner, M. (2009). A case of test impact: Cheating on the College English Test in China. *Language testing matters: Investigating the wider social and educational impact of assessment*, 59-76.

Hunt, E., Janamsetty, R., Kinares, C., Koh, C., Sanchez, A., Zhan, F., ... & Oh, P. (2019, November). Machine learning models for paraphrase identification and its applications on plagiarism detection. In 2019 IEEE International Conference on Big Knowledge (ICBK) (pp. 97-104). IEEE.

Idouglid, L., & Tkatek, S. (2023). Word Embedding Methods of Text Processing in Big Data: A Comparative Study. In Artificial Intelligence and Smart Environment: ICAISE'2022 (pp. 831-836). Cham: Springer International Publishing.

Indriyanto, I., & Sumitra, I. D. (2019, November). Measuring the level of plagiarism of thesis using vector space model and cosine similarity methods. In IOP Conference Series: Materials Science and Engineering (Vol. 662, No. 2, p. 022111). IOP Publishing.

Iqbal, Z., Anees, M., Khan, R., Hussain, I. A., Begum, S., Rashid, A., ... & Hussain, F. (2021). Cheating during examinations: Prevalence, consequences, contributing factors and prevention. *International Journal of Innovation, Creativity, and Change*, *15*(6), 601-609.

Irving, R. W. (2004). Plagiarism and collusion detection using the Smith-Waterman algorithm. *University of Glasgow*, *9*.

Küppers, R., & Conrad, S. (2012, September). A Set-Based Approach to Plagiarism Detection. In CLEF (Online Working Notes/Labs/Workshop).

Leonardo, B., & Hansun, S. (2017). Text documents plagiarism detection using Rabin-Karp and Jaro-Winkler distance algorithms. *Indonesian Journal of Electrical Engineering and Computer Science*, *5*(2), 462-471.

Mansoor, M. N., & Al-Tamimi, M. S. (2022). Computer-based plagiarism detection techniques: A comparative study. International Journal of Nonlinear Analysis and Applications, 13(1), 3599-3611.

Maurer, H. A., Kappe, F., & Zaka, B. (2006). Plagiarism-A survey. Journal of Universal Computer Science, 12(8), 1050-1084.

Mazov, N. A., Gureev, V. N., & Kosyakov, D. V. (2016). On the development of a plagiarism detection model based on citation analysis using a bibliographic database. Scientific and Technical Information Processing, 43, 236-240.

McCabe, D., Butterfield, K., & Trevino, L. (2012). Cheating in College: Why Students Do It and What Educators Can Do about It. The Johns Hopkins University Press.

Meo, S. A., & Talha, M. (2019). Turnitin: Is it a text matching or plagiarism detection tool? Saudi Journal of Anaesthesia, 13(1), 48-51.

Myers, E. W. (1986). An O(ND) difference algorithm and its variations. Algorithmica, 1(1-4), 251-266.

Naik, R. R., Landge, M. B., & Mahender, C. N. (2015). A review on plagiarism detection tools. International Journal of Computer Applications, 125(11), 16-22.

Noorbehbahani, F., Mohammadi, A., & Aminazadeh, M. (2022). A systematic review of research on cheating in online exams from 2010 to 2021. *Education and Information Technologies*, *27*(6), 8413-8460.

Nugroho, Y. S., Hata, H., & Matsumoto, K. (2020). How different are different diff algorithms in Git? Use--histogram for code changes. Empirical Software Engineering, 25, 790-823.

Park, C. (2003). In other (people's) words: Plagiarism by university students--literature and lessons. *Assessment & evaluation in higher education*, *28*(5), 471-488.

Pecorari, D. (2008). Academic writing and plagiarism: A linguistic analysis. Bloomsbury Publishing.

Qualifications and Curriculum Authority. (2007). Regulatory principles for e-assessment. https://publications.parliament.uk/pa/cm200607/cmselect/cmeduski/memo/test&ass/ucm3102paper4.pdf.

Ramos, J. (2003, December). Using TF-IDF to determine word relevance in document queries. In Proceedings of the first instructional conference on machine learning (Vol. 242, No. 1, pp. 29-48).

Saptono, R., Prasetyo, H., & Irawan, A. (2018). Combination of cosine similarity method and conditional probability for plagiarism detection in the thesis documents vector space model. Journal of Telecommunication, Electronic and Computer Engineering (JTEC), 10(2-4), 139-143.

Sjölund, M. (2021, September). Evaluating a Tree Diff Algorithm for Use in Modelica Tools. In Modelica Conferences (pp. 529-537).

So'ud, M. D. N. (2016). The Effect of Cheating in English Examinations on the Process of the Pedagogical Evaluation,17(4), 145-155.

Stein, B., Lipka, N., & Prettenhofer, P. (2011). Intrinsic plagiarism analysis. Language Resources and Evaluation, 45, 63-82.

Su, Z., Ahn, B. R., Eom, K. Y., Kang, M. K., Kim, J. P., & Kim, M. K. (2008, June). Plagiarism detection using the Levenshtein distance and Smith-Waterman algorithm. In *2008 3rd International Conference on Innovative Computing Information and Control* (pp. 569-569). IEEE.

Vijaymeena, M. K., & Kavitha, K. (2016). A survey on similarity measures in text mining. Machine Learning and Applications: An International Journal, 3(2), 19-28.

Wan, Y K and Li, H T (2006) A study on college students cheating in examinations, Journal of Yuxi Teachers College, 22 (3), 86–90.

Appendix 1: Categorising Similarity yet Difference in Texts

Example 1: Both texts exactly the same. *Similarity score 1.0*

Candidate Name 1	Candidate Name 2
I love summer. My favorite month is August. I eat watermelons in August. I hate winter.	I love summer. My favorite month is August. I eat watermelons in August. I hate winter.

Example 2: Changing the order of one sentence. *Similarity score 1.0*

Candidate Name 1	Candidate Name 2
~~I love summer.~~ My favorite month is August. I eat watermelons in August. I hate winter.	My favorite month is August. I eat watermelons in August. I hate winter. I love summer.

Example 3: Changing the order of two sentences. *Similarity score 1.0*

Candidate Name 1	Candidate Name 2
~~I love summer.~~ My favorite month is August.~~I hate winter.~~ I eat watermelons in August.	My favorite month is August. I eat watermelons in August. I hate winter. I love summer.

Example 4: Changing the order of all sentences. *Similarity score 1.0*

Candidate Name 1	Candidate Name 2
I ~~love summer. I eat watermelons in August~~. My favorite month is August.I ~~hate winter~~.	I hate winter. My favorite month is August. I love summer. I eat watermelons in August.

s

Example 5: Making typos: “favoritemonth” instead of “favorite month”. *Similarity score 0.81*

Candidate Name 1	Candidate Name 2
I ~~love summer. I eat watermelons in~~ August ~~.My favorite month is August.I hate winter.~~	I hate winter. August is my favoritemonth. I love summer. I eat watermelons in August.

“Favoritemonth“ and “ favorite month “ are considered two different words, by the scoring algorithm that is why the score changes

Language Cert
languagecert.org