
# Project Group 142: Code Cracking: Who Murdered The Somerton Man?

## Final Thesis

**Supervisor:** **Prof. Derek Abbott**
**Dr. Matthew Berryman**

Author: Yifan Ma
ID: a1658524

1. Introduction ..... 3
2. Background ..... 3
    2.1 About the Case ..... 3
    2.2 Previous Works ..... 5
        2.2.1 Australian Department of Defence ..... 5
        2.2.2 Previous Groups in the University of Adelaide ..... 5
    2.3 Motivation ..... 5
3. Technique Details ..... 5
    3.1 Hamming Distance and Levenshtein Distance ..... 5
    3.2 Simhash Algorithms ..... 6
        3.2.1 Brief Introduction of Simhash ..... 6
        3.2.2 Why Simhash? ..... 6
4. Tests ..... 9
    4.1 Levenshtein Distance Test ..... 10
        4.1.1 Data Processing ..... 10
        4.1.2 Preparation Test ..... 10
        4.1.3 Main Test ..... 12
        4.1.4 Conclusion of the Levenshtein Distance Test ..... 15
    4.2 2-grams Simhash Test ..... 15
        4.2.1 Data Processing ..... 15
        4.2.2 Preparation Test ..... 16
        4.2.3 Main Test ..... 17
        4.2.4 Conclusion of the Simhash Test ..... 18
    4.3 Summary of Tests ..... 18
5. Conclusions ..... 19
6. Acknowledgements and Future Works ..... 19
7. References ..... 19

# 1. Introduction

In this project, data comparison and statistical data analysis techniques were applied to crack a set of mysterious code extracted from an unsolved possible murder case. Data comparison is a branch of computer science, which calculates the differences and similarities among data objects, personified in this case: among a large amount of strings. Data analysis is to extract useful information from data to help generating conclusion and giving supports to it. The goal of this project is to find out in which language the code was written.

Initially, the contributions accomplished by previous project groups were reviewed and an insight was found to be a perfect beginning point of this project: it was suggested and supported that the code was very likely to be initialism in English, meaning that the code consists of first letters of a set of English words [1]. It was an amazing guess but unfortunately two problems were found after the reviewing of their works. Despite that the method applied to calculate similarities was lacking of convincing, it had not been updated for a long time. Besides, the test numbers were insufficient to draw out any conclusions.

Considering that similarity check algorithm is the most important component in this project, a large amount of time was spent on finding and testing a reliable algorithm. Several similarity check algorithms were researched and two of them were picked out to carry out the test. The carefully selected two algorithms should be good enough to discuss this case.

To deal with the other problem the test was designed to repeat multiple many times in order to get unbiased result. Writing scripts in Java simplified the repeating tests significantly. At the meantime, the hint from Professor Derek Abbott suggests to reduce the number of languages being considered in tests, which makes it possible to concentrate on the testing of several highly suspected languages.

The contributions in this project were listed as follow:

Selected and realized the Levenshtein Distance and the Simhash Algorithms using Java, which give output similarity values of two input text strings.

Classified and normalized the text materials for performing tests to suit the selected algorithms.

Applied statistical analysis to tests results and gain sufficient support that the code was initialisms written in English.

Paper Summary: The next section (Section 2) contains the detailed background of the project. Section 3 explains the principles and implementations of two algorithms. The design, implementation and analysis of tests are presented In Section 4. Conclusion is arranged to Section 5. Acknowledgements and future works are presented in Section 6. References are in Section 7.

# 2. Background

## 2.1 About the Case

The project is aiming to break an unsolved case of an unknown man found dead on Somerton beach, Glenelg. The victim, also known as the Somerton Man was found dead at 6.45 on 1st December, 1948.

After investigations the SA Police found a small wrinkled piece of paper in the man's pocket, with the Persian phrase “Tamám Shud” (translated to English: *to the end*)printed on it. The paper scratch was later confirmed to be part of an uncommon version of the *Rubaiyat of Omar Khayyam* book. Soon after the man's being found, the book was also found in the rear seat of an unlocked car parked 3km away from the location of the corpse.

Figure 1 The Somerton Man [2].

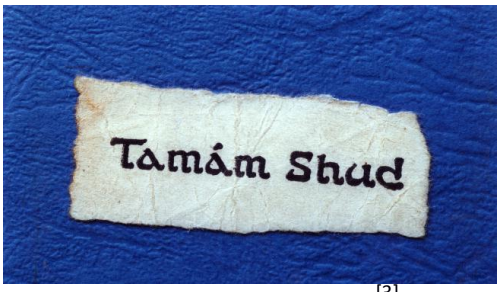



Figure 2 the paper scrap [3].

Despite the language, the book was not much different from ordinary books, but what really interesting was the handwriting of a set of mysterious code found on the back of the book. Considering that the main theme of the *Rubaiyat of Omar Khayyam* was about that people should not have any regrets when they die, combined with the meaning of the phrase on the paper scratch, it is widely believed that the book is strongly related to the dead man. Hence the case was widely known as the Tamam Shud case [4].

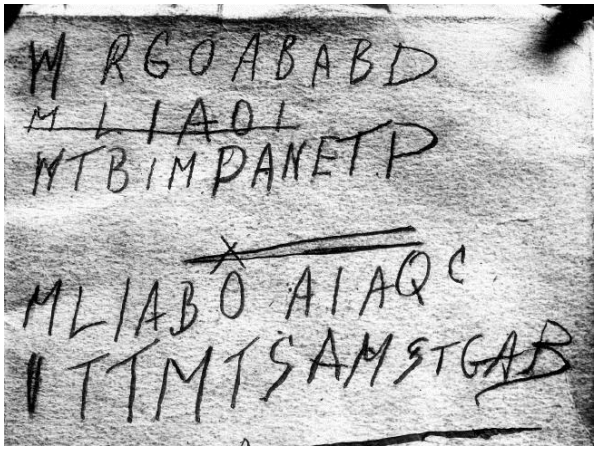



Figure 3. Handwriting found on the book [5]

As the figure above displayed, the code consists of fifty capital letters. For the sake of getting as many samples as possible the six crossed out letters are also included. There has been a dispute about the ambiguous “M” or “W” letters appear in the code, as a result two different versions of the code was tested (one with “M” and the other with “W”).

## 2.2 Previous Works

### 2.2.1 Australian Department of Defence

In response to the request from journalist Stuart Littlemore the Australian Department of Defence had worked on cracking the code left in the Tamám Shud case. Unfortunately after a time of working the cryptographers defined the code as unable to crack. The code was said to either "have insufficient symbols" or it was just a meaningless product generated under a "disturbed mind" [6].

### 2.2.2 Previous Groups in the University of Adelaide

As Professor Derek Abbott has been following this case for a long time, there were several project groups that had been working on this unsolved case from 2009 to the present. The conclusions drawn out by previous groups are: the code is unlikely to be generated randomly, the code is unlikely to be initial letters from words, the book *Rubaiyat of Omar Khayyam* was unlikely to be used as a one-time pad for encryption, the original language of the code is likely to be English, the code is unlikely to be initialisms extracted from poems, the book *Rubaiyat of Omar Khayyam* was not used as a straight substitution one-time pad for encryption and the code was not created using the *Rubaiyat of Omar Khayyam* as a one-time pad.[7][8][9][10][11][12]

## 2.3 Motivation

As the previous chapter stated, this project is relating to an unsolved possible murder case which has been remaining in the public's interest for almost 70 years. Just like Professor Derek Abbott, all the group members that have joined in this case must be extremely curious about the mysterious case. The desire of finding out the truth would be the most important motivation of undertaking this project. Another motivation comes from the ethical consideration. The victim has been resting in West Terrance Cemetery without a name for decades. It will be meaningful if the identity of the victim could be unveiled. This is also for the victim's family whom lost their relative and probably had no idea about it.

# 3. Technique Details

Here introduces several key concepts and techniques that have been applied in this project. As the main purpose of this thesis should not be algorithm engineering, the algorithm part is made as simplified as possible.

## 3.1 Hamming Distance and Levenshtein Distance

The two concepts were introduced from Information Theory. Both of them describe the amount of differences between two strings. The Levenshtein Distance is applied to calculate the difference

between two strings that consist of letters, while the Hamming Distance is used to compare two binary strings with same length.

The Hamming distance measures the minimum time of calculations (substitutions, precisely) required to transform string A into string B. For example, the Hamming distance between “1010” and “0010” is 1 as it requires substituting the first bit '1' in the first string with '0'. In this case the calculations of Hamming distance are based on pure binary strings so the Hamming distance can be easily expressed as *H(a, b) = a XOR b.*

The Levenshtein Distance, also known as edit distance, is an enhanced version of the Hamming Distance. It not only counts substitution, also it considers insertions and deletions. The Levenshtein distance between two words is the smallest calculation times of **substitutions, insertions, and deletions** of symbols that are used to transform one string into another.

Here is an example demonstrating the calculation of the Levenshtein distance, substitution is marked as **s** and **d** stands for deletion, **i** for insertion. String1: INTENSION, string2: EXECUSION.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| I | N | T | E | # | N | S | I | O N |
| \| | \| | \| | \| | \| | \| | \| | \| | \| |
| # | E | X | E | C | U | S | I | O N |
| \| | \| | \| | \| | \| | \| | \| | \| | \| |
| d | s | s | — | i | s | ——— | | |

Table1. Levenshtein distance

According to Table1, the minimum cost to turn string1 into string2 is 5: 3 substitutions, 1 deletion and 1 insertion. As the Levenshtein Distance considers three kinds of calculations, the complexity is inherently higher than the Hamming Distance.

## 3.2 Simhash Algorithms

### 3.2.1 Brief Introduction of Simhash

The Simhash algorithm was originally invented by Moses Charikar. It was invented to estimate the similarities of a large volume of data [13]. Later the Simhash was applied by Google as their duplicate removal algorithm to deal with Google's massive data. Charikar's algorithm has been proved to be practically useful for identifying near-duplicates in web documents belonging to a multi-billion page repository [14] in Google's thesis. The idea of the Simhash algorithm are extremely condensed, it is even easier than the algorithm of finding all fingerprints with Hamming Distance less than k in Google's thesis mentioned above. As the algorithm performs well in similarity check, it is adapted here to calculate the similarities of 2-grams strings in tests.

### 3.2.2 Why Simhash?

Traditional similarity check algorithms use Vector Space Model to separate documents into

individual terms, allocate these terms into their corresponding vectors in multidimensional space. Each dimension indicates an individual term. Values of these vectors are calculated by a specific algorithm based on the terms, mostly based on the occurrence frequencies of terms, high frequency terms will be assigned a relatively larger value. Algorithms may differ but the Vector Space Model is static. After the modeling text will be transformed into a set of vectors.

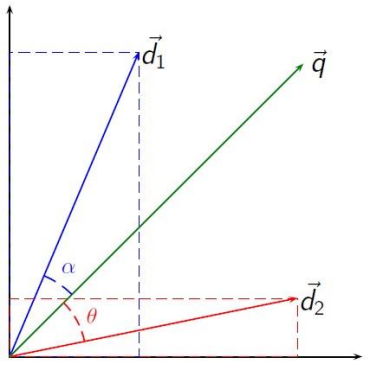



Figure 4 Vector Space Model

Figure 4 illustrates an example of the modeling of Vector Space Model, $d_1$ and $d_2$ represent two texts that have already been modeled. The two documents' similarity can be found by calculating the Cosine Similarity of their corresponding vectors (dot product of the two vectors divided by the product of the two vectors' Euclidean lengths):

$$\textbf{\textit{Similarity} } \langle d_1, d_2 \rangle = \cos(\theta_{d1,d2}).$$

Notice that the usage of a 2-D dimension was just for illustration purpose. In practice the number of dimension can be a much higher value, but the principle will be all the same.

It has been proved that the Vector Space Model brings accurate estimation of the similarities, but this comes at the expensive of the exceptional high complexity in both time domain and space domain. Modeling long texts into vectors will certainly take up a large volume of storage; on top of that, each similarity value is generated by calculating the cosine value of two vectors. Obviously it is not a wise choice when dealing with data that contains a large amount of terms. Using SimHash algorithm could reduce the complexity significantly while preserve the accurate estimation of similarities.

Before explaining the Simhash, it is necessary to introduce the Hash function. A Hash function mappings different data of arbitrary size into totally different hash values of a fixed size. After hash mapping each term will be allocated a unique hash value as its fingerprint. A well-defined Hash function should be collision-resistant, which means that it is impossible to find two data sets that will generate identically the same Hash values. Also, the Hash function should be sensitive to trivial changes. Even the string changes only one bit, the Hash value will be totally different.

Here explains the Simhash algorithm in details. The figure below demonstrates the general processes of calculating the Simhash fingerprint of a given document.

1. Firstly the original document (big blue box) will be separated into n individual terms (small blue bars). The rule of separating can be customized in different cases. Here, in the next tests the document will be separated into 2-grams letter groups.

2. Now the separation has been done. For each term, calculate its corresponding weight ($w_1$, $w_2$, …

$w_n$). Normally the weight is determined by the frequency of occurrence of each term. The weight can also be customized to meet specific requirements.

3. For each individual term and its corresponding weight: Apply hash mapping to each term (the length of hash mapping (n) could be adjusted by adjusting the segmentation method in step1, Google used 64-bits hash mapping in its webpage duplication remove program), then multiple the n-bits hash mapping result with the corresponding weight, '0' in hash mapping result will be treated as '-1' and '1' stays unchanged.

4. Finally, add all the results generated in the previous step together. Here the result is a string consists of n numbers. The final fingerprint is also an n-bit long binary string. For each number in the string, if it is positive then set its corresponding bit in the final result to '1', otherwise set the bit to '0'.

*5. The similarities of two strings are then generated by calculating the Hamming Distance of two Simhash strings.

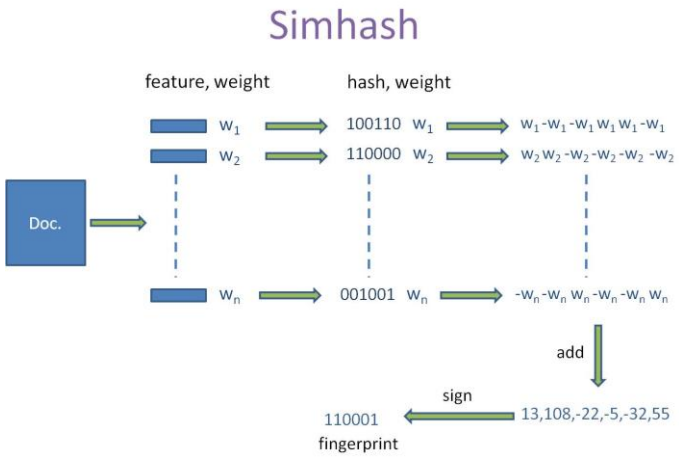



Figure 5 Simhash Procedures [16]

This is how the Simhash algorithm mappings a given long text into an n-bit long fingerprints. After the previous illustration, the geometric meaning of this algorithm is quite explicit: Firstly it mappings the long text into an n-dimension space, each individual term is transformed to a vector in the space. By using hash mapping, all the terms could be guaranteed to be transformed into a set of vectors which can be seen as nearly uniformly distributed. By multiplying weights and summing up together, the result can be called as a “sum-vector”. The “sum-vector” is then compressed by mapping positive values to 1 and others to 0, this operation is actually preserving the quadrant information of the “sum-vector”. Assume that n = 64, a 64-bit long fingerprint can express as many as $2^{64}$ quadrants, which seems to be enough to represent a specific document.

Theoretically the algorithm is reasonable, but the reason why the n-bit fingerprint has the ability to manifest similarities between documents is still unknown. Nor did Charikar (The inventor of the Simhash algorithm) give out any justification. Nevertheless, tests that have been done illustrates that the Simhash algorithm actually does a good job in similarity estimations. Here presents a simple example of similarity estimation based on the Simhash algorithm and the Hamming Distance.

String A (initialism of the lyrics of “Bohemian Rhapsody” by Queen), String B (copied from String A with 3 letters modified) and String C (an arbitrary string) are strings being tested in the Simhash

algorithm.

String A: *ITTRLITJFCIALNEFROYELUTTSASIJAPBINNSBIECEGALHLLATWBDRMTMTM*

String B: *ITTRQITJFCIALNEFRQYELUTTSASIJAPBINNSBIECEZALHLLATWBDRMTMTM*

String C: *KAYCDRKBPQOGVTAACDUQKXJNZNZMXCBNUKPHVODWUUSQGJZFFYUKHBDMFY*

After the 2-grams separation, the Simhash of each strings were presented below (red characters indicate the differences):

SimHash of A: 10011100100010000111100001101011011110001101111000 1001

SimHash of B: 10011100110010000111100001111011011110001101111000 1001

SimHash of A: 10011100100010000111100001101011011110001101111000 1001

SimHash of C: 10011010101101010110110001100101001110000111111111 0010 10

By observing, the Hamming Distance between two Simhash strings of A and B should be 3, which is low enough to indicate that the two strings are extremely similar. Actually they are extremely similar (only 3 letters out of 59 were different). While the Hamming Distance between A and C is 21, indicating that the two strings A and C are unlikely to have any similarities.

# 4. Tests

There are mainly two aspects of tests in this project, the Direct Comparison Test and the 2-grams Comparison Test. Nine different languages are selected and tested, they are English, Italian, French, German, Portuguese, Latin, Spanish, Turkish and Polish.

As this section may be too long and disturbing, here presents a simple chart for the navigation usage in Figure 6.

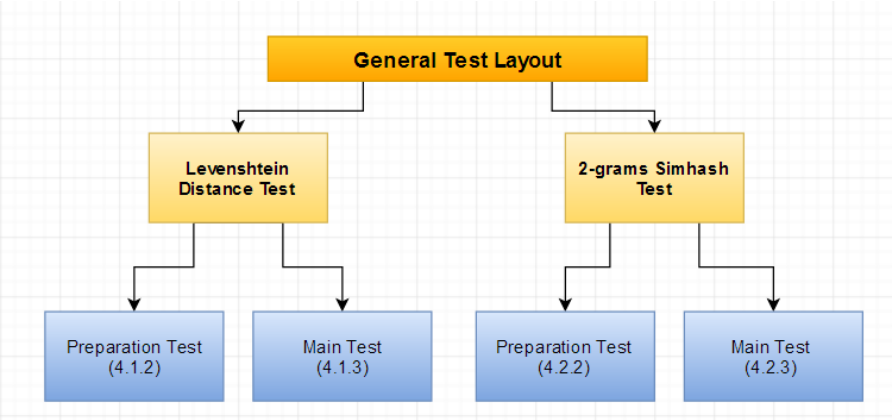



Figure 6 Test Layout

## 4.1 Levenshtein Distance Test

The comparison method of this section is calculating the Levenshtein Distance between two initialism strings directly. Materials used for comparison, length of string and comparison time may vary based on the purposes of setting up each test.

### 4.1.1 Data Processing

All the materials for this test were transformed into Initialisms format, with all punctuations and spaces removed, for example:

Original Text: "Snow is falling, all around me, children playing, having fun…"

Initialisms Format: "SIFARMCPHF…"

Initialisms were then divided into segmentations of a given length. Each segmentation is referred as 'unit' in the next sections.

### 4.1.2 Preparation Test

It is always necessary to check if the method for tests is effective before starting any tests. Hence the purpose of this section is **to prove that the Levenshtein distance algorithm has the ability to revealing the differences between different languages.**

Units of a fixed length extracted from English version and eight different language versions of the War and Peace were compared. To make it comprehensive, there are six groups of tests with different length varies from 50 letters/unit to 800 letters/unit and in each length group there are 100 times of comparison (**10 units of one language compared with 10 units of another language one by one using a double 'for loop'** ). Each unit was uniquely selected from texts without any overlapping. Results of these tests are normalized with the unit of length = 50.

The table below is for a quick reference.

| Materials | War and Peace versus War and Peace |
|---|---|
| Unit | Consecutive Initialisms of a fixed length. |
| Unit Length | Ranging from 50 letters / Unit to 800 letters / Unit. |
| Time of Comparisons | 100 |
| Operation | Calculate the Levenshtein Distance between two Unit |

The result of the tests was presented in boxplots in the figure below (Language names were abbreviated, En stands for English, It-Italian, Fr-French, Ge-German, Pq-Portuguese, La-Latin, Sp-Spanish, Tr-Turkish and Po-Polish). Each box is a set of the output Levenshtein Distances between two texts. For example, the leftmost box represents the test results of 10 English units versus another 10 English units; same rule applies for the others.

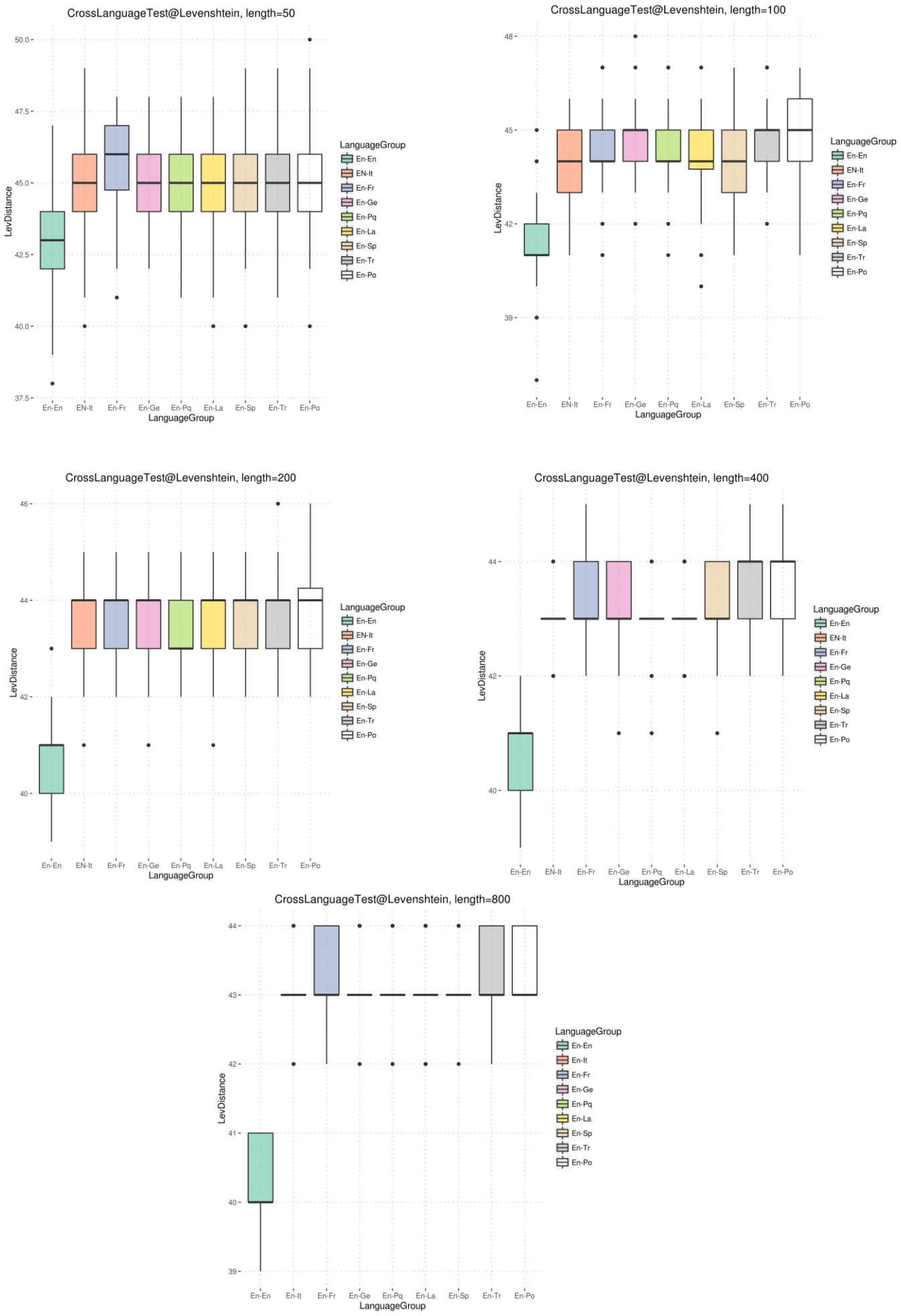



Figure Group1. Levenshtein Preparation Test Result

By analyzing the five box plots above in the Figure Group 1, the following conclusions can be inferred:

1. In all the five figures, the leftmost boxes are significantly lower than the others, which mean that the Levenshtein distances between English strings are much lower than those between English strings and strings of other languages. **Hence, the Levenshtein Distance is able to manifest the difference between different languages.**

2. As the unit length grows the boxes become lower and more compact (median value and standard deviation shrink when unit length goes up). Some of them even converged into a specific value. In addition, the difference between the last two plots (unit length = 400 and unit length = 800) is trivial. It means that **the Levenshtein distances tend to be stable as the length of the unit increases.**

3. When increasing the unit size, the trends of the nine language groups are highly similar: median values of the Levenshtein Distances decrease as the unit size increases. This implies that **the performance of the Levenshtein Distance algorithm is independent with different languages.**

### 4.1.3 Main Test

The previous section has proved that the Levenshtein Distance actually can reveal the differences between languages. Based on this, the test in this section is **to find out the difference between the mysterious code and texts of varies language.**

In the experimental group (line plots in figures), two versions the mysterious code were compared with 100 units of different languages extracted from the War and Peace. As the code is only of 50 letters long, there is no need to carry out the test in different unit length.

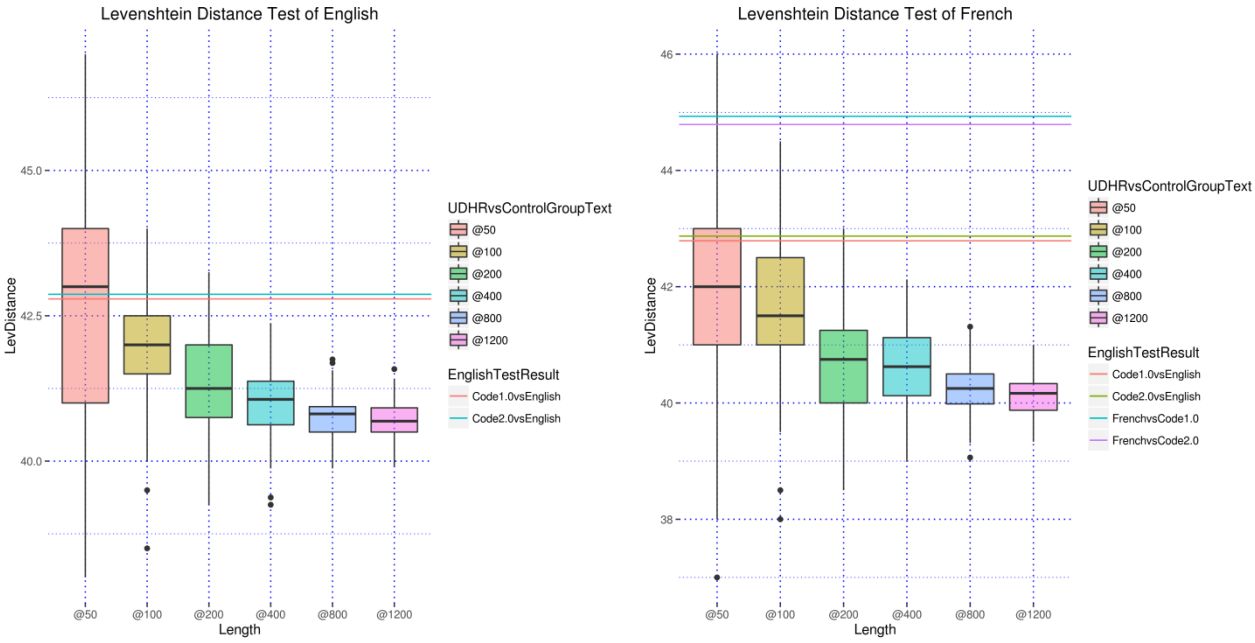
In the comparison group (box plots in figures), units extracted from the UDHR were compared with the same materials in the experimental group. Like the preparation test, this test was also divided into 6 different length groups. 100 times of comparison was made, the 10 vs 10 double for loop structure was also preserved here.

The reason why Latin language was not tested is because that the Latin version of the War and Peace was not found, and lacking of material for the control group test would affect the confidence.

| | |
|---|---|
| Materials | Codes/UDHR versus War and Peace respectfully |
| Unit | Consecutive Initialisms of a fixed length. |
| Unit Length | Ranging from 50 letters / Unit to 1200 letters / Unit. |
| Time of Comparisons | 100 |

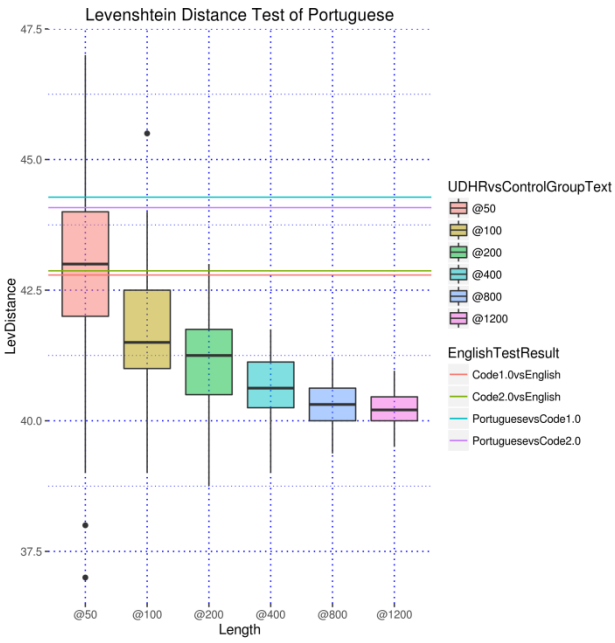
| Operation | Calculate the Levenshtein Distance between two Unit |
|---|---|

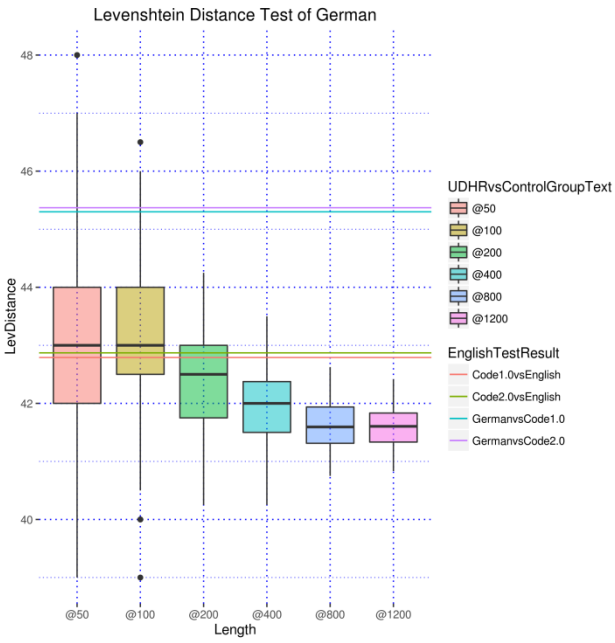
Results are presented in the following figures. The horizontal line indicates the mean (average value) of the experimental group's result and the boxes indicate results of the comparison group. For example, in the first plot: the six boxes represent the test result of UDHR units in English versus War and Peace units in English in six different length groups. Two lines indicate the test result of two versions of code versus units of War and Peace in English. The "Codes vs English" test results (red and green lines) appear in every plot as a reference.









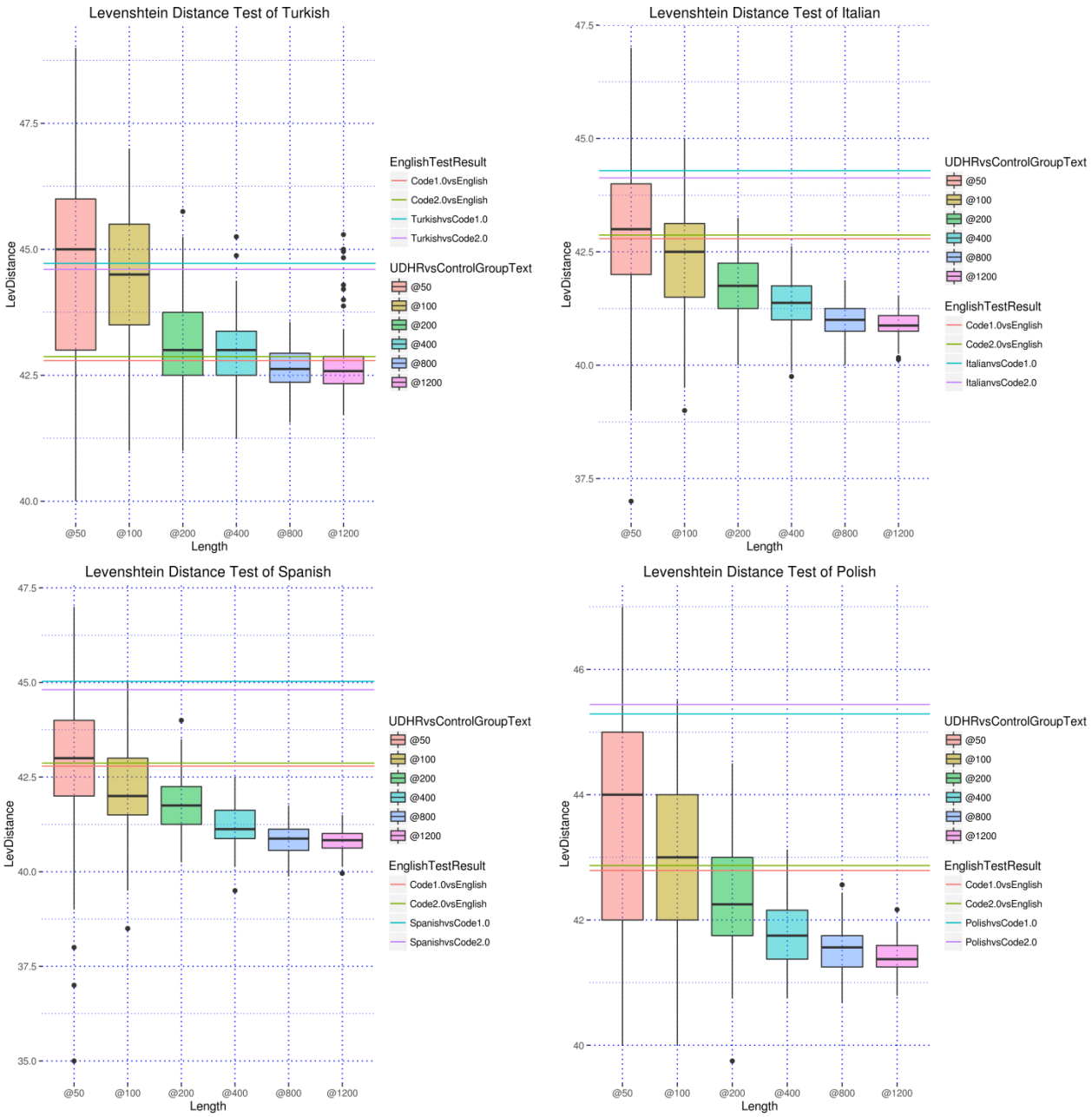



Figure Group 2. Result of Levenshtein Main Test

The following interesting phenomenon could be observed from the Figure group2 above:

1. Observing within each plot: both the median value and the standard deviation fall as the unit length increases. This trend fits reasonably well with the trend of the preparation test (the one mentioned in the 2nd conclusion of section 4.1.2).

2. Observing from the second plot to the last plot: there are distinct gaps between the Levenshtein Distances of codes versus English text (red and green lines), and the Levenshtein Distances of codes versus other languages (blue and purple lines).

3. Among all the languages tested except English and Turkish tests, results of the experimental group are consistently higher than results of the comparison group. In addition, the experimental group's result in Turkish was significant higher the one in English.

4. Observing inside of the English test plot and the Turkish test plot: both the two groups of lines located around the median value of test result in 50 letters length group.

Conclusion:

According to the aforementioned phenomenon, the corresponding inferences can be drawn out:

1. The similarity of two trends reveals that the Levenshtein Distance algorithm does not sensitive to different testing materials. On the other hand, the relatively higher standard deviation in 50 letters group implies the higher randomness in small length tests. Notice that results of the experimental group are literally of 50-letters group. This reduces the credibility of the whole test.

2. Within the scope of the experimental group, the mysterious code has relatively smaller Levenshtein Distance with English text compared to those with other languages.

3. The differences between results of experimental and comparison groups in six languages (German, Italian, French, Portuguese, Spanish and Polish) imply that the code is unlikely to be one of these languages.

4. In English and Turkish test results, the overlapped results from experimental and comparison groups imply the consistencies of the Levenshtein Distances between 'Code vs War and Peace' and 'UDHR vs War and Peace'.

### 4.1.4 Conclusion of the Levenshtein Distance Test

According to the previous analysis, the code has more possibility to be initialisms of English or Turkish text, rather than of texts in other languages. In addition, considering the higher Levenshtein Distance in "code vs Turkish" test, being initialisms of English is more possible than being initialisms of Turkish.

On the other hand, the credibility of the test was challenged by the existence of randomness in small length group test. Hence the conclusion that the code is more likely to be English should be considered cautiously.

## 4.2 2-grams Simhash Test

Though the previous test has drawn out some useful conclusions, its lacking of credibility is still unacceptable. As a result, the 2-grams test with another algorithm (SimHash algorithm) has been designed and implemented.

The test is expected to give supports to the conclusion drawn out in the previous Levenshtein Distance Test. In the meantime, it is expected to have a better credibility.

### 4.2.1 Data Processing

Another difference was that texts for this test were rearranged into the 2-grams format based on the initialisms format in the previous test. For example:

Original String: “The visions dancing in my mind the early dawn the shades of time …”

Initialisms format: “TVDIMMTEDTSOT ...”

2-grams format: “TV VD DI IM MM MT TE ED DT TS SO OT T…”

### 4.2.2 Preparation Test

Just like before, the preparation test was designed to check the performance of algorithm. Unlike the Levenshtein Algorithm, the grouping of test based on unit length is meaningless. This is because of the nature of the Simhash algorithm. For details please refer to section 3.2.2.

The preparation test was based on the UDHR. In the first test: it compared 50 units extracted from the English version of UDHR with the whole text of UDHR (2-grams formatted as well) in both English and other languages, by firstly turning each unit to its corresponding Simhash string, then calculating the Hamming Distance between two Simhash strings. In the second test, the same method and layout are used again to compare 50 units with the whole text of UDHR in the same kind of language. To make it simple, test 1 is a cross-language test while test2 is a same-language test. Results were presented in the two box plots below (apologize for the being out of order of the x-axis; please make comparisons according to the column names under each box):

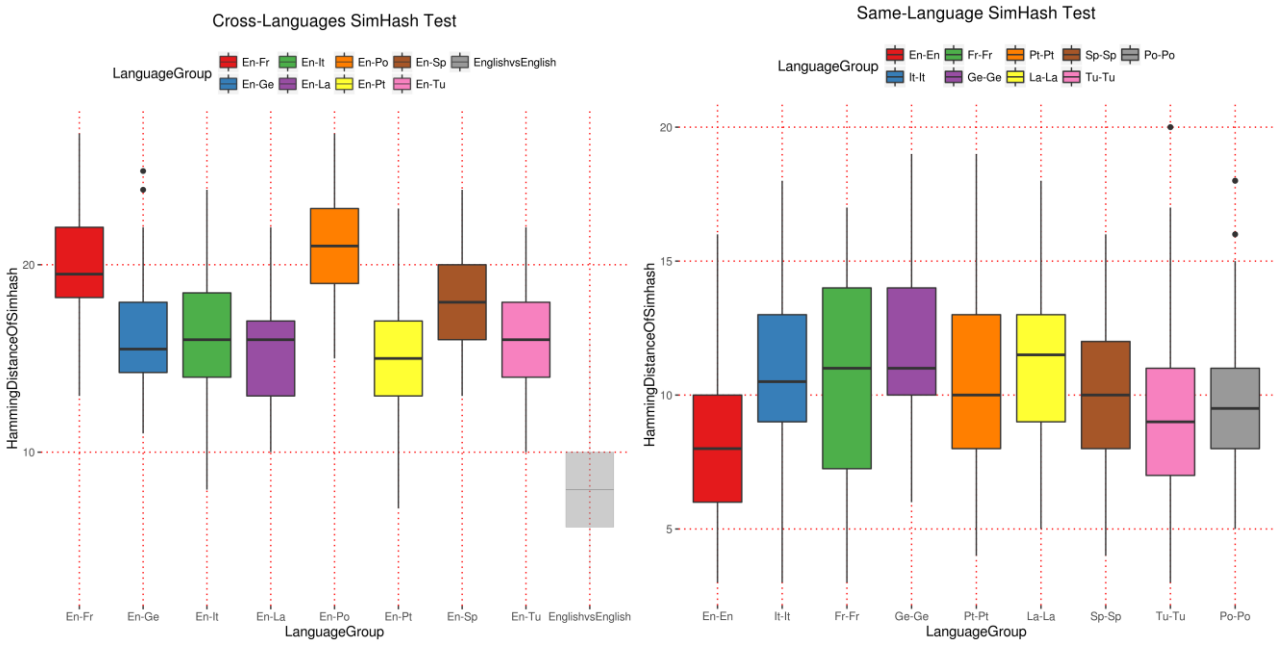



Figure 20 and 21.

Comparing inside the cross-language group (Figure 20): the data set of “English versus English” (the more transparent box located at the bottom-right of the first figure) is considerably lower than other cross-language data sets (colored boxes in the first figure).

Comparing across the two figures (Figure 20 and 21): it is easy to find out that the cross-language group has generally higher results than the same-language group (*Color schemes in two figures are not the same , please refer to the column names when doing comparison*). Median values of boxes in same-language group are all significantly lower than those in the cross-language group. All third

quartiles (Q3) in the same-language group are lower than first quartiles (Q3) of the corresponding language group in the cross-language group.

In addition, data sets in figures above are less biased compared to those in Levenshtein Test. Distributions of data in each boxplots are quite compact. The Simhash Algorithm is not sensitive to different languages. These facts give extra credibility to the Simhash algorithm.

**Based on the aforementioned observations, it is reasonable to draw out the conclusion that the Simhash Algorithm has an excellent ability of distinguishing different kinds of languages.**

### 4.2.3 Main Test

Here, the two versions of the mysterious code are tested against the UDHR in different languages. For each version of code, it was compared to different language versions of the UDHR. Results are presented in the first bar chart. Pink represents the first version of the code and blue represents the other one. (Each bar actually consists of two sub-bars, length of sub-bar represents each result, and there is no overlapping between sub-bars).

As there is little difference between pink and blue bars, the average value in each language group is taken. For comparison purpose, these average values are combined with the same-language test result, and presented in the second figure below (bars in the left figure are turned to horizontal lines in the right figure, which represent the mean value of the Hamming Distance set between code and a specific language):

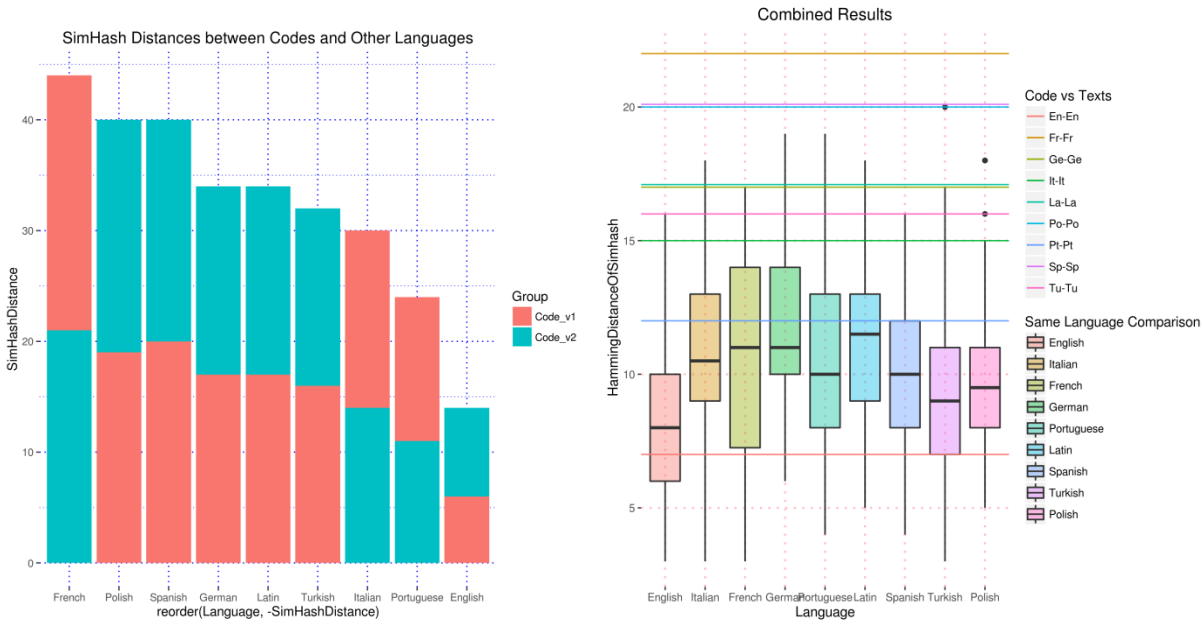



Figure 22 and 23

Considering that the comparison between two versions of code and the UDHR texts may not be sufficient to draw out any convincing conclusion, the two versions of code are compared again with 12000 letters long War and Peace text in each language using the same Simhash method. As the two versions of code have extremely similar test results against a specific language, the two versions' results are added together in each language group. Result of the test is shown below:

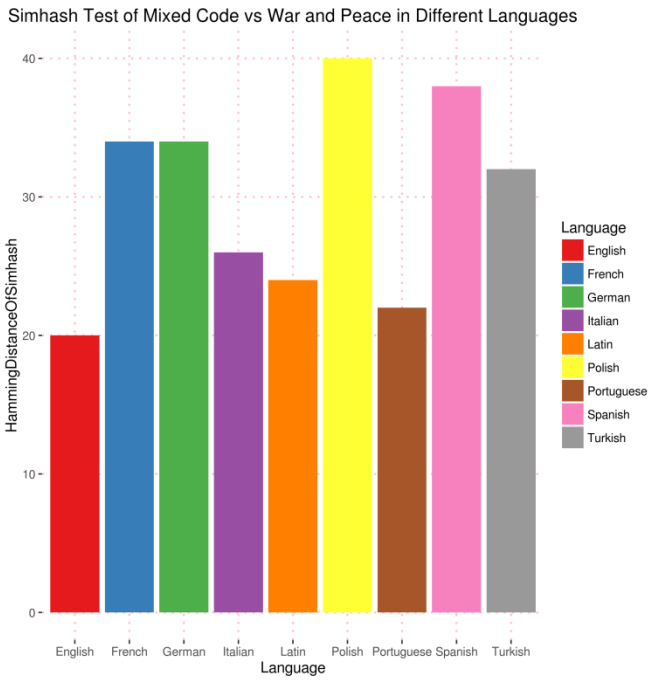



Figure 24

Figure 22 clearly showed that the Hamming Distance between English and the code is the minimum; figure 24 also presents the same characteristic.

In Figure 23, there are two horizontal lines reside inside of their corresponding boxes: English and Portuguese. But compared with Portuguese, the red line that represents the result of code vs English is relatively closer to the median value of the English vs English test result set.

### 4.2.4 Conclusion of the Simhash Test

Based on these facts, the conclusion is that the mysterious code is more likely to be English, rather than other eight languages being tested.

## 4.3 Summary of Tests

In Section 4, there are generally two groups of tests presented: the Levenshtein Test and the 2-grams Simhash Test.

The Levenshtein Test has given the opinion that the mysterious code is more likely to be English initialisms; but its credibility was challenged by the randomness appeared in the 50-length group test. On the other hand, the opinion inferred from the Levenshtein Test has gained vigorous support by the result in the 2-grams Simhash Test.

For the secondary suspected languages which are Turkish and Portuguese respectfully in two tests. By comparing results of two tests together the suspicion can be removed. As the Turkish test result in the 2-grams Simhash test was not as conspicuously high as the one in the Levenshtein Test, nor did the Portuguese test in the Levenshtein test perform abnormally.

**Now, there are enough evidences to support the final conclusion that the code consists of Initialism of English.**

# 5. Conclusions

In this project, two new algorithms have been researched and introduced to check the similarities between the mysterious code and texts of different languages. Algorithms have been implemented in Java. Text materials for each test have been gathered and arranged into ideal formats. Individual test for each algorithm has been designed and implemented. After analyzing test results the conclusion that **the code consists of Initialism of English** has been drawn out.

# 6. Acknowledgements and Future Works

This project was supported by the School of Electrical & Electronic Engineering of the University of Adelaide. The author would like to thank all the staffs who have been supporting the Final Year Project for a long time. The author would like to thank all the members in previous project groups; they had made great contributions to the project. At the very end, the author would like to gratefully acknowledge the technical support and guidance from Prof. Derek Abbott and Dr. Matthew Berryman, the project will be nothing without their generous advices and helps.

Being restricted by time, a lot of meaningful works were not performed. The author will keep following up this interesting project. The next several tasks are expected to be finished:

1. Extend the two tests to other languages which have not been tested.

2. Perform more n-grams test using the Simhash Algorithm.

3. Increase the number of tests in the 50-letters group Levenshtein test to see if the randomness can be avoided.

# 7. References

[1]. L. Griffith and P. Varsos. (2013). Semester B Final Report 2013 – Cipher Cracking . Available: http://www.adelaidenow.com.au/news/south-australia/somerton-man-mystery-new-details-revealed-of-jo-thomson-nurse-in-the-case/news-story/4c6bccbd2318584ad0cc6daaf3d8abd4

[2]. Renato Castello, “New twist in Somerton Man mystery as fresh claims emerge,” Sunday Mail SA, November 23th, 2013. Access via Internet:

http://www.adelaidenow.com.au/news/south-australia/new-twist-in-somerton-man-mystery-as-fresh-claims-emerge/story-fni6uo1m-1226766905157

[3]. Lynton Grace, “Somerton Man mystery: New details revealed of Jo Thomson, nurse in the case”, The Advertiser, 29th May 2015. Access via Internet:

http://www.adelaidenow.com.au/news/south-australia/somerton-man-mystery-new-details-revealed-of-jo-thomson-nurse-in-the-case/news-story/4c6bccbd2318584ad0cc6daaf3d8abd4

[4]. From Wikipedia, the Taman Shud Case. Access via Internet:

https://en.wikipedia.org/wiki/Tamam_Shud_case

[5]. From Internet:

http://ciphermysteries.com/wp-content/uploads/sites/6/2014/01/SomertonManCode-wikipedia.jpg

[6]. Inside Story, presented by Stuart Littlemore, ABC TV, screened at 8 pm, Thursday, August 24th, 1978.

[7]. A. Turnbull and D. Bihari. (2009). Final Report 2009: Who killed the Somerton man? Available:

https://www.eleceng.adelaide.edu.au/personal/dabbott/wiki/index.php/Final_report_2009:_Who_killed_the_Somerton_man%3F

[8]. K. Ramirez and L-V. Michael. (2010). Final Report 2010 . Available:

https://www.eleceng.adelaide.edu.au/personal/dabbott/wiki/index.php/Final_Report_2010

[9]. S. Maxwell and P. Johnson. (2011). Final Report 2011 . Available:

https://www.eleceng.adelaide.edu.au/personal/dabbott/wiki/index.php/Final_Report_2011

[10]. A. Duffy and T. Stratfold. (2012). Final Report 2012 . Available:

https://www.eleceng.adelaide.edu.au/personal/dabbott/wiki/index.php/Final_Report_2012

[11]. L. Griffith and P. Varsos. (2013). Semester B Final Report 2013 – Cipher Cracking . Available:

https://www.eleceng.adelaide.edu.au/personal/dabbott/wiki/index.php/Semester_B_Final_Report_2013_-_Cipher_cracking

[12]. N. Gencarelli and J-K. Yang. (2015). Semester B Final Report 2015 – Cipher Cracking . Available:

https://www.eleceng.adelaide.edu.au/personal/dabbott/wiki/index.php/Final_Report/Thesis_2015

[13]. MS. Charikar. (2002). Similarity estimation techniques from rounding algorithms. Available:

https://www.cs.princeton.edu/courses/archive/spr04/cos598B/bib/CharikarEstim.pdf

[14]. G.S. Manku, A. Jain and A. Das Sarma. (2007). Detecting Near-Duplicates for Web Crawling. Conference on World Wide Web.

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.78.7794&rep=rep1&type=pdf

[15]. From Wikipedia, the Vector Space Model. Access via Internet:
https://en.wikipedia.org/wiki/Vector_space_model#/media/File:Vector_space_model.jpg

[16]. From Internet:

http://static.oschina.net/uploads/img/201308/30125158_L1CI.jpg